



Informatica® Multidomain MDM
10.5 HotFix 2

Configuration Guide

Informatica Multidomain MDM Configuration Guide
10.5 HotFix 2
October 2023

© Copyright Informatica LLC 2001, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, the Informatica logo, and ActiveVOS are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. 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. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2024-02-12

Table of Contents

Preface	25
Informatica Resources.	25
Informatica Network.	25
Informatica Knowledge Base.	25
Informatica Documentation.	25
Informatica Product Availability Matrices.	26
Informatica Velocity.	26
Informatica Marketplace.	26
Informatica Global Customer Support.	26
 Part I: Introduction.....	 27
 Chapter 1: Informatica MDM Hub Administration.....	 28
Informatica MDM Hub Administration Overview.	28
Phases in Informatica MDM Hub Administration.	28
Startup Phase.	29
Configuration Phase.	29
Production Phase.	29
 Chapter 2: Getting Started with the MDM Hub Console.....	 30
Overview.	30
About the MDM Hub Console.	30
Start the Hub Console.	30
Navigating the MDM Hub Console.	32
Toggling Between the Processes and Workbenches Views.	32
Starting a Tool in the Workbenches View.	32
Acquire Locks to Change Metadata.	33
Changing the Target Database.	35
Logging in as a Different User.	35
Changing the Password for a User.	36
Using the Navigation Tree in the Navigation Pane.	36
Adding, Editing, and Removing Objects Using Command Buttons.	38
Customizing the MDM Hub Console interface.	39
Showing Version Details.	39
Informatica MDM Hub Workbenches and Tools.	40
Tools in the Configuration Workbench.	40
Tools in the Model Workbench.	41
Tools in the Security Access Manager Workbench.	41
Tools in the Data Steward Workbench.	42
Tools in the Utilities Workbench.	42

Chapter 3: Configuring International Data Support.	43
Configuring International Data Support Overview.	43
Configuring a Unicode Database (Oracle Only).	43
Configuring Match Settings for Non-US Populations.	44
Configuring Encoding for Match Processing.	44
Using Multiple Populations Within a Single Base Object.	45
Configuring the ANSI Code Page in the Windows Registry.	45
Cleanse Settings for Unicode.	46
Locale Recommendations for UNIX When Using UTF-8.	46
Troubleshooting Corrupted Data.	46
Configuring Language in Oracle Environments.	47
Syntax for NLS_LANG.	47
Configuring NLS_LANG in the Windows Registry.	47
Configuring NLS_LANG as an Environment Variable (Windows).	48
Setting the LANG Environment Variables and Locales (UNIX).	48
 Part II: Configuring Hub Console Tools.....	50
 Chapter 4: Configuring Access to Hub Console Tools.....	51
Configuring Access to Hub Console Tools Overview.	51
User Configuration.	51
User Access to Tools and Processes.	52
Starting the Tool Access Tool.	52
Granting User Access to Tools and Processes.	52
Revoking User Access to Tools and Processes.	52
 Chapter 5: Implementing Custom Buttons in Hub Console Tools.....	53
Overview.	53
About Custom Buttons in the Hub Console.	53
What Happens When a User Clicks a Custom Button.	54
How Custom Buttons Appear in the Hub Console.	54
Adding Custom Buttons.	54
Writing a Custom Function.	55
Controlling the Custom Button Appearance.	57
Deploying Custom Buttons.	57
 Part III: Building the Data Model.....	59
 Chapter 6: About the Hub Store	60
Overview.	60
Databases in the Hub Store.	60
How Hub Store Databases Are Related.	61

Creating Hub Store Databases.	61
Version Requirements.	61
Chapter 7: Configuring Operational Reference Stores and Data Sources.	62
Configuring Operational Reference Stores and Data Sources Overview.	62
Before You Begin.	62
About the Databases Tool.	63
Starting the Databases Tool.	63
Configuring Operational Reference Stores.	63
Operational Reference Store Connection Properties for Microsoft SQL Server.	64
Operational Reference Store Connection Properties for Oracle.	64
Operational Reference Store Connection Properties for IBM DB2.	66
Registering an Operational Reference Store.	67
Editing Operational Reference Store Registration Properties.	67
Editing Operational Reference Store Properties.	69
Operational Reference Store Connections.	70
Changing Passwords.	71
Password Encryption.	72
Production Mode for an Operational Reference.	72
Unregistering an Operational Reference Store.	73
Dropping an Operational Reference Store in IBM DB2.	73
Data Source Configuration.	73
Managing Data Sources in WebLogic.	74
Creating Data Sources.	74
Removing Data Sources.	74
Chapter 8: Building the Schema.	75
Overview.	75
Before You Begin.	75
About the Schema.	75
Types of Tables in an Operational Reference Store.	76
Requirements for Defining Schema Objects.	78
Starting the Schema Manager.	93
Configuring Base Objects.	94
Relationships Between Base Objects and Other Tables in the Hub Store.	95
Process Overview for Defining Base Objects.	95
Base Object Columns.	96
Cross-Reference Tables.	97
History Tables.	101
Base Object Properties.	101
Creating Base Objects.	104
Editing Base Object Properties.	105
Custom Indexes for Base Objects.	105

Viewing the Impact Analysis of a Base Object.	107
Deleting Base Objects.	107
Configuring Columns in Tables.	107
About Columns.	108
Navigating to the Column Editor.	112
Adding Columns.	113
Importing Column Definitions From Another Table.	113
Editing Column Properties.	114
Changing the Column Display Order.	115
Deleting Columns.	115
Configuring Foreign-Key Relationships Between Base Objects.	116
About Foreign Key Relationships.	116
Process Overview for Defining Foreign-Key Relationships.	116
Adding Foreign Key Relationships.	117
Editing Foreign Key Relationships.	117
Relationship Details.	118
Configuring Lookups for Foreign-Key Relationships.	118
Deleting Foreign Key Relationships.	119
Viewing Your Schema.	119
Starting the Schema Viewer.	119
Zooming In and Out of the Schema Diagram.	120
Switching Views of the Schema Diagram.	121
Navigating to Related Design Objects and Batch Jobs.	121
Configuring Schema Viewer Options.	121
Saving the Schema Diagram as a JPG Image.	122
Printing the Schema Diagram.	122
Chapter 9: Queries and Packages.	124
Queries and Packages Overview.	124
Queries Tool.	125
Packages Tool.	125
Maintenance of Queries and Packages.	126
Query Groups.	127
Adding a Query Group.	127
Editing a Query Group.	127
Deleting a Query Group.	127
Generic Queries.	128
Adding a Generic Query.	128
Refining a Generic Query.	129
Viewing the Query Results.	134
Viewing the Impact of a Query.	134
Deleting a Query.	134
Custom Queries.	135

SQL Syntax for Custom Queries.	135
SQL Validation.	136
Adding a Custom Query.	136
Editing a Custom Query.	137
Packages.	137
Display Packages.	137
Update Packages.	137
Adding a Package.	138
Editing a Package.	139
Refreshing a Package After Changing Queries.	139
Deleting a Package.	140
Specifying Join Queries.	140
Chapter 10: Timeline.	141
Overview.	141
Guidelines.	142
Example.	142
Record Versions.	144
Record Versions Example.	144
Timeline Granularity.	145
History and State Management.	146
Timeline Enforcement Rules.	146
Effective Period Computation.	147
Rule 1. Add a Record without an Effective Period.	147
Rule 2. Add a Record with an Effective Period.	147
Rule 3. Add a Record Version for an Effective Period.	148
Rule 4. Add a Record Version that Intersects an Effective Period and Has an Extended Start Date.	148
Rule 5. Add a Record Version that Intersects an Effective Period and Has an Earlier End Date.	149
Rule 6. Add a Record Version that is Contained within an Effective Period.	149
Rule 7. Add a Record Version to Contain an Effective Period.	150
Rule 8. Add a Record Version with a Noncontiguous Effective Period.	151
Rule 9. Add a Record Version in the Pending State within an Effective Period.	151
Rule 10. Add a Record Version when a Record Version is Locked.	152
Rule 11. Add a Record Version when a Version is in the Pending State.	152
Rule 12. Delete or Update a Record Version in a Contiguous Base Object.	153
Rule 13. Update Data.	153
Rule 14. Update Effective Period.	154
Rule 15. Add an Effective Period.	154
Configuring the Base Object Timeline.	155
Step 1. Enable the Timeline in the Hub Console.	155
Step 2. Configure the Properties File.	156
Load Multiple Versions of a Record In a Batch Job.	156

Set Up Load Batch to Load Multiple Versions of a Record.	156
Batch Load of Multiple Record Versions Example.	157
Edit the Effective Period of a Record Version.	158
Increase the Effective Period of a Record Version.	158
Decrease the Effective Period of a Record Version.	159
Add a Record Version.	160
Add a Record Version Example.	161
Update Data in a Record.	161
Update Record Data Example.	161
Update a Relationship.	162
Update a Custom Column of a Relationship Record Example.	162
Update a System Column of a Relationship Record Example.	163
End a Relationship.	164
End a Relationship Example.	164
Delete a Relationship Period.	165
Delete a Relationship Example.	165
Delete All Relationship Periods.	166
Delete All Relationship Periods Example.	166
Using Timeline Extract.	167
Configure Timeline Extract Properties.	168

Chapter 11: State Management and BPM Workflow Tools. 169

State Management and BPM Workflow Tools Overview.	169
Example.	170
State Management in the MDM Hub.	170
Record States.	170
Protection of Pending Records.	172
Rules for Loading Data.	173
Record States and Base Object Record Value Survivorship.	173
BPM Workflow Tools.	174
Configure the MDM Hub for ActiveVOS.	174
Enabling State Management.	175
Adding a Workflow Engine.	175
Setting the Primary and Secondary Workflow Adapters.	176
Configure User Roles.	176
Enabling Match on Pending Records.	179
Message Triggers for State Transitions.	179
Enabling Message Triggers for State Transitions.	180
Record Promotion.	180
Promoting Records in the Data Steward Tools.	180
Setting Up a Promote Batch Job Using the Batch Viewer.	181
Setting Up a Promote Batch Job Using the Batch Group Tool.	181

Chapter 12: Data Encryption.....	182
Data Encryption Overview.	182
Data Encryption Architecture.	182
Data Encryption Restrictions.	183
Data Encryption Utilities.	183
Configuring Data Encryption.	184
Step 1. Implement DataEncryptor Interface.	184
Step 2. Configure the Data Encryption Properties File.	185
Step 3. Configure Data Encryption for the Hub Server.	185
Step 4. Configure Data Encryption for the Process Server.	186
Services Integration Framework API Requests and Responses.	186
Sample Data Encryption Properties File.	187
 Chapter 13: Hierarchies.....	 189
Hierarchies Overview.	189
Hierarchy Example.	190
About Configuring Hierarchies.	190
Before You Begin.	191
Overview of Configuration Steps.	191
Preparing Your Data for Hierarchy Manager.	191
Use Case Example of How to Prepare Data for Hierarchy Manager.	192
Scenario.	192
Methodology.	192
Creating the HM Repository Base Objects.	196
Uploading Default Entity Icons.	197
Configuring Entity Icons.	197
Adding Entity Icons.	197
Modifying Entity Icons.	197
Deleting Entity Icons.	198
Entities.	198
Entity Base Objects.	198
Entity Base Object Example.	199
Create Entity Base Objects.	200
Converting Base Objects to Entity Base Objects.	201
Entity Types.	202
Entity Type Example.	203
Creating Entity Types.	204
Editing Entity Types.	205
Deleting Entity Types.	205
Display Options for Entities.	205
Reverting Entity Base Objects to Base Objects.	206
Hierarchy Types.	206

Adding Hierarchies.	207
Deleting Hierarchies.	207
Relationship Objects.	207
Relationship Base Objects.	208
Creating Relationship Base Objects.	209
Converting Base Objects to Relationship Base Objects.	209
Reverting Relationship Base Objects to Base Objects.	210
Foreign Key Relationship Base Objects.	211
Creating Foreign Key Relationship Base Objects.	211
Relationship Types.	212
Relationship Type Example.	213
Creating Relationship Types.	214
Editing Relationship Types.	215
Deleting Relationship Types.	215
Packages.	215
Hierarchy Manager Data Configuration.	216
Creating Entity, Relationship, and FK Relationship Object Packages.	216
Assigning Packages to Entity or Relationship Types.	218
About Profiles.	219
Adding Profiles.	219
Editing Profiles.	220
Validating Profiles.	220
Copying Profiles.	221
Deleting Profiles.	221
Deleting Relationship Types from a Profile.	221
Deleting Entity Types from a Profile.	222
Assigning Packages to Entity and Relationship Types.	222
Chapter 14: Hierarchy Manager Tutorial.	223
Hierarchy Configuration Overview.	223
About the Tutorial Example.	225
Foreign Key Relationships.	225
Step 1. Create the Product Entity Base Object.	225
Step 2. Create the Entity Types.	226
Create the Product Entity Type.	228
Create the Product Group Entity Type.	229
Step 3. Create the Product Relationship Object.	229
Step 4. Create the Relationship Types.	230
Create the Product Group is Parent of Product Group Relationship Type.	232
Create the Product Group is Parent of Product Relationship Type.	232
Step 5. Create a Hierarchy Type.	233
Step 6. Add the Relationship Types to the Hierarchy Profile.	234
Step 7. Create the Packages.	234

Create the Product Entity Object Package.	234
Create the Product Relationship Package.	235
Step 8. Assign the Packages.	236
Assign the PKG Product Package to the Product Entity Type.	236
Assign the PKG Product Package to the Product Group Entity Type.	237
Assign the PKG Product Rel Package to the Product Group is Parent of Product Group Relationship Type.	237
Assign the PKG Product Rel Package to the Product Group is Parent of Product Relationship	239
Step 9. Configure the Display of Data in Hierarchy Manager.	239
Configure the Product Entity Type Label.	240
Configure the Label for the Product Group Entity Type.	241
Configure the Product Entity Type Tooltip Text.	241
Configure the Product Group Entity Type Tooltip Text.	242
Configure the Product Group is Parent of Product Group Relationship Type Tooltip Text.	243
Configure the Product Group is Parent of Product Relationship Type Tooltip Text.	244
Configure the List of Entities.	245
Configure the List of Relationships.	245
Configure the Entity Search Fields.	246
Configure the Relationship Search Fields.	247
Configure the Entity Search Results.	249
Configure the Relationship Search Results.	250
Configure the Entity Details in Hub Console Hierarchy Manager.	251
Configure the Entity Details for IDD Hierarchy Manager.	252
Configure the Relationship Details.	253
Configure the Editable Entity Fields.	254
Relationship Field Editability.	254
Configure the Entity Creation Fields.	255
Configure the Relationship Creation Fields.	256
Hierarchy Management.	257
Part IV: Configuring the Data Flow.	258
Chapter 15: MDM Hub Processes.	259
MDM Hub Processes Overview.	259
About Informatica MDM Hub Processes.	259
Overall Data Flow for Batch Processes.	259
Consolidation Status for Base Object Records.	260
Cell Data Survivorship and Order of Precedence.	261
ROWID_OBJECT Survivorship.	262
Land Process.	262
About the Land Process.	262
Managing the Land Process.	263
Stage Process.	264

Load Process.	265
About the Load Process.	265
Tables Associated with the Load Process.	265
Initial Data Loads and Incremental Loads.	266
Trust Settings and Validation Rules.	266
Run-time Execution Flow of the Load Process.	267
Other Considerations for the Load Process.	271
Tokenize Process.	271
Match Tokens and Match Keys.	272
Match Key Tables.	272
Example Match Keys.	273
Tokenize Process Applies to Fuzzy-match Base Objects Only	273
Key Concepts for the Tokenize Process.	273
Optimizing the Tokenize and Merge Process Performance.	275
Clean Up Match Key Tables.	275
Match Process.	277
Match Rules.	279
Exact-match and Fuzzy-match Base Objects.	279
Support Tables Used in the Match Process.	280
Population Sets.	280
Matching for Duplicate Data.	280
Build Match Groups and Transitive Matches.	281
Maximum Matches for Manual Consolidation.	281
External Match Jobs.	281
Distributed Process Server.	281
Handling Application Server or Database Server Failures.	281
Consolidate Process.	282
About the Consolidate Process.	282
Consolidation Options.	283
Consolidation and Workflow Integration.	284
Publish Process.	284
JMS Models Supported by Informatica MDM Hub.	284
Publish Process for Outbound Distribution of Reconciled Data.	284
Publish Process Message Triggers.	284
Outbound JMS Message Queues.	285
ORS-specific XML Message Schemas.	285
Run-time Flow of the Publish Process.	286
Chapter 16: Configuring the Land Process.	287
Configuring the Land Process Overview.	287
Configuring Source Systems.	287
About Source Systems.	287
Starting the Systems and Trust Tool.	288

Source System Properties.	289
Adding Source Systems.	289
Editing Source System Properties.	290
Removing Source Systems.	290
Configuring Landing Tables.	290
About Landing Tables.	291
Landing Table Columns.	291
Landing Table Properties.	291
Adding Landing Tables.	292
Editing Landing Table Properties.	292
Removing Landing Tables.	293
Chapter 17: MDM Hub Staging.	294
MDM Hub Staging Overview.	294
MDM Hub Staging Tables.	295
MDM Hub Stage Process Tables.	295
Staging Table Properties.	296
Mapping Properties.	297
MDM Hub Staging Prerequisites.	297
Add Staging Tables.	297
Map Columns Between a Landing Table and Staging Tables.	300
Data is Either Cleansed or Passed Through Unchanged.	301
Starting the Mappings Tool.	302
Creating a Mapping.	303
Mapping the Primary Key.	304
Mapping Columns.	306
Filtering Records in Mappings.	307
Maintaining Mappings that Use Cleanse Functions.	308
Loading by Row ID.	308
Configure Audit Trail and Delta Detection.	309
Configuring the Audit Trail for a Staging Table.	309
Configuring Delta Detection for a Staging Table.	310
Staging Table Management.	313
Changing Properties in Staging Tables.	313
Jumping to the Source System for a Staging Table.	313
Removing Staging Tables.	314
Mappings Management.	314
Editing Mapping Properties.	314
Copying Mappings.	314
Jumping to a Schema.	315
Testing Mappings.	315
Removing Mappings.	315

Chapter 18: Hard Delete Detection.	316
Hard Delete Detection Overview.	316
Hard Delete Detection Types.	317
Delete Flag Values of Base Objects.	317
Trust and Validation Rules.	317
Hard Delete Detection Table.	318
Configuring Hard Delete Detection.	319
Specifying the Primary Key Columns for Hard Delete Detection	322
Direct Delete.	323
Hard Delete Detection Configuration for Direct Delete Flagging.	323
Setting Up Hard Delete Detection for Direct Delete with End Date.	324
Direct Delete with End Date Code Example.	324
Example of Hard Delete Detection for Direct Delete.	325
Consensus Delete.	326
Hard Delete Detection Configuration for Consensus Delete Flagging.	326
Setting Up Hard Delete Detection for Consensus Delete with End Date.	327
Consensus Delete with End Date Code Example.	328
Example of Hard Delete Detection for Consensus Delete.	328
Using Hard Delete Detection within User Exits.	331
 Chapter 19: Data Cleansing Configuration.	 333
Data Cleansing Configuration Overview.	333
Set Up Data Cleansing in the MDM Hub.	333
Configure Process Servers for Data Cleansing.	333
Modes of Cleanse Operations.	334
Distributed Data Cleansing.	334
Cleanse Requests.	335
Starting the Process Server Tool.	335
Process Server Properties.	335
Adding a Process Server.	337
Enabling Secured Communications for Process Servers.	337
Editing Process Server Properties.	338
Deleting a Process Server.	338
Testing the Process Server Configuration.	338
Configure Cleanse Functions.	338
Starting the Cleanse Functions Tool.	339
Cleanse Function Types.	339
Cleanse Function Properties.	340
Overview of Configuring Cleanse Functions.	340
Configuring User Cleanse Libraries.	340
Configuring Java Cleanse Libraries.	341
Adding Regular Expression Functions.	342

Configure Graph Functions.	342
Testing Functions.	347
Using Conditions in Cleanse Functions.	347
About Conditional Execution Components.	347
When to Use Conditional Execution Components.	347
Adding Conditional Execution Components.	348
Configuring Cleanse Lists.	348
About Cleanse Lists.	348
Adding Cleanse Lists.	348
Cleanse List Properties.	349
Editing Cleanse List Properties.	351

Chapter 20: Configuring the Load Process 354

Overview.	354
Before You Begin.	354
Configuration Tasks for Loading Data.	355
Configuring Staging Tables.	355
Staging Table Columns.	355
Preserve Source System Keys.	357
Specify the Highest Reserved Key.	357
Highest Reserved Key Example.	357
Enable Cell Update.	358
Properties for Columns in Staging Tables.	358
Changing Properties in Staging Tables.	360
Lookups for Foreign Key Columns.	361
Configuring Initial Data Load.	362
Configuring Trust for Source Systems.	362
About Trust.	362
Trust Properties.	364
Considerations for Setting Trust Values.	365
Column Trust.	366
Configuring Validation Rules.	368
About Validation Rules.	368
Enabling Validation Rules for a Column.	369
Navigating to the Validation Rules Node.	370
Validation Rule Properties.	370
Adding Validation Rules.	373
Editing Validation Rule Properties.	373
Changing the Sequence of Validation Rules.	374
Removing Validation Rules.	374

Chapter 21: Configuring the Match Process. 375

Before You Begin.	375
---------------------------	-----

Configuration Tasks for the Match Process.	375
Understanding Your Data.	375
Base Object Properties Associated with the Match Process.	376
Configuration Steps for Defining Match Rules.	376
Configuring Base Objects with International Data.	376
Distributed Match Configuration.	376
Configuring Data Load.	377
Navigating to the Match/Merge Setup Details Dialog.	377
Configuring Match Properties for a Base Object.	378
Setting Match Properties.	378
Match Properties.	378
Supporting Long ROWID_OBJECT Values.	382
Configuring Match Paths for Related Records.	382
Match Paths.	382
Foreign Key Relationships and Filters.	382
Relationship Base Objects.	383
Inter-Table Paths.	383
Example Base Objects for Inter-Table Paths.	383
Columns in the Example Base Objects	384
Example Configuration Steps.	385
Intra-Table Paths.	385
Example Base Object for Intra-Table Paths.	385
Columns in the Example Base Object	386
Create a Relationship Base Object.	386
Example Configuration Steps.	386
Navigating to the Paths Tab.	387
Configuring Filters for Match Paths.	388
Configuring Path Components.	389
Display Name.	390
Physical Name.	390
Allow Missing Child Records.	390
Constraints.	391
Adding Path Components.	391
Editing Path Components.	391
Deleting Path Components.	391
Configuring Match Columns.	392
About Match Columns.	392
Configuring Match Columns for Fuzzy-match Base Objects.	395
Configuring Match Columns for Exact-match Base Objects.	398
Match Rule Sets.	400
Match Rule Set Properties.	401
Navigating to the Match Rule Set Tab	403

Adding Match Rule Sets.	404
Editing Match Rule Set Properties.	404
Renaming Match Rule Sets.	404
Deleting Match Rule Sets.	405
Configuring Match Column Rules for Match Rule Sets.	405
Prerequisites for Configuring Match Column Rules	405
Match Column Rules Differ Between Exact-Match and Fuzzy-Match Base Objects	405
Specifying Consolidation Options for Matched Records	406
Match Rule Properties for Fuzzy-match Base Objects Only	406
Match Column Properties for Match Rules	414
Requirements for Exact-match Columns in Match Column Rules.	421
Command Buttons for Configuring Column Match Rules.	421
Adding Match Column Rules.	422
Editing Match Column Rules.	423
Deleting Match Column Rules.	424
Changing the Execution Sequence of Match Column Rules	424
Specifying Consolidation Options for Match Column Rules	425
Configuring the Match Weight of a Column	425
Configuring Segment Matching for a Column	425
Configuring Primary Key Match Rules.	426
About Primary Key Match Rules.	426
Adding Primary Key Match Rules.	427
Editing Primary Key Match Rules.	427
Deleting Primary Key Match Rules.	428
Investigating the Distribution of Match Keys.	428
About Match Keys Distribution	428
Navigating to the Match Keys Distribution Tab.	429
Components of the Match Keys Distribution Tab	429
Filtering Match Keys	430
Excluding Records from the Match Process.	431
Proximity Search.	431
Configuring Proximity Search.	432
Lightweight Matching.	433
Chapter 22: Match Rule Configuration Example.	435
Match Rule Configuration Example Overview.	435
Match Rule Configuration Scenario.	436
Configuring Match Rules.	437
Step 1. Review the Data.	437
Step 2. Identify the Base Objects for the Match Job.	438
Step 3. Configure Match Properties.	438
Configuring Match Properties.	438
Step 4. Define the Match Path.	439

Adding Match Path Components.	440
Step 5. Define Match Columns.	444
Defining Match Columns.	444
Step 6. Define a Match Rule Set.	449
Defining a Match Rule Set.	450
Step 7. Add Match Rules.	451
Adding Match Rules.	451
Step 8. Set Merge Options for Match Rules.	454
Setting a Match Rule as an Automerge Match Rule.	454
Step 9. Review the Match Properties.	455
Reviewing the Match Properties.	456
Step 10. Test the Match Rules.	457
 Chapter 23: Search with Elasticsearch.	459
Search with Elasticsearch Overview.	459
Search with Elasticsearch Architecture.	459
Installing and Configuring Search.	460
Step 1. Install and Set Up Elasticsearch.	461
Complete Pre-Installation Tasks.	461
Install Elasticsearch.	462
Configure the Elasticsearch Java Virtual Machine (JVM).	462
Configure the Elasticsearch Properties File.	463
Secure Elasticsearch (Optional).	464
Install Analysis Plugins.	464
Configure Stop Words, Synonyms, and Character Mappings.	464
Start Elasticsearch.	465
Upgrade Elasticsearch.	466
Step 2. Configure the MDM Hub Properties for Search.	466
Configure the Hub Server Properties.	466
Configure the Process Server Properties.	468
Step 3. Configure Search by Using the Provisioning Tool.	470
Configure the Elasticsearch Cluster.	470
Create Custom Elasticsearch Index Settings (Optional).	471
Configure the Searchable Fields.	474
Configure the Search or Query Results Display.	478
Configure the Layout to Display Similar Records (Optional).	479
Step 4. Validate the Operational Reference Store.	480
Step 5. Index the Search Data.	481
Creating Keystores, Truststore, and Certificates (Optional).	481
 Chapter 24: Configuring the Consolidate Process.	483
Configuring the Consolidate Process Overview.	483
Consolidation Settings.	483

Immutable Rowid Object.	483
Distinct Systems.	484
Unmerge Child When Parent Unmerges (Cascade Unmerge).	484
Changing Consolidation Settings.	487

Chapter 25: Pending Control Table. 489

Chapter 26: Configuring the Publish Process. 490

Publish Process Overview.	490
Configuration Steps for the Publish Process.	491
Starting the Message Queues Tool.	491
Configuring Global Message Queue Settings.	491
Configuring Message Queue Servers.	492
Message Queue Server Properties.	492
Adding Message Queue Servers.	493
Editing Message Queue Server Properties.	493
Deleting Message Queue Servers.	494
Configuring Outbound Message Queues.	494
About Message Queues.	494
Message Queue Properties.	494
Adding Message Queues to a Message Queue Server.	494
Editing Message Queue Properties.	495
Deleting Message Queues.	495
Configuring Parallel Processing of JMS Messages.	496
Configuring JMS Security.	496
Message Trigger Configuration.	496
Types of Events for Message Triggers.	497
Best Practices for Message Triggers.	498
Message Trigger System Properties.	499
Adding Message Triggers.	499
Editing Message Triggers.	500
Deleting Message Triggers.	500
Disabling Message Queue Polling.	501
JMS Message XML Reference.	501
Generating ORS-specific XML Message Schemas.	501
Elements in an XML Message.	501
Filtering Messages.	503
Example XML Messages.	503
Legacy JMS Message XML Reference.	514
Message Fields for Legacy XML.	515
Filtering Messages for Legacy XML.	515
Example Messages for Legacy XML.	515

Part V: Executing Informatica MDM Hub Processes.....	527
Chapter 27: Using Batch Jobs.	528
Using Batch Jobs Overview.	528
Batch Job Thread Configuration.	529
Multi-threaded Batch Job Process.	529
Multi-threaded Batch Job Example.	529
Multi-threaded Batch Performance.	530
Multi-threaded Batch Job Properties.	530
Launching Batch Jobs.	530
Support Tables Used By Batch Jobs.	531
Running Batch Jobs in Sequence.	531
Populating Landing Tables Before Running Batch Jobs.	531
Match Jobs and Subsequent Consolidation Jobs.	531
Loading Data from Parent Tables First.	532
Loading Data for Objects With Foreign Key Relationships.	532
Best Practices for Working With Batch Jobs.	532
Limiting the Parallel Degree for Gathering Statistics in Oracle environments.	533
Batch Job Creation.	533
Batch Jobs That Are Created Automatically.	533
Batch Jobs That Are Created When Changes Occur.	534
Information-Only Batch Jobs (Not Run in the Hub Console).	534
Process Server Configuration.	535
Running Batch Jobs Using the Batch Viewer Tool.	535
Batch Viewer Tool.	535
Starting the Batch Viewer Tool.	535
Grouping by Table, Data, or Procedure Type.	536
Running Batch Jobs Manually.	536
Viewing Job Execution Logs.	538
Clearing the Job Execution History.	543
Running Batch Jobs Using the Batch Group Tool.	543
About Batch Groups.	543
Starting the Batch Group Tool.	544
Configuring Batch Groups.	544
Refreshing the Batch Groups List.	548
Executing Batch Groups Using the Batch Group Tool.	548
Filtering Execution Logs By Status.	552
Deleting Batch Groups.	552
Batch Jobs Reference.	552
Alphabetical List of Batch Jobs.	553
Accept Non-Matched Records As Unique.	554
Auto Match and Merge Jobs.	554

Automerge Jobs.	555
Batch Unmerge.	556
BVT Snapshot Jobs.	556
External Match Jobs.	557
Generate Match Tokens Jobs.	560
Initially Index Smart Search Data Jobs.	561
Key Match Jobs.	562
Load Jobs.	562
Manual Merge Jobs.	566
Manual Unmerge Jobs.	567
Match Jobs.	567
Match Analyze Jobs.	569
Match for Duplicate Data Jobs.	571
Multi Merge Jobs.	571
Promote Jobs.	571
Recalculate Base Object Jobs.	573
Recalculate BVT Jobs.	573
Reset Match Table Jobs.	573
Revalidate Jobs.	573
Stage Jobs.	574
Synchronize Jobs.	574
Chapter 28: User Exits.	576
User Exits Overview.	576
User Exit Processing.	577
User Exit JAR Files.	578
Implementing the User Exit JAR File.	578
Uploading User Exits to the MDM Hub.	578
Removing User Exits from the MDM Hub.	578
UserExitContext Class.	579
Stage Process User Exits.	580
Post-landing User Exit.	581
Pre-stage User Exit.	582
Post-stage User Exit.	582
Load Process User Exits.	583
Post-load User Exit.	584
Match Process User Exits.	585
Pre-match User Exit.	586
Post-match User Exit.	586
Merge Process User Exits.	587
Post-merge User Exit.	587
Unmerge Process User Exits	588
Pre-unmerge User Exit.	589

Post-unmerge User Exit.	589
Task Management User Exits.	590
AssignTasks User Exit Interface.	590
GetAssignableUsersForTask User Exit Interface.	591
Using Services Integration Framework APIs Within User Exits.	591
Creating a User Exit to Call a Services Integration Framework API.	592
User Exit Example.	592
Services Integration Framework APIs.	593
Guidelines for Implementing User Exits.	595
Part VI: Configuring Application Access.	596
Chapter 29: ORS-specific APIs.	597
ORS-specific APIs Overview.	597
Performance Considerations.	598
Supported Repository Objects.	598
ORS-Specific SIF API Properties.	598
Repository Objects Statuses.	599
Archive Table.	599
Generating and Deploying an ORS-Specific SIF API.	600
Renaming an ORS-specific SIF API.	600
Downloading an ORS-Specific Client JAR File.	600
Using ORS-Specific Client JAR Files with SIF SDK.	601
Removing an ORS-Specific SIF API.	601
Chapter 30: ORS-specific Message Schemas.	602
ORS-specific Message Schemas Overview.	602
About the JMS Event Schema Manager Tool.	602
Starting the JMS Event Schema Manager Tool.	603
Starting the SIF Manager Tool.	603
Generating and Deploying ORS-specific Schemas.	604
Downloading an XSD File.	604
Finding Out-of-Sync Objects.	604
Auto-searching for Out-of-Sync Objects.	605
Chapter 31: Viewing Registered Custom Code.	606
Overview.	606
User Objects.	606
Starting the User Object Registry Tool.	607
Viewing User Exits.	607
About User Exits.	607
Viewing User Exits.	607
Viewing Custom Java Cleanse Functions.	607

About Custom Java Cleanse Functions.	608
How Custom Java Cleanse Functions Are Registered.	608
Viewing Registered Custom Java Cleanse Functions.	608
Viewing Custom Button Functions.	608
About Custom Button Functions.	608
How Custom Button Functions Are Registered.	608
Viewing Registered Custom Button Functions	609
Appendix A: MDM Hub Properties.	610
MDM Hub Properties Overview.	610
Hub Server Properties.	610
Sample Hub Server Properties File.	628
Process Server Properties.	630
Sample Process Server Properties File.	638
Operational Reference Store Properties.	639
Appendix B: Viewing Configuration Details.	641
Viewing Configuration Details Overview.	641
Starting the Enterprise Manager.	641
Properties in Enterprise Manager.	642
C_REPOS_DB_RELEASE Table.	642
Environment Report.	644
Saving the MDM Hub Environment Report.	644
Viewing Version History in Enterprise Manager.	644
Using Application Server Logs.	644
Application Server Log Levels.	645
Log File Rolling.	645
Configuring Application Server Logs.	646
Using Hub Console Log For the Client.	646
Appendix C: Row-level Locking.	647
Row-level Locking Overview.	647
About Row-level Locking.	647
Default Behavior.	647
Types of Locks.	648
Considerations for Using Row-level Locking.	648
Configuring Row-level Locking.	648
Enabling Row-level Locking on an ORS.	648
Configuring Lock Wait Times.	649
Locking Interactions Between SIF Requests and Batch Processes.	649
Interactions When API Batch Interoperability is Enabled.	649
Interactions When API Batch Interoperability is Disabled.	650

Appendix D: MDM Hub Logging.....	651
MDM Hub Logging Overview.	651
Configuring the Logging Settings.	652
Hub Console Log.	652
Hub Server Log.	652
Process Server Log.	653
Entity 360 Log.	653
Provisioning Tool Log.	653
 Appendix E: Table Partitioning.....	 654
Support for Table Partitioning.	654
 Appendix F: Collecting MDM Environment Information with the Product Usage Toolkit.....	 655
Collecting MDM Environment Information with the Product Usage Toolkit Overview.	655
System Configuration Information.	656
MDM Hub Environment Information.	656
Enabling MDM Hub Data Collection on the Hub Server.	657
Enabling MDM Hub Data Collection on the Process Server.	657
Disabling MDM Hub Data Collection on the Hub Server.	658
Disabling MDM Hub Data Collection on the Process Server.	658
 Appendix G: Glossary.....	 659
 Index.	 688

Preface

Follow the instructions in the Informatica® *Multidomain MDM Configuration Guide* to configure the Informatica MDM Hub system. Learn how to use the MDM Hub Console tools to build data models, configure data flows, execute processes and configure application access.

This guide assumes that you have read the *Multidomain MDM Overview Guide* and have a basic understanding of MDM Hub architecture and key concepts.

Informatica Resources

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

Informatica Network

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

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

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

Informatica Knowledge Base

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

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

Informatica Documentation

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

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

Informatica Product Availability Matrices

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

Informatica Velocity

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

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

Informatica Marketplace

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

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:

<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>.

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

Part I: Introduction

This part contains the following chapters:

- [Informatica MDM Hub Administration, 28](#)
- [Getting Started with the MDM Hub Console, 30](#)
- [Configuring International Data Support, 43](#)

CHAPTER 1

Informatica MDM Hub Administration

This chapter includes the following topics:

- [Informatica MDM Hub Administration Overview, 28](#)
- [Phases in Informatica MDM Hub Administration, 28](#)

Informatica MDM Hub Administration Overview

Informatica MDM Hub administrators have primary responsibility for the configuration of the Informatica MDM Hub system.

Administrators access Informatica MDM Hub through the Hub Console, which comprises a set of tools for managing an Informatica MDM Hub implementation.

Informatica MDM Hub administrators use the Hub Console to perform the following tasks:

- Build the data model and other objects in the Hub Store.
- Configure and execute Informatica MDM Hub data management processes.
- Configure external application access to Informatica MDM Hub functionality and resources.
- Monitor ongoing operations.
- Maintain logs needed by Global Customer Support for troubleshooting the Informatica MDM Hub.

Phases in Informatica MDM Hub Administration

The administration phases may vary for your Informatica MDM Hub implementation based on your organization's methodology.

MDM Hub administration can include the following phases:

1. Startup. Install and set up MDM Hub.
2. Configuration. Build and test MDM Hub functionality.
3. Production. Deploy, tune, and maintain the environment.

Startup Phase

The startup phase involves installing and configuring core Informatica MDM Hub components: Hub Store, Hub Server, Process Server, and cleanse adapters.

For instructions on installing the Hub Store, Hub Server, and Process Server, see the *Multidomain MDM Installation Guide* for your application server. For instructions on setting up a cleanse adapter to integrate a supported external cleanse engine with the MDM Hub, see the *Multidomain MDM Cleanse Adapter Guide*.

Note: The instructions in this chapter assume that you have already completed the startup phase and are ready to begin configuring your Informatica MDM Hub implementation.

Configuration Phase

After Informatica MDM Hub has been installed and set up, administrators can begin configuring and testing Informatica MDM Hub functionality—the data model and other objects in the Hub Store, data management processes, external application access, and so on.

This phase involves a dynamic, iterative process of building and testing Informatica MDM Hub functionality to meet the stated requirements of an organization. The bulk of the material in this chapter refers to tasks associated with the configuration phase.

After a schema has been sufficiently built and the Informatica MDM Hub has been properly configured, developers can build external applications to access Informatica MDM Hub functionality and resources. For instructions on developing external applications, see the *Multidomain MDM Services Integration Framework Guide*.

Production Phase

After Informatica MDM Hub implementation has been sufficiently configured and tested, administrators deploy the Informatica MDM Hub in a production environment.

In addition to managing ongoing Informatica MDM Hub operations, this phase can involve performance tuning to optimize the processing of actual business data.

Note: The MDM Hub administrator must provide access to all necessary log files to permit timely troubleshooting.

CHAPTER 2

Getting Started with the MDM Hub Console

This chapter includes the following topics:

- [Overview, 30](#)
- [About the MDM Hub Console, 30](#)
- [Start the Hub Console, 30](#)
- [Navigating the MDM Hub Console, 32](#)
- [Informatica MDM Hub Workbenches and Tools, 40](#)

Overview

This chapter introduces the MDM Hub Console and provides a high-level overview of the tools involved in configuring your Informatica MDM Hub implementation.

About the MDM Hub Console

Administrators and data stewards can access Informatica MDM Hub features using the Informatica MDM Hub user interface, which is called the Hub Console. The Hub Console comprises a set of tools. You can perform a specific action or a set of related actions with each tool.

Note: The available tools in the Hub Console depend on your Informatica license agreement.

Start the Hub Console

To access the MDM Hub, start the Hub Console by using an HTTP or HTTPS connection.

Before you start the Hub Console, ensure that you have the following information:

- Host name and port number for the URL

- User name and password
 - An SSL certificate on the client machine, if you want to access the Hub Console through an HTTPS connection
1. Open a browser window and enter the following URL:
`http://<MDM Hub host>:<port number>/cmx/`
 The Hub Console launch page appears.
 2. Enter your user name and password, and then click **Download**.
 The MDM Hub application JAR file that is required to launch the Hub Console downloads.
Note: If you cannot download the MDM Hub application JAR file, contact your MDM administrator. The administrator can distribute the JAR file from the following directory: `<MDM Hub installation directory>/hub/server/resources/hub`
 3. Run the application JAR file.
Note: If you do not have an SSL certificate on the client machine, and want to access the Hub Console through an HTTPS connection, you must install it. To do so, you can use the following procedure:
 - Import the certificate to the Java keystore of your local client machine, by running the following command:
`keytool -import -trustcacerts -alias <certificate alias name> -file <certificate alias file> -keystore <local java cacerts keystore location>`
 - Pass the location and password of the truststore file that contains the certificate, by running the following command:
`java -Djavax.net.ssl.trustStore=<truststore file location> -Djavax.net.ssl.trustStorePassword=<truststore_password> -jar hubConsole.jar`
 Use a separate truststore that contains all the custom trust certificates, rather than the default cacert file. Obtain the certificates from the team that maintains the application server. The server might use either a self signed certificate or security certificate. Download the .jar file only if there is a version change on the server. Each time you download the .jar file, launch it using the same command.
 4. To specify the maximum memory allocation pool for the application, run the following command:
`java -Xmx<n>G -jar hubConsole.jar`
 Where `<n>` is the maximum memory allocation in GB.
 The **Informatica MDM Hub Login** dialog box appears.
 5. Enter your user name and password.
 6. If you want to connect to a specific Hub Server node or if you use a load balancer or a reverse proxy server, override the pre-configured connection parameters in the Connection Property field.
 Enter the parameters in the following format:
`<host name>:<port name>`
 Where, host name and port name are either the host name and port name of the Hub Server or that of the load balancer or the reverse proxy server that you use.
 7. Click **OK**.
 The **Change database** dialog box appears.
 8. Select the target database.
 The target database is the MDM Hub Master Database.
 9. Select a language from the list, and click **Connect**.
 The Hub Console user interface appears in the language that you select. If you need to change the language in which the Hub Console user interface appears, restart the Hub Console with the language of your choice.

Navigating the MDM Hub Console

The Hub Console is a collection of tools that you use to configure and manage your Informatica MDM Hub implementation.

Each tool allows you to focus on a particular area of your Informatica MDM Hub implementation.

Toggling Between the Processes and Workbenches Views

Informatica MDM Hub groups tools in two different ways:

View	Description
By Workbenches	Similar tools are grouped together by workbench. A workbench is a logical collection of related tools.
By Processes	Tools are grouped into a logical workflow that walks you through the tools and steps required to complete a task.

You can click the tabs at the left side of the Hub Console window to toggle between the **Processes** and **Workbenches** views.

Note: When you log into Informatica MDM Hub, you see only those workbenches and processes that contain the tools that your Informatica MDM Hub security administrator has authorized you to use.

Workbenches View

To view tools by workbench:

- Click the **Workbenches** tab on the left side of the Hub Console window.
The Hub Console displays a list of available workbenches on the **Workbenches** tab. The **Workbenches** view organizes Hub Console tools by similar functionality.
The workbench names and tool descriptions are metadata-driven, as is the way in which tools are grouped. It is possible to have customized tool groupings.

Processes View

To view tools by process:

- Click the **Processes** tab on the left side of the Hub Console window.
The Hub Console displays a list of available processes on the **Processes** tab. Tools are organized into common sequences or processes.
Processes step you through a logical sequence of tools to complete a specific task. The same tool can belong to multiple processes and can appear many times in one process.

Starting a Tool in the Workbenches View

Start the Hub Console from the **Workbenches** view.

- In the **Workbenches** view, expand the workbench that contains the tool that you want to start.
- If necessary, expand the workbench node to show the tools associated with that workbench.

3. Click the tool.

If you selected a tool that requires a different database, the Hub Console prompts you to select it. All tools in the Configuration workbench, Databases, Users, Security Providers, Tool Access, Message Queues, Repository Manager, Enterprise Manager, and Workflow Manager require a connection to the MDM Hub Master Database. All other tools require a connection to an Operational Reference Store.

The Hub Console displays the tool that you selected.

Acquire Locks to Change Metadata

To change the metadata in the Hub Store, you need to acquire a lock on the repository tables through the Hub Console.

All the tools that you access through the Hub Console, except the Data Steward tools, are in the read-only mode. To use the tools to make changes to the metadata in the Hub Store, you must acquire a lock on the repository tables.

To simultaneously make changes to the Hub Store, acquire locks for an exclusive user or for multiple users. You can force the release of write or exclusive locks held by other users.

Types of Locks

The **Write Lock** menu provides two types of locks.

The following table describes the types of locks that you can access in the Hub Console:

Type of Lock	Description
exclusive lock	Allows only one user to make changes to the underlying Operational Reference Store, preventing any other users from changing the Operational Reference Store while the exclusive lock is in effect.
write lock	Allows multiple users to making changes to the underlying metadata at the same time. Write locks can be obtained on the MDM Hub Master Database or on an Operational Reference Store.

Note: You cannot acquire a lock an Operational Reference Store that is in production mode. If an Operational Reference Store is in production mode and you try to acquire a write lock, you will see a message stating that you cannot acquire the lock.

Tools that Require a Lock

You need to acquire a lock on tools in the MDM Hub Master Database and the Operational Reference Store before you can make configuration changes to the databases.

You need to acquire a lock on the following tools to make configuration changes to the MDM Hub Master Database:

- Databases
- Repository Manager
- Message Queues
- Security Providers
- Tool Access
- Users

You need to acquire a lock on the following tools to make configuration changes to the Operational Reference Store:

- Batch Group
- Cleanse Functions
- Hierarchies
- Hierarchy Manager
- Mappings
- Packages
- Process Server
- Queries
- Roles
- Schema Manager
- Schema Viewer
- Secure Resources
- SIF Manager
- Systems and Trust
- Users and Groups

Note: The Data Manager, Merge Manager, and Hierarchy Manager do not require write locks. For more information about these tools, see the *Multidomain MDM Data Steward Guide*. The Audit Manager also does not require write locks.

Automatic Lock Expiration

The Hub Console refreshes the lock every 60 seconds on the current connection. The user can manually release a lock. If a user switches to a different database while holding a lock, then the lock is automatically released. If the Hub Console is terminated, then the lock expires after one minute.

Server Caching and MDM Hub Console Locks

When no locks are in effect in the Hub Console, the Hub Server caches metadata and other configuration settings for performance reasons. When a Hub Console user acquires a write lock or exclusive lock, caching is disabled, the cache is emptied, and Informatica MDM Hub retrieves this information from the database instead. When all locks are released, caching is enabled again.

When more than one Hub Console uses the same Operational Reference Store, a write lock on a Hub Server does not disable caching on the other Hub Servers.

Acquiring a Write Lock

Write locks allow multiple users to edit data in the Hub Console at the same time.

However, write locks do not prevent those users from editing the same data at the same time. In such cases, the most recently-saved changes remain.

1. Click **Write Lock > Acquire Lock**.
 - If the lock has already been acquired by someone else, then the login name and machine address of that person appears.

- If the Operational Reference Store is in production mode, a message appears explaining that you cannot acquire the lock.
 - If the lock is acquired successfully, then the tools are in read-write mode. Multiple users can have a write lock per Operational Reference Store or in the MDM Hub Master Database.
2. When you are finished, click **Write Lock > Release Lock**.

Acquiring an Exclusive Lock

You can acquire an exclusive lock in the Hub Console.

1. Click **Write Lock > Clear Lock** to clear any write locks held by other users.
2. Click **Write Lock > Acquire Exclusive Lock**.
If the Operational Reference Store is in production mode, a message appears explaining that you cannot acquire the exclusive lock.
3. When you are finished, click **Write Lock > Release Lock**.

Releasing a Lock

You can release a lock in the Hub Console.

- Click **Write Lock > Release Lock**.

Clearing Locks

You can clear locks in the Hub Console. Clear locks only when required because other users are not warned to save changes before their write locks are released.

- Click **Write Lock > Clear Lock**.

The Hub Console releases any locks on the Operational Reference Store.

Changing the Target Database

The status bar at the bottom of the Hub Console window shows the name of the target database that you are connected to and the user name that you used to log in.

1. On the status bar, click the database name.
The Hub Console prompts you to choose a target database.
2. Select the MDM Hub Master Database or the Operational Reference Store that you want to connect to.
3. Click **Connect**.

Logging in as a Different User

You can log in as a different user in the Hub Console.

1. Choose one of the following options:
 - Click the user name on the status bar.
 - Click **Options > Re-Login As**.
2. Specify the user name and password for the user account that you want to use.

3. Click **OK**.

Changing the Password for a User

You can change the password for the user that is currently logged in to the Hub Console.

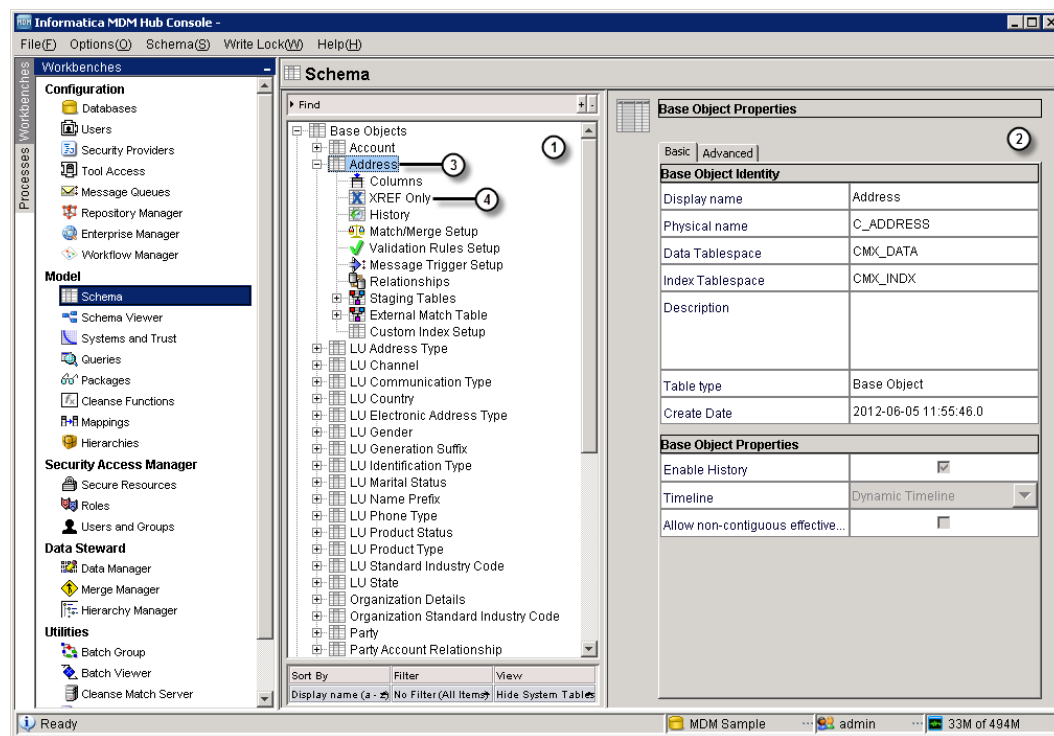
1. Click **Options > Change Password**.
2. Specify the password that you want to use.
3. Click **OK**.

Using the Navigation Tree in the Navigation Pane

Use the navigation tree in the Hub Console to view and manage a hierarchical collection of objects.

The navigation tree is in the **Navigation** pane of the Hub Console. Each named object in the navigation tree is represented as a node. A node that contains other nodes is called a parent node. A node that belongs to a parent node is called a child node.

The following figure shows the Hub Console interface:



1. Navigation pane
2. Properties pane
3. Parent node
4. Child node

Showing and Hiding Child Nodes

To show child nodes beneath a parent node:

- Click the plus (+) sign next to the parent node.

To hide child nodes beneath a parent node:

- Click the minus (-) sign next to the parent node.

Sorting by Display Name

The display name is the name of an object as it appears in the navigation tree. You can change the order in which the objects are displayed in the navigation tree by clicking **Sort By** in the tree options area and selecting the appropriate sort option.

Choose from the following sort options:

- **Display Name (a-z)** sorts the objects in the tree alphabetically according to display name.
- **Display Name (z-a)** sorts the objects in the tree in descending alphabetical order according to display name.

Filter Options

You can filter the items shown in the navigation tree by clicking the **Filter** area at the bottom of the **Navigation** pane and selecting the appropriate filter option.

Choose from the following filter options:

- **No Filter (All Items)**. Removes any filter that was previously defined.
- **One Item**. Displays a drop-down list above the navigation tree from which to select an item.
For example, in the Schema Manager, you can choose Table type or Table.
If you choose Table type, you click the down arrow to display a list of table types from which to select for your filter.
- **Some Items**. Allows you to select one or more items.

Filtering Items

When you select the **One Item** filter option or the **Some Items** filter option, you can choose the items that you want to filter.

For example, in the Schema Manager, you can choose tables based on either the table type or table name. When you select **Some Items**, the Hub Console displays the **Define Item Filter** button above the navigation tree.

1. Click the **Define Item Filter** button.
2. Select the items that you want to include in the filter, and then click **OK**.

Changing the Item View

Certain tools in the Hub Console show a **View** or **View By** area below the navigation tree.

- In the Schema Manager, you can show or hide the public Informatica MDM Hub items by clicking the **View** area below the navigation tree and choosing the appropriate command.
For example, you can view all system tables.
- In the Mappings tool, you can view items by mapping, staging table, or landing table.
- In the Packages tool, you can view items by package or by table.
- In the Users and Groups tool, you can display sub groups and sub users.
- In the Batch Viewer, you can group jobs by table, date, or procedure type.

Searching for Items

When there is no filter, or when the **Some Items** filter is selected, the **Navigation** pane includes a **Find** box. Use the **Find** box to find items by full or partial name. For example, in the Schema Manager, you can find all tables that contain "Lookup" in the table name. The find process is case-sensitive.

1. Click the **Find** box.
The **Find** window opens.
2. Type the name or the first few letters of the name that you want to find.
3. Click the **F3 - Find** button.
The first matching item in the tree is highlighted.
4. To go to the next matching item, click the **F3 - Find** button again.
5. To close the **Find** window, click the **Find** box.

Running Commands On Objects in the Navigation Tree

To run commands on an object in the navigation tree, do one of the following:

- Right-click an object name to display a pop-up menu of commands that you can perform on the object.
- Select an object in the navigation tree, and then choose a command from the Hub Console menu at the top of the window.





For example, in the Schema Manager, you can right-click on certain types of objects in the navigation tree to see a pop-up menu of the commands available for the selected object.

Adding, Editing, and Removing Objects Using Command Buttons

You can use command buttons to add, edit, and delete objects in the Hub Console.

Command Buttons

If you have access to create, modify, or remove objects in the Hub Console window and if you have acquired a write lock, you might see some or all of the following command buttons in the **Properties** pane. There are other command buttons as well.

Button	Name	Description
	Add	Add a new object.
	Edit	Edit a property for the selected item in the Properties pane. Indicates that the property is editable.
	Remove	Remove the selected item.
	Save	Save changes.

Note: To see a description about what a command button does, hold the mouse over the button to display a tooltip.

Adding Objects

You can add an object in the Hub Console.

1. Acquire a write lock.
2. Click the **Add** button.

The Hub Console displays an **Add object** window, where object is the name of the type of object that you are adding.

3. Specify the object properties.
4. Click **OK**.

Editing Object Properties

You can edit the properties of an object in the Hub Console.

1. Acquire a write lock and select the object that you want to edit.
2. For each property that you want to edit, click the **Edit** button next to it and specify the new value.
3. Click the **Save** button to save your changes.

Removing Objects

You can remove objects in the Hub Console.

1. Acquire a write lock and select the object that you want to remove.
2. Click the **Remove** button.

If the object is dependent on or related to other objects, the **Impact Analyzer** dialog box appears. If the object is not dependent on or related to other objects, the **Informatica MDM Hub Console** dialog box appears.

3. Choose the appropriate option.

Customizing the MDM Hub Console interface

You can customize the Hub Console interface.

1. Click **Options > Options**.

The **Options** dialog box appears.

2. Specify the options you want from the following tabs:

- **General** tab: Specify whether to show wizard welcome screens or whether to save window sizes and positions.
- **Quick Launch** tab: Specify tools that you want to appear as icons in the quick launch bar below the menu.

Showing Version Details

You can show version details about the installed version of Multidomain MDM.

1. In the Hub Console, choose **Help > About**.

The Informatica Multidomain MDM dialog box opens.

2. Click **Installation Details**.
The **Installation Details** dialog box opens.
3. Click **Close**.
4. Click **Close**.








Informatica MDM Hub Workbenches and Tools

This section provides an overview of the Informatica MDM Hub workbenches and tools.









Tools in the Configuration Workbench

The configuration workbench contains tools to help you with the hub configuration.




The following table lists the tools and their associated icons in the configuration workbench:

Icon	Tool Name	Description
	Databases	Register and manage Operational Reference Store.
	Users	Define users and specify which databases they can access. Manage global and individual password policies. Note that MDM Hub supports external authentication for users, such as LDAP.
	Security Providers	Configure security providers, which are third-party organizations that provide security services (authentication, authorization, and user profile services) for users accessing MDM Hub.
	Tool Access	Define which Hub Console tools and processes a user can access. By default, new user accounts do not have access to any tools until access is explicitly assigned.
	Message Queues	Define inbound and outbound message queue interfaces to MDM Hub.
	Repository Manager	Validate Operational Reference Store metadata, promote changes between repositories, import objects into repositories, and export repositories. For more information, see the <i>Multidomain MDM Repository Manager Guide</i> .
	Enterprise Manager	View configuration details and version information for the Hub Server, Process Server, the MDM Hub Master Database, and Operational Reference Store.




Tools in the Model Workbench

Icon	Tool Name	Description
	Schema	Define base objects, relationships, history and security requirements, staging and landing tables, validation rules, match criteria, and other data model attributes.
	Schema Viewer	View and navigate the current schema.
	Systems and Trust	Name the source systems that can provide data for consolidation in Informatica MDM Hub. Define the trust settings associated with each source system for each base object column.
	Queries	Define query groups and queries used by packages.
	Packages	Define packages (table views).
	Cleanse Functions	Define cleanse functions to perform on your data.
	Mappings	Map cleansing function outputs to target columns in staging tables.
	Hierarchies	Set up the structures required to view and manipulate data relationships in Hierarchy Manager.

Tools in the Security Access Manager Workbench







Icon	Tool Name	Description
	Secure Resources	Manage secure resources in MDM Hub. Configure the status (Private, Secure) for each MDM Hub resource, and define resource groups to organize secure resources.
	Roles	Define roles and privilege assignments to resources and resource groups. Assign roles to users and user groups.
	Users and Groups	Manage the users and user groups within a single Hub Store.

Tools in the Data Steward Workbench

Icon	Tool Name	Description
	Data Manager	Manage the content of consolidated data, view cross-references, edit data, view history and unmerge consolidated records. Note: You cannot use the Data Manager tool to edit or update data in encrypted columns.
	Merge Manager	Review and merge the matched records that have been queued for manual merging.
	Hierarchy Manager	Define and manage hierarchical relationships in the Hub Store.

For more information about the tools in the Data Steward workbench, see the *Multidomain MDM Data Steward Guide*.

Tools in the Utilities Workbench

Icon	Tool Name	Description
	Batch Group	Configure and run batch groups, which are collections of individual batch jobs, such as Stage, Load, and Match jobs, that can be executed with a single command.
	Batch Viewer	Execute batch jobs to cleanse, load, match or auto-merge data, and view job logs.
	Process Server	View Process Server information, including name, port, server type, and whether server is on or offline.
	Audit Manager	Configure auditing and debugging of application requests and message queue events.
	SIF Manager	Generate Operational Reference Store-specific Services Integration Framework (SIF) request APIs. SIF Manager generates and deploys the code to support SIF request APIs for packages, remote packages, mappings, and cleanse functions in an Operational Reference Store. Once generated, the Operational Reference Store-Specific APIs are available as a Web service and through the Siperian API JAR (siperian-api.jar).
	User Object Registry	View registered user exits, custom Java cleanse functions, and custom GUI functions for an Operational Reference Store.

CHAPTER 3

Configuring International Data Support

This chapter includes the following topics:

- [Configuring International Data Support Overview, 43](#)
- [Configuring a Unicode Database \(Oracle Only\), 43](#)
- [Configuring Match Settings for Non-US Populations, 44](#)
- [Configuring the ANSI Code Page in the Windows Registry, 45](#)
- [Cleanse Settings for Unicode, 46](#)
- [Locale Recommendations for UNIX When Using UTF-8, 46](#)
- [Troubleshooting Corrupted Data, 46](#)
- [Configuring Language in Oracle Environments, 47](#)

Configuring International Data Support Overview

If you have data from multiple countries, you can configure character sets in an MDM Hub implementation. The database that you use must support the character set you choose.

If you have an Oracle environment that has data from multiple countries with different character sets, you must use Unicode Transfer Format (UTF-8) encoding. You must also configure the NLS_LANG setting to specify the locale behavior of the client Oracle software.

Configuring a Unicode Database (Oracle Only)

If you use Oracle databases in your MDM Hub implementation, you must configure the character set that you want to use. If your implementation uses mixed locale information, such as data from multiple countries with different character sets, you must configure the MDM Hub and the database to use Unicode Transfer Format

(UTF-8) encoding. However, if the database contains data from a single locale, a UTF-8 database is probably not required.

1. Create a UTF-8 database and choose the following settings:

- **database character set:** AL32UTF8
- **national character set:** AL16UTF16

Note: Oracle recommends to use AL32UTF8 as the database character set for Oracle 10g. For information about the previous Oracle releases, refer to the Oracle documentation.

2. On both the server and the client, set **NLS_LANG** to match the database character set. For example, if you are in the United States, set NLS_LANG to the following value:

```
AMERICAN_AMERICA.AL32UTF8
```

However, if you have a Japanese implementation, set NLS_LANG to the following value:

```
JAPANESE_JAPAN.AL32UTF8
```

3. Ensure that you configure regional font settings on the client. For example, if the data is from China, install Chinese fonts.
4. If you use a multi-byte character set, to support Unicode values, change the following setting in the C_REPOS_DB_RELEASE table:

```
column_length_in_bytes_ind = 0
```

Configuring Match Settings for Non-US Populations

If the MDM Hub implementation uses a non-US population, configure the population set and enable encoding for match processing. A population set encapsulates intelligence about name, address, and other identification information that is typical for a particular population. For more information about population sets, see [“Population Sets” on page 280](#).

The MDM Hub includes a `demo.ysp` file and a `<population>.ysp` file. The `demo.ysp` file contains a population for demonstration purposes only and must not be used for actual match rules. In your MDM Hub implementation, use the `<population>.ysp` population file for which you purchased a license. If you do not have a population file, contact Informatica Global Customer Support to get a population file that is appropriate for your implementation.

Decide on a population set based on the following considerations:

- If the data is exclusively from one country and Informatica provides a population set for that country, then use that population.
- If the data is mostly from one country with small amounts of mixed data from one or more countries, use the majority population.
- If the data is from different countries with large amounts of mixed data, consider whether it is meaningful to match across such a disparate set of data. If so, then use the `international` population.

For more information about enabling a match population, see the *Multidomain MDM Installation Guide*.

Configuring Encoding for Match Processing

To enable processing of UTF-8 characters during matching, edit the Process Server settings.

1. Use a text editor to open the following file: `<MDM Hub Installation Directory>\hub\cleanse\resources\cmxcleanse.properties`

2. Manually add the following setting:

```
cmx.server.match.server_encoding = 1
```

Using Multiple Populations Within a Single Base Object

You can use multiple populations within a single base object in the MDM Hub.

This is useful if data in a base object comes from different populations. For example, 70% of the records come from the United States and 30% come from China. Populations can vary on a record-by-record basis.

To use multiple populations within a base object, perform the following steps:

1. Contact Informatica Global Customer Support to get the applicable `<population>.ysp` files for your implementation, along with instructions for enabling the population.
2. For each population that you want to use, enable it in the `C_REPOS_SSA_POPULATION` metadata table.
3. Copy the applicable population files to the following location:

On UNIX. `<MDM Hub installation directory>/hub/cleanse/`

On Windows. `<MDM Hub installation directory>\cleanse\resources\match`

4. Restart the application server.
5. In the Schema Manager, add a VARCHAR column named `SIP_POP` to the base object that contains the population to use for each record.

Note: The width of the VARCHAR column must fit the largest population name in use. A width of 30 is sufficient for most implementations.

6. Configure the match column as an exact match column with the name of `SIP_POP`.
7. For each record in the base object that uses a population that is not the default, in the `SIP_POP` column enter the name of the population to use instead.

You can specify values for the `SIP_POP` column in one of the following ways:

- Add the UTF-8 data in the landing tables.
- Use cleanse functions that calculate the values during the stage process.
- Invoke SIF requests from external applications.
- Edit the column values manually through the Data Manager tool.

Note: Data in the `SIP_POP` column is not case sensitive, but the MDM Hub processes invalid values, such as NULL values or empty strings, using the default population.

8. Run the Generate Match Tokens process on the base object to update the match key table.
9. Run the match process on the base object.

Note: The match process compares only records that share the same population. For example, the match process compares Chinese records with Chinese records, and American records with American records.

Configuring the ANSI Code Page in the Windows Registry

On Windows, configure the ANSI code page through the Windows Registry Editor.

1. From a command prompt, type `regedit`, and then click **OK**.

2. Navigate to the following registry entry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\ACP
```

3. To change the ANSI code page, configure locale and language settings in the Windows Control Panel. The instructions differ based on which version of Windows you run. For detailed instructions, see your Microsoft Windows documentation.

Cleanse Settings for Unicode

If you use Informatica Address Verification cleanse libraries, ensure that you have the right database and the unlock code for Informatica Address Verification. You must get the Informatica Address Verification database for all countries needed for your implementation. Contact Informatica Global Customer Support for details.

If you use Trillium, ensure that you use the correct template to create the project. To determine which countries are supported, see the Trillium installation documentation. You can get country-specific projects from Trillium directly.

Locale Recommendations for UNIX When Using UTF-8

Many UNIX systems use incompatible character encodings to represent their local alphabets as binary data. This means that, for example, a string of text from a Korean system cannot be viewed in a Chinese system. However, you can configure UNIX systems to use UTF-8 encoding for any language. UTF-8 text encoding supports multiple languages so that one language does not interfere with another.

To configure the system locale settings to use UTF-8, perform the following steps:

1. Run the following command:

```
locale -a
```

2. Determine whether you can find a locale for your language with the suffix `.utf8`.

```
localedef -f UTF-8 -i en_US en_US.utf8
```

3. If you have a locale that allows UTF-8, instruct the UNIX system to use that locale.

```
Export LC_ALL="en_US.utf8"  
export LANG="en_US.utf8"  
export LANGUAGE="en_US.utf8"
```

Troubleshooting Corrupted Data

If you use SQL*Loader to load data that is represented in an international character set, and the loaded data is corrupted, you can fix this issue by setting a property in the `cmxcleanse.properties` file.

1. Navigate to the following directory:

```
<MDM installation directory>/hub/cleanse/resources
```

2. In an editor, open the file `cmxcleanse.properties`.
3. At the end of the file, type the following property and specify the Unicode identifier for the international character set that matches the data you want to load.

```
cmx.server.stage.sqlldr.charset=AL32UTF8
```
4. Save the file.
5. Run the stage process.
The stage process generates a control file for SQL*Loader with the specified character set.
6. Use the control file when you reload the data.

Configuring Language in Oracle Environments

To specify the locale behavior of your client Oracle software, you must set your NLS_LANG setting, which specifies the language, territory, and the character set of your client. How you configure the NLS_LANG setting depends on the operating system.

Windows

You can configure the NLS_LANG setting through the Windows Registry Editor or as an environment variable.

UNIX

You must set the LANG environment variables and install the required locales.

Syntax for NLS_LANG

The NLS_LANG setting uses the following format:

```
NLS_LANG = <LANGUAGE>_<TERRITORY>.<CHARACTERSET>
```

The following table describes the parameters:

Parameter	Description
LANGUAGE	Specifies the language used for Oracle messages, in addition to the names of days and months.
TERRITORY	Specifies monetary and numeric formats, in addition to territory and conventions for calculating week and day numbers.
CHARACTERSET	Controls the character set that the client application uses. Alternatively, this parameter matches your Windows code page or can be set to UTF-8 for a Unicode application.

Note: The character set defined with the NLS_LANG setting does not change the client character set. Instead, the Oracle database converts data to the defined character set. The NLS_LANG setting never inherits the character set from the server.

Configuring NLS_LANG in the Windows Registry

On Windows systems, ensure that you have set an NLS_LANG registry subkey for each Oracle Home.

You can modify this subkey through the Windows Registry Editor.

1. From a command prompt, type `regedit`, and then click **OK**.
2. Navigate to the following registry entry:

For Oracle 10g:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_<Oracle_Home_name>
```

3. Edit the **NLS_LANG** subkey.

Configuring NLS_LANG as an Environment Variable (Windows)

You can set NLS_LANG as a system or user environment variable in the system properties, although Informatica does not recommend this approach. All Oracle Homes use the configured setting.

To check and modify system or user environment variables, perform the following steps:

1. Right-click **My Computer**, and then select **Properties**.
2. On the **Advanced** tab, click **Environment Variables**.

The User variables list contains the settings for the currently logged-in Windows user.

The System variables list contains system-wide variables for all users.

3. Change settings as needed.

Because these environment variables take precedence over the parameters specified in the Windows Registry, avoid setting Oracle parameters at this location unless you have a good reason. In particular, note that the ORACLE_HOME parameter is set on UNIX but not on Windows.

Setting the LANG Environment Variables and Locales (UNIX)

To ensure consistent processing of UTF-8 data throughout the MDM Hub, set the correct locales and the LANG environment variables for UNIX servers that host the application server.

On all the systems in the MDM Hub, including databases and servers, set the following environment variables:

- `export LC_ALL=en_US.UTF-8`
- `export LANG=en_US.UTF-8`
- `export LANGUAGE=en_US.UTF-8`

For Oracle environments, set the following environment variable:

- `export NLS_LANG=AMERICAN_AMERICA.AL32UTF8`

For example, the default LANG environment variable for the United States is `export LANG=en_US`

Therefore, when you use UTF-8, use the following command to configure the LANG environment variable:

```
export LANG=en_US.UTF-8
```

If multiple applications are installed on a machine, and all correct locales are also installed, then you can set the correct environment variable for the profile that starts the application. If the same user profile starts multiple applications, then you can set the environment variable locally in the start-up script of the applications. This ensures that the environment variable is applied locally, and only within the context of the application process.

Usually, all LANG environment variables have the same setting, but you might use different settings. For example, if the interface language is in English but the data that you must sort is in French, then set

LC_MESSAGES to en_US and LC_COLLATE to fr_FR. If you do not need to use different LANG settings, then set LC_ALL or LANG.

An application uses the following rules to determine the locale to be used:

- If the LC_ALL environment variable is defined and is not null, then the application uses the value of LC_ALL.
- If the appropriate component-specific environment variable, such as LC_COLLATE, is set and is not null, then the application uses the value of this environment variable.
- If the LANG environment variable is defined and is not null, then the application uses the value of LANG.
- If the LANG environment variable is not set or is null, then the application uses an implementation-dependent default locale.

Note: If you must use different locales for different scenarios, then do not set LC_ALL.

Part II: Configuring Hub Console Tools

This part contains the following chapters:

- [Configuring Access to Hub Console Tools, 51](#)
- [Implementing Custom Buttons in Hub Console Tools, 53](#)

CHAPTER 4

Configuring Access to Hub Console Tools

This chapter includes the following topics:

- [Configuring Access to Hub Console Tools Overview, 51](#)
- [User Configuration, 51](#)
- [User Access to Tools and Processes, 52](#)

Configuring Access to Hub Console Tools Overview

You can control how MDM Hub users access Hub Console tools. For example, data stewards might have access to only the Data Manager and Merge Manager tools.

You use the Tool Access tool in the Configuration workbench to configure access to Hub Console tools. To use the Tool Access tool, you must be connected to the master database.

The Tool Access tool applies only to MDM Hub users who are not configured as administrators. For more information about user configuration, see the *Multidomain MDM Security Guide*.

User Configuration

You can create, edit, and delete users in the MDM Hub.

Use the Users tool in the Configuration workbench to configure user accounts for MDM Hub users, as well as to change passwords and enable external authentication. The latest information about the latest password and who changed the password is maintained. Password history is not available.

Informatica recommends that you do not use other means to import user information to the MDM Hub directly because it can cause issues in the Hub Store.

You can also use the Users tool to configure user accounts for external application users. An external application user is a MDM Hub user who accesses MDM Hub data indirectly through trusted third-party applications.

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

User Access to Tools and Processes

Use the Tool Access tool in the Configuration workbench to configure access to Hub Console tools.

Starting the Tool Access Tool

To start the Tool Access tool:

1. In the Hub Console, connect to the master database, if you have not already done so.
2. Expand the Configuration workbench and click **Tool Access**.

The Hub Console displays the Tool Access tool.

Granting User Access to Tools and Processes

To grant user access to Hub Console tools and processes for a specific MDM Hub user:

1. Acquire a write lock.
2. In the Tool Access tool, scroll the User list and select the user that you want to configure.
3. Do one of the following:
 - In the **Available processes** list, select a process to which you want to grant access.
 - In the **Available workbenches** list, select a workbench that contains the tools to which you want to grant access.

4. Click **Add tool** or **Add process**.

The Tool Access tool adds the selected tool or process to the **Accessible tools and processes** list. When you grant access to a process, the MDM Hub grants access to any tool that the process uses. When you grant access to a tool, the MDM Hub grants access to any process that uses the tool.

The user can access these processes and tools for every Operational Reference Store to which they have access. You cannot give a user access to one tool for one Operational Reference Store and another tool for a different Operational Reference Store.

Note: If you do not want to grant access to all tools in a workbench, expand the associated workbench in the **Accessible tools and processes** list, and revoke access for selected tools.

Revoking User Access to Tools and Processes

To revoke user access to Hub Console tools and processes for a specific MDM Hub user:

1. Acquire a write lock.
2. In the Tool Access tool, scroll the User list and select the user that you want to configure.
3. Scroll the Accessible tools and processes list and select the process, workbench, or tool to which you want to revoke access.

To select a tool, expand the associated workbench.

4. Click **Remove tool** or **Remove process**.

The Tool Access tool prompts you to confirm that you want to remove the access.

5. Click **Yes**.

The Tool Access tool removes the selected item from the Accessible tools and processes list. When you revoke access to a process, the MDM Hub revokes access to any tool that the process uses. When you revoke access to a tool, the MDM Hub revokes access to any process that uses the tool.

CHAPTER 5

Implementing Custom Buttons in Hub Console Tools

This chapter includes the following topics:

- [Overview, 53](#)
- [About Custom Buttons in the Hub Console, 53](#)
- [Adding Custom Buttons, 54](#)
- [Controlling the Custom Button Appearance, 57](#)
- [Deploying Custom Buttons, 57](#)

Overview

This chapter explains how, in a Informatica MDM Hub implementation, you can add custom buttons to tools in the Hub Console that allow you to invoke external services on demand.

About Custom Buttons in the Hub Console

In your Informatica MDM Hub implementation, you can provide Hub Console users with custom buttons that can be used to extend your Informatica MDM Hub implementation.

Custom buttons can provide users with on-demand, real-time access to specialized data services. Custom buttons can be added to Merge Manager and Hierarchy Manager.

Custom buttons can give users the ability to invoke a particular external service (such as retrieving data or computing results), perform a specialized operation (such as launching a workflow), and other tasks. Custom buttons can be designed to access data services by a wide range of service providers, including—but not limited to—enterprise applications (such as CRM or ERP applications), external service providers (such as foreign exchange calculators, publishers of financial market indexes, or government agencies), and even Informatica MDM Hub itself (for more information, see the *Multidomain MDM Services Integration Framework Guide*).

For example, you could add a custom button that invokes a specialized cleanse function, offered as a Web service by a vendor, that cleanses data in the customer record that is currently selected in the Merge Manager screen. When the user clicks the button, the underlying code would capture the relevant data from the selected record, create a request (possibly including authentication information) in the format expected

by the Web service, and then submit that request to the Web service for processing. When the results are returned, the Hub displays the information in a separate Swing dialog (if you created one and if you implemented this as a client custom function) with the customer rowid_object from Informatica MDM Hub.

Custom buttons are not installed by default, nor are they required for every Informatica MDM Hub implementation. For each custom button you need to implement a Java interface, package the implementation in a JAR file, and deploy it by running a command-line utility. To control the appearance of the custom button in the Hub Console, you can supply either text or an icon graphic in any Swing-compatible graphic format (such as JPG, PNG, or GIF).

What Happens When a User Clicks a Custom Button

When a user selects a customer record then clicks a custom button in the Hub Console, the Hub Console invokes the request, passing content and context to the Java external (custom) service. Examples of the type of data include record keys and other data from a base object, package information, and so on. Execution is asynchronous—the user can continue to work in the Hub Console while the request is processed.

The custom code can process the service response as appropriate—log the results, display the data to the user in a separate Swing dialog (if custom-coded and the custom function is client-side), allow users to copy and paste the results into a data entry field, execute real-time PUT statements of the data back into the correct business objects, and so on.

How Custom Buttons Appear in the Hub Console

This section shows how custom buttons, once implemented, will appear in the Merge Manager and Hierarchy Manager tools of the Hub Console.

Custom Buttons in the Merge Manager

Custom buttons are displayed to the right of the top panel of the Merge Manager, in the same location as the regular Merge Manager buttons.

Custom Buttons in the Hierarchy Manager

Custom buttons are displayed in the top part of the top panel of the Hierarchy Manager screen, in the same location as other Hierarchy Manager buttons.

Adding Custom Buttons

To add a custom button to the Hub Console in your Informatica MDM Hub implementation, complete the following tasks:

1. Determine the details of the external service that you want to invoke, such as the format and parameters for request and response messages.
2. Write and package the business logic that the custom button will execute.
3. Deploy the package so that it appears in the applicable tool(s) in the Hub Console.

Once an external service button is visible in the Hub Console, users can click the button to invoke the service.

Writing a Custom Function

To build an external service invocation, you write a custom function that executes the application logic when a user clicks the custom button in the Hub Console. The application logic implements the following Java interface:

```
com.siperian.mrm.customfunctions.api.CustomFunction
```

To learn more about this interface, see the Javadoc that accompanies your Informatica MDM Hub distribution.

Server-Based and Client-Based Custom Functions

Execution of the application logic occurs on either:

Environment	Description
Client	UI-based custom function—Recommended when you want to display elements in the user interface, such as a separate dialog that displays response information.
Server	Server-based custom button—Recommended when it is preferable to call the external service from the server for network or performance reasons.

Example Custom Functions

This section provides the Java code for two example custom functions that implement the `com.siperian.mrm.customfunctions.api.CustomFunction` interface. The code simply prints (on standard error) information to the server log or the Hub Console log.

Example Client-Based Custom Function

The name of the client function class for the following sample code is `com.siperian.mrm.customfunctions.test.TestFunction`.

```
package com.siperian.mrm.customfunctions.test;
import java.awt.Frame;
import java.util.Properties;

import javax.swing.Icon;
import com.siperian.mrm.customfunctions.api.CustomFunction;

public class TestFunctionClient implements CustomFunction {

    public void executeClient(Properties properties, Frame frame, String username,
String password, String orsId, String baseObjectRowid, String baseObjectUid, String
packageRowid, String packageUid, String[] recordIds) {
        System.err.println("Called custom test function on the client with the following
parameters:");
        System.err.println("Username/Password: '" + username + "/" + password + "'");
        System.err.println(" ORS Database ID: '" + orsId + "'");
        System.err.println("Base Object Rowid: '" + baseObjectRowid + "'");
        System.err.println(" Base Object UID: '" + baseObjectUid + "'");
        System.err.println("    Package Rowid: '" + packageRowid + "'");
        System.err.println("    Package UID: '" + packageUid + "'");
        System.err.println("        Record Ids: ");
        for(int i = 0; i < recordIds.length; i++) {
            System.err.println("        '"+recordIds[i]+'");
        }
        System.err.println("        Properties: " + properties.toString());
    }

    public void executeServer(Properties properties, String username, String password,
```

```

String orsId, String baseObjectRowid, String baseObjectUid, String packageRowid,
String packageUid, String[] recordIds) {
    System.err.println("This method will never be called because getExecutionType()
returns CLIENT_FUNCTION");
}
    public String getActionText() { return "Test Client"; }
    public int getExecutionType() { return CLIENT_FUNCTION; }
    public Icon getGuiIcon() { return null; }
}

```

Example Server-Based Function

The name of the server function class for the following code is `com.siperian.mrm.customfunctions.test.TestFunctionClient`.

```

package com.siperian.mrm.customfunctions.test;
import java.awt.Frame;
import java.util.Properties;
import javax.swing.Icon;

import com.siperian.mrm.customfunctions.api.CustomFunction;

/**
 * This is a sample custom function that is executed on the Server.
 * To deploy this function, put it in a jar file and upload the jar file
 * to the DB using DeployCustomFunction.
 */
public class TestFunction implements CustomFunction {
    public String getActionText() {
        return "Test Server";
    }
    public Icon getGuiIcon() {
        return null;
    }
    public void executeClient(Properties properties, Frame frame, String username,
String password, String orsId, String baseObjectRowid, String baseObjectUid,
String packageRowid, String packageUid, String[] recordIds) {
        System.err.println("This method will never be called because getExecutionType()
returns SERVER_FUNCTION");
    }
    public void executeServer(Properties properties, String username, String password,
String orsId, String baseObjectRowid, String baseObjectUid, String packageRowid,
String packageUid, String[] recordIds) {
        System.err.println("Called custom test function on the server with the following
parameters:");
        System.err.println("Username/Password: '" + username + "'/' + password + "'");
        System.err.println(" ORS Database ID: '" + orsId + "'");
        System.err.println("Base Object Rowid: '" + baseObjectRowid + "'");
        System.err.println(" Base Object UID: '" + baseObjectUid + "'");
        System.err.println("    Package Rowid: '" + packageRowid + "'");
        System.err.println("    Package UID: '" + packageUid + "'");
        System.err.println("    Record Ids: ");
        for(int i = 0; i < recordIds.length; i++) {
            System.err.println("    '"+recordIds[i]+'");
        }
        System.err.println("    Properties: " + properties.toString());
    }
    public int getExecutionType() {
        return SERVER_FUNCTION;
    }
}

```

Controlling the Custom Button Appearance

To control the appearance of the custom button in the Hub Console, you implement one of the following methods in the `com.siperian.mrm.customfunctions.api.CustomButton` interface:

Method	Description
<code>getActionText</code>	Specify the text for the button label. Uses the default visual appearance for custom buttons.
<code>getGuilcon</code>	Specify the icon graphic in any Swing-compatible graphic format (such as JPG, PNG, or GIF). This image file can be bundled with the JAR file for this custom function.

Custom buttons are displayed alphabetically by name in the Hub Console.

Deploying Custom Buttons

Before you can see the custom buttons in the Hub Console, you need to explicitly add them using the `DeployCustomFunction` utility from the command line.

To deploy custom buttons:

1. Open a command prompt.
2. Run the `DeployCustomFunction` utility, which loads and registers a JAR file that a user has created.

Note: To run `DeployCustomFunction`, two JAR files must be in the CLASSPATH—`siperian-server.jar` and the JDBC driver (in this case, `ojdbc14.jar`)— with directory paths that point to these files.

Specify following command at the command prompt:

```
java -cp siperian-server.jar; ojdbc14.jar
com.siperian.mrm.customfunctions.dbadapters.DeployCustomFunction
```

Reply to the prompts based on the configured settings for your Informatica MDM Hub implementation.

For example:

```
Database Type:oracle
Host:localhost
Port(1521):
Service:orcl
Username:ds_uil
Password:!!cmx!!
(L)ist, (A)dd, (U)pdate, (C)hange Type, (S)et Properties, (D)elele or (Q)uit:l
No custom actions
(L)ist, (A)dd, (U)pdate Jar, (C)hange Type, (S)et Properties, (D)elele or (Q)uit:q
```

3. At the respective prompts, specify the following information (based on the configured settings for your Informatica MDM Hub implementation):
 - Database host
 - Port
 - Service
 - Login username (schema name)
 - Login password
4. When prompted, specify database connection information: database host, port, service, login username, and password.

5. The DeployCustomFunction tool displays a menu of the following options.

Label	Description
(L)ist	Displays a list of currently-defined custom buttons.
(A)dd	Adds a new custom button. The DeployCustomFunction tool prompts you to specify: <ul style="list-style-type: none">- the JAR file for your custom button- the name of the custom function class that implements the com.siperian.mrm.customfunctions.api.CustomFunction interface- the type of the custom button: m—Merge Manager, d—Data Manager, h—Hierarchy Manager (you can specify one or two letters)
(U)pdate	Updates the JAR file for an existing custom button. The DeployCustomFunction tool prompts you to specify: <ul style="list-style-type: none">- the rowID of the custom button to update- the JAR file for your custom button- the name of the custom function class that implements the com.siperian.mrm.customfunctions.api.CustomFunction interface- the type of the custom button: m—Merge Manager, h—Hierarchy Manager (you can specify one or two letters)
(C)hange Type	Changes the type of an existing custom button. The DeployCustomFunction tool prompts you to specify: <ul style="list-style-type: none">- the rowID of the custom button to update- the type of the custom button: m—Merge Manager, and /or h—Hierarchy Manager (you can specify one or two letters)
(S)et Properties	Specify a properties file, which defines name/value pairs that the custom function requires at execution time (name=value). The DeployCustomFunction tool prompts you to specify the properties file to use.
(D)elete	Deletes an existing custom button. The DeployCustomFunction tool prompts you to specify the rowID of the custom button to delete.
(Q)uit	Exits the DeployCustomFunction tool.

6. When you have finished choosing your actions, choose **(Q)uit**.
7. Refresh the browser window to display the custom button you just added.
8. Test your custom button to ensure that it works properly.

Part III: Building the Data Model

This part contains the following chapters:

- [About the Hub Store , 60](#)
- [Configuring Operational Reference Stores and Data Sources, 62](#)
- [Building the Schema, 75](#)
- [Queries and Packages, 124](#)
- [Timeline, 141](#)
- [State Management and BPM Workflow Tools, 169](#)
- [Data Encryption, 182](#)
- [Hierarchies, 189](#)
- [Hierarchy Manager Tutorial, 223](#)

CHAPTER 6

About the Hub Store

This chapter includes the following topics:

- [Overview, 60](#)
- [Databases in the Hub Store, 60](#)
- [How Hub Store Databases Are Related, 61](#)
- [Creating Hub Store Databases, 61](#)
- [Version Requirements, 61](#)

Overview

The Hub Store is where business data is stored and consolidated in Informatica MDM Hub.

The Hub Store contains common information about all of the databases that are part of your Informatica MDM Hub implementation.

Databases in the Hub Store

The Hub Store is a collection of databases that includes:

Element	Description
Master Database	Contains the Informatica MDM Hub environment configuration settings—user accounts, security configuration, Operational Reference Store registry, message queue settings, and so on. A given Informatica MDM Hub environment can have only one Master Database. The default name of the Master Database is CMX_SYSTEM. In the Hub Console, the tools in the Configuration workbench (Databases, Users, Security Providers, Tool Access, and Message Queues) manage configuration settings in the Master Database.
Operational Reference Store (Operational Reference Store)	Database that contains the master data, content metadata, the rules for processing the master data, the rules for managing the set of master data objects, along with the processing rules and auxiliary logic used by the Informatica MDM Hub in defining the best version of the truth (BVT). An Informatica MDM Hub configuration can have one or more ORS databases. The default name of an Operational Reference Store is CMX_ORS.

Users for Hub Store databases are created globally—within the Master Database—and then assigned to specific Operational Reference Stores. The Master Database also stores site-level information, such as the number of incorrect log-in attempts allowed before a user account is locked out.

How Hub Store Databases Are Related

An Informatica MDM Hub implementation contains one Master Database and zero or more ORSs.

If no ORS exists, then only the Configuration workbench tools are available in the Hub Console. A Informatica MDM Hub implementation can have multiple Operational Reference Stores, such as separate Operational Reference Stores for development and production, or separate Operational Reference Stores for each geographical location or for different parts of the organization.

You can access and manage multiple Operational Reference Stores from one Master Database. The Master Database stores the connection settings and properties for each Operational Reference Store.

Note: An Operational Reference Store can be registered in only one Master Database. Multiple Master Databases cannot share the same Operational Reference Store. A single Operational Reference Store cannot be associated with multiple Master Databases.

Creating Hub Store Databases

Databases are initially created and configured when you install Informatica MDM Hub.

- To create the Master Database and one ORS, you must run the `setup.sql` script.
- To create an individual ORS, you must run the `setup_ors.sql` script.

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

Version Requirements

Different versions of the Informatica MDM Hub cannot operate together in the same environment.

All components of your installation must be the same version, including the Informatica MDM Hub software and the databases in the Hub Store.

If you want to have multiple versions of Informatica MDM Hub at your site, you must install each version in a separate environment. If you try to work with a different version of a database, you will receive a message telling you to upgrade the database to the current version.

CHAPTER 7

Configuring Operational Reference Stores and Data Sources

This chapter includes the following topics:

- [Configuring Operational Reference Stores and Data Sources Overview, 62](#)
- [Before You Begin, 62](#)
- [About the Databases Tool, 63](#)
- [Starting the Databases Tool, 63](#)
- [Configuring Operational Reference Stores, 63](#)
- [Data Source Configuration, 73](#)

Configuring Operational Reference Stores and Data Sources Overview

You can use the Databases tool in the Hub Console to configure Operational Reference Stores and data sources for the Hub Store. After you create an Operational Reference Store, you must register and define it in the Databases tool. Also, you can use the Databases tool to create or remove data sources.

Before You Begin

Before you begin, you must install the MDM Hub, create the MDM Hub Master Database and at least one Operational Reference Store. See the instructions in the *Multidomain MDM Installation Guide* to create the MDM Hub Master database and Operational Reference Store.

About the Databases Tool

After you create the Hub Store, use the Databases tool in the Hub Console to register and define Operational Reference Stores.

You use the Databases tool to register an Operational Reference Store so that the MDM Hub can connect to it. Registration of an Operational Reference Store stores the database connection properties in the MDM Hub Master Database.

You use the Databases tool to create a data source for the Operational Reference Store. An Operational Reference Store data source contains a set of properties for the Operational Reference Store. It contains properties such as the location of the database server, the name of the database, the network protocol used to communicate with the server, the database user ID and password.

Note: The Databases tool refers to an Operational Reference Store as a database.

Starting the Databases Tool

Start the Databases tool in the Hub Console.

1. In the Hub Console, connect to the MDM Hub Master Database.
2. Expand the Configuration workbench, and then click **Databases**.

The Hub Console displays the Databases tool in which registered Operational Reference Stores appear.

The Databases tool displays the following database information:

Database Information	Description
Number of databases	Number of Operational Reference Stores defined in the Hub Store.
Database List	List of registered MDM Hub Operational Reference Store databases.
Database Properties	Database properties for the Operational Reference Store that you select.

Configuring Operational Reference Stores

You can use the Databases tool of the Hub Console to configure an Operational Reference Store in the Hub Store.

If you need assistance with configuring the Operational Reference Store, consult with your database administrator. For more information about Operational Reference Stores, see the *Multidomain MDM Installation Guide*.

Operational Reference Store Connection Properties for Microsoft SQL Server

When you register an Operational Reference Store on Microsoft SQL Server, configure the following connection properties:

Database Display Name

Name for the Operational Reference Store that must appear in the Hub Console.

Machine Identifier

Prefix given to keys to uniquely identify records from the Hub Store instance.

Database hostname

IP address or name of the server that hosts the Microsoft SQL Server database.

Port

Port of the Microsoft SQL Server database. The default is 1433.

Database Name

Name of the Operational Reference Store.

User name

User name for the Operational Reference Store. By default, this is the user name that you specify when you create the Operational Reference Store. This user owns all of the Operational Reference Store database objects in the Hub Store.

Note: You do not need to provide the user name for Microsoft SQL Server.

Password

Password associated with the user name for the Operational Reference Store.

DDM connection URL

Optional. URL for the Dynamic Data Masking server. Leave empty if you do not use Dynamic Data Masking.

You can also configure the following additional properties on the **Summary** page:

Connection URL

Connect URL. The Connection Wizard generates the connect URL by default.

Create datasource after registration

Select to create the datasource on the application server after registration.

Operational Reference Store Connection Properties for Oracle

When you register an Operational Reference Store on Oracle, configure the following connection properties:

Database Display Name

Name for the Operational Reference Store that must appear in the Hub Console.

Machine Identifier

Prefix given to keys to uniquely identify records from the Hub Store instance.

Database hostname

IP address or name of the server that hosts the Microsoft SQL Server database.

SID

Oracle System Identifier that refers to the instance of the Oracle database running on the server. This field appears if you selected the `SID` connection type.

Service

Name of the Oracle service used to connect to the Oracle database. This field appears if the you selected the `Service` connection type.

Port

The TCP port of the Oracle listener running on the Oracle database server. The default is 1521.

Oracle TNS Name

Name by which the database is known on your network as defined in the `TNSNAMES.ORA` file of the application server.

For example: `mydatabase.mycompany.com`.

You set the Oracle TNS name when you install the Oracle database. For more information about the Oracle TNS name, see the Oracle documentation.

Schema Name

Name of the Operational Reference Store.

Note: The **Schema Name** and the **User Name** are both the names of the Operational Reference Store that you specified when you creates the Operational Reference Store. If you need this information, consult your database administrator.

Password

Password associated with the user name for the Operational Reference Store.

For Oracle, this password is not case sensitive.

By default, this is the password that you specify when you create the Operational Reference Store.

DDM connection URL

Optional. URL for the Dynamic Data Masking server. Leave empty if you do not use Dynamic Data Masking.

You can also configure the following additional properties on the **Summary** page:

Connection URL

Connect URL. The Connection Wizard generates the connect URL by default. The following examples show the format of the connect URL:

Service connection type:

```
jdbc:oracle:thin:@//database_host:port/service_name
```

SID connection type:

```
jdbc:oracle:thin:@//database_host:port/sid
```

You can specify a custom URL of a service connection type. Example:

```
jdbc:oracle:thin:@//orclhost:1521/mdmorcl.mydomain.com
```

Create datasource after registration

Select to create the datasource on the application server after registration.

Note: If you do not select the option, you must manually configure the data source.

Operational Reference Store Connection Properties for IBM DB2

When you register an Operational Reference Store on IBM DB2, configure the following connection properties:

Database Display Name

Name for the Operational Reference Store that must appear in the Hub Console.

Machine Identifier

Prefix given to keys to uniquely identify records from the Hub Store instance.

Database server name

IP address or name of the server that hosts the IBM DB2 database.

Database name

Name of the database that you create.

Database hostname

IP address or name of the server that hosts the IBM DB2 database.

Port

The TCP port for the IBM DB2 database server. The default is 5000.

Schema Name

Name of the Operational Reference Store.

Note: The **Schema Name** and the **User Name** are both the names of the Operational Reference Store that you specified when you created the Operational Reference Store. If you need this information, consult your database administrator.

User name

User name for the Operational Reference Store. By default, this is the user name that you specify in the script that you use to create the Operational Reference Store. This user owns all the Operational Reference Store database objects in the Hub Store.

Password

Password associated with the user name for the Operational Reference Store.

For IBM DB2, the password is case sensitive.

By default, this is the password that you specify when you create the Operational Reference Store.

DDM connection URL

Optional. URL for the Dynamic Data Masking server. Leave empty if you do not use Dynamic Data Masking.

You can also configure the following additional properties on the **Summary** page:

Connection URL

Connect URL. The Connection Wizard generates the connect URL by default. The following example shows the format of the connect URL:

```
jdbc:db2://database_host:port/db_name
```

Create datasource after registration

Select to create the datasource on the application server after registration.

Note: If you do not select the option, you must manually configure the data source.

Registering an Operational Reference Store

You can register an Operational Reference Store through the Hub Console. You cannot register an Operational Reference Store with more than one MDM Hub Master Database.

Note: An ORS password can contain up to 25 characters. If you set a password that exceeds this limit, the ORS cannot be registered.

1. Start the Hub Console.
The **Change database** dialog box appears.
2. Select the MDM Hub Master database, and click **Connect**.
3. Start the **Databases** tool under the Configuration workbench.
4. Acquire a write lock.
5. Click the **Register database** button.
The **Informatica MDM Hub Connection Wizard** appears and prompts you to select the database type.
6. Select the type of database, and click **Next**.
7. If you create a connection to an Oracle database, select the connection method and click **Next**:
 - Select **Service** to connect to Oracle using the service name.
 - Select **SID** to connect to Oracle using the Oracle System ID.

Note: For more information about Service and SID names, see the Oracle documentation.

The **Connection Properties** page appears.
8. Configure connection properties for the database.
9. Review your changes on the **Summary** page and specify additional connection properties.
10. Click **Finish**.
The **Registering Database** dialog box appears.
11. Click **OK**.
The MDM Hub registers the Operational Reference Store.
12. Select the Operational Reference Store that you registered, and click the **Test database connection** button to test the database settings.
If you use WebSphere, restart WebSphere before you test the database connection.
The **Test Database** dialog box displays the result of the database connection test.
13. Click **OK**.

The Operational Reference Store is registered, and the connection to the database is tested.

Note: When you register an Operational Reference Store that is used elsewhere, and if the Operational Reference Store has Process Servers registered, then other servers might not register. You need to re-register one of the Process Servers. This updates the data in c_repos_db_release.

Editing Operational Reference Store Registration Properties

You can edit certain Operational Reference Store registration properties. Unregister and re-register the Operational Reference Store with new properties if you want to edit properties that cannot be edited.

1. Start the Databases tool.
2. Acquire a write lock.
3. Select the Operational Reference Store that you want to configure.

4. Click the **Edit database connection properties** button.

The **Register Database** dialog box for the selected Operational Reference Store appears.

5. Edit the database registration settings.

- Edit the following database registration settings for Oracle:

Property	Description
Database Display Name	Name for the Operational Reference Store as that must appear in the Hub Console.
Machine Identifier	Prefix assigned to keys to uniquely identify records from the Hub Store instance.
Oracle TNS name	Name by which the database is known on the network. The TNS name is defined in the <code>TNSNAMES.ORA</code> file.
Password	Password associated with the user name that was specified when the Operational Reference Store was created.
DDM connection URL	Optional. URL for the Dynamic Data Masking server. Leave empty if you do not use Dynamic Data Masking.

- Edit the following database registration settings for Microsoft SQL Server:

Property	Description
Database Display Name	Name for the Operational Reference Store as that must appear in the Hub Console.
Machine Identifier	Prefix assigned to keys to uniquely identify records from the Hub Store instance.
Password	Password associated with the user name that was specified when the Operational Reference Store was created.
DDM connection URL	Optional. URL for the Dynamic Data Masking server. Leave empty if you do not use Dynamic Data Masking.

6. To update the data source on the application server with the settings that you change, enable the **Update datasource after registration** option, and click **OK**.

The Hub Console prompts you to reset the data source parameters and the corresponding connection pool parameters to the default values.

7. To reset to the default data source and connection pool parameters, click **Yes**. To retain the data source and connection pool parameters, click **No**.

The **Update Database Registration** dialog box appears.

8. Click **OK**.

The Databases tool saves your changes.

9. Test the updated database connection settings.

Editing Operational Reference Store Properties

You can change the properties for a registered Operational Reference Store.

1. Start the **Databases** tool.
2. Acquire a write lock.
3. Select the Operational Reference Store that you want to configure.

The Databases tool displays the database properties for the selected Operational Reference Store.

The following table describes the database properties:

Property	Description
Database Type	Type of database such as Oracle, or Microsoft SQL Server.
Database ID	<p>Identification for the Operational Reference Store. The MDM Hub uses the database ID in SIF requests. The database ID lookup is case-sensitive.</p> <p>The database ID for Oracle uses the following format:</p> <ul style="list-style-type: none">- If the Oracle connection type is SID, the format is <code>hostname-sid-databasename</code>.- If the Oracle connection type is Service, the format is <code>hostname-sid-databasename</code>. <p>The database ID for Microsoft SQL Server uses the following format:</p> <p><code>hostname-databasename</code></p> <p>When you register an Operational Reference Store, the MDM Hub normalizes the host, server, and database names.</p> <p>The MDM Hub converts the host name to lowercase, and the database name to uppercase, which is the standard for database objects such as schemas and tables.</p>
JNDI Datasource Name	<p>The data source JNDI name for the Operational Reference Store that you select. This is the JNDI name that the MDM Hub configures for the JDBC connection on the application server.</p> <p>The JNDI data source name for Oracle uses the following format:</p> <ul style="list-style-type: none">- If the Oracle connection type is SID, the format is <code>jdbc/siperian-hostname-sid-databasename-ds</code>.- If the Oracle connection type is Service, the format is <code>jdbc/siperian-servicename-databasename-ds</code>. <p>The JNDI data source for Microsoft SQL Server uses the following format:</p> <p><code>jdbc/siperian-hostname-databasename-ds</code></p>
Machine Identifier	Prefix assigned to keys to uniquely identify records from the Hub Store instance.
GETLIST Limit (records)	Limit for the number of records returned through SIF search requests such as <code>searchQuery</code> , <code>searchMatch</code> , and <code>getLookupValues</code> . Default is 200 and the maximum value is 5,999.
Production Mode	<p>Specifies whether the Operational Reference Store is in production mode or normal mode. If disabled, authorized users can edit metadata for the Operational Reference Store in the Hub Console. Default is disabled.</p> <p>If enabled, users cannot edit the metadata for the Operational Reference Store. If a user attempts to acquire a write lock on an Operational Reference Store in production mode, the Hub Console displays a message explaining that the lock cannot be obtained.</p> <p>Note: Only an MDM Hub administrator user can enable or disable the production mode for an Operational Reference Store.</p>

Property	Description
Transition Mode	Specifies whether the Operational Reference Store is running in transition mode. Transition mode is available if you enable production mode for the Operational Reference Store. If enabled, users can perform the promote actions of the Repository Manager. If disabled, users cannot perform the promote actions of the Repository Manager.
Batch API Interoperability	Specifies whether the MDM Hub can use row-level locking for the Operational Reference Store to concurrently execute SIF API write calls and batch operations. If enabled, row-level locking is available for SIF API write calls, improving concurrency and performance. If disabled, row-level locking is unavailable.
ZDT Enabled	Specifies whether the Operational Reference Store is running in Zero Downtime (ZDT) mode. If enabled, the Operational Reference Store runs in ZDT mode. If disabled, the Operational Reference Store does not runs in ZDT mode.

- To change a property, click the **Edit** button next to it, and edit the property.
- Click the **Save** button to save the changes.

If you enable production mode for an Operational Reference Store, the Databases tool displays a lock icon next to the Operational Reference Store in the list.

Operational Reference Store Connections

You must ensure that you are able to connect to the Operational Reference Store.

When you test the Operational Reference Store connection, the Test Database command tests for the following connection properties:

- Database connection parameters
- Existence of a data source
- Data source connection
- Validity of the Operational Reference Store version

Testing Operational Reference Store Connections

You can test the connection to an Operational Reference Store.

- Start the **Databases** tool.
- Select the Operational Reference Store that you want to test.
- Click the **Test database connection** button.

The **Test Database** dialog box appears.

Note: For WebSphere, if the test connection fails through the Hub Console, verify the connection from the WebSphere Console. The JNDI name is case sensitive and must match what the Hub Console generates.

- Click **OK**.

Changing Passwords

You can change passwords for the MDM Hub Master Database or an Operational Reference Store after you install the MDM Hub.

Changing the Password for the MDM Hub Master Database in an Oracle Environment

To change the password for the MDM Hub Master Database in an Oracle environment, change the password on the database server, and then, update the password on the application server.

1. On the database server, change the password for the CMX_SYSTEM database.
2. Log in to the administration console for your application server.
3. Edit the datasource connection information to specify the CMX_SYSTEM password you created in step 1. Save your changes.

Changing the Password for the MDM Hub Master Database in an IBM DB2 Environment

To change the password for the MDM Hub Master Database in an IBM DB2 environment, change the password in the operating system, and then, update the password on the application server.

1. In the operating system environment, change the password for CMX_SYSTEM.
2. Log in to the administration console for your application server.
3. Edit the datasource connection information to specify the CMX_SYSTEM password you created in step 1. Save your changes.

Changing the Password for an Operational Reference Store

You can change the password for an Operational Reference Store.

1. If you are in an IBM DB2 environment, or in a Microsoft SQL Server environment with operating system authentication enabled, change the operation system password. The password must match the password for the Operational Reference Store.
2. On the database server, change the password for the Operational Reference Store schema.
3. Start the Hub Console and select MDM Hub Master Database as the target database.
4. Start the **Databases** tool.
5. Acquire a write lock.
6. Select the Operational Reference Store that you want to configure.
7. In the **Database Properties** panel, make a note of the JNDI datasource name for the selected Operational Reference Store.
8. Log into the administration console for your application server. Edit the datasource connection information for the Operational Reference Store, specifying the new password for the noted JNDI Datasource name, and then saving your changes.
9. In the Databases tool, test the connection to the database.

Password Encryption

To successfully change the schema password, you must change it in the data sources defined in the application server.

The password is not encrypted, because the application server protects it. In addition to updating the data sources on the application server, you must encrypt and store the password in various tables.

Encrypting a Password for the Schema

You can encrypt a database schema password to secure it.

- To encrypt a database schema password, run the following command from a command prompt:

```
java -classpath siperian-api.jar;siperian-common.jar;siperian-server.jar
com.delos.util.PublicKeyBasedEncryptionHelper <plain text password> <Hub Server
installation directory>
```

The results are echoed to the terminal window:

```
Plaintext Password: password
Encrypted Password: encrypted password
```

Updating the Password for the Schema

You can update the MDM Hub Master Database password or Operational Reference Store password.

1. To update your Master Database password or Operational Reference Store password, connect as the `cmx_system` user and run the following statement:

On Oracle and IBM Db2.

```
UPDATE C_REPOS_DATABASE SET PASSWORD = '<new_password>' WHERE USER_NAME =
<user_name>;
COMMIT;
```

On Microsoft SQL Server.

```
UPDATE [dbo].[C_REPOS_DATABASE] SET PASSWORD = '<new_password>' WHERE USER_NAME =
<user_name>
```

2. Restart the application server.

Production Mode for an Operational Reference

MDM Hub administrators can use the Hub Console to enable production mode for an Operational Reference Store. When the Operational Reference Store is in production mode, the design of the Operational Reference Store is locked.

If an Operational Reference Store is in production mode, you cannot acquire a write lock if you are a user without administrative privileges. You cannot make changes to the schema definition in the Operational Reference Store. If you attempt to acquire a lock on an Operational Reference Store that is in production mode, the Hub Console displays a message to explain that the lock cannot be obtained because the Operational Reference Store is in production mode.

Enabling Or Disabling Production Mode for an Operational Reference

Use the Hub Console to enable production mode for an Operational Reference Store.

To enable the production mode for an Operational Reference Store, you must have administrative privileges to run the Databases tool and be able to obtain a lock on the Master Database.

1. Log into the Hub Console with administrative privileges to the MDM Hub implementation.

2. Start the **Databases** tool.
3. Clear exclusive locks on the Operational Reference Store.
You cannot enable or disable production mode for an Operational Reference Store if the Operational Reference Store has an exclusive lock.
4. Acquire a write lock.
5. Select the Operational Reference Store that you want to configure.
The Databases tool displays the database properties for the Operational Reference Store that you select.
6. Enable or disable the **Production Mode** option.
7. Click the **Save**.

Unregistering an Operational Reference Store

You can use the Databases tool to unregister an Operational Reference Store (ORS). When you unregister an ORS, the MDM Hub removes the connection information to ORS from the MDM Hub Master Database. When you unregister an ORS, the MDM Hub also removes the data source definition from the application server environment.

1. From the MDM Hub Console, click **Write Lock > Acquire Lock**.
2. From the **Configuration** workbench, select the **Databases** tool.
The **Database Information** page appears.
3. From the list of databases, select the ORS to unregister.
4. Click **Unregister database**. If you use WebLogic, enter the user name and password for the application server.
The Databases tool prompts you to confirm that you want to unregister the ORS.
5. Click **Yes**.

Dropping an Operational Reference Store in IBM DB2

If you do not want to use an Operational Reference Store, you can remove it from the database by dropping it.

1. Use the MDM Hub Console to unregister the Operational Reference Store.
2. Open a IBM DB2 command window.
3. Enter the following command:

```
CALL SYSPROC.ADMIN_DROP_SCHEMA('<Operational Reference Store name>', NULL,
'ERRORSCHEMA', 'ERRORTABLE')
```

The Operational Reference Store specified in the command is dropped.

Data Source Configuration

Every Operational Reference Store requires a data source definition in the application server environment. In MDM Hub, a data source specifies properties for an Operational Reference Store, such as the location of the database server, the name of the database, the database user ID, and password.

An MDM Hub data source points to a JDBC resource defined in the application server environment. To learn more about JDBC data sources, see the application server documentation.

Managing Data Sources in WebLogic

For WebLogic, whenever you attempt to add, delete, or update a data source, the MDM Hub prompts you to specify the WebLogic administrative user name and password.

If you perform multiple operations in the **Databases** tool, the dialog box retains the last user name that was entered, but always requires you to enter the password.

Creating Data Sources

You might need to explicitly create a data source if you use a different application server to create an Operational Reference Store, or if you did not create a data source when you register the Operational Reference Store.

1. Start the **Databases** tool.
2. Acquire a write lock.
3. Right-click the Operational Reference Store in the databases list, and then click **Create Datasource**.
4. If you use WebLogic, enter the WebLogic user name and password when prompted.

When the default password `ChangeMe` is configured in the `cmxserver.properties` file as `cmx.server.database.authentication.method=windowsauthentication`, you will see the following characters in the **Password** field: `*****`. The application will use the Windows authentication method to connect to the data source.

1. The Databases tool creates the data source and displays a progress message.
5. Click **OK**.

Removing Data Sources

If you registered an Operational Reference Store and configured a data source, you can use the Databases tool to manually remove the data source definition from the application server.

After you remove the data source definition, however, the Operational Reference Store still appears in the Hub Console. To completely remove the Operational Reference Store from the Hub Console, unregister it.

1. Start the **Databases** tool.
2. Acquire a write lock.
3. Right-click an Operational Reference Store in the databases list, and then click **Remove Datasource**.
4. If you run WebLogic, enter the WebLogic user name and password when prompted.

The Databases tool removes the data source and displays a progress message.

5. Click **OK**.

CHAPTER 8

Building the Schema

This chapter includes the following topics:

- [Overview, 75](#)
- [Before You Begin, 75](#)
- [About the Schema, 75](#)
- [Starting the Schema Manager, 93](#)
- [Configuring Base Objects, 94](#)
- [Configuring Columns in Tables, 107](#)
- [Configuring Foreign-Key Relationships Between Base Objects, 116](#)
- [Viewing Your Schema, 119](#)

Overview

This chapter explains how to design and build your schema in Informatica MDM Hub.

Before You Begin

Before you begin, you must install Informatica MDM Hub and create the Hub Store, including an Operational Reference Store, by using the instructions in the *Multidomain MDM Installation Guide*.

About the Schema

The schema is the data model that is used in the MDM Hub implementation.

The MDM Hub does not impose or require any particular schema. The schema exists inside the MDM Hub and is independent of the source systems providing data to the MDM Hub.

Note: The process of schema design for your MDM Hub implementation is outside the scope of this document. It is assumed that you have developed a data model, using industry-standard data modeling methodologies, which caters to the requirements of your organization.

The Informatica schema is a flexible, repository-driven model that supports the data structure of any vertical business sector. The Hub Store is the database that underpins the MDM Hub functionality. Every MDM Hub installation has a Hub Store, which includes one MDM Hub Master Database and one or more Operational Reference Store databases. Based on the configuration of your system, you can have multiple Operational Reference Store databases in an installation. For example, you could have a development Operational Reference Store, a testing Operational Reference Store, and a production Operational Reference Store.

Before you begin to implement the schema, you must understand the basic structure of the underlying MDM Hub schema and the components that comprise it.

Note: You must use Hub Console tools to define and manage the consolidated schema. You cannot make changes directly to the database. For example, you must use the Schema Manager to define tables and columns.

Types of Tables in an Operational Reference Store

An Operational Reference Store contains tables that you configure and system support tables.

Configurable Tables

You can use the configurable tables to model business reference data. You must explicitly create and configure these tables.

The following table describes the types of MDM Hub tables that you can configure:

Type of Table	Description
Base object	Used to store data for a central business entity, such as customer, product, or employee, or a lookup table such as country or state. In a base object, you can consolidate data from multiple source systems and use trust settings to determine the most reliable value of each base object cell. You can define one-to-many relationships between base objects. You need to create and configure the base objects.
Landing table	Used to receive batch loads from a source system. You need to create and configure the landing tables.
Staging table	Used to load data into a base object. You define mappings between landing tables and staging tables to specify how data must be cleansed and standardized when it is moved from a landing table to a staging table. You need to create and configure the staging tables.

Infrastructure Tables

The MDM Hub infrastructure tables manage and support the flow of data in the Hub Store. The MDM Hub creates, configures, and maintains the MDM Hub infrastructure tables whenever you configure base objects.

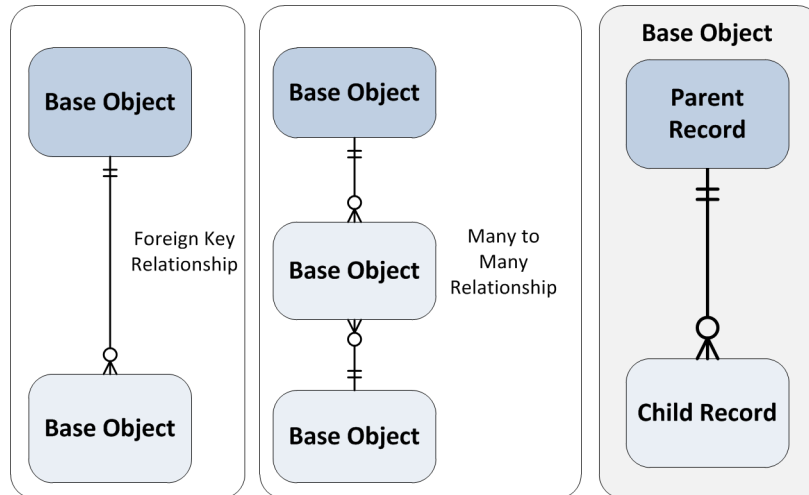
The following table describes the types of MDM Hub infrastructure tables:

Type of Table	Description
Cross-reference table	<p>Used to track the origin of each record in the base object.</p> <p>Cross-reference tables are named according to the following pattern: C_<base object name>_XREF</p> <p>where <base object name> is the root name of the base object. For example, if the base object name is PARTY, the cross-reference table name is C_PARTY_XREF. When you create a base object, the MDM Hub creates a cross-reference table to store information about data that comes from source systems.</p>
History table	<p>Used if history is enabled for a base object.</p> <p>History tables are named according to the following pattern:</p> <ul style="list-style-type: none"> - C_<base object name>_HIST. Base object history table. - C_<base object name>_HXRF. Cross-reference history table. <p>where <base object name> is the root name of the base object. For example, a base object named PARTY has the C_PARTY_HIST and the C_PARTY_HXRF history tables.</p> <p>The MDM Hub creates and maintains different types of history tables. The history tables provide detailed change tracking options, including merge and unmerge history, history of the pre-cleansed data, history of base objects, and the cross-reference history.</p>
Match key table	<p>Contains the match keys that the MDM Hub generates for all base object records. The match key tables are named according to the following pattern: C_<base object name>_STRP</p> <p>where <base object name> is the root name of the base object. For example, if the base object name is PARTY, the match key table name is C_PARTY_STRP.</p>
Match table	<p>Contains the pairs of matched records in the base object resulting from the execution of the match process on the base object. The match tables are named according to the following pattern: C_<base object name>_MTCH</p> <p>where <base object name> is the root name of the base object. For example, if the base object name is PARTY, the match table name is C_PARTY_MTCH.</p>
External match table	<p>Contains the data for external match jobs. You can use the following types of external match tables:</p> <ul style="list-style-type: none"> - EMI table. External match input table. Contains records to match against the records in the base object. - EMO table. External match output table. Contains the output data for External Match jobs. Each row in the EMO table represents a pair of matched records. One of the pairs of records is from the EMI table and one is from the base object. <p>The external match tables are named according to the following pattern:</p> <ul style="list-style-type: none"> - C_<base object name>_EMI - C_<base object name>_EMO <p>where <base object name> is the root name of the base object. For example, if the base object name is PARTY, the match table names are C_PARTY_EMI and C_PARTY_EMO.</p>
Temporary tables	<p>The MDM Hub creates temporary tables that it needs to process data during batch jobs. After the MDM Hub completes processing data, the batch processes remove the temporary tables because they are no longer needed. If the batch process, application server, or the database server fails, temporary tables may not be deleted.</p> <p>Temporary tables have the prefix T\$_. After the batch process completes, if the repository contains temporary tables that the batch process created, you can manually remove them. Some temporary tables do not have the prefix T\$_ prefix. For example, the cleanse temporary table has the suffix _CL instead, and the delta temporary table has the _DLT suffix instead.</p>

Supported Relationships Among Data

The MDM Hub supports one:many and many:many relationships between tables, in addition to hierarchical relationships between records in the same base object. In the MDM Hub, you can define relationships between records in multiple ways.

The following image shows the different relationships between base objects:



The following table describes relationship types between data:

Type of Relationship	Description
Foreign key relationship between base objects	One base object, the child table, contains a foreign key column with values that match the values in the primary key column of another base object, the parent table.
Records within the same base object	Within a base object, records are related to each other hierarchically. You can define many-to-many relationships within the base object.

After you configure the relationships in the Hub Console, you can use the relationships to configure match column rules by defining match paths between records.

Requirements for Defining Schema Objects

This section describes requirements for configuring schema objects.

Make Schema Changes Only in the Hub Console

The MDM Hub maintains schema consistency, provided that all model changes are done using the Hub Console tools, and that no changes are made directly to the database. The MDM Hub provides all the tools required for maintaining the schema.

Think Before You Change the Schema

Schema changes can involve risk to data and should be approached in a managed and controlled manner. You should plan the changes to be made and analyze the impact of the changes before making them. You must also back up the database before making any changes.

You Must Have a Write Lock to Change the Schema

In order to make any changes to the schema, you must have a write lock.

Rules for Database Object Names

Database object names cannot be longer than 24 characters.

Reserved Strings for Database Object Names

The MDM Hub creates metadata objects that use prefixes and suffixes added to the names you use for base objects. In order to avoid confusion and possible data loss, database object names must not use the following strings as either standalone names or parts of column names (prefixes or suffixes).

_BVTB	_OPL	_TMGB	BVTX_	TCMO_	TMF_
_BVTC	_ORAW	_TMIN	BVTXC_	TCRN_	TMMA_
_BVTV	_STRP	_TML0	BVTXV_	TCRO_	TMR_
B01	_STRPT	_TMMA	CLC_	TCSN_	TPBR_
B02	_T	_TMNX	COCs	TCSO_	TRBX_
_C	_TBKF	_TMP0	CSC_	TCVN_	TUCA_
_CL	_TBVB	_TMST	CTL	TCVO_	TUCC_
C_REPOS_	_TBVC	_TNPMA	EXP_	TCXN_	TUCF_
C_REPAR_	_TBVV	_TPMA	GG	TCXO_	TUCR_
_D	_TC0	_TPRL	HMRG	TDCC_	TUCT_
_DLT	_TC1	_TRAW	LNK	TDEL_	TUCX_
_EMI	_TDEL	_TRLG	M	TDUMP_	TUDL_
_EMO	_TEMI	_TRLT	PRL	TFK_	TUGR_
_GOV	_TEM0	_TSD	S_	TFX_	TUHM_
_HIST	_TEMP	_TSI	T_verify_	TGA_	TUID_
_HUID	_TEST	_TSNU	TBDL_	TGB_	TUK_
_HXRF	_TGA	_TSTR	TBOX_	TGB1_	TUPT_
_JOBS	_TGA1	_TUID	TBXR_	TGC_	TUTR_
_L	_TGB	_TVXR	TCBN_	TGC1_	TVXRD_
_LINK	_TGB1	_VCT	TCBO_	TGD_	TXDL_
_LMH	_TGC	_XREF	TCCN_	TGF_	TXPR_
_LMT	_TGC1	BV0_	TCCO_	TGM_	V_
_MTBM	_TMG0	BV1_	TCGN_	TGMD_	-
_MTCH	_TMG1	BV2_	TCGO_	TGV_	-

_MTFL	_TMG2	BV3_	TCHN_	TGV1_	-
_MTFU	_TMG3	BV5_	TCHO_	TLL	-
_MVLE	_TMGA	BVLNK_	TCMN_	TMA_	-

Reserved Column Names

No part of a user-defined column name can be a reserved word or reserved column name.

For example, LAST_UPDATE_DATE is a reserved column name and must not be used for user-defined columns. Also, you must not use LAST_UPDATE_DATE as part of any user-defined column name, such as S_LAST_UPDATE_DATE.

If you use a reserved column name, a warning message is displayed. For example:

```
"The column physical name "XREF_LUD" is a reserved name. Reserved names cannot be used."
```

You cannot use the following column names in part or in whole for base object column names:

AFFECTED_LEVEL_CODE	PERIOD_REFERENCE TIME
AFFECTED_ROWID_COLUMN	PK_SRC_OBJECT
AFFECTED_ROWID_OBJECT	PKEY_SRC_OBJECT
AFFECTED_ROWID_XREF	PKEY_SRC_OBJECT1
AFFECTED_SRC_VALUE	PKEY_SRC_OBJECT2
AFFECTED_TGT_VALUE	PREFERRED_KEY_IND
AUTOLINK_IND	PROMOTE_IND
AUTOMERGE_IND	PUT_UPDATE_MERGE_IND
CASE_INSENSITIVE_INDX_IND	REPOINTED_IND
CHECKIN_ID	ROOT_IND
CONSOLIDATION_IND	ROU_IND
CREATE_DATE	ROWID
CREATOR	ROWID_GROUP
CROSS_PERIOD_ID	ROWID_JOB
CTL_ROWID_OBJECT	ROWID_KEY_CONSTRAINT
DATA_COUNT	ROWID_MATCH_RULE
DATA_ROW	ROWID_OBJECT
DELETED_BY	ROWID_OBJECT1

DELETED_DATE	ROWID_OBJECT2
DELETED_IND	ROWID_OBJECT_MATCHED
DEP_PKEY_SRC_OBJECT	ROWID_OBJECT_NUM
DEP_ROWID_SYSTEM	ROWID_PERIOD
DIRTY_IND	ROWID_SYSTEM
ERROR_DESCRIPTION	ROWID_TASK
FILE_NAME	ROWID_USER
FIRSTV	ROWID_XREF
GENERATED_XREF	ROWID_XREF1
GROUP_ID	ROWID_XREF2
GVI_NO	ROWKEY
HIST_CREATE_DATE	RULE_NO
HIST_UPDATE_DATE	SDSRCFLG
HSI_ACTION	SEQ
HUB_STATE_IND	SOURCE_KEY
INTERACTION_ID	SOURCE_NAME
INVALID_IND	SRC_LUD
LAST_ROWID_SYSTEM	SRC_ROWID
LAST_UPDATE_DATE	SRC_ROWID_OBJECT
LASTV	SRC_ROWID_XREF
LOST_VALUE	SSA_DATA
MATCH_REVERSE_IND	SSA_KEY
MERGE_DATE	STRIP_DATE
MERGE_OPERATION_ID	TGT_ROWID_OBJECT
MERGE_UPDATE_NULL_ALLOW_IND	TIMELINE_ACTION
MERGE_VIA_UNMERGE_IND	TOTAL_BO_IND
MRG_SRC_ROWID_OBJECT	TREE_UNMERGE_IND
MRG_TGT_ROWID_OBJECT	UNLINK_IND

NULL_INDICATOR_BITMAP	UNMERGE_DATE
NUM_CONTR	UNMERGE_IND
OLD_AFFECTED	UNMERGE_OPERATION_ID
ONLINE_IND	UPDATED_BY
ORIG_ROWID_OBJECT	WIN_VALUE
ORIG_ROWID_OBJECT_MATCHED	XREF_LUD
ORIG_TGT_ROWID_OBJECT	-

You cannot use the following column names in part or in whole for landing table column names:

DEP_PKEY_SRC_OBJECT	ROWID_JOB
ERROR_DESCRIPTION	SRC_ROWID
PKEY_SRC_OBJECT	VERSION_SEQ
ROWID	ONLINE_IND

You cannot use the following words as column names:

CLASS	ROWID
-------	-------

Reserved Words in Oracle environments

You cannot use the following words as column names in an Oracle environments:

ABORT	DIGITS	MAXLOGMEMBERS	ROLES
ACCEPT	DISABLE	MAXTRANS	ROLLBACK
ACCESS	DISMOUNT	MAXVALUE	ROW
ADD	DISPOSE	MIN	ROWID
ADMIN	DISTINCT	MINEXTENTS	ROWLABEL
AFTER	DO	MINUS	ROWNUM
ALL	DOUBLE	MINVALUE	ROWS
ALLOCATE	DROP	MLSLABEL	ROWTYPE
ALTER	DUMP	MOD	RUN
ANALYZE	EACH	MODE	SAVEPOINT

AND	ELSE	MODIFY	SCHEMA
ANY	ELSIF	MOUNT	SCN
ARCHIVE	ENABLE	NATURAL	SECTION
ARCHIVELOG	END	NEXT	SEGMENT
ARRAY	ENTRY	NEXTVAL	SELECT
ARRAYLEN	ESCAPE	NOARCHIVELOG	SEPARATE
AS	EVENTS	NOAUDIT	SEQUENCE
ASC	EXCEPT	NOCACHE	SESSION
ASSERT	EXCEPTION	NOCOMPRESS	SET
ASSIGN	EXCEPTIONS	NOCYCLE	SHARE
AT	EXCEPTION _INIT	NOMAXVALUE	SHARED
AUDIT	EXCLUSIVE	NOMINVALUE	SIZE
AUTHORIZATION	EXEC	NONE	SMALLINT
AVG	EXECUTE	NOORDER	SNAPSHOT
BACKUP	EXISTS	NORESETLOGS	SOME
BASE_TABLE	EXIT	NORMAL	SORT
BECOME	EXPLAIN	NOSORT	SPACE
BEFORE	EXTENT	NOT	SQL
BEGIN	EXTERNALLY	NOTFOUND	SQLBUF
BETWEEN	FALSE	NOWAIT	SQLCODE
BINARY_INTEGER	FETCH	NULL	SQLERRM
BLOB	FILE	NUMBER	SQLERROR
BLOCK	FLUSH	NUMBER_BASE	SQLSTATE
BODY	FOR	NUMERIC	START
BOOLEAN	FORCE	OF	STATEMENT
BY	FOREIGN	OFF	STATEMENT_ID
CACHE	FORM	OFFLINE	STATISTICS
CANCEL	FORTRAN	OLD	STDDEV

CASCADE	FOUND	ON	STOP
CASE	FREELIST	ONLINE	STORAGE
CHANGE	FREELISTS	ONLY	SUBTYPE
CHAR	FROM	OPEN	SUCCESSFUL
CHARACTER	FUNCTION	OPTIMAL	SUM
CHAR_BASE	GENERIC	OPTION	SWITCH
CHECK	GO	OR	SYNONYM
CHECKPOINT	GOTO	ORDER	SYSDATE
CLOB	GRANT	OTHERS	SYSTEM
CLOSE	GROUP	OUT	TABAUTH
CLUSTER	GROUPS	OWN	TABLE
CLUSTERS	HAVING	PACKAGE	TABLES
COBOL	IDENTIFIED	PARALLEL	TABLESPACE
COLAUTH	IF	PARTITION	TASK
COLUMN	IMMEDIATE	PCTFREE	TEMPORARY
COLUMNS	IN	PCTINCREASE	TERMINATE
COMMENT	INCLUDING	PCTUSED	THEN
COMMIT	INCREMENT	PLAN	THREAD
COMPILE	INDEX	PLI	TIME
COMPRESS	INDEXES	POSITIVE	TO
CONNECT	INDICATOR	PRAGMA	TRACING
CONSTANT	INITIAL	PRECISION	TRANSACTION
CONSTRAINT	INTRANS	PRIMARY	TRIGGER
CONSTRAINTS	INSERT	PRIOR	TRIGGERS
CONTENTS	INSTANCE	PRIVATE	TRUE
CONTINUE	INT	PRIVILEGES	TRUNCATE
CONTROLFILE	INTEGER	PROCEDURE	UID
COUNT	INTERSECT	PROFILE	UNDER

CRASH	INTO	PUBLIC	UNION
CREATE	IS	QUOTA	UNIQUE
CURRENT	KEY	RAISE	UNLIMITED
CURRVAL	LANGUAGE	RANGE	UNTIL
CURSOR	LAYER	RAW	UPDATE
CYCLE	LEVEL	READ	USE
DATABASE	LIKE	REAL	USER
DATAFILE	LIMITED	RECORD	USING
DATA_BASE	LINK	RECOVER	VALIDATE
DATE	LISTS	REFERENCES	VALUES
DBA	LOCK	REFERENCING	VARCHAR
DEBUGOFF	LOGFILE	RELEASE	VARCHAR 2
DEBUGON	LONG	REMR	VARIANCE
DEC	LOOP	RENAME	VIEW
DECIMAL	MANAGE	RESETLOGS	VIEWS
DECLARE	MANUAL	RESOURCE	WHEN
DEFAULT	MAX	RESTRICTED	WHENEVER
DEFINITION	MAXDATAFILES	RETURN	WHERE
DELAY	MAXEXTENTS	REUSE	WHILE
DELETE	MAXINSTANCES	REVERSE	WITH
DELTA	MAXLOGFILES	REVOKE	WORK
DESC	MAXLOGHISTORY	ROLE	WRITE
-	-	-	XOR

Reserved Words in IBM DB2 environments

You cannot use the following words as column names in IBM DB2 environments:

ABSOLUTE	DESTROY	LOCALTIME	SAVE
ACTION	DESTRUCTOR	LOCALTIMESTAMP	SAVEPOINT

ADA	DETERMINISTIC	LOCATOR	SCHEMA
ADD	DIAGNOSTICS	LOWER	SCOPE
ADMIN	DICTIONARY	MAP	SCROLL
AFTER	DISCONNECT	MATCH	SEARCH
AGGREGATE	DISK	MAX	SECOND
ALIAS	DISTINCT	MDMALIAS	SECTION
ALL	DISTRIBUTED	MDMNODE	SELECT
ALLOCATE	DOMAIN	MIN	SEQUENCE
ALTER	DOUBLE	MINUTE	SESSION
AND	DROP	MODIFIES	SESSION_USER
ANY	DUMMY	MONTH	SET
ARE	DUMP	NAMES	SETS
ARRAY	DYNAMIC	NATIONAL	SETUSER
AS	EACH	NATURAL	SHUTDOWN
ASC	ELSE	NCHAR	SIZE
ASSERTION	END	NCLOB	SMALLINT
AT	END-EXEC	NEXT	SOME
AUTHORIZATION	EQUALS	NO	SPACE
AVG	ERRLVL	NOCHECK	SPECIFIC
BACKUP	ESCAPE	NONCLUSTERED	SPECIFICTYPE
BEFORE	EVERY	NONE	SQL
BEGIN	EXCEPT	NOT	SQLCA
BETWEEN	EXCEPTION	NULL	SQLCODE
BINARY	EXEC	NULLIF	SQLERROR
BIT	EXECUTE	NUMERIC	SQL EXCEPTION
BIT_LENGTH	EXISTS	OBJECT	SQLSTATE
BLOB	EXIT	OCTET_LENGTH	SQLWARNING
BOOLEAN	EXTERNAL	OF	START

BOTH	EXTRACT	OFF	STATEMENT
BREADTH	FALSE	OFFSETS	STATIC
BREAK	FETCH	OLD	STATISTICS
BROWSE	FILE	ON	STRUCTURE
BULK	FILLFACTOR	ONLY	SUBSTRING
BY	FIRST	OPEN	SUM
CALL	FOR	OPENDATASOURCE	SYSTEM_USER
CASCADE	FOREIGN	OPENQUERY	TABLE
CASCADED	FORTRAN	OPENROWSET	TEMPORARY
CASE	FOUND	OPENXML	TERMINATE
CAST	FREE	OPERATION	TEXTSIZE
CATALOG	FREETEXT	OPTION	THAN
CHAR	FREETEXTTABLE	OR	THEN
CHARACTER	FROM	ORDER	TIME
CHARACTER_LENGTH	FULL	ORDINALITY	TIMESTAMP
CHAR_LENGTH	FUNCTION	OUTER	TIMEZONE_HOUR
CHECK	GENERAL	OUTPUT	TIMEZONE_MINUTE
CHECKPOINT	GET	OVER	TO
CLASS	GLOBAL	OVERLAPS	TOP
CLOB	GO	PAD	TRAILING
CLOSE	GOTO	PARAMETER	TRAN
CLUSTERED	GRANT	PARAMETERS	TRANSACTION
COALESCE	GROUP	PARTIAL	TRANSLATE
COLLATE	GROUPING	PASCAL	TRANSLATION
COLLATION	HAVING	PATH	TREAT
COLUMN	HOLDLOCK	PERCENT	TRIGGER
COMMIT	HOST	PLAN	TRIM
COMPLETION	HOURL	POSITION	TRUE

COMPUTE	IDENTITY	POSTFIX	TRUNCATE
CONNECT	IDENTITYCOL	PRECISION	TSEQUEL
CONNECTION	IDENTITY_INSERT	PREFIX	UNDER
CONSTRAINT	IF	PREORDER	UNION
CONSTRAINTS	IGNORE	PREPARE	UNIQUE
CONSTRUCTOR	IMMEDIATE	PRESERVE	UNKNOWN
CONTAINS	IN	PRIMARY	UNNEST
CONTAINSTABLE	INCLUDE	PRINT	UPDATE
CONTINUE	INDEX	PRIOR	UPDATETEXT
CONVERT	INDICATOR	PRIVILEGES	UPPER
CORRESPONDING	INITIALIZE	PROC	USAGE
COUNT	INITIALLY	PROCEDURE	USE
CREATE	INNER	PUBLIC	USER
CROSS	INOUT	RAISERROR	USING
CUBE	INPUT	READ	VALUE
CURRENT	INSENSITIVE	READS	VALUES
CURRENT_DATE	INSERT	READSTEXT	VARCHAR
CURRENT_ROLE	INT	REAL	VARIABLE
CURRENT_TIME	INTEGER	RECONFIGURE	VARYING
CURRENT_TIMESTAMP	INTERSECT	RECURSIVE	VIEW
CURRENT_USER	INTERVAL	REF	WAITFOR
CURSOR	INTO	REFERENCES	WHEN
CYCLE	IS	REFERENCING	WHENEVER
DATA	ISOLATION	RELATIVE	WHERE
DATABASE	ITERATE	REPLICATION	WHILE
DATE	JOIN	RESTORE	WITH
DAY	KEY	RESTRICT	WITHOUT
DBCC	KILL	RESULT	WORK

DEALLOCATE	LANGUAGE	RETURN	WRITE
DEC	LARGE	RETURNS	WRITETEXT
DECIMAL	LAST	REVOKE	YEAR
DECLARE	LATERAL	RIGHT	ZONE
DEFAULT	LEADING	ROLE	
DEFERRABLE	LEFT	ROLLBACK	
DEFERRED	LESS	ROLLUP	
DENY	LEVEL	ROUTINE	
DEPTH	LIKE	ROW	
DEREF	LIMIT	ROWCOUNT	
DESC	LINENO	ROWGUIDCOL	
DESCRIBE	LOAD	ROWS	
DESCRIPTOR	LOCAL	RULE	

Reserved Words in Microsoft SQL Server environments

You cannot use the following words as column names in Microsoft SQL Server environments:

ABSOLUTE	DESC	LEVEL	ROLLUP
ACTION	DESCRIBE	LIKE	ROUTINE
ADA	DESCRIPTOR	LIMIT	ROW
ADD	DESTROY	LINENO	ROWCOUNT
ADMIN	DESTRUCTOR	LOAD	ROWGUIDCOL
AFTER	DETERMINISTIC	LOCAL	ROWS
AGGREGATE	DIAGNOSTICS	LOCALTIME	RULE
ALIAS	DICTIONARY	LOCALTIMESTAMP	SAVE
ALL	DISCONNECT	LOCATOR	SAVEPOINT
ALLOCATE	DISK	LOWER	SCHEMA
ALTER	DISTINCT	MAP	SCOPE
AND	DISTRIBUTED	MATCH	SCROLL

ANY	DOMAIN	MAX	SEARCH
ARE	DOUBLE	MIN	SECOND
ARRAY	DROP	MINUTE	SECTION
AS	DUMMY	MODIFIES	SELECT
ASC	DUMP	MONTH	SEQUENCE
ASSERTION	DYNAMIC	NAMES	SESSION
AT	EACH	NATIONAL	SESSION_USER
AUTHORIZATION	ELSE	NATURAL	SET
AVG	END	NCHAR	SETS
BACKUP	END-EXEC	NCLOB	SETUSER
BEFORE	EQUALS	NEXT	SHUTDOWN
BEGIN	ERRLVL	NO	SIZE
BETWEEN	ESCAPE	NOCHECK	SMALLINT
BINARY	EVERY	NONCLUSTERED	SOME
BIT	EXCEPT	NONE	SPACE
BIT_LENGTH	EXCEPTION	NOT	SPECIFIC
BLOB	EXEC	NULL	SPECIFICTYPE
BOOLEAN	EXECUTE	NULLIF	SQL
BOTH	EXISTS	NUMERIC	SQLCA
BREADTH	EXIT	OBJECT	SQLCODE
BREAK	EXTERNAL	OCTET_LENGTH	SQLERROR
BROWSE	EXTRACT	OF	SQL EXCEPTION
BULK	FALSE	OFF	SQLSTATE
BY	FETCH	OFFSETS	SQLWARNING
CALL	FILE	OLD	START
CASCADE	FILLFACTOR	ON	STATEMENT
CASCADED	FIRST	ONLY	STATIC
CASE	FOR	OPEN	STATISTICS

CAST	FOREIGN	OPENDATASOURCE	STRUCTURE
CATALOG	FORTRAN	OPENQUERY	SUBSTRING
CHAR	FOUND	OPENROWSET	SUM
CHARACTER	FREE	OPENXML	SYSTEM_USER
CHARACTER_LENGTH	FREETEXT	OPERATION	TABLE
CHAR_LENGTH	FREETEXTABLE	OPTION	TEMPORARY
CHECK	FROM	OR	TERMINATE
CHECKPOINT	FULL	ORDER	TEXTSIZE
CLASS	FUNCTION	ORDINALITY	THAN
CLOB	GENERAL	OUT	THEN
CLOSE	GET	OUTER	TIME
CLUSTERED	GLOBAL	OUTPUT	TIMESTAMP
COALESCE	GO	OVER	TIMEZONE_HOUR
COLLATE	GOTO	OVERLAPS	TIMEZONE_MINUTE
COLLATION	GRANT	PAD	TO
COLUMN	GROUP	PARAMETER	TOP
COMMIT	GROUPING	PARAMETERS	TRAILING
COMPLETION	HAVING	PARTIAL	TRAN
COMPUTE	HOLDLOCK	PASCAL	TRANSACTION
CONNECT	HOST	PATH	TRANSLATE
CONNECTION	HOURL	PERCENT	TRANSLATION
CONSTRAINT	IDENTITY	PLAN	TREAT
CONSTRAINTS	IDENTITYCOL	POSITION	TRIGGER
CONSTRUCTOR	IDENTITY_INSERT	POSTFIX	TRIM
CONTAINS	IF	PRECISION	TRUE
CONTAINSTABLE	IGNORE	PREFIX	TRUNCATE
CONTINUE	IMMEDIATE	PREORDER	TSEQUEL
CONVERT	IN	PREPARE	UNDER

CORRESPONDING	INCLUDE	PRESERVE	UNION
COUNT	INDEX	PRIMARY	UNIQUE
CREATE	INDICATOR	PRINT	UNKNOWN
CROSS	INITIALIZE	PRIOR	UNNEST
CUBE	INITIALLY	PRIVILEGES	UPDATE
CURRENT	INNER	PROC	UPDATETEXT
CURRENT_DATE	INOUT	PROCEDURE	UPPER
CURRENT_ROLE	INPUT	PUBLIC	USAGE
CURRENT_TIME	INSENSITIVE	RAISERROR	USE
CURRENT_TIMESTAMP	INSERT	READ	USER
CURRENT_USER	INT	READS	USING
CURSOR	INTEGER	READTEXT	VALUE
CYCLE	INTERSECT	REAL	VALUES
DATA	INTERVAL	RECONFIGURE	VARCHAR
DATABASE	INTO	RECURSIVE	VARIABLE
DATE	IS	REF	VARYING
DAY	ISOLATION	REFERENCES	VIEW
DBCC	ITERATE	REFERENCING	WAITFOR
DEALLOCATE	JOIN	RELATIVE	WHEN
DEC	KEY	REPLICATION	WHENEVER
DECIMAL	KILL	RESTORE	WHERE
DECLARE	LANGUAGE	RESTRICT	WHILE
DEFAULT	LARGE	RESULT	WITH
DEFERRABLE	LAST	RETURN	WITHOUT
DEFERRED	LATERAL	RETURNS	WORK
DELETE	LEADING	REVOKE	WRITE
DENY	LEFT	RIGHT	WRITETEXT

DEPTH	LESS	ROLE	YEAR
DEREF	-	ROLLBACK	ZONE

Reserved Special Characters

The MDM Hub accepts all special characters, but some special characters cause issues in Informatica Data Director. For this reason, do not use the following special characters in MDM Hub attribute display names:

- ! (exclamation point)
- ' (single quote)
- # (number sign)
- & (ampersand)
- < (less than symbol)

However, to use the & and < special characters in your SIF CleansePut and Put requests, you can replace them with the following valid string values:

- Replace & with `&`
- Replace < with `<`

Adding Columns for Technical Reasons

For purely technical reasons, you might want to add columns to a base object. For example, for a segment match, you must add a segment column.

To avoid confusion, Informatica recommends that you differentiate columns added to base objects for technical reasons from columns added for other business reasons. To filter out columns added for technical reasons, prefix the column names with a specific identifier, such as `CSTM_`.

Starting the Schema Manager

You use the Schema Manager in the Hub Console to define the schema, staging tables, and landing tables.

The Schema Manager is also used to define rules for match and merge, validation, and message queues.

To start the Schema Manager:

1. In the Hub Console, expand the Model workbench, and then click **Schema**.
2. The Hub Console displays the Schema Manager.

The Schema Manager is divided into two panes.

Pane	Description
Navigation pane	Shows (in a tree view) the core schema objects: base objects and landing tables. Expanding an object in the tree shows you the property groups available for that object.
Properties pane	Shows the properties for the selected object in the left-hand pane. Clicking any node in the schema tree displays the corresponding properties page (that you can view and edit) in the right-hand pane.

You must use the Schema Manager when defining tables in an Operational Reference Store.

Configuring Base Objects

In the MDM Hub, central business entities such as customers, accounts, or products, are represented in tables called base objects. A base object is a table in the Hub Store that contains collections of data about individual entities.

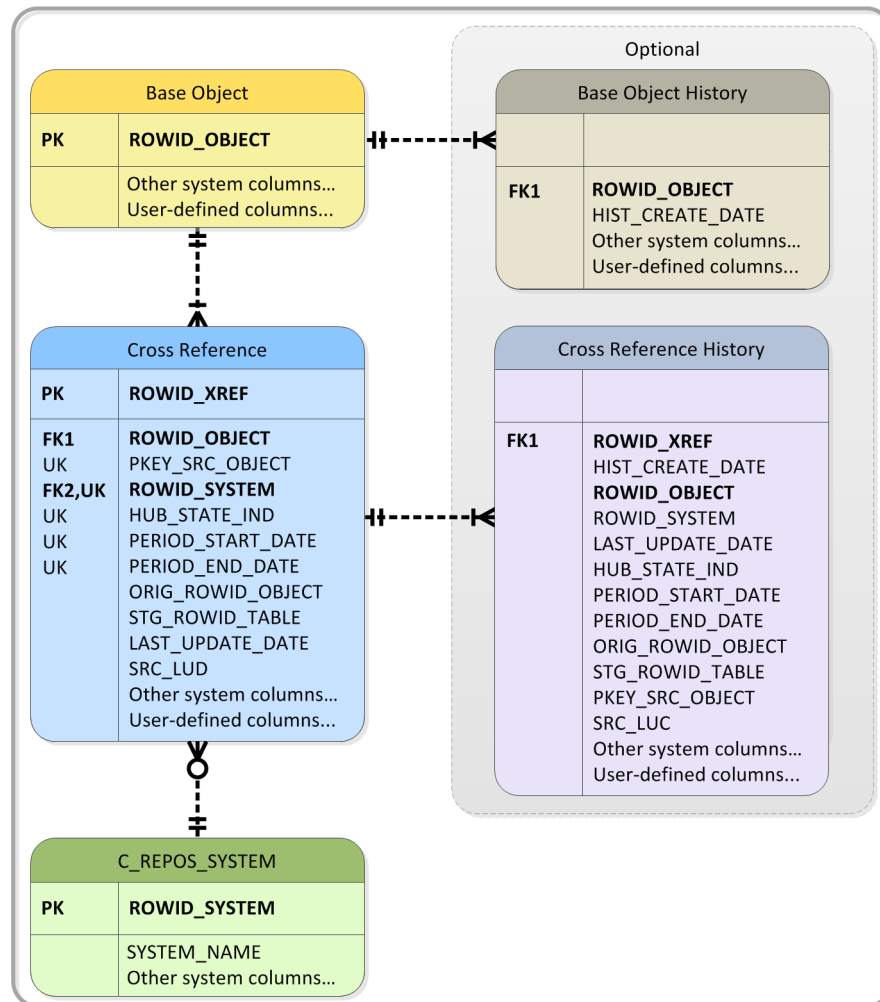
Each entity has a single master record. The master record is the best version of the truth. An individual entity might have records in the base object that contain multiple versions of the truth that need to be consolidated into the master record. Consolidation is the process of merging duplicate records into a single consolidated record that contains the most reliable cell values from all of the source records.

Use the Schema Manager in the Hub Console to define base objects. You cannot configure them directly in the database.

When you make changes to the schema, metadata validation generates a warning and the MDM Hub adds an entry to the C_REPOS_MET_VALID_RESULT table.

Relationships Between Base Objects and Other Tables in the Hub Store

The following figure shows base objects in relation to other tables in the Hub Store.



Process Overview for Defining Base Objects

Use the Schema Manager to define base objects.

- Using the Schema Manager, create a base object table.
The Schema Manager automatically adds system columns.
- Add the user-defined columns that will contain business data.
Note: Column names cannot be longer than 26 characters.
- While configuring column properties, specify the columns that will use trust to determine the most reliable value when different source systems provide different values for the same cell.
- For the base object, create one staging table for each source system. For each staging table, select the base object columns that you need to include.
- Create landing tables that you need to store data from source systems.

6. Map the landing tables to the staging tables.

If any column needs data cleansing, specify the cleanse function in the mapping.

Each staging table must get its data from one landing table (with any intervening cleanse functions), but the same landing table can provide data to more than one staging table. Map the primary key column of the landing table to the PKEY_SRC_OBJECT column in the staging table.

7. Populate each landing table with data using an ETL tool or some other process.

Base Object Columns

Base objects have system columns and user-defined columns. System columns are columns that the Schema Manager creates and maintains. User-defined columns are columns that the user adds.

The following table describes the base object system columns:

Physical Name	MDM Hub Datatype (Size)	Description
ROWID_OBJECT	CHAR (14)	Primary key. Informatica MDM Hub assigns a unique value when a record is inserted into the base object.
CREATOR	VARCHAR (50)	User or process that created the record.
CREATE_DATE	TIMESTAMP	Date on which the record was created.
UPDATED_BY	VARCHAR (50)	User or process responsible for the most recent update on the record.
LAST_UPDATE_DATE	TIMESTAMP	Date of the most recent update to any cell on the record.
CONSOLIDATION_IND	INT	Integer value indicating the consolidation state of this record. Valid values are: <ul style="list-style-type: none">- 1 = unique. The record represents the best version of the truth.- 2 = ready for consolidation.- 3 = ready for match; this record is a match candidate for the currently executing match process.- 4 = available for match. The record is new and needs to undergo the match process.- 9 = on hold. A data steward has put this record on hold.
DELETED_IND	INT	Reserved for future use.
DELETED_BY	VARCHAR (50)	Reserved for future use.
DELETED_DATE	TIMESTAMP	Reserved for future use.
LAST_ROWID_SYSTEM	CHAR (14)	Identifier of the system responsible for the update to the most recently processed cell in the base object record. LAST_ROWID_SYSTEM is a foreign key that references the ROWID_SYSTEM column in the C_REPOS_SYSTEM table.
DIRTY_IND	INT	The DIRTY_IND system column is deprecated. Instead, the MDM Hub uses a system table called the dirty table to determine the records that it needs to tokenize.

Physical Name	MDM Hub Datatype (Size)	Description
INTERACTION_ID	INT	For state-enabled base objects. INTERACTION_ID is an interaction identifier that protects a pending cross-reference record from updates that are not part of the same process as the original cross-reference record.
HUB_STATE_IND	INT	For state-enabled base objects. Integer value indicating the state of this record. Valid values are: <ul style="list-style-type: none"> - 0=Pending - 1=Active - -1=Deleted The default is 1.
CM_DIRTY_IND	INT	Indicates if data changed when you use the zero downtime process to upgrade.

Note: System columns such as `sourceKey` and `ROWID_OBJECT` must not include special characters such as ~ and ' in their value.

Cross-Reference Tables

This section describes cross-reference tables in the Hub Store.

About Cross-Reference Tables

Each base object has one associated cross-reference (XREF) table. The XREF table tracks the lineage and, if timeline is enabled, the records versions for different effective periods.

Informatica MDM Hub automatically creates a cross-reference table when you create a base object. Informatica MDM Hub uses cross-reference tables to translate all source system identifiers into the appropriate ROWID_OBJECT values.

Records in Cross-Reference Tables

Each row in the cross-reference table represents a separate record from a source system. If multiple sources provide data for a single column (for example, the phone number comes from both the CRM and ERP systems), then the cross-reference table contains separate records from each source system. For timeline-enabled base objects, the cross-reference table also contains separate records for each version of a base object record. Each base object record has one or more associated cross-reference records.

The cross-reference record contains:

- An identifier for the source system that provided the record.
- The primary key value of that record in the source system.
- The most recent cell values provided by that system.
- The original ROWID_OBJECT value of the record.
- The original GBID value of the record, if applicable.
- The start date and end date of the period of effectiveness of the record, if applicable.

Load Process and Cross-Reference Tables

The load process populates cross-reference tables. During load inserts, new records are added to the cross-reference table. During load updates, changes are written to the affected cross-reference records.

Data Steward Tools and Cross-Reference Tables

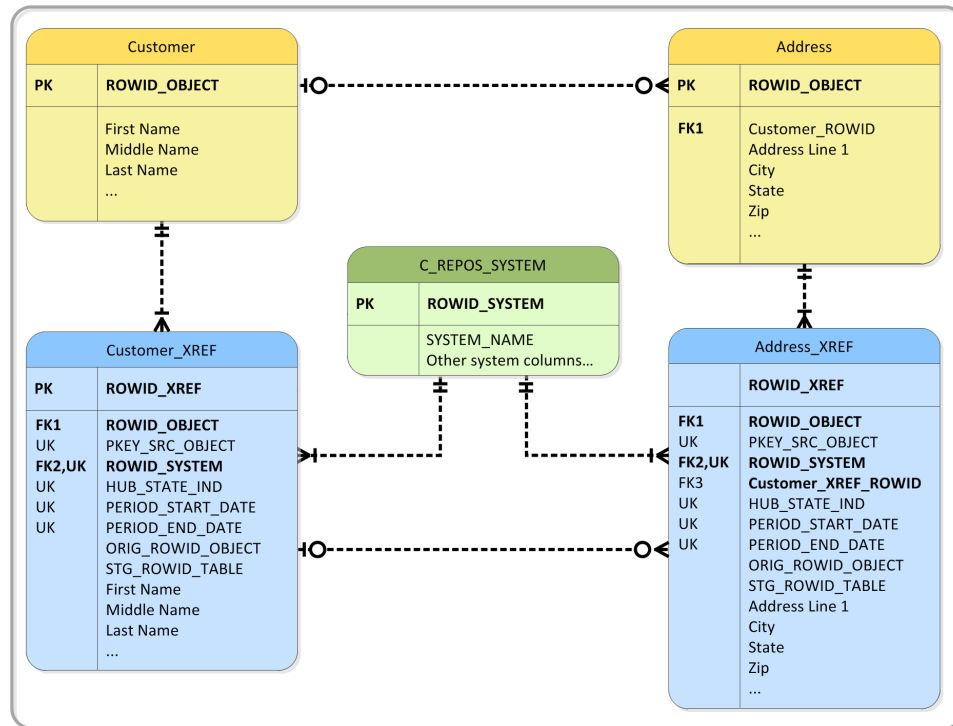
Cross-reference records are visible in the Merge Manager and can be modified using the Data Manager. For more information, see the *Multidomain MDM Data Steward Guide*.

Relationships Between Base Objects and Cross-Reference Tables

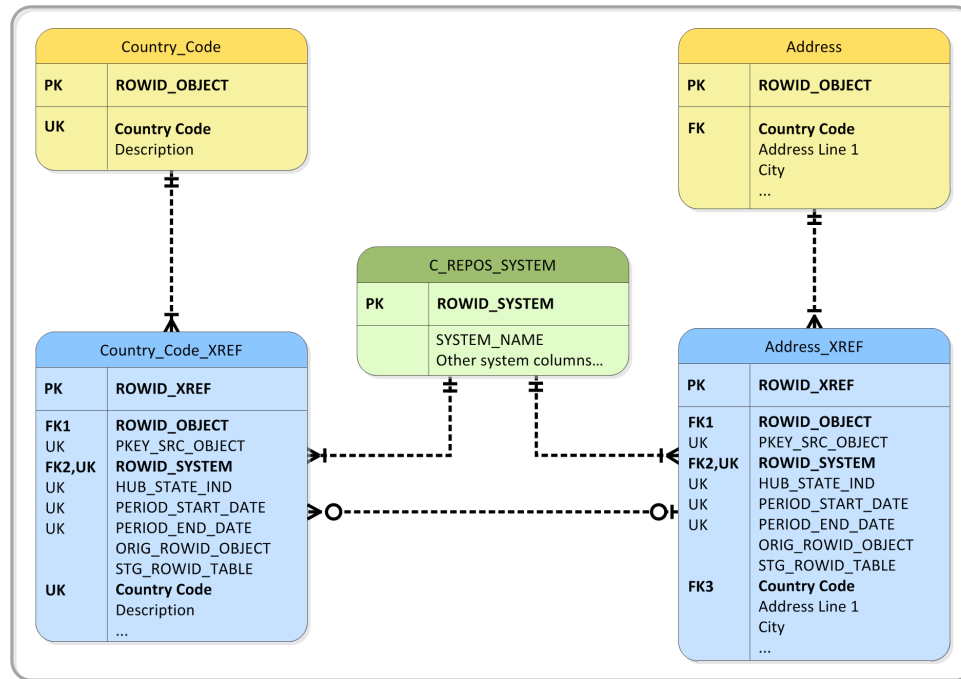
The base object and cross-reference tables can be linked in two ways:

- By ROWID.
- By a unique key.

The following figure shows an example of the relationships between base objects, cross-reference tables, and C_REPOS_SYSTEM based on ROWID.



The following figure shows an example of the relationships between base objects, cross-reference tables, and C_REPOS_SYSTEM based on a unique key.



Cross-reference Table Columns

Cross-reference tables have the following system columns:

Note: Cross-reference tables have a unique key representing the combination of the PKEY_SRC_OBJECT and ROWID_SYSTEM columns.

Physical Name	MDM Hub Datatype (Size)	Description
ROWID_XREF	NUMBER (38)	Primary key that uniquely identifies this record in the cross-reference table.
PKEY_SRC_OBJECT	VARCHAR2 (255)	<p>Primary key value from the source system. You must concatenate keys that contain multiple fields or multiple columns into a single key value. Use the MDM Hub internal cleanse process or an external cleanse process such as an ETL tool or some other data loading utility to concatenate the keys.</p> <p>For admin system cross-reference records that are a result of an edit, the PKEY_SRC_OBJECT is SYS0:<ROWID_OBJECT>, where ROWID_OBJECT is the row ID of the edited record.</p> <p>If the PKEY_SRC_OBJECT is SYS0:<ROWID_OBJECT> and the PUT_UPDATE_MERGE_IND is 1, MDM Hub deletes the cross-reference record during unmerge.</p>
ROWID_SYSTEM	CHAR (14)	Foreign key to C_REPOS_SYSTEM, which is the MDM Hub repository table that stores an MDM Hub identifier and description of each source system that can populate the ORS.
ROWID_OBJECT	CHAR (14)	Foreign key to the base object. The MDM Hub assigns ROWID_OBJECT to the associated base object record.

Physical Name	MDM Hub Datatype (Size)	Description
ORIG_ROWID_OBJECT	CHAR (14)	The original ROWID_OBJECT of the cross-reference record.
<i>GBID_Column_Name_GOV</i>	VARCHAR or INT	The original global identifier value.
STG_ROWID_TABLE	CHAR (14)	Name of the staging table whose lookup configuration was used when the MDM Hub loaded the cross-reference record.
<i>S_Foreign_Key_Column_Name</i>	VARCHAR (255)	A shadow column that contains the value used to look up the foreign key column.
SRC_LUD	TIMESTAMP	Last source update date. The MDM Hub updates SRC_LUD when the source system sends an update to the cross-reference record.
CREATOR	VARCHAR2 (50)	User or process responsible for creating the cross-reference record.
CREATE_DATE	TIMESTAMP	Date on which the cross-reference record was created.
UPDATED_BY	VARCHAR2 (50)	User or process responsible for the most recent update to the cross-reference record.
LAST_UPDATE_DATE	TIMESTAMP	Date of the most recent update to any cell in the cross-reference record. The MDM Hub updates LAST_UPDATE_DATE as applicable during the load process and consolidation process.
DELETED_IND	NUMBER (38)	Reserved for future use.
DELETED_BY	VARCHAR2 (50)	Reserved for future use.
DELETED_DATE	TIMESTAMP	Reserved for future use.
PERIOD_START_DATE	TIMESTAMP	Start date of an effective period of a record. A PERIOD_START_DATE value is required for cross-reference records of timeline-enabled base objects.
PERIOD_END_DATE	TIMESTAMP	End date of an effective period of a record. A PERIOD_END_DATE value is required for cross-reference records of timeline-enabled base objects.
PUT_UPDATE_MERGE_IND	NUMBER (38)	When the value is 1, indicates that a Put API call updated that particular record by specifying the ROWID_OBJECT value in the Put API request.
INTERACTION_ID	NUMBER (38)	For state-enabled base objects. Interaction identifier that is used to protect a pending cross-reference record from updates that are not part of the same process as the original cross-reference record.

Physical Name	MDM Hub Datatype (Size)	Description
HUB_STATE_IND	NUMBER (38)	For state-enabled base objects. Integer value indicating the state of the record. Valid values are: <ul style="list-style-type: none"> - 0 = Pending - 1 = Active - -1 = Deleted The default value is 1.
PROMOTE_IND	NUMBER (38)	For state-enabled base objects. Integer value indicating the promotion status. The promote process uses PROMOTE_IND to determine whether to promote the record to an ACTIVE state. Valid values are: <ul style="list-style-type: none"> - 0 = Do not promote the record. - 1 = Promote this record to ACTIVE. The MDM Hub does not change PROMOTE_IND to 0 if the promote process does not successfully promote the record.

History Tables

History tables reside in the Hub Store.

If you enable history for a base object, Informatica MDM Hub maintains history tables for the base objects and cross-reference tables. Informatica MDM Hub provides detailed change-tracking options, such as merge and unmerge history, history of the pre-cleansed data, history of the base object, and the cross-reference history in history tables.

When you run an automerge job, the Informatica MDM Hub creates a record in the history table for each merge operation. Similarly, each time the Informatica MDM Hub updates a foreign key in a child base object, a record is inserted in the corresponding history table.

Base Object History Tables

A history-enabled base object has a single history table (named *C_baseObjectName_HIST*) that contains historical information about data changes in the base object. Whenever a record is added or updated in the base object, a new record is inserted into the base object history table to capture the event.

Cross-Reference History Tables

A history-enabled base object has a single cross-reference history table (named *C_baseObjectName_HXRF*) that contains historical information about data changes in the cross-reference table. Whenever a record changes in the cross-reference table, a new record is inserted into the cross-reference history table to capture the event.

Base Object Properties

This section describes the basic and advanced properties for base objects.

Basic Base Object Properties

This section describes the basic base object properties.

Item Type

The type of table that you must add. Select **Base Object**.

Display Name

The name of the base object as it must display in the Hub Console. Enter a descriptive name. Ensure that the display name is less than 67 characters.

Physical Name

The actual name of the table in the database. Informatica MDM Hub suggests a physical name for the table based on the display name that you enter. Ensure that you do not use any reserved name suffixes.

Data Tablespace

The name of the data tablespace. Read-only.

Index Tablespace

The name of the index tablespace. Read-only.

Description

A brief description of the base object.

Enable History

Specifies whether history is enabled for the base object. If enabled, Informatica MDM Hub keeps a log of records that are inserted, updated, or deleted for this base object. You can use the information in history tables for audit purposes.

Timeline

Specifies whether timeline is enabled for the base object. The default is **No Timeline**. Select **Dynamic Timeline** to enable timeline. If timeline is enabled, the MDM Hub manages versions of base object records, including entities and relationships.

Allow non-contiguous effective periods

Uses non-contiguous effective periods for records. Enable to specify non-contiguous effective periods for records. Disable to use contiguous effective periods for records. Default is disabled.

Advanced Base Object Properties

This section describes the advanced base object properties.

Complete Tokenize Ratio

When the percentage of the records that have changed is higher than the complete tokenize ratio, a complete re-tokenization is performed. If the number of records to be tokenized does not exceed this threshold, then Informatica MDM Hub deletes the records requiring re-tokenization from the match key table, calculates the tokens for those records, and then reinserts them into the match key table. The default value is 60.

Note: Deleting can be a slow process. However, if your Process Server is fast and the network connection between the Process Server and the database server is also fast, then you may test with a much lower tokenization threshold (such as 10%). This will enable you to determine whether there are any gains in performance.

Allow constraints to be disabled

During the initial load/updates—or if there is no real-time, concurrent access—you can disable the referential integrity constraints on the base object to improve performance. The default value is 1, signifying that constraints are disabled.

Duplicate Match Threshold

This parameter is used only with the Match for Duplicate Data job for initial data loads. The default value is 0. To enable this functionality, this value must be set to 2 or above. For more information, see the *Multidomain MDM Data Steward Guide*.

Load Batch Size

The load process inserts and updates batches of records in the base object. The load batch size specifies the number of records to load for each batch cycle (default is 1000000).

Max Elapsed Match Minutes

This specifies the timeout (in minutes) when running a match rule. If the time limit is reached, then the match process (whenever a match rule is run, either manually or through a batch job) will exit. If a match process is run as part of a batch job, the system should move onto the next match. It will stop if this is a single match process. The default value is 20. Increase this value only if the match rule and data are very complex. Generally, rules are able to complete within 20 minutes (the default).

Parallel Degree

This specifies the degree of parallelism set on the base object table and its related tables. It does not take effect for all batch processes, but can have a beneficial effect on performance when it is used. However, its use is constrained by the number of CPUs on the database server machine, as well as the amount of memory available. The default value is 1.

Note: You cannot configure the degree of parallelism in a Microsoft SQL Server environment.

Requeue On Parent Merge

The default value is NONE. You can set one of the following values for the Requeue on Parent Merge property:

- **NONE.**
If the value is set to NONE for a child base object, in the event of a merging parent, the consolidation indicator of the child record does not change.
- **UNCONSOLIDATED ONLY.**
If the value is set to UNCONSOLIDATED ONLY for a child base object, in the event of a merging parent, the consolidation indicator of child records change to 4, except for records with consolidation indicator set to 1. To requeue child records with consolidation indicator set to 1, the Requeue on Parent Merge setting must be manually set to 2.
- **ALL.**
If the value is set to ALL for a child base object, in the event of a merging parent, the consolidation indicator of all child records, including those with a consolidation indicator of 1, changes to 4 so they can be matched again.

Generate Match Tokens on Load

Enables the token generation procedure to run after a load procedure completes. You can enable generating match tokens after a load process for base objects that do not have dependent children. You must load dependent children before you can tokenize the parent. Disable generating match tokens after a load process if you have a limited window within which you must perform the load process. Default is disabled.

Match Flag Audit Table

Specifies whether a match flag audit table is created.

- If enabled, then an audit table (*BusinessObjectName_FMHA*) is created and populated with the user ID of the user who, in Merge Manager, queued a manual match record for automerging.
- If disabled, then the Updated_By column is set to the userID of the person who executed the Automerge batch job.

API “lock wait” interval (seconds)

Specifies the maximum number of seconds that a SIF request will wait to obtain a row-level lock. Applies only if row-level locking is enabled for an ORS.

Batch “lock wait” interval (seconds)

Specifies the maximum number of seconds that a batch job will wait to obtain a row-level lock. Applies only if row-level locking is enabled for an ORS.

Enable State Management

Specifies whether Informatica MDM Hub manages the system state for records in the base object. By default, state management is disabled. Select (check) this check box to enable state management for this base object in support of approval workflows. If enabled, this base object is referred to in this document as a *state-enabled base object*.

Note: If the base object has custom query, when you disable state management on the base object, you will always get a warning pop-up window, even when the `hub_state_ind` is not included in the custom query.

Enable History of Cross-Reference Promotion

For state-enabled base objects, specifies whether Informatica MDM Hub maintains the promotion history for cross-reference records that undergo a state transition from PENDING (0) to ACTIVE (1). By default, this option is disabled.

Base Object Style

The base object style is merge style. You can use the merge style base object with the match and merge capabilities of the MDM Hub. It is selected by default.

Lookup Indicator

Specifies how values are retrieved in the MDM Hub Informatica Data Director.

- If enabled, then the Informatica Data Director displays drop-down lists of lookup values.
- If disabled, then the Informatica Data Director displays a search wizard that prompts users to select a value from a data table.

Creating Base Objects

Create base objects in your schema.

1. Start the Schema Manager.
2. Acquire a write lock.
3. Right-click in the left pane of the Schema Manager and choose **Add Item** from the popup menu.
The Schema Manager displays the Add Table dialog box.
4. Specify the basic base object properties.
5. Click **OK**.

The Schema Manager creates the new base table in the Operational Reference Store (ORS), along with any support tables, and then adds the new base object table to the schema tree.

Editing Base Object Properties

You can edit the properties of an existing base object. After you edit base object properties, validate the Operational Reference Store.

Note: If you do not validate the Operational Reference Store, and an application uses business entity services to retrieve data, the service call might fail with errors.

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, select the base object that you want to modify.
The Schema Manager displays the Basic tab of the Base Object Properties page.
4. Edit basic base object properties using the following instructions:
 - Click the **Edit** button and specify the new value in the Display Name field.
 - Click the **Edit** button and specify the new value in the Description field.
 - Select the **Enable History** check box to have Informatica MDM Hub keep a log of records that are inserted, updated, or deleted. The history table is used for auditing.
 - Select one of the following choices from the Timeline drop-down list:
 - **No Timeline** to disable timeline for the base object
 - **Dynamic Timeline** to enable timeline for the base object
 - Select the **Allow non-contiguous effective periods** check box to allow non-contiguous effective periods for base object records, including entities and relationship. This property can be set only for timeline-enabled base objects.
5. To modify other base object properties, click the **Advanced** tab.
6. Specify the advanced properties for this base object.
7. In the left pane, click Match/Merge Setup beneath the base object's name.
8. Specify the match / merge object properties. At a minimum, consider configuring the following properties:
 - maximum number of matches for manual consolidation
 - number of rows per match job batch cycleTo edit a property, click the **Edit** button and enter a new value.
9. Click the **Save** button to save your changes.

When you make changes to the schema, metadata validation generates a warning and the MDM Hub adds an entry to the C_REPOS_MET_VALID_RESULT table.

Custom Indexes for Base Objects

You can create custom indexes to improve batch job and SIF API performance.

Informatica MDM Hub creates system indexes for primary keys and unique columns. Optionally, you can create custom indexes for the other columns to increase performance. The values in a custom index might not be unique.

For example, an external application might call the SearchQuery SIF API request to search a base object by last name. If you create a custom index on the last name column, search performance improves.

Batch jobs drop and re-create system indexes and registered custom indexes to increase performance. Use the RegisterCustomIndex SIF API to register the custom indexes.

You cannot create a custom index if a global business identifier index exists for the base object. Base objects have a global business identifier index when you enable any of the base object columns to be a global business identifier.

Because of Microsoft SQL Server limitations, you cannot create a custom index with more than 16 columns with MDM Hub running on Microsoft SQL Server.

Navigating to the Custom Index Setup Node

To navigate to the custom index setup node:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Base Objects** node, and then expand the node of the base object that you want to work with.
4. Click the **Custom Index Setup** node.

The Schema Manager displays the Custom Index Setup page.

Creating a Custom Index

To create a custom index for a base object, configure the custom index setup for the base object. You can create more than one custom index for a base object.

1. Under the **Model** workbench, select **Schema**.
2. In the Schema Manager, navigate to the Custom Index Setup node for the base object that you want to work with.
3. Click the **Add** button.
The Schema Manager creates a custom index named `NI_C_<base_object_name>_inc`, where *inc* is an incremental number.
4. From the list of base object columns in the **Columns in Index** pane, select the columns that you want in the custom index. Click the **Save** button.

Editing a Custom Index

To change a custom index, you must delete the existing custom index and add a new custom index with the columns that you want.

Deleting a Custom Index

To delete a custom index:

1. In the Schema Manager, navigate to the Custom Index Setup node for the base object that you want to work with.
2. In the Indexes list, select the custom index that you want to delete.
3. Click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
4. Click **Yes**.

Creating a Custom Index Outside the MDM Hub

To create a custom index outside the MDM Hub, you can use the database utility for your database.

You can create an index outside the MDM Hub to support a specialized operation. For example, you can create a function-based index such as `Upper(Last_Name)` in the index expression. If you create a custom index outside the MDM Hub, you cannot register the index. The MDM Hub cannot maintain unregistered indexes for you. If you do not maintain the index, batch job performance could decrease.

Viewing the Impact Analysis of a Base Object

The Schema Manager allows you to view all of the tables, packages, and queries associated with a base object.

You would typically do this before deleting a base object to ensure that you do not delete other associated objects by mistake.

Note: If a column is deleted from a base query, then the dependent queries and packages are completely removed.

To view the impact analysis for a base object:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, select the base object that you want to view.
4. Right-click the mouse and choose **Impact Analysis**.

The Schema Manager displays the Table Impact Analysis dialog box.

5. Click **Close**.

Deleting Base Objects

To delete a base object:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, select the base object that you want to delete.
4. Right-click the mouse and choose **Remove**.

The Schema Manager prompts you to confirm deletion.

5. Choose **Yes**.

The Schema Manager asks you whether you want to view the impact analysis before deleting the base object.

6. Choose **No** if you want to delete the base object without viewing the impact analysis.

The Schema Manager removes the deleted base object from the schema tree.

Configuring Columns in Tables

After you have created a table (base object or landing table), you use the Schema Manager to define the columns for that table. You *must* use the Schema Manager to define columns in tables—you cannot configure them directly in the database.

Note: In the Schema Manager, you can also view the columns for cross-reference tables and history tables, but you cannot edit them.

About Columns

This section provides general information about table columns.

Types of Columns in ORS Tables

Tables in the Hub Store contain two types of columns:

Column	Description
System columns	A column that Informatica MDM Hub automatically creates and maintains. System columns contain metadata.
User-defined columns	Any column in a table that is not a system column. User-defined columns are added in the Schema Manager and usually contain business data.

Note: The system columns contain Informatica MDM Hub metadata. Do not alter Informatica MDM Hub metadata in any way. Doing so will cause Informatica MDM Hub to behave in unpredictable ways and you can lose data.

Data Types for Columns

The MDM Hub has a set of data types for columns. The MDM Hub data types map directly to Oracle, IBM DB2, and Microsoft SQL Server data types. For more information about the database data types, see the product documentation for your database.

Note: To change the data types of the columns, use the Hub Console. Do not change data types in the database. If you change data types in the database, the metadata validation process might encounter issues.

The following table shows how the MDM Hub data types map to the database data types:

MDM Hub	Oracle	IBM DB2	Microsoft SQL Server
CHAR	CHAR	CHARACTER	NCHAR ¹
VARCHAR	VARCHAR2	VARCHAR	NVARCHAR
NVARCHAR2	NVARCHAR2	VARGRAPHIC	NVARCHAR
NCHAR	NCHAR	GRAPHIC	NCHAR ¹
DATE	DATE	TIMESTAMP	DATETIME2
TIMESTAMP ²	TIMESTAMP	TIMESTAMP	DATETIME2

MDM Hub	Oracle	IBM DB2	Microsoft SQL Server
NUMBER	NUMBER	DECIMAL	NUMERIC
INT	INTEGER	DECIMAL (31,0)	BIGINT

1. For Global Identifier (GBID) columns, use VARCHAR instead of CHAR or NCHAR to avoid issues that might be caused by row width limitations in Microsoft SQL Server.

2. For system columns with dates, the data type is TIMESTAMP. For user-defined columns with dates, the TIMESTAMP data type is not available. Use DATE instead.

Column Properties

You can configure the properties of Informatica MDM Hub columns.

Note: In the MDM Hub, an empty string is the equivalent of a null value regardless of the database type that contributes the empty string.

Informatica MDM Hub columns have the following properties:

Display Name

The name of the column that the Hub Console displays.

Physical Name

The name of the column in the base object. The Hub Console suggests a physical name for the column based on the display name.

The physical column name cannot be a reserved column name and it cannot include the dollar sign '\$' character.

Nullable

If enabled, the column can contain null values. If you do not enable the Nullable property, you must specify a default value.

When the Nullable property is disabled for columns, if a record is updated in Data Director or during a SIF PUT operation, a cross-reference record is created with values only for the updated fields. All the other cross-reference record fields, including the fields that must not be null, have null values. During the best version of the truth (BVT) calculation for the record, null values can be the winner for fields that must not be null, and an error occurs.

To ensure that the BVT calculation takes into account the fields that must not be null, set the `cmx.server.put.autopopulate.missing.user.columns.bo.list` property in the `cmxserver.properties` file. Set the property value to a comma-separated list of the names of base objects that have columns with the Nullable property disabled. When the property is set, the MDM Hub updates the null values in the cross-reference record with values from the associated base object. This ensures that during BVT calculation, null values are not the winner for fields that must not be null.

Data Type

The data type of the column. For character data types, you can specify the length property. For certain numeric data types, you can specify the precision property and scale property.

Length

For character data types, specify the number of characters allowed.

Precision

For numeric data types, specify the number of digits allowed in the number, including all decimal places.

Scale

For numeric data types, specify the number of digits allowed after the decimal point.

Has Default

If enabled, you can specify a default value.

Default

The default value for the column. Informatica MDM Hub uses the default value if the column is not nullable and no value is provided for the column.

If you enable the Unique property or if you disable the Nullable property, you must specify a default value for the column.

Trust

If enabled, Informatica MDM Hub uses the source system value with the highest trust score.

If not enabled, Informatica MDM Hub uses the most recently updated value.

Unique

If enabled, Informatica MDM Hub enforces unique constraints on the column. If unique constraint is enabled, ensure that duplicate values do not exist in the unique column. Most organizations use the primary key from the source system for the lookup value. Informatica MDM Hub rejects any record with a duplicate value in the column when you enable the Unique property. You must configure a default value for the column if you enable the Unique property.

If you enable the Unique property for consolidated base objects, an insert into the base object could fail because Informatica MDM Hub might load the same key from different systems. Ensure you have unique keys for this column across all source systems.

Validate

If enabled, Informatica MDM Hub applies validation rules during the load process to downgrade trust scores for cell values that are not valid. You must configure validation rules if you enable the Validate property.

Apply Null Values

Set this property to whatever default you want to use when a null value is the most trustworthy value for this column. A process uses this property when the process cannot determine the setting of the **Allow Null Update** property for the column in a staging table.

- True. When enabled, if the most trustworthy value for this column is a null value, a process can write the null value to the base object record.
- False. Default. When disabled, a process cannot write a null value to the base object record as long as another source system contributes a non-null value for this column.

The **Apply Null Values** property works the same way as the **Allow Null Update** property. For more information about the **Allow Null Update** property, see [“Properties for Columns in Staging Tables” on page 358](#).

GBID

If enabled, Informatica MDM Hub uses this column as the Global Business Identifier (GBID) for the base object. You can configure any number of GBID columns for API access and for batch loads.

On Oracle, if GBID is enabled, you must configure the column as an INT datatype or as a CHAR, NCHAR, VARCHAR, and NVARCHAR2 datatype with a length of 255.

On IBM DB2, if GBID is enabled, you must configure the column as an INT datatype or as a VARCHAR, and NVARCHAR datatype with a length of 255.

On Microsoft SQL Server, if GBID is enabled, you must configure the column as an INT datatype or as a VARCHAR, and NVARCHAR2 datatype with a length of 255.

If you enable GBID for any base object column, Informatica MDM Hub creates an index for the base object. Informatica MDM Hub does not allow you to create a custom index for the base object because the base object has an index.

Putable

If enabled for system columns, batch jobs and SIF API requests can insert or update the data in the system column. You cannot enable the Putable property for the following system columns:

- ROWID_OBJECT
- CONSOLIDATION_IND
- HUB_STATE_IND
- LAST_ROWID_SYSTEM
- CM_DIRTY_IND

If a Put or Cleanse Put API request tries to update a system column but you do not enable the Putable property, the API request fails.

User-defined columns and the INTERACTION_ID and LAST_UPDATE_DATE system columns are always putable.

Global Identifier (GBID) Columns

A Global Business Identifier (GBID) column contains common identifiers (key values) that allow you to uniquely and globally identify a record based on your business needs. Examples include:

- Identifiers defined by applications external to the MDM Hub, such as ERP (SAP or Siebel customer numbers) or CRM systems.
- Identifiers defined by external organizations, such as industry-specific codes (AMA numbers, DEA numbers, and so on), or government-issued identifiers (social security number, tax ID number, driver's license number, and so on).

Note: To be configured as a GBID column, the column must be an integer, CHAR, VARCHAR, NCHAR, or NVARCHAR column type. A non-integer column must be exactly 255 characters in length.

In the Schema Manager, you can define multiple GBID columns in a base object. For example, an employee table might have columns for social security number and driver's license number, or a vendor table might have a tax ID number.

A Master Identifier (MID) is a common identifier that is generated by a system of reference or system of record that is used by others (for example, CIF, legacy hubs, the MDM Hub, counterparty hub, and so on). In the MDM Hub, the MID is the ROWID_OBJECT, which uniquely identifies individual records from various source systems.

GBIDs do not replace the ROWID_OBJECT. GBIDs provide additional ways to help you integrate your MDM Hub implementation with external systems, allowing you to query and access data through unique identifiers of your own choosing (using SIF requests, as described in the *Multidomain MDM Services Integration Framework Guide*). In addition, by configuring GBID columns using already-defined identifiers, you can avoid the need to custom-define identifiers.

GBIDs help with the *traceability* of your data. Traceability is keeping track of the data so that you can determine its *lineage*—which systems, and which records from those systems, contributed to consolidated records. When you define GBID columns in a base object, the Schema Manager creates two columns in the

cross-reference table, <GBID_column_name> and <GBID_column_name>_GOV, to track the current and original GBID values.

For example, suppose two of your customers (both of which had different tax ID numbers) merged into a single company, and one tax ID number survived while the other one became obsolete. If you defined the taxID number column as a GBID, the MDM Hub could help you track both the current and historical tax ID numbers so that you could access data (via SIF requests) using the historical value.

Note: The MDM Hub does not perform any data verification or error detection on GBID columns. If the source system has duplicate GBID values, then those duplicate values will be passed into the MDM Hub.

Columns in Staging Tables

The columns for staging tables cannot be defined using the column editor. Staging table columns are a special case, as they are based on some or all columns in the staging table's target object. You use the Add/Edit Staging Table window to select the columns on the target table that can be populated by the staging table. Informatica MDM Hub then creates each staging table column with the same data types as the corresponding column in the target table.

Maximum Number of Columns for Base Objects

A base object cannot have more than 200 user-defined columns if it will have match rules that are configured for automatic consolidation.

Navigating to the Column Editor

To configure columns for base objects and landing tables:





1. Start the Schema Manager.
2. Acquire a write lock.
3. Expand the schema tree for the object to which you want to add columns.
4. Select **Columns**.






The Schema Manager displays column definitions in the Properties pane.

The Column Editor displays a "lock" icon next to system columns if you select the **Show system columns** check box.

Command Buttons in the Column Editor

The Properties pane in the Column Editor contains the following command buttons:

Button	Name	Description
	Add Column	Adds new columns.
	Remove Column	Removes existing columns.
	Edit Column Description	Allows editing the description of the selected column.
	Move Up	Moves the selected column up in the display order.

Button	Name	Description
	Move Down	Moves the selected column down in the display order.
	Import Schema	Allows adding new columns by importing column definitions from another table.
	Expand table columns view	Expands the table columns view.
	Restore table columns view	Restores the table columns view.
	Save	Saves changes to the column definitions.

Showing or Hiding System Columns

You can toggle the Show System Columns check box to show or hide system columns.

Expanding the Table Columns View

You can expand the properties pane to display all the column properties in a single pane. By default, the Schema Manager displays column definitions in a contracted view.

To show the expanded table columns view:

- Click the **Expand table columns view** button.
The Schema Manager displays the expanded table columns view.

To show the default table columns view:

- Click the **Restore table columns view** button.
The Schema Manager displays the default table columns view.

Adding Columns

To add a column:

1. Navigate to the column editor for the table that you want to configure.
2. Acquire a write lock.
3. Click the **Add** button.
The Schema Manager displays an empty row.
4. For each column, specify its properties.
5. Click the **Save** button to save the columns you have added.

Importing Column Definitions From Another Table

To import some of the column definitions from another table:

1. Navigate to the column editor for the table that you want to configure.
2. Acquire a write lock.
3. Click the **Import Schema** button.

The Import Schema dialog is displayed.

4. Specify the connection properties for the schema that you want to import.

If you need more information about the connection information to specify here, contact your database administrator.

5. Click **Next**.

Note: The database you enter does not need to be the same as the Informatica ORS that you're currently working in, nor does it need to be a Informatica ORS.

The only restriction is that you cannot import from a relational database that is a different type from the one in which you are currently working. For example, if your database is an Oracle database, then you can import columns only from another Oracle database.

The Schema Manager displays a list of the tables that are available for import.

6. Select that table that you want to import.
7. Click **Next**.

The Schema Manager displays a list of columns for the selected table.

8. Select the column(s) you want to import.
9. Click **Finish**.
10. Click the **Save** button to save the column(s) that you have added.

Editing Column Properties

Once columns are added and saved, you can change certain column properties.

Note: After you define a table and save changes, you cannot reduce the length of a CHAR, VARCHAR, NCHAR, or NVARCHAR2 field and change the scale or precision of a NUMBER field.

If you make schema changes after the tables are populated with data, manage such changes to columns in a planned and controlled fashion, and ensure that the appropriate database backups are done before making changes.

1. Navigate to the column editor for the table that you want to configure.
2. Acquire a write lock.
3. You can change the following properties for each column:

Property	Description
Display Name	Name for the column that will display in the Hub Console.
Length	You can only increase the length of a CHAR, VARCHAR, NCHAR, or NVARCHAR2 field.
Default	Used if no value is provided for the column but the column cannot be NULL. Note: Enabling Default does not have any affect on records that were loaded before Default was enabled. Existing NULL values remain NULL. Reload the data, preferably from the staging table, to ensure the values in the column are not NULL.

Property	Description
Trust	<p>Specifies whether a column is a trusted column.</p> <p>You need to synchronize metadata if you enable trust. If you enable trust for a column on a table that already contains data, you are warned that your trust settings have changed and that you need to run the trust Synchronization batch job in the Batch Viewer tool before doing any further loads to the table.</p> <p>Informatica MDM Hub automatically ensures that the Synchronization job is available in the Batch Viewer tool.</p> <p>You must execute the synchronization process before you run any more Load jobs. Otherwise, the trusted values used to populate the column will be incorrect.</p> <p>If you disable trust, columns from some of the underlying metadata tables are removed and results in loss of data.</p> <p>If you inadvertently disable trust and save that change, you should correct your error by enabling trust again and immediately running the Synchronization job to recreate the metadata.</p>
Unique	<p>Specifies whether a column has unique values. Enable the Unique property if the column should contain unique values. If you enable the Unique property for a column, ensure that duplicate values do not exist in the column. You cannot enable the Unique property for a column that contains duplicate values. Do not use the Unique property, particularly on base objects that might be merged.</p>
Validate	<p>Specifies whether a column needs to be validated. If you disable validation, metadata is lost for the associated column.</p>
Putable	<p>Enable the Putable property for system columns into which you want to put data (insert or update) using SIF requests and by using batch jobs run through the Hub Console. Applies to any system column except ROWID_OBJECT, CONSOLIDATION_IND, HUB_STATE_IND, LAST_ROWID_SYSTEM, and CM_DIRTY_IND.</p>

- Click the **Save** button to save your changes.

Changing the Column Display Order

You can move columns up or down in the display order. Changing the display order does not affect the physical table in the database.

To change the column display order:

- Navigate to the column editor for the table that you want to configure.
- Acquire a write lock.
- Select the column that you want to move.
- Do one of the following:
 - Click the **Move Up** button to move the selected column up in the display order.
 - Click the **Move Down** button to move the selected column down in the display order.
- Click the **Save** button to save your changes.

Deleting Columns

Deleting columns should be approached with extreme caution. Any data loaded into a column is lost when the column is removed. Deleting a column can be a slow process due to the number of underlying tables that could be affected.

To delete a column from base objects and landing tables:

1. In the Hub Console, start the Schema tool.
2. Navigate to the column editor for the table that you want to configure.
3. Acquire a write lock.
4. Select the column that you want to delete from the list of column definitions in the Properties pane.
5. Click the **Remove Column** button.
The column is analyzed and the Impact Analyzer is displayed.
6. Click **OK** in the Impact Analyzer to delete the column.
The Schema tool removes the column.
7. Click the **Save** button to save your changes.

Configuring Foreign-Key Relationships Between Base Objects

This section describes how to configure foreign key relationships between base objects in your Informatica MDM Hub implementation.

For a general overview of foreign key relationships, see [“Process Overview for Defining Foreign-Key Relationships” on page 116](#).

About Foreign Key Relationships

In Informatica MDM Hub, a foreign key relationship establishes an association between two base objects through matching columns. In a foreign-key relationship, one base object, the child, contains a foreign key column, which contains values that match values in the primary key column of another base object, the parent. If the foreign key of a child base object points to ROWID_OBJECT of the parent base object, the foreign key of the associated child cross-reference table points to ROWID_XREF of the associated parent cross-reference table.

Types of Foreign Key Relationships in Operational Reference Store Tables

There are two types of foreign-key relationships in Hub Store tables.

Type	Description
system foreign key relationships	Automatically defined and enforced by Informatica MDM Hub to protect the referential integrity of your schema.
user-defined foreign key relations	Custom foreign key relationships that are manually defined according to the instructions later in this section.

Process Overview for Defining Foreign-Key Relationships

To create a foreign-key relationship:

1. Create the parent table.
2. Create the child table.

3. Define the foreign key relationship between them.

If the child table contains generated keys from the parent table, the load process copies the appropriate primary key value from the parent table into the child table.

Adding Foreign Key Relationships

You can add a foreign key relationship between two base objects.

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand a base object (the base object that will be the child in the relationship).
4. Right-click **Relationships**.

The Schema Manager displays the Properties tab of the Relationships page.

5. Click the **Add** button.

The Schema Manager displays the Add Relationship dialog.

6. Define the new relationship by selecting:
 - a column in the Relate from tree, and
 - a column in the Relate to tree
7. If you want, check (select) the **Create associated index** check box if you want to create an index on this a foreign key relationship. Metadata is defined in the Operational Reference Store that an index exists.
8. Click **OK**.
9. Click the **Diagram** tab to view the foreign key relationship diagram.
10. Click the **Save** button to save your changes.

Note: After you have created a relationship, if you go back and try to create another relationship, the column is not displayed because it is in use. When you delete the relationship, the column will be displayed.

Editing Foreign Key Relationships

You can change only the Lookup Display Name in a foreign key relationship. To change any other properties, you need to delete the relationship, add it again, and specify the properties you want.

To edit the lookup display name for a foreign key relationship between two base objects:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand a base object and right-click **Relationships**.

The Schema Manager displays the Properties tab of the Relationships page.
4. On the **Properties** tab, click the foreign key relationship whose properties you want to view.

The Schema Manager displays the relationship details.
5. Click the **Edit** button next to the Lookup Display Name and specify the new value.
6. If you want, select the **Has Associated Index** check box to add an index on this foreign key relationship, or clear it to remove an existing index.
7. Click the **Save** button to save your changes.

Relationship Details

You can view the details of relationships between tables and foreign keys in the Relationship Details screen.

Description

Description of the relationship between the foreign key and the object.

Constraint Table

Name of the table where the foreign key constraint is defined.

Constraint Column

Name of the column where the foreign key constraint is defined.

Parent Table

Name of the parent table for the foreign key.

Parent Column

Name of the parent column, which is in the parent table, for the foreign key.

Relationship Enforcement

- **Restrict.** The MDM Hub restricts deletion of records from the parent tables when the associated child records are available. To delete a record from the parent table, delete the associated record from the child table first.
- **Delete Cascade.** The MDM Hub deletes all appropriate metadata when the parent record is deleted from C_REPOS_TABLE. If you delete a record from the parent table, the MDM Hub deletes the record from the parent table and associated records from the child table.

Type

- **Enforced.** The MDM Hub creates enforced relationships. Enforced relationships are database constraints in the database. For example, the relationship between a base object and its cross-reference table is enforced. Problems with an enforced relationship are reported from the database layer.
- **Virtual.** The user creates virtual relationships. The MDM Hub does not create constraints for virtual relationships. For virtual relationships, the MDM Hub stores the metadata for the foreign key relationships internally. Problems with a virtual relationship are only reported when you run the application server.

Has Associated Index

Select the **Has Associated Index** check box to add an index to a foreign key relationship. Clear the **Has Associated Index** check box to remove an existing index from a foreign key relationship.

Lookup Display Name

The name that displays in the Hub Console for the lookup table. You can change the display name by clicking the **Edit** button.

Configuring Lookups for Foreign-Key Relationships

After you create a foreign key relationship, you can configure a lookup for the column. A lookup causes Informatica MDM Hub to retrieve a data value from a parent table during the load process. For example, if an Address staging table includes a CONSUMER_CODE_FK column, you could have Informatica MDM Hub perform a lookup to the ROWID_OBJECT column in the Consumer base object and retrieve the ROWID_OBJECT value of the associated parent record in the Consumer table.

Deleting Foreign Key Relationships

You can delete any user-defined foreign key relationship that has been added. You cannot delete the system foreign key relationships that Informatica MDM Hub automatically defines and enforces to protect the referential integrity of your schema.

To delete a foreign-key relationship between two base objects:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand a base object and right-click **Relationships**.
4. On the Properties tab, click the foreign-key relationship that you want to delete.
5. Click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
6. Click **Yes**.
The Schema Manager deletes the foreign key relationship.
7. Click the **Save** button to save your changes.

Viewing Your Schema

You can use the Schema Viewer tool in the Hub Console to visualize the schema in an Operational Reference Store. The Schema Viewer is particularly helpful for visualizing a complex schema.

Starting the Schema Viewer

Note: The Schema Viewer can also be launched from within the Repository Manager, as described in the *Multidomain MDM Repository Manager Guide*. Once started, however, the instructions for using the Schema Viewer are the same, regardless of where it was launched from.

To start the Schema Viewer tool:

- In the Hub Console, expand the Model workbench, and then click **Schema Viewer**.
The Hub Console starts the Schema Viewer and loads the data model, showing a progress dialog.
After the data model loads, the Hub Console displays the Schema Viewer tool.








Panes in the Schema Viewer

The Schema Viewer is divided into two panes.

Pane	Description
Diagram pane	Shows a detailed diagram of your schema.
Overview pane	Shows an abstract overview of your schema. The gray box highlights the portion of the overall schema diagram that is currently displayed in the diagram pane. Drag the gray box to move the display area over a particular portion of your schema.

Command Buttons in the Schema Viewer

The Diagram Pane in the Schema Viewer contains the following command buttons:

Button	Name	Description
	Zoom In	Zooms in and magnifies a smaller area of the schema diagram.
	Zoom Out	Zooms out and displays a larger area of the schema diagram.
	Zoom All	Zooms out to displays the entire schema diagram.
	Layout	Toggles between a hierarchic and orthogonal view.
	Options	Shows or hides column names and controls the orientation of the hierarchic view.
	Save	Saves the schema diagram.
	Print	Prints the schema diagram.

Zooming In and Out of the Schema Diagram

You can zoom in and out of the schema diagram.

Zooming In

To zoom into a portion of the schema diagram:

- Click the **Zoom In** button.
The Schema Viewer magnifies a portion of the screen.

Note: The gray highlight box in the Overview Pane has grown smaller to indicate the portion of the schema that is displayed in the diagram pane.

Zooming Out

To zoom out of the schema diagram:

- Click the **Zoom Out** button.
The Schema Viewer zooms out of the schema diagram.

Note: The gray box in the Overview Pane has grown larger to indicate a larger viewing area.

Zooming All

To zoom all of the schema diagram, which means that the entire schema diagram is displayed in the Diagram Pane:

- Click the **Zoom All** button.
The Schema Viewer zooms out to display the entire schema diagram.

Switching Views of the Schema Diagram

The Schema Viewer displays the schema diagram in two different views:

- Hierarchic view (the default)
- Orthogonal view

Toggling Views

To switch between the hierarchic and orthogonal views:

- Click the **Layout** button.
The Schema Viewer displays the other view.

Navigating to Related Design Objects and Batch Jobs

Right-click on an object in the Schema Viewer to display the context menu.

The context menu displays the following commands.

Command	Description
Go to BaseObject	Launches the Schema Manager and displays this base object with an expanded base object node.
Go to Staging Table	Launches the Schema Manager and displays the selected staging table under the associated base object.
Go to Mapping	Launches the Mappings tool and displays the properties for the selected mapping.
Go to Job	Launches the Batch Viewer and displays the properties for the selected batch job.
Go to Batch Groups	Launches the Batch Group tool.

Configuring Schema Viewer Options

To configure Schema Viewer options:

1. Click the **Options** button.
The Schema Viewer displays the Options dialog.

2. Specify the options you want.

Pane	Description
Show column names	Controls whether column names appear in the entity boxes. Check (select) this option to display column names in the entity boxes. Uncheck (clear) this option to hide column names and display only entity names in the entity boxes.
Orientation	Controls the orientation of the schema hierarchy. One of the following values: <ul style="list-style-type: none">- Top to Bottom (default)—Hierarchy goes from top to bottom, with the highest-level node at the top.- Bottom to Top—Hierarchy goes from bottom to top, with the highest-level node at the bottom.- Left to Right—Hierarchy goes from left to right, with the highest-level node at the left.- Right to Left—Hierarchy goes from right to left, with the highest-level node at the right.

3. Click **OK**.

Saving the Schema Diagram as a JPG Image

To save the schema diagram as a JPG image:

1. Click the **Save** button.
The Schema Viewer displays the Save dialog.
2. Navigate to the location on the file system where you want to save the JPG file.
3. Specify a descriptive name for the JPG file.
4. Click **Save**.
The Schema Viewer saves the file.

Printing the Schema Diagram

To print the schema diagram:

1. Click the **Print** button.
The Schema Viewer displays the Print dialog.
2. Select the print options that you want.

Pane	Description
Print Area	Scope of what to print: Print All—Print the entire schema diagram. Print viewable—Print only the portion of the schema diagram that is currently visible in the Diagram Pane.
Page Settings	Page output options, such as media, orientation, and margins.
Printer Settings	Printer options based on available printers in your environment.

3. Click **Print**.

The Schema Viewer sends the schema diagram to the printer.

CHAPTER 9

Queries and Packages

This chapter includes the following topics:

- [Queries and Packages Overview, 124](#)
- [Query Groups, 127](#)
- [Generic Queries, 128](#)
- [Custom Queries, 135](#)
- [Packages, 137](#)

Queries and Packages Overview

In MDM Hub, a query is a request to retrieve data from the Hub Store. The request is in the form of an SQL SELECT statement. When you run a query, the MDM Hub sends the query SQL statement to the database that contains the Hub Store, and the database returns the results of the query. A package is a public view of the results of the query.

Generic Queries

A generic query is a type of query that you define by using a query wizard and building blocks. No SQL knowledge is required. The Queries tool generates an SQL SELECT statement from the selected building blocks. The generated SELECT statement works with all supported databases.

Custom Queries

A custom query is a type of query where you define your own SQL SELECT statement. When you want to use database-specific SQL syntax and grammar, create a custom query.

Query Groups

A query group is a user-defined container for queries. Use query groups to organize your queries so that they are easier to find and run.

Tip: If you create query groups before you create queries, you can select a group when you create a query.

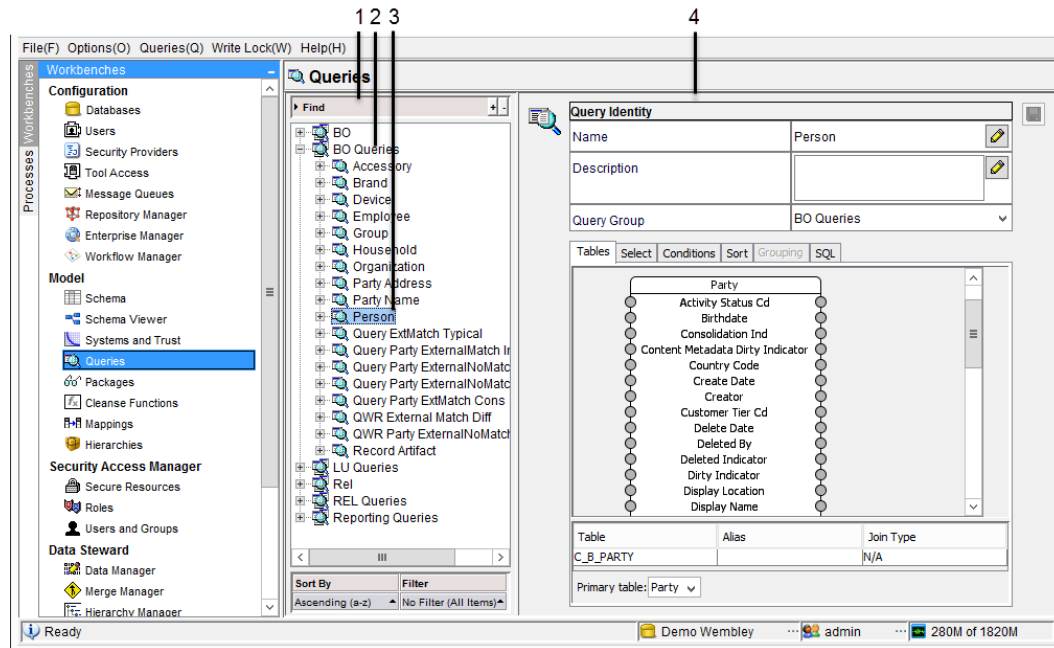
Packages

A package is a public view of the results of a query. Data stewards use packages with the Data Manager and Merge Manager tools. You can also use packages with external applications.

Queries Tool

Use the Queries tool to add, edit, and delete generic queries, custom queries, and query groups. You can also view the results of a query or view the packages that depend on the query.

The following image shows the Queries tool with a query selected:

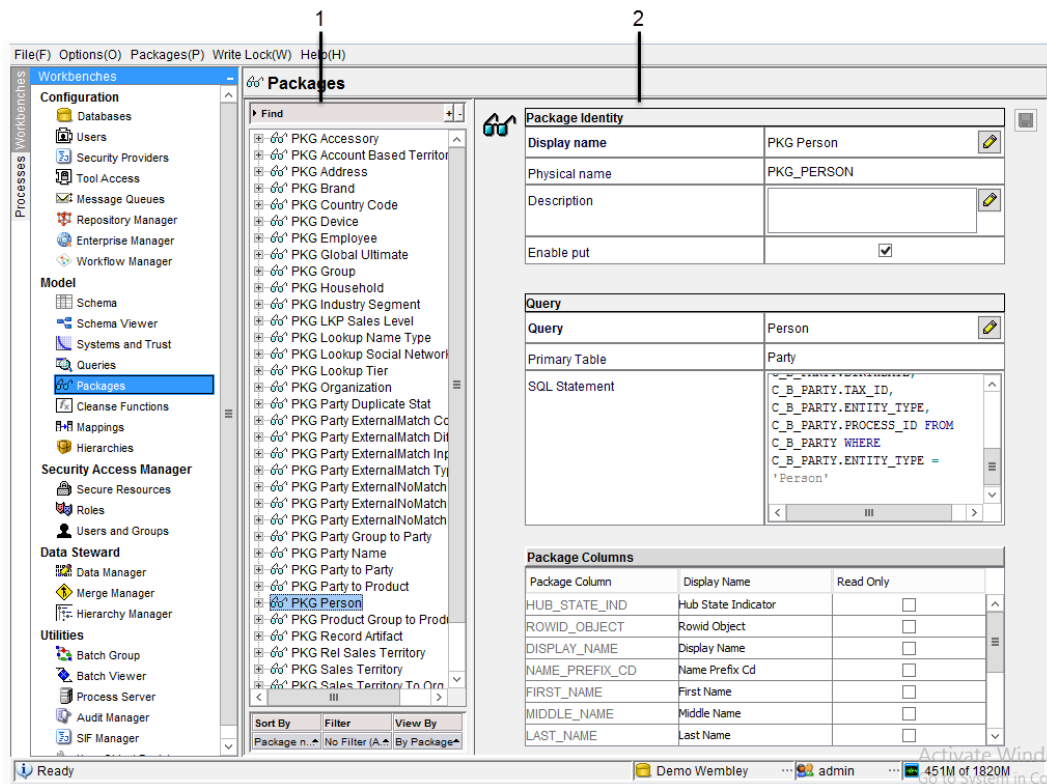


1. Navigation pane. Contains user-defined query groups and queries.
2. Query group.
3. Query.
4. Properties pane. Contains the properties for the selected query group or query.

Packages Tool

Use the Packages tool to add, edit, and delete packages.

The following image shows the Packages tool with a package selected:



1. Navigation pane. Contains user-defined packages.
2. Properties pane. Contains the properties for the selected package.

Maintenance of Queries and Packages

If you change the database schema after you create queries and packages, the MDM Hub updates queries and packages. The update depends on the type of schema change. You might need to edit some queries and packages manually.

The following table lists the types of schema changes, describes the actions taken by the MDM Hub, and notes any actions that you need to perform:

Schema Change	Generic Queries and Packages	Custom Queries and Packages
Revised column name	Updates the generic queries and packages to use the revised column name.	Updates the custom queries and packages to use the revised column name.
Deleted column	Removes the deleted column from the generic queries and packages.	Custom queries are not updated. The custom queries and packages that use the deleted column are no longer valid. Note: You must edit the queries and packages to remove the deleted column.
Deleted base object	Deletes the generic queries and packages that use the deleted base object.	Custom queries are not updated. The custom queries and packages that use the deleted base object are no longer valid. Note: You must delete the queries and the dependent packages.

Query Groups

A query group is a user-defined container for queries. Use query groups to organize your queries so that they are easier to find and use.

For example, you might want to create separate groups for the following types of queries:

- Generic queries that are suitable for use in update packages.
- Queries that select from more than one base object table.
- Queries that select from lookup tables.

Adding a Query Group

Use query groups to organize your queries.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, right-click and click **New Query Group**.
4. In the Add Query Group window, enter a name for the query group and, optionally, enter a description.
5. Click **OK**.

The query group appears in the navigation pane in alphabetical order.

Editing a Query Group

You can edit the query group name and description.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, select a query group.
The properties appear in the properties pane.
4. To edit a property, click its **Edit** icon, edit the text, and click the **Accept Edit** icon.
5. Click the **Save** icon.

Deleting a Query Group

If a query group contains queries, you must move or delete the queries before you can delete the group.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, expand the target query group.
4. If the query group contains queries, either move or delete the queries in the group.
Tip: To move a query, edit the query and select another query group.
5. Right-click the empty query group and click **Delete Query Group**.

Generic Queries

You build generic queries by using the building blocks in the Queries tool. You do not need to know SQL to create generic queries.

The building blocks that you select specify the criteria to use to retrieve data, including table names, column names, and a set of conditions. Queries can also include instructions about how to sort the results and group the results. The MDM Hub generates an SQL statement from the building blocks. The SQL statement works with any database.

Adding a Generic Query

If you want to use the building blocks to construct a query that can be understood by any of the supported databases, add a generic query.

Tip: If you want to code an SQL statement, add a custom query instead.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. Optionally, in the navigation pane, select the query group to which you want to add the query.
4. Right-click in the navigation pane and click **New Query**.

The **New Query Wizard** opens.

5. If you see a **Welcome** screen, click **Next**.
6. Specify the general query properties, and click **Next**:

Property	Description
Query name	Type a descriptive name for the query.
Description	Optionally, type a description for the query.
Query Group	Optionally, select a different query group.
Select primary table	Select the table from which you want to retrieve data.

7. If you want to retrieve a subset of columns, select the columns:
 - a. In the **Select query columns** screen, select the columns to include and clear the check boxes for all other columns.
 - b. If you intend to use the query in a PUT-enabled package, select the **Rowid Object** column.

Note: The Rowid Object is a required column for PUT-enabled packages.

8. Click **Finish**.

The query appears within the selected query group.
9. To view the results of the query, in the navigation pane, expand the query and click **View**.

The Queries tool displays the results of the query.

To further refine the query, edit the query and use the query building blocks.

Refining a Generic Query

After you create a generic query, you can refine the query by using the building blocks. Edit the query to open the building blocks. Each building block is contained on a tab in the properties pane.

The following table lists the tabs, describes the building blocks, and identifies the equivalent SQL syntax:

Tab Name	Building Block Description	SQL Syntax
Tables	Tables associated with this query.	FROM clause
Select	Columns associated with this query. You can add functions and constants to the columns.	SELECT <list of columns>
Conditions	Conditions associated with this query. Determines selection criteria for individual records.	WHERE clause
Sort	Sort order for the results of this query.	ORDER BY clause
Grouping	Grouping for the results of this query.	GROUP BY clause
SQL	Displays the SQL statement that is generated from the selected building blocks.	SELECT statement with all clauses

Editing a Generic Query

To refine the query criteria, edit the query.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, select the query.
The query properties appear in the properties pane.
4. To change the name or description, click its **Edit** icon, edit the text, and click the **Accept Edit** icon.
5. To change the query group, select a group from the Query Group list.
6. To refine the query criteria, select a tab and define the building blocks.

RELATED TOPICS:

- [“Selecting Additional Tables” on page 130](#)
- [“Selecting Columns” on page 130](#)
- [“Defining Functions” on page 131](#)
- [“Defining Constants” on page 131](#)
- [“Defining Comparison Conditions” on page 132](#)
- [“Defining a Sort Order for Results” on page 133](#)
- [“Defining a Grouping for Results” on page 133](#)

Selecting Additional Tables

When you created the generic query, you selected the primary table that you want to query. When you edit the query, you can add additional tables. You can also define foreign key relationships between the tables.

Note: If you intend to use a query with an update package, do not select additional tables. The purpose of an update package is to update the data in the primary table.

1. In the properties pane, click the **Tables** tab.
2. Click the **Add** icon.

Tip: If the Add icon is not available, the query is associated with an update package. Update packages can reference only one table.

3. In the **Select table to add** dialog box, select the table and click **OK**.

The table appears in the view area and in the list. In the list, the Join Type is Cross.

4. Optionally, create a foreign key relationship between the tables:

- a. In the view area, find the columns that you want to relate. The columns must be compatible for a join.

- b. From a circle beside one column, drag a connector line to a circle beside the other column.

In the list, the Join Type changes to Inner.

- c. Optionally, change the Join Type by selecting a type from the list.

Caution: When a query contains more than one relationship, only one of the relationships can be an outer join.

5. Add other tables as required.

If you add a table or relationship in error, you can remove it.

Option	Description
Remove a relationship	In the view area, right-click the connector line, and click Delete .
Remove a table	In the view area, select the table, and click the Delete icon. You cannot delete the primary table.

6. Click the **Save** icon.

Selecting Columns

You can restrict the query to a subset of columns from each table. If you added columns that you want to remove, you can delete them from the query.

1. In the properties pane, click the **Select** tab.
2. Click the **Add** icon.

The **Add Table Column** dialog box opens.

3. Expand the lists of table columns.
4. Select the columns to include in the query.
5. Click **OK**.

The selected columns appear in the table.

6. Optionally, you can reorder or remove columns.

Option	Description
Reorder	Select the column, and click the Move Up or Move Down icon.
Remove	Select the column, and click the Delete icon.

7. Click the **Save** icon.

Defining Functions

You can add aggregate functions to the columns in your queries. For example, you can use COUNT, MIN, or MAX on the data retrieved from the column. You can also edit and remove functions from the query.

The following code sample shows a function in an SQL statement:

```
select col1, COUNT(col2) as c1 from table_name
```

1. In the properties pane, click the **Select** tab.
2. Click the **Add Function** icon.
The **Add Function** dialog box opens.
3. Select the column.
4. Select the function.
5. Click **OK**.
The function appears in the table.
6. Optionally, you can edit, reorder, or remove functions.

Option	Description
Edit	Select the function, and click the Edit icon.
Reorder	Select the function, and click the Move Up or Move Down icon.
Remove	Select the function, and click the Delete icon.

7. Click the **Save** icon.

Defining Constants

You can add constants that apply to the column data retrieved by the query. You can also edit and remove constants from the query.

1. In the properties pane, click the **Select** tab.
2. Click the **Add Constant** icon.
The **Add Constant** dialog box opens.
3. Select the data type.
4. If the Value field appears, enter a value that respects the data type.
5. Click **OK**.
The constant appears in the table.

6. Optionally, you can edit, reorder, or remove constants.

Option	Description
Edit	Select the constant, and click the Edit icon.
Reorder	Select the constant, and click the Move Up or Move Down icon.
Remove	Select the constant, and click the Delete icon.

7. Click the **Save** icon.

Defining Comparison Conditions

You can define comparison conditions in a query. A comparison condition has a column, an SQL comparison operator, and another column or a constant. When you run the query, the query compares the values in the column against the condition, and returns records that satisfy the condition.

Note: When the data in a column is encrypted, you cannot create conditions that use strings or wildcard characters.

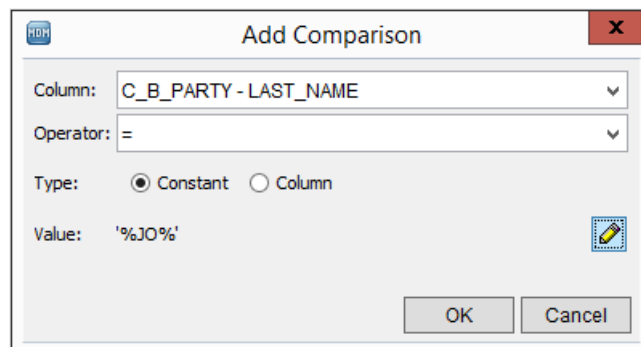
1. In the properties pane, click the **Conditions** tab.
2. Click the **Add** icon.

The **Add Comparison** dialog box opens.

3. In the Column field, select the column on which you want to define a comparison.
4. In the Operator field, select an SQL comparison operator.
5. Select the target of the comparison, which can be another column or a constant.

- If you select **Column**, select the column from the Edit Column list.
- If you select **Constant**, the default value is NULL. To change the value, click the **Edit** icon, select a data type, and if the Value field appears, enter a value.

For example, the following image shows a comparison condition that compares values in the column LAST_NAME with the string %JO%:



When you run the query, the query returns records with values like "Johnson", "Vallejo", and "Major".

6. Click **OK**.

The condition appears in the table.

- Optionally, you can edit, reorder, or remove conditions.

Option	Description
Edit	Select the condition, and click the Edit icon.
Reorder	Select the condition, and click the Move Up or Move Down icon.
Remove	Select the condition, and click the Delete icon.

- Click the **Save** icon.

Defining a Sort Order for Results

You can specify how you want the database to sort the results of the query. You choose the columns on which to perform the sort.

- In the properties pane, click the **Sort** tab.
- Click the **Add** icon.
The **Add Table Column** dialog box opens.
- Find the table and expand the list of columns.
- Select the columns to include in the sort.
- Click **OK**.

The selected columns appear in the Sort table. By default, columns are sorted in ascending order.

- If you want to sort a column in descending order, clear the check box in the **Ascending** column.
- Optionally, you can reorder or remove columns.

Option	Description
Reorder	Select the column, and click the Move Up or Move Down icon.
Remove	Select the column, and click the Delete icon.

- Click the **Save** icon.

Defining a Grouping for Results

You can specify how you want the database to group the results of the query. You choose the columns to include in the grouping.

- In the properties pane, click the **Grouping** tab.
- Click the **Add** icon.
The **Add Table Column** dialog box opens.
- Expand the lists of table columns.
- Select the columns to include in the group.
- Click **OK**.

The selected columns appear in the table.

6. Optionally, you can reorder or remove columns.

Option	Description
Reorder	Select the column, and click the Move Up or Move Down icon.
Remove	Select the column, and click the Delete icon.

7. Click the **Save** icon.

Viewing the SQL for a Query

For generic queries, the **Queries** tool generates an SQL statement from the building blocks you specified. The SQL statement is generated each time you update the query.

- To view the generated SQL statement, in the properties pane, click the **SQL** tab.

Viewing the Query Results

You can preview the results of a query within the Queries tool.

1. In the **Model** workbench, click **Queries**.
2. In the navigation pane, expand the query group that contains the query.
3. Expand the query.
4. Click **View**.

The Queries tool displays the results of the query.

Viewing the Impact of a Query

For each query, you can view its dependencies. For example, you can see the packages that use the query.

1. In the **Model** workbench, click **Queries**.
2. In the navigation pane, expand the query group that contains the query.
3. Right-click the query and click **Impact Analysis**.
The Impact Analysis dialog box opens.
4. Review the system objects that use the query.
5. Click **Close**.

Deleting a Query

If you no longer use a query, delete it.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, expand the query group that contains the query.
4. Right-click the query and click **Impact Analysis**.
The **Impact Analyzer** dialog box opens.

5. If the Packages section contains packages, make a note of the package names, and click **Close**.
The **Impact Analyzer** dialog box closes.
6. If the query was linked to packages, delete the packages.
 - a. In the **Model** workbench, click **Packages**.
 - b. Right-click a package name and click **Delete Package**. Repeat to remove all dependent packages.
 - c. In the **Model** workbench, click **Queries**.
7. In the navigation pane, right-click the query and click **Delete Query**.

Custom Queries

A *custom query* is a query for which you supply the SQL statement directly, rather than building it. Custom queries can be used in display packages. You can create custom queries against any object.

If you use a proxy user in an IBM DB2 environment, ensure that the proxy user has access to the objects in a query. If the Repository Manager does not have metadata for all the objects in a query, the Repository Manager migration can result in privilege warnings for a query. To fix the warnings, manually grant the proxy user access to the objects.

SQL Syntax for Custom Queries

The Hub Console applies some restrictions to the SQL syntax that you can use in a custom query. Beyond these restrictions, use the SQL syntax and grammar that the database supports.

The following table describes the restrictions for SQL syntax:

SQL	Restriction
Statements	A custom query must be a SELECT statement. Other SQL statements are not supported.
Column Names	Column names can contain alphanumeric characters and underscores. Spaces and other special characters are not supported.
Aliases	Alias names can contain alphanumeric characters and special characters. Spaces are not supported.
Constant Columns	To add a constant column, which is enclosed in single quotes, you must use an alias. For example, the following query uses the alias <code>const_alias</code> : <pre>SELECT ID, 'CONST_COL' AS const_alias FROM c_party</pre>
Aggregate Functions	To add an aggregate function with a special character, you must use an alias. For example, the following query uses the alias <code>new_rowid</code> : <pre>SELECT rowid_object*0 AS new_rowid FROM c_party</pre>

SQL Validation

When you save a custom query, the MDM Hub performs client-side validation of the SQL statement and then sends the SQL statement to the database for further validation.

Client-side SQL Validation

The MDM Hub validates that the SQL statement meets the SQL syntax required by the MDM Hub. For example, the client-side validation process checks that the statement begins with the keyword **SELECT** and the column names do not contain spaces or special characters. If the validation process finds an error, the process displays an error message.

Database-side SQL Validation

When the database receives the SQL statement, the database verifies that the syntax and grammar of the SQL statement meet the requirements of the database. If the SQL statement generates an error, the database returns the error to the MDM Hub, and the Hub Console displays the database error.

Adding a Custom Query

You can use database-specific SQL syntax in a custom query. Ensure that the SQL statement conforms to the syntactical requirements of both the database and the MDM Hub Console.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. If you defined query groups, select the group to which you want to add the query.
4. Right-click the group and click **New Custom Query**.
The Queries tool displays the New Custom Query Wizard.
5. If you see a Welcome screen, click **Next**.
6. Set the following custom query properties:

Property	Description
Query name	Type a descriptive name for this query.
Description	Optionally, type a description for this query.
Query Group	Optionally, change the query group that you selected.

7. Click **Finish**.
The query appears in the navigation pane under the selected query group.
8. Click the **Edit** icon next to the SQL field.
9. Enter the SQL query according to the syntax rules for your database environment.
10. Click the **Save** icon.
The Hub Console saves the query and sends the query to the database. If the database finds an error in the SQL statement, the Queries tool displays the database error message. Fix the errors and save your changes.
11. To view the results of the query, in the navigation pane, expand the query and click **View**.
The Queries tool displays the results of the query.

Editing a Custom Query

You can edit a custom query.

1. In the **Model** workbench, click **Queries**.
2. Acquire a write lock.
3. In the navigation pane, select the custom query.
The properties and SQL statement appear in the properties pane.
4. To edit a property, click its **Edit** icon, edit the text, and click the **Accept Edit** icon.
5. Click the **Save** icon.
The Queries tool validates your query settings and prompts you if it finds errors.

RELATED TOPICS:

- [“Deleting a Query” on page 134](#)
- [“Viewing the Impact of a Query” on page 134](#)
- [“Viewing the Query Results” on page 134](#)

Packages

A package is a public view of one or more queries. You use packages with the data steward tools in the Hub Console and with external applications that use services to access master data.

You can create two kinds of packages: display packages and update packages. Display packages define read-only views of master data. Update packages define views from which authorized users can make changes to master data.

When you want to restrict a package to authorized users, make the package a secure resource. You manage all secure resources from the Roles tool. For more information, see the *Multidomain MDM Security Guide*.

Display Packages

If you want users to view the query results from the data steward tools or from an external application, create display packages. Users can view the query results, but cannot make changes to the data.

For each authorized role, set read privileges for the display packages that the role can use. Do not enable any other privileges. For more information about secure resources and privileges, see the *Multidomain MDM Security Guide*.

Update Packages

An update package is also known as a PUT-enabled package, a putable package, or a merge package. When you create an update package, you enable the PUT API within the package. A data steward can use the update package to add, change, or merge master data.

Create update packages when you want authorized users to be able to perform any of the following actions:

- Update data in records from the Data Manager or from an external application
- Insert records from the Data Manager or from an external application
- Merge records from the Merge Manager or from an external application

For each authorized role, set privileges for the update packages that the role can use. Grant the read privilege and at least one of the create, update, or merge privileges. For more information about secure resources and privileges, see the *Multidomain MDM Security Guide*.

Requirements for Update Packages

Update packages have some requirements at the package level and at the query level.

Package-level requirements

An update package must adhere to the following requirements:

- Select a generic query for the update package. Custom queries and other packages are not supported with update packages.
- Select the **Enable Put** option.
- Access only the tables and relationships defined within the generic query. The update package cannot contain joins to other tables.

Query-level requirements

The generic query must adhere to the following requirements:

- Select only a primary table. The generic query cannot contain additional tables.
- If you select individual columns, you must add the ROWID_OBJECT column.
- The generic query cannot contain system tables, constant columns, aggregate functions, or groupings.

Adding a Package

You can create either a display package or an update package.

Tip: If you want to create an update package, before you begin, create a generic query that meets the requirements for update packages.

1. In the **Model** workbench, click **Packages**.
2. Acquire a write lock.
3. Right-click in the navigation pane and click **New Package**.
The **New Package Wizard** opens.
4. If you see a **Welcome** screen, click **Next**.
5. Set the following properties:

Properties	Description
Display Name	Type a descriptive name for the package. This name appears in the navigation pane.
Physical Name	Optionally, change the suggested physical name. The wizard suggests a physical name that is based on the display name you specified.
Description	Optionally, type a description for the query.
Query Group	Optionally, select a different query group.
Enable PUT	To create an update package, select this option. To create a display package, clear this option.

Properties	Description
Secure Resource	To restrict who can use the package, select this option. You use the Roles tool to assign user roles to secured packages.

- Click **Next**.

The **New Package Wizard** displays the **Select Query** dialog box.

- Select the query that you want to use in the package.

Remember: For update packages, you must select a generic query. For display packages, you can select a generic query or a custom query.

- To use an existing query, select the query from the list.
- To create a query, click **New Query** and create the query.
- To create a query group, click **New Query Group** and create the query group.

- Click **Finish**.

- To preview the package results, in the navigation pane, expand the package and click **View**.

The Packages tool displays a preview of the package.

Tip: If the package fails to generate and you selected a custom query, verify that the custom query adheres to the SQL syntax constraints.

Editing a Package

You can change package properties and the underlying query.

- In the **Model** workbench, click **Packages**.
- Acquire a write lock.
- In the navigation pane, select the package.
The properties appear in the properties pane.
- To edit a text property, click its **Edit** icon, edit the text, and click the **Accept Edit** icon.
- Click the **Save** icon.
- Optionally, you can edit the query for this package.
 - In the navigation pane, expand the package.
 - Click **Query**.
 - Edit the query.
- To preview the package results, in the navigation pane, expand the package and click **View**.
The Packages tool displays a preview of the package.

Refreshing a Package After Changing Queries

If you change a query, refresh all packages that use the query.

Note: After a refresh, if a package remains out of synch with the query, select or clear columns to match the query.

- In the **Model** workbench, click **Packages**.
- Acquire a write lock.

3. In the navigation pane, expand the package.
4. Click **Refresh**.

Deleting a Package

When you no longer need a package, delete the package to remove the dependency from the underlying query.

1. In the **Model** workbench, click **Packages**.
2. Acquire a write lock.
3. In the navigation pane, right-click the package and click **Delete Package**.

Specifying Join Queries

You can create a package that permits data stewards to view base object information, along with information from the other tables, in the Data Manager or Merge Manager.

1. Create an update package that queries a base object.
2. Create a query to join the update package with the other tables.
3. Create a display package based on the query you just created.

CHAPTER 10

Timeline

This chapter includes the following topics:

- [Overview, 141](#)
- [Guidelines, 142](#)
- [Example, 142](#)
- [Record Versions, 144](#)
- [Timeline Granularity, 145](#)
- [History and State Management, 146](#)
- [Timeline Enforcement Rules, 146](#)
- [Configuring the Base Object Timeline, 155](#)
- [Load Multiple Versions of a Record In a Batch Job, 156](#)
- [Edit the Effective Period of a Record Version, 158](#)
- [Add a Record Version, 160](#)
- [Update Data in a Record, 161](#)
- [Update a Relationship, 162](#)
- [End a Relationship, 164](#)
- [Delete a Relationship Period, 165](#)
- [Delete All Relationship Periods, 166](#)
- [Using Timeline Extract, 167](#)

Overview

You can manage data change events of business entities and their relationships through timeline management. A data change event is a change to data that is effective for a time period. When data change events occur, the MDM Hub creates versions of an entity instead of overwriting the existing data.

You can define the data change events or versions of business entities and their relationships based on their effective periods. Data changes occur over time and are independent of their relationship to other data. The changes to data result in new effective periods or updates to an existing or a future effective period. Use timeline management to track the changes to effective periods of data.

To manage data events of business entities, such as customer address, phone number, and their relationships, you can enable the timeline. To maintain the effective periods for base object records, the

MDM Hub uses cross-reference tables that are associated with the base objects for which you enabled the timeline.

You enable the timeline for base objects when you configure base object properties in the Hub Console. When you enable the timeline, the MDM Hub tracks past, present, and future changes to data, such as the address of a customer.

Timeline is enabled by default for hierarchy management relationship base objects.

Base objects for which you enable the timeline contain a version of data that pertains to the effective period that you specify. The base objects might not reflect the current values. The version of data in base objects at any point in time depends on the contributing cross-reference tables that might change with time. You can use SIF API calls to get records for the current effective date. Additionally, you can pass any effective date to retrieve the data for the effective date that you specify.

When an incoming change affects the current effective period of the base object, the MDM Hub uses the current trust settings to recalculate BVT. Then the MDM Hub updates the base object record.

Note: The Hub Server always uses the current trust settings to calculate the best version of the truth (BVT) for past, present, or future data.

After you enable the timeline for a base object, you cannot disable it. If you enable the timeline for a populated base object, the Hub Server sets the effective period start date and end date to null.

Guidelines

When you configure the timeline for base objects, the MDM Hub manages data change events of base objects and their relationships.

The timeline for relationship base objects is enabled by default but not for entity base objects. You cannot disable the timeline for relationship base objects. Also, you cannot disable the timeline for any base object after the timeline is enabled.

When you enable the timeline for base objects, history and state management are enabled by default. You cannot disable history or state management for base objects that have the timeline enabled.

The MDM Hub does not automatically update the best version of the truth (BVT) for base objects that have the timeline enabled. You can view the BVT of a record for the effective period that you specify.

Example

Your organization has the CUSTOMER base object for which you enable timeline. The CUSTOMER base object contains a record for John Smith who lived in the city of Los Angeles effective 31 January 2011 to 20 October 2013, currently lives in San Francisco effective 21 October 2013, and will live in Las Vegas effective 25 November 2015. The MDM Hub tracks past, present, and future changes to data, such as the address of John Smith.

Record with Past Data

The CUSTOMER base object contains the record for John Smith with a Los Angeles address where he resided in the past. The cross-reference table contains the record of John Smith with the period between which he lived in Los Angeles.

The cross-reference record for John Smith shows that he lived in Los Angeles in the past.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	Los Angeles	January 31, 2011	October 20, 2013

The base object record for John Smith displays the best version of the truth (BVT) for the effective date that you specify, such as 12 February 2012.

Rowid	Object	Customer ID	First Name	Last Name	City
2		25	John	Smith	Los Angeles

Record with Past and Present Data

The CUSTOMER base object contains the record for John Smith with an address that is the BVT for the date that you specify. The cross-reference table contains a record for John Smith. The cross-reference table shows the period between which he lived in Los Angeles and a record with the period between which he lived in San Francisco.

The cross-reference record for John Smith shows that he lived in Los Angeles in the past.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	Los Angeles	January 31, 2011	October 20, 2013
2	2	25	John	Smith	San Francisco	October 21, 2013	November 24, 2015

The base object record for John Smith displays the BVT for the period that you specify, such as 12 February 2012.

Rowid	Object	Customer ID	First Name	Last Name	City
2		25	John	Smith	Los Angeles

Record with Past, Present, and Future Data

The CUSTOMER base object contains the record for John Smith with an address that is the BVT for the date that you specify. The cross-reference table contains the record of John Smith with the period for which he lived in Los Angeles.

The cross-reference records for John Smith, who lived in Los Angeles, lives in San Francisco, and will live in Las Vegas from 25 November 2015.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	Los Angeles	January 31, 2011	October 20, 2013
2	2	25	John	Smith	San Francisco	October 21, 2013	November 24, 2015
3	2	25	John	Smith	Las Vegas	November 25, 2015	null

The base object record for John Smith displays the BVT for the period that you specify, such as 12 February 2012.

Rowid	Object	Customer ID	First Name	Last Name	City
2		25	John	Smith	Los Angeles

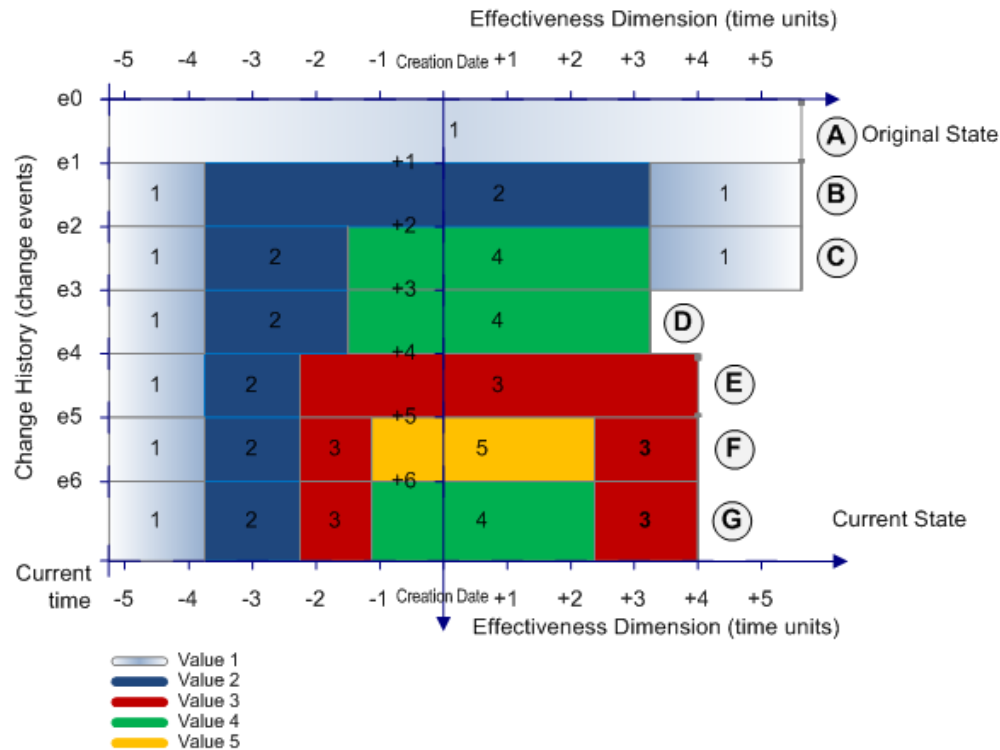
Record Versions

When the timeline is enabled for a base object, you can have a two-dimensional visibility into data. The two-dimensional visibility into data is based on the effective period and history of records. The effective period is the period of time for which a record is effective. You can define the effective period for a record by specifying the start date and the end date of the record. History is a past data event in the life of a record. To view the history of a record, specify a date from the past during which the record was effective.

Record Versions Example

Your organization has a record present in a base object for which the timeline is enabled. The timeline of the record includes multiple data change events, such as e0, e1, e2, e3, e4, e5, and e6. The data change events, which are new or updated record values result in new cross-reference record versions that are effective for the specified effective period.

The following image shows the overall effective period of the record along its historical lifespan and its change history:



The record undergoes the following changes in its life span:

- A. The original record with value 1 is effective all the time because no effective period is specified.
- B. The MDM Hub adds a new version of the record with a new value 2 for a specific effective period.
- C. The MDM Hub adds another new version of the record for the record with a new value 4 for a new effective period.
- D. The MDM Hub removes a version of the record for a specific effective period with value 1.
- E. The MDM Hub adds a version of the record with a new value 3 for a specific effective period. The MDM Hub removes the version of the record with value 4 with a specific effective period that spans the

effective period of the new version. The MDM Hub updates the effective period of the record version with value 2.

- F. The MDM Hub adds a new version of the record with a new value 5 for a new effective period. The MDM Hub creates two versions of the record with value 3 for the record for two new effective periods.
- G. The value of the record version changes from 5 to 4 but the effective period does not change.

Timeline Granularity

Timeline granularity is the time measurement that you want to use to define effective periods for versions of records. For example, you can choose the effective periods to be in years, months, or seconds.

You can configure the timeline granularity of year, month, day, hour, minute, or seconds to specify effective periods of data in the MDM Hub implementation. You can configure the timeline granularity that you need when you create or update an Operational Reference Store.

Important: The timeline granularity that you configure cannot be changed.

When you specify an effective period in any timeline granularity, the system uses the database time locale for the effective periods. To create a version that is effective for one timeline measurement unit, the start date and the end date must be the same.

The following table describes the timeline granularity options that you can configure:

Timeline Granularity	Description
Year	When the timeline granularity is year, you can specify the effective period in the year format, yyyy, such as 2010. An effective start date of a record starts at the beginning of the year and the effective end date ends at the end of the year. For example, if the effective start date is 2013 and the effective end date is 2014, then the record would be effective from 01/01/2013 to 31/12/2014.
Month	When the timeline granularity is month, you can specify the effective period in the month format, mm/yyyy, such as 01/2013. An effective start date of a record starts on the first day of a month. The effective end date of a record ends on the last day of a month. For example, if the effective start date is 02/2013 and the effective end date is 04/2013, the record is effective from 01/02/2013 to 30/04/2013.
Day	When the timeline granularity is day, you can specify the effective period in the date format, dd/mm/yyyy, such as 13/01/2013. An effective start date of a record starts at the beginning of a day, that is 12:00. The effective end date of the record ends at the end of a day, which is 23:59. For example, if the effective start date is 13/01/2013 and the effective end date is 15/04/2013, the record is effective from 12:00 on 13/01/2013 to 23:59 on 15/04/2013.
Hour	When the timeline granularity is hour, the effective period includes the year, month, day and hour. The timeline format is dd/mm/yyyy hh, such as 13/01/2013 15. An effective start date of a record starts at the beginning of an hour of a day. The effective end date of the record ends at the end of the hour that you specify. For example, if the effective start date is 13/01/2013 15 and the effective end date is 15/04/2013 10, the record is effective from 15:00 on 13/01/2013 to 10:59 on 15/04/2013.

Timeline Granularity	Description
Minute	When the timeline granularity is minute, the effective period includes the year, month, day, hour, and minute. The format timeline format is dd/mm/yyyy hh:mm, such as 13/01/2013 15:30. An effective start date of a record starts at the beginning of a minute. The effective end date of the record ends at the end of the minute that you specify. For example, if the effective start date is 13/01/2013 15:30 and the effective end date is 15/04/2013 10:45, the record is effective from 15:30:00 on 13/01/2013 to 10:45:59 on 15/04/2013.
Second	When the timeline granularity is second, the effective period includes the year, month, day, hour, minute, and second. The timeline format is dd/mm/yyyy hh:mm:ss, such as 13/01/2013 15:30:45. An effective start date of a record starts at the beginning of a second. The effective end date ends at the end of the second that you specify. For example, if the effective start date is 13/01/2013 15:30:55 and the effective end date is 15/04/2013 10:45:15, the record is effective from 15:30:55:00 on 13/01/2013 to 10:45:15:00 on 15/04/2013.

History and State Management

History and state management are enabled by default for base objects that have the timeline enabled and for the associated relationship base objects.

When the timeline is enabled for a base object, you cannot disable history or state management for the base object. The MDM Hub maintains the history of changes that pertain to the timeline in the history tables that are associated with cross-reference tables. The MDM Hub manages the states of records, such as active, pending, or deleted, in the associated cross-reference tables.

When a record that you update requires approval, a data change event occurs and the MDM Hub saves the record as PENDING. The data change event due to the update has an impact on the cross-reference version. The MDM Hub uses a state management interaction ID to prevent modification to the cross-reference version. The impact on the associated cross-reference version might be due to the overlap of time periods from the same source system. When you promote a PENDING cross-reference, the state of the cross-reference record changes to ACTIVE. If you soft delete a record, the state changes to DELETED.

Timeline Enforcement Rules

When you define and maintain the timeline information, to manage the timelines of business entities and relationships, the MDM Hub enforces pre-defined timeline enforcement rules. The MDM Hub applies a set of rules to the period start and end dates to manage effective periods during the load and put operations.

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

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

Effective Period Computation

The MDM Hub computes the overall effective period of the base object record.

When you use a load or put operation on a base object for which you track data change events, specify period start and end dates for each source record. A base object record can have contributors from multiple source systems or primary key sources or both. To compute the overall effective period of the base object record, the MDM Hub aggregates the effective periods of records from all the source systems.

The following image shows a base object that has contributors from two source systems, System 1 with PKey 1 and System 2 with PKey 2:

System 1/ PKey Source 1	<div>Value 1</div> <div>Value 2</div>
System 2/ PKey Source 2	<div>Value 4</div> <div>Value 5</div>
Overall	<div>1</div> <div>1/4</div> <div>2/4</div> <div>2/5</div> <div>5</div>

In the preceding image, System 1 has values 1 and 2 and System 2 has values 4 and 5 for different effective periods that overlap. To compute the overall effective period of the record, the MDM Hub aggregates the effective periods of records from both the source systems. The result of aggregating the effective periods of records is the Best Version of Truth (BVT) for each resultant effective period. The overall effective values 1, 1/4, 2/4, 2/5, and 5 that are represented in the image are the BVT for the specific effective periods.

Note: You cannot change the rules for the management of effective periods.

Rule 1. Add a Record without an Effective Period

If a cross-reference record does not have period start and period end dates, and no existing cross-reference records are present, the record is in effect all the time.

The following image shows a cross-reference record that is inserted without period start and period end dates, and no existing cross-reference records are present:

Before	No Record
The Change	<div>Initial Value</div>
After	<div>Initial Value</div>

Rule 2. Add a Record with an Effective Period

If a cross-reference record has period start and period end dates, and no existing cross-reference records are present, the MDM Hub inserts the record with the effective period that you specify.

The following image shows a cross-reference record that is inserted with period start and period end dates, and no existing cross-reference records are present:

Before	No Record
The Change	Initial Value
After	Initial Value

Rule 3. Add a Record Version for an Effective Period

You can add a record version for an existing effective period.

If the following conditions exist, rule 3 applies:

- The period start and period end dates are specified and a cross-reference record with the same effective period exists.
- The cross-reference record that is inserted and the existing cross-reference record belong to the same source system.

Rule 3 ensures the following MDM Hub behavior:

- The cross-reference record is updated.
- The effective period is not changed.

The following image shows a cross-reference record that is inserted with period start and period end dates when a cross-reference record with the same effective period exists:

Before	Initial Value
The Change	Other Value
After	Other Value

Rule 4. Add a Record Version that Intersects an Effective Period and Has an Extended Start Date

You can add a record version that intersects an effective period and has an extended start date.

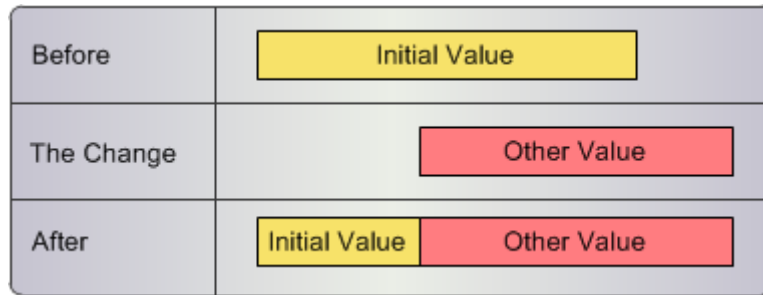
If the following conditions exist, rule 4 applies:

- The specified effective period intersects the effective period of an existing cross-reference record.
- The changed period start date is later than the period start date of the existing cross-reference record.

Rule 4 ensures the following MDM Hub behavior:

- The period end date of the existing cross-reference version is updated with 'changed period start date - 1' in terms of the chosen timeline unit and the initial period start date is not changed.
- A new version of the cross-reference record is inserted with the specified effective period.

The following image shows a cross-reference record with effective period that intersects the effective period of an existing cross-reference record and has a later period start date:



Rule 5. Add a Record Version that Intersects an Effective Period and Has an Earlier End Date

You can add a record version that intersects an effective period and has an earlier end date.

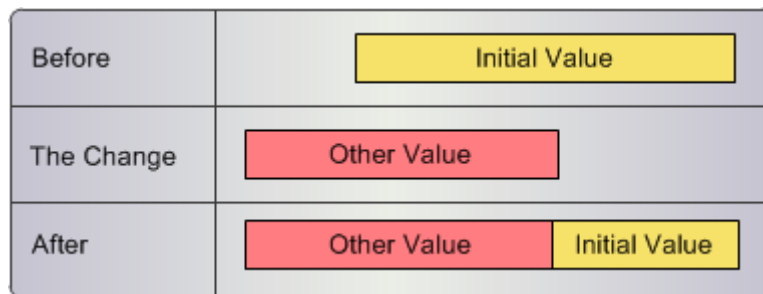
If the following conditions exist, rule 5 applies:

- The specified effective period intersects the effective period of an existing cross-reference record.
- The changed end date is before the existing end date of the cross-reference record.

Rule 5 ensures the following MDM Hub behavior:

- The period start date of the existing cross-reference version is updated with the changed end date incremented by 1 in terms of the chosen timeline measurement unit.
- The original period end date of the existing cross-reference version is not changed.
- A new version of the cross-reference record is added with the specified effective period.

The following image shows a cross-reference record with effective period that intersects the effective period of an existing cross-reference record and has an earlier period end date:



Rule 6. Add a Record Version that is Contained within an Effective Period

You can add a record version that is contained within an effective period.

If the following conditions exist, rule 6 applies:

- The effective period of a new cross-reference version is contained in the effective period of an existing cross-reference record.
- The changed period start date is after the period start date of the existing cross-reference record.
- The changed period end date is before the period end date of the existing cross-reference record.

Rule 6 ensures the following MDM Hub behavior:

- The period end date of the existing cross-reference version is updated with changed date decremented by 1 in terms of the chosen timeline measurement unit.
- The period start date of the existing cross-reference version remains unchanged.
- A new version of the cross-reference record is inserted with the specified effective period.
- A second cross-reference record version is inserted with the period start date set to the changed end date incremented by 1 in terms of the chosen timeline measurement unit.
- The end date of the second cross-reference record version is set to the period end date of the existing cross-reference record.

The following image shows a cross-reference record with effective period that is contained in the effective period of an existing cross-reference record:

Before	Initial Value
The Change	Other Value
After	Initial Value Other Value Initial Value

Rule 7. Add a Record Version to Contain an Effective Period

You can add a record version that can contain the existing effective period of a record version.

If the following conditions exist, rule 7 applies:

- The effective period of an existing cross-reference version starts after or on the same date.
- The effective period of the existing cross-reference version ends before or on the same date as the new cross-reference version.

Rule 7 ensures the following MDM Hub behavior:

- The existing cross-reference version with an effective period that starts after or on the same date and ends before or on the same date as the new cross-reference version is deleted.
- The new cross-reference version is inserted with the specified effective period.
- For all the other existing cross-reference versions, rule 4 and 5 are applied.

The following image shows an existing cross-reference record with effective period that starts after and ends before the effective period of the new cross-reference record:

Before	Initial Value 1 Initial Value 2 Initial Value 3
The Change	Other Value
After	Initial Value 1 Other Value Initial Value 3

Rule 8. Add a Record Version with a Noncontiguous Effective Period

You can add a record version with a noncontiguous effective period.

If the following conditions exist, rule 8 applies:

- The specified effective period is not contiguous with the effective period of an existing cross-reference record.
- The changed period end date is before the period start date or the changed period start date is after the period end date of the existing cross-reference data.

Rule 8 ensures the following MDM Hub behavior:

- The existing version of the cross-reference record remains unchanged.
- If the base object can have noncontiguous effective periods, a new version of the cross-reference record is inserted with the specified effective period.
- If the base object must have contiguous effective periods, a new version of the cross-reference record is not inserted and an error is generated.

The following image shows cross-reference records with noncontiguous effective periods:

Before	Initial Value	
The Change		Other Value
After	Initial Value	Other Value

Rule 9. Add a Record Version in the Pending State within an Effective Period

You can add a record version in the pending state to be contained within an existing effective period.

Rule 9 applies when the effective period of an existing cross-reference record spans the effective period of a new PENDING cross-reference version.

Rule 9 ensures the following MDM Hub behavior:

- The existing version of the cross-reference record is not changed, but is locked by an interaction ID, which is used during promotion.
- The new cross-reference version is inserted with the specified effective period and in the PENDING state.
- When the pending cross-reference version is promoted, rules 1 to 8 are applied.

The following image shows that the effective period of an existing cross-reference record spans the effective period of a new PENDING cross-reference version:

Before	Initial Value
The Change	Other Value
After	Initial Value Other Value

Rule 10. Add a Record Version when a Record Version is Locked

You can add a record version when an existing record version is locked by an interaction ID.

If a change based on rules 1 to 9 affects an existing cross-reference record that is locked by an interaction ID, the MDM Hub restricts the change.

The following image shows that if a new cross-reference record is inserted when a cross-reference record that is locked by an interaction ID exists, no change occurs:

Before	Initial Value 1 Initial Value 2
The Change	Other Value
After	Initial Value 1 Initial Value 2

Rule 11. Add a Record Version when a Version is in the Pending State

You can add a record version in the active or pending state when an existing version is in the pending state.

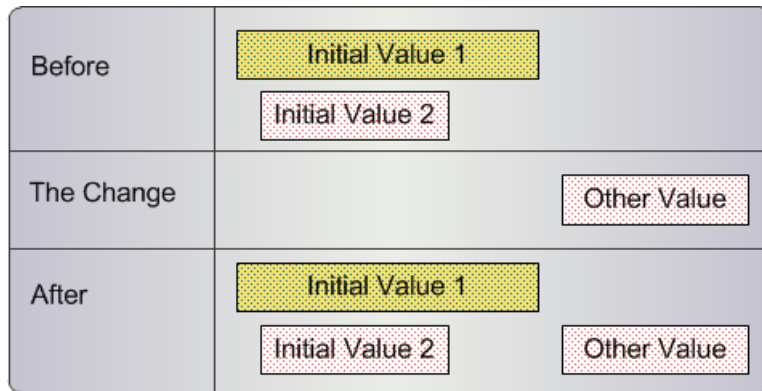
If the following conditions exist, rule 11 applies:

- The existing cross-reference record is in the PENDING state.
- A cross-reference record is inserted in the ACTIVE or PENDING state and the effective period of the cross-reference record does not intersect any locked records.

Rule 11 ensures the following MDM Hub behavior:

- The existing cross-reference version remains unchanged.
- The new version of the cross-reference record is inserted with the specified effective period and without a change in state.

The following image shows that the existing cross-reference record is in the PENDING state and a new cross-reference record is inserted in the PENDING state:



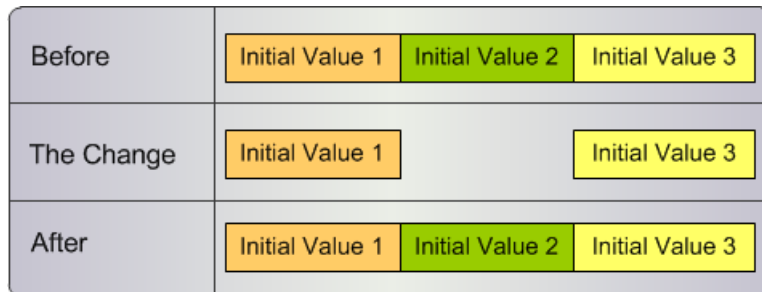
Rule 12. Delete or Update a Record Version in a Contiguous Base Object

You cannot delete or update a record version that breaks the contiguity of record versions in a contiguous base object.

If the base object property setting does not allow noncontiguous effective periods, rule 12 ensures the following MDM Hub behavior:

- An effective period that breaks contiguity cannot be deleted.
- Any change to the effective period of a cross-reference record that breaks contiguity cannot be saved.

The following image shows cross-reference records with contiguous effective periods before and after a version of the cross-reference record is deleted:



Rule 13. Update Data

You can update the data in an existing record version.

If the following conditions exist, rule 13 applies:

- The effective period of the existing cross-reference record does not change.
- The existing cross-reference data changes.
- The timeline action is set to 1.

Rule 13 ensures the following MDM Hub behavior:

- The existing version of the cross-reference record remains unchanged.
- The data in the existing version of the cross-reference record is updated.

The following image shows data update for an existing record version:

Before	Initial Value
The Change	Updated Value
After	Updated Value

Rule 14. Update Effective Period

You can update the effective period of a record version.

If the following conditions exist, rule 14 applies:

- The effective period of the existing cross-reference record is updated to either increase or decrease the effective period.
- The data in the existing cross-reference record is unchanged.
- The timeline action is set to 2.

Rule 14 ensures the following MDM Hub behavior:

- The existing version of the cross-reference record is updated.
- The existing version of the cross-reference record has the effective period that you specify with no change to data.
- If the base object can have noncontiguous effective periods and you increase the effective period of a record version, the effective period of the prior adjacent and the later adjacent record versions decrease. To avoid overlapping of record versions, the MDM Hub decreases the effective period of the adjacent record version.
- If the base object can have noncontiguous effective periods and you decrease the effective period of a record version, the MDM Hub introduces a gap between record versions.
- If the base object must have contiguous effective periods, the MDM Hub extends or decreases the effective period of the adjacent record to maintain contiguity.

The following image shows an update to the effective period when contiguity is enabled for the base object:

Before	Initial Value 1	Initial Value 2
The Change	Updated Value 1	
After	Updated Value 1	Updated Value 2

Rule 15. Add an Effective Period

You can add a record version with a new effective period.

If the following conditions exist, rule 15 applies:

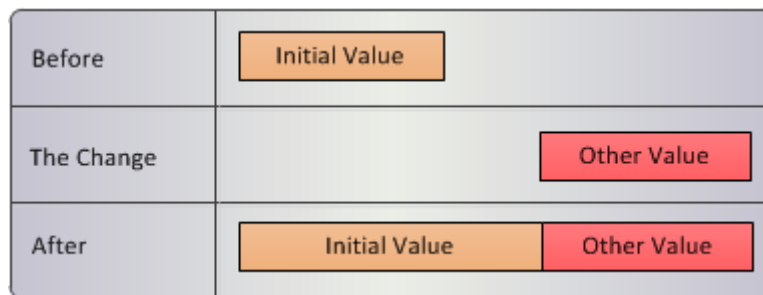
- A record version with a new effective period is added to the cross-reference table.

- The specified effective period creates a gap between effective periods.
- The base object property setting does not allow noncontiguous effective periods.
- The timeline action is set to 4.

Rule 15 ensures the following MDM Hub behavior:

- The cross-reference record with the new effective period is loaded.
- If you want to fill gaps when contiguity is broken, the effective period of the existing record version is extended to fill the gap between record versions.
- If you do not want to fill gaps when contiguity is broken, the new record version is not inserted and an error is generated. During a batch load, the record is moved to the reject table.

The following image shows the addition of a record to the cross-reference table when contiguity is enabled for the associated base object:



Configuring the Base Object Timeline

To configure the timeline for records in a base object, enable the timeline in the MDM Hub and the Hub Server properties file.

1. Enable the timeline for a base object in the Hub Console.
2. Configure the MDM Hub properties file.

Step 1. Enable the Timeline in the Hub Console

Enable the timeline for each base object that requires a timeline.

1. Open the Model workbench and click **Schema**.
2. Acquire a write lock.
3. In the Schema tree, select a base object.
The Schema Manager displays the Basic tab of the Base Object Properties page.
4. From the Timeline list, select **Dynamic Timeline**.
5. Configure the effective periods.
 - If you need contiguous effective periods for records, disable the **Allow non-contiguous effective periods** check box.
 - If you need noncontiguous effective periods for records, enable the **Allow non-contiguous effective periods** check box.

6. Click **Save**.

The MDM Hub enables the timeline for the base object.

Step 2. Configure the Properties File

If you enable the timeline for base objects, configure the maximum number of records to use during the best version of the truth (BVT) calculation. When you specify effective dates for BVT calculation, the SearchQuery, SearchHMQQuery, GetOneHop, and GetEntityGraph SIF APIs use the maximum number of records that you configure during the calculation.

1. Open the `cmxserver.properties` file in the following directory:

On UNIX. `<infamdm installation directory>/hub/server/resources`

On Windows. `<infamdm installation directory>\hub\server\resources`

2. Add the following properties:

```
searchQuery.buildBvtTemp.MaxRowCount  
sif.search.result.querytemptableTimeToLive.seconds
```

The default for `searchQuery.buildBvtTemp.MaxRowCount` is 10000. The default for `sif.search.result.querytemptableTimeToLive.seconds` is 30.

3. Save and close the file.

Load Multiple Versions of a Record In a Batch Job

You can load multiple versions of a record from a staging table to a cross-reference table that is associated with a base object in one batch job. Ensure that you enable the base object property for contiguity of effective periods of records.

When the MDM Hub loads multiple record versions, it rejects record versions that break the contiguity of effective periods of the record. To load multiple record versions without affecting the contiguity of effective periods of the records, configure the MDM Hub to load the record versions in a sequence that maintains contiguity.

Set Up Load Batch to Load Multiple Versions of a Record

Configure the MDM Hub to load multiple versions of records in a single load batch job.

1. Configure contiguous effective periods for the base object records.

- a. Open the Model workbench and click **Schema**.

- b. Acquire a write lock.

- c. In the Schema tree, select a base object.

The Schema Manager displays the Basic tab of the Base Object Properties page.

- d. Disable the **Allow non-contiguous effective periods** check box.

2. Configure sequencing of record versions to load to base objects.
 - a. Open the `cmxserver.properties` file in the following directory:
 On UNIX. `<infamdm installation directory>/hub/server/resources`
 On Windows. `<infamdm installation directory>\hub\server\resources`
 - b. Add the following property:

```
cmx.server.batch.load.smart_resequencing = true
```
 - c. Save and close the file.

Batch Load of Multiple Record Versions Example

Your organization has the CUSTOMER base object for which you track data change events. The CUSTOMER base object contains a record for John Smith who lives in New York effective 21 March 2014 and will be in New York until 15 January 2015. You want to load record versions for other effective periods in the past and in the future.

You want to load records with the following effective periods from the staging table into the cross-reference table associated with the CUSTOMER base object:

- 31 January 2011 to 20 October 2013
- 21 October 2013 to 20 March 2014
- 16 January 2015 to 24 November 2015
- 25 November 2015 to null

To load multiple versions of records in a single load batch job, configure contiguity of effective periods of records in the base object. Also, you enable the MDM Hub to load record versions in a sequence that maintains contiguity with the existing record versions.

Before the load batch job, the cross-reference has one record for John Smith that shows that he lives in New York.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period	Start date	Period	End Date
1		2		25		John	Smith	New York	March 21, 2014		January 15, 2015	

After the load batch job, the cross-reference table has multiple, contiguous record versions for John Smith.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period	Start date	Period	End Date
3		2		25		John	Smith	Los Angeles	January 31, 2011		October 20, 2013	
2		2		25		John	Smith	San Francisco	October 21, 2013		March 20, 2014	
1		2		25		John	Smith	New York	March 21, 2014		January 15, 2015	
4		2		25		John	Smith	Austin	January 16, 2015		November 24, 2015	
5		2		25		John	Smith	Las Vegas	November 25, 2015		null	

The record shows that John Smith lived at Los Angeles and San Francisco, lives at New York, and will live in Austin and Las Vegas after the multiple versions of the record are loaded.

Edit the Effective Period of a Record Version

You can edit the effective period of a record version. You edit the effective period value if you saved an incorrect value for a record version. You can change the effective period of a record by changing the start date and end date of a record version.

When you edit the effective period of a record version to start later or to end earlier than the date you specified, you decrease the effective period. When you edit the effective period of a record version to start earlier or to end later than the date you specified, you increase the effective period.

You can edit the effective period of a record version when contiguity of records is enabled in the associated base object. The MDM Hub extends or decreases the effective period of any record version that is adjacent to the record version that you edit to maintain contiguity. If noncontiguous effective periods can exist in a base object, the record versions that are adjacent are affected when you extend the effective period.

To edit effective periods of record versions during a load batch job, perform the following settings:

- **TIMELINE_ACTION.** Set the value of the **TIMELINE_ACTION** staging table column to 2. When you set the property to 2, the MDM Hub can edit the effective period of a record version. The **TIMELINE_ACTION** column is mapped from the corresponding landing table column.
- **TIMELINE_FILL_ON_GAP.** Ensures that contiguity between the effective dates of record versions is maintained when you edit the effective periods of record versions. Set the property in the **C_REPOS_TABLE** of the Operational Reference Store or in the Staging Table properties of the Hub Console. If set to `true`, when you can add a new record version to the base object, the MDM Hub maintains the contiguity between effective periods of record versions. If set to `false`, the MDM Hub rejects any addition of record version that breaks the contiguity between effective periods of record versions. The default is `false`.

Note: You can effect edits to effective periods through data from a state management override system or through the source system in which the record originated.

Increase the Effective Period of a Record Version

You can edit the end date or start date to increase the effective period of a record version. When the base object record versions are contiguous, the MDM Hub decreases the effective period of the record version that is adjacent to the record version you edit.

You can increase the effective period when you extend the end date. After you extend the end date, the record version that you edit can overlap with an adjacent record version. The overlap is with a record version that is after the record version that you edit. The MDM Hub extends the start date of the adjacent record version. To ensure contiguity without the overlapping of record versions, the effective period of the adjacent record version decreases.

You can increase the effective period when you move the start date to an earlier date. After you move the start date, the record version can overlap with an adjacent record version. The overlap is with a record version that is before the record version that you edit. The MDM Hub moves the end date of the adjacent record version to an earlier date. To ensure contiguity without the overlapping of record versions, the effective period of the adjacent record version decreases.

If you do not enable contiguity of records in base objects, the adjacent record versions remain unchanged unless there is a record version overlap.

Increase Effective Period Example

Your organization has the **CUSTOMER** base object for which you track data change events. The cross-reference table associated with the **CUSTOMER** base object contains a record for John Smith who lives in

New York effective 21 March 2014 to 30 November 2014. The cross-reference table also contains another version of the record that is effective from 1 December 2014. You want to edit the record version that is effective from 21 March 2014 to 30 November 2014 to be effective up to 28 February 2015.

Before you edit the first record version, the second record version for John Smith is effective from 1 December 2014.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	New York	March 21, 2014	November 30, 2014
2	2	25	John	Smith	Los Angeles	December 1, 2014	null

When you edit the effective period of the record version to end on 28 February 2015 instead of 30 November 2014, the effective period is extended. The start date of the adjacent record version changes from 1 December 2014 to 1 March 2015 to avoid overlapping of record versions.

After you extend the start date of the first record version, the MDM Hub updates the second record version to start after the first version ends.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	New York	March 21, 2014	February 28, 2015
2	2	25	John	Smith	Los Angeles	March 1, 2015	null

Decrease the Effective Period of a Record Version

You can edit the start date or the end date to decrease the effective period of a record version. When the base object record versions are contiguous, the MDM Hub extends the effective period of the record version that is adjacent to the record version that you edit.

You can decrease the effective period when you extend the start date. If you have a record version before and adjacent to the record version that you edit, a gap is created between the record versions. The MDM Hub extends the end date of the record version with which the gap is created to maintain contiguity.

You can decrease the effective period by moving the end date to an earlier date. If you have a record version after and adjacent to the record version that you edit, a gap is created between the record versions. To ensure contiguity, the MDM Hub moves the start date of the adjacent record version to an earlier date.

If you do not enable contiguity of records in base objects, the adjacent record versions remain unchanged.

Decrease Effective Period Example

Your organization has the CUSTOMER base object for which you track data change events. The CUSTOMER base object contains a record for John Smith who lives in New York effective 21 March 2014. You want to maintain contiguity of effective periods. You load a record version that is effective from 1 December 2014. Finally, you want to edit the record version that is effective from 1 December 2014 to be effective from 1 March 2015.

The cross-reference record for John Smith shows that he lives in New York from 21 March 2014.

Rowid	XREF	Rowid	Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1		2		25	John	Smith	New York	March 21, 2014	null

When you load the record version that is effective from 1 December 2014, the existing record version ends on 30 November 2014. The effective period of the existing record version decreases to avoid overlap and maintain contiguity.

After you load the record version that is effective from 1 December 2014, the cross-reference table shows that the existing record version for John Smith ends on 30 November 2014.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	New York	March 21, 2014	November 30, 2014
2	2	25	John	Smith	Los Angeles	December 1, 2014	null

When you edit the record version that is effective from 1 December 2014 to be effective from 1 March 2015, the effective period decreases. Also, a gap is introduced between effective periods of the two record versions. The MDM Hub extends the end date of the record version that is adjacent to the version that you edit. The end date changes from 30 November 2014 to 28 February 2015 to fill the gap.

After you edit the effective date of the record version from 1 December 2014 to 1 March 2015.

Rowid XREF	Rowid Object	Customer ID	First Name	Last Name	City	Period Start date	Period End Date
1	2	25	John	Smith	New York	March 21, 2014	February 28, 2015
2	2	25	John	Smith	Los Angeles	March 1, 2015	null

Add a Record Version

You can add a version of a record that is effective for a new period.

If you enable contiguity of records in the base object, the MDM Hub ensures that the record versions have contiguous effective dates. The MDM Hub extends the effective period of the record version that is adjacent to the record version that you add to maintain contiguity.

Configure the staging table column to add new record versions and C_REPOS_TABLE column to ensure contiguity of record versions.

To add record versions during a load batch job, configure the following settings:

- **TIMELINE_ACTION.** Set the value of the TIMELINE_ACTION staging table column to 4. When you set the property to 4, the MDM Hub can add new record versions. The TIMELINE_ACTION column is mapped from the corresponding landing table column.
- **TIMELINE_FILL_ON_GAP.** Ensures that contiguity between the effective dates of record versions is maintained when you add new record versions. Set the property in the C_REPOS_TABLE of the Operational Reference Store or in the Staging Table properties of the Hub Console. If set to `true`, when you can add a new record version to the base object, the MDM Hub maintains the contiguity between effective periods of record versions. If set to `false`, the MDM Hub rejects any addition of record version that breaks the contiguity between effective periods of record versions. The default is `false`.

Add a Record Version Example

Your organization has the CUSTOMER base object for which you track data change events. The CUSTOMER base object contains a record for John Smith who lives in New York effective 21 March 2014. You want to maintain contiguity of effective periods. You load a record version that is effective from 1 December 2014.

Before you load a new record version, the record for John Smith shows that he lives in New York from 21 March 2014.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period Start date	Period End Date
1		2		25		John	Smith	New York	March 21, 2014	null

When you load the record version that is effective from 1 December 2014, the effective period of the adjacent record version decreases. The MDM Hub adjusts the effective end date of the existing record to avoid the overlap of record versions and to maintain contiguity.

After you load the record version that is effective from 1 December 2014, the cross-reference table shows that the existing record version for John Smith ends on 30 November 2014.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period Start date	Period End Date
1		2		25		John	Smith	New York	March 21, 2014	November 30, 2014
2		2		25		John	Smith	Los Angeles	December 1, 2014	null

Update Data in a Record

When you update the data in a record, the MDM Hub changes the data in the record. It does not create a new record version.

Update Record Data Example

Your organization has the CUSTOMER base object for which you track data change events. The CUSTOMER base object contains a record for John Smith who lives in New York effective 21 March 2014 to 15 January 2015. You want to update the city that he lives in to Newark.

Before the data update, the cross-reference record for John Smith shows that he lives in New York.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period Start date	Period End Date
1		2		25		John	Smith	New York	March 21, 2014	January 15, 2015

After the data update, the cross-reference record for John Smith shows that he lives in Newark.

Rowid	XREF	Rowid	Object	Customer	ID	First Name	Last Name	City	Period Start date	Period End Date
1		2		25		John	Smith	Newark	March 21, 2014	January 15, 2015

The MDM Hub updates the existing record version with the correct city name, Newark. The effective period for the record of John Smith does not change.

Update a Relationship

When you update relationship details in a relationship base object record, the MDM Hub either inserts or updates a record version.

If you update relationship details in the custom columns of a relationship base object record, the MDM Hub updates the cross-reference record associated with the relationship record.

If you update relationship details in the system columns of a relationship base object record, the MDM Hub inserts new record versions to the associated cross-reference table. System columns are columns such as Relationship Type, Hierarchy Type, Period Start Date, and Period End Date. Also, if you update a relationship from a source system that does not have a corresponding cross-reference record, the MDM Hub adds a new record version to the cross-reference table.

Update a Custom Column of a Relationship Record Example

Your organization has the PARTY base object that contains the record for Mary Adam and the record for the ABC organization. The relationship between Mary Adam and the ABC organization is defined in the associated PARTY REL relationship base object. You update the value in the Rel Desc column, which is a custom column, from `Individual` to `Organization`.

Note: You configured Informatica Data Director to use the IDD source system and the updates are performed in the Hierarchy view.

When Updates Are from the SFDC Source System

When the update to the Rel Desc column is from a SFDC source system record, the MDM Hub updates the RL PARTY CROSS-REFERENCE table in the following ways:

- The MDM Hub does not change the existing cross-reference record.
- The MDM Hub inserts a new cross-reference record with no change to the start date and end date.
- The MDM Hub updates the values of the custom column and the Rowid System column in the new cross-reference record with values from the contributing source system.

The RL PARTY CROSS-REFERENCE table shows a record for which you want to change the value of the Rel Desc custom column.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Rel Desc	Period	Start Date	Period	End Date
SFDC		414722		2402007		2402147		Individual	null			null

After you change the value of the Rel Desc column from `Individual` to `Organization`, the RL PARTY CROSS-REFERENCE table shows the existing record and a new one.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Rel Desc	Period	Start Date	Period	End Date
SFDC		414722		2402007		2402147		Individual	null			null
IDD		414722		2402007		2402147		Organization	null			null

When Updates Are from the IDD Source System

When the update to the Rel Desc column is from a record that is from the IDD source system, the MDM Hub updates the RL PARTY CROSS-REFERENCE table in the following ways:

- The MDM Hub updates the value of the custom column in the existing cross-reference record.
- The MDM Hub does not change the start date and end date of the cross-reference record.

The RL PARTY CROSS-REFERENCE table shows the record for which you want to change the value of the Rel Desc custom column.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Rel Desc	Period	Start Date	Period	End Date
IDD		414722		2402007		2402147		Individual	null			null

After you change the value of the Rel Desc column from Individual to Organization, the RL PARTY CROSS-REFERENCE table shows the updated record.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Rel Desc	Period	Start Date	Period	End Date
IDD		414722		2402007		2402147		Organization	null			null

Update a System Column of a Relationship Record Example

Your organization has the PARTY base object that contains the record for Mary Adam, and the record for the ABC organization. The relationship between Mary Adam and the ABC organization is defined in the associated PARTY REL relationship base object. You update the period start date and period end date.

Note: You configured Informatica Data Director to use the IDD source system and the updates are performed in the Hierarchy view.

You update the period start date from null to 1 January 2016 and period end date from null to 1 January 2017.

When Updates Are from the SFDC Source System

When the update to the period start date and the period end date is from an SFDC source system record, the MDM Hub updates the RL PARTY CROSS-REFERENCE table in the following ways:

- The MDM Hub does not change the existing cross-reference records.
- The MDM Hub inserts a new cross-reference record with the start date set to 1 January 2016 and the end date set to 1 January 2017.

The RL PARTY CROSS-REFERENCE table shows an active record for which you want to change the period start date and period end date.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Hub State Indicator	Rel Desc	Period	Start Date	Period	End Date
SFDC		414722		2402007		2402147		Active	Individual	null			null

After you change the period start date and period end date, the RL PARTY CROSS-REFERENCE table shows the existing record and a new record.

Rowid	System	Rowid	Object	Party	ID1	Party	ID2	Hub State Indicator	Rel Desc	Period	Start Date	Period	End Date
SFDC		414722		2402007		2402147		Active	Individual	null			null
IDD		414722		2402007		2402147		Active	Individual	1 January 2016		1 January 2017	

When Updates Are from the IDD Source System

When the update to the period start date and the period end date is from the IDD source system record, the MDM Hub updates the RL PARTY CROSS-REFERENCE table in the following ways:

- The MDM Hub does not change the existing cross-reference records.
- The MDM Hub inserts a new cross-reference record with the end date set to 31 December 2015.

- The MDM Hub inserts a new cross-reference record with the start date set to 1 January 2016 and the end date set to 1 January 2017.
- The MDM Hub inserts a new cross-reference record with the start date set to 2 January 2017 and the end date set to null.

The RL PARTY CROSS-REFERENCE table shows active records for which you want to change the period start date and period end date.

Rowid System	Rowid Object	Party	ID1Party	ID2Hub Indicator	State	Rel Desc	Period Start Date	Period End Date
IDD	414722	2402007	2402147	Active		Individual	null	null

After you change the period start date and period end date, the RL PARTY CROSS-REFERENCE table shows the existing record and three new records.

Rowid System	Rowid Object	Party	ID1Party	ID2Hub Indicator	State	Rel Desc	Period Start Date	Period End Date
IDD	414722	2402007	2402147	Active		Individual	null	null
IDD	414722	2402007	2402147	Active		Individual	null	31 December 2015
IDD	414722	2402007	2402147	Active		Individual	1 January 2016	1 January 2017
IDD	414722	2402007	2402147	Active		Individual	2 January 2017	null

End a Relationship

You can end a relationship between two records. When you end a relationship, the MDM Hub inserts a new record version in the cross-reference table associated with the relationship base object. Also, the MDM Hub sets the deleted indicator for the record to -999999999 and updates the hub state indicator to deleted. The period start date of the new record version is set to the date on which you end the relationship.

If the record belongs to a state management override system, the period start date of the new record version is set to the date on which you end the relationship incremented by one timeline unit. For example, if the period end date is January 31, 2014, and the timeline granularity is one day, the period start date of the state management override system record version is February 1, 2014.

End a Relationship Example

Your organization has the PARTY base object that contains a record for Mary Adam. The PARTY GROUP base object contains the details of the Adam household to which Mary belongs. The relationship between the PARTY and the PARTY GROUP base object is defined in the associated relationship base object PARTY GROUP REL. On May 17, 2015, Mary informs you that she is not a part of the Adam household anymore. You need to update Mary's record to end the relationship with the Adam household.

You set an end date for Mary's relationship with the Adam household. When you set an end date for Mary's relationship record, the MDM Hub adds a record in the deleted state to the cross-reference table associated with the relationship base object, RL PARTY GROUP CROSS-REFERENCE.

The following changes occur in the cross-reference table, RL PARTY GROUP CROSS-REFERENCE, that is associated with the PARTY GROUP REL relationship base object:

- A record is inserted for the relationship between Mary and the Adam household.
- The record is effective from May 18, 2015.

- The deleted indicator is set to -999999999.
- The hub state indicator is set to deleted.
- The end date of the original record is updated to May 17, 2015.

Before you end Mary's relationship with the Adam household, the cross-reference table shows a record that is effective from October 6, 2006 to December 31, 2999.

Rowid XREF	Rowid Object	Party ID	Deleted Indicator	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63	22	147		Active	28	October 6, 2006	December 31, 2999

After you end Mary's relationship with the Adam household, the cross-reference table shows that starting May 18, 2015, Mary is not a member of the Adam household.

Rowid XREF	Rowid Object	Party ID	Deleted Indicator	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63	22	147		Active	28	October 6, 2006	May 17, 2015
621	22	147	-999999999	Deleted	28	May 18, 2015	

Delete a Relationship Period

You can delete a relationship record version that is effective for a specific period. When you delete a relationship record version, the MDM Hub inserts a new record version in the cross-reference table associated with the relationship base object. The effective period of the record version is the same as the effective period of the relationship record version that you delete. Also, the MDM Hub sets the deleted indicator for the record version to -999999999, and the hub state indicator to deleted.

Delete a Relationship Example

Your organization has the PARTY base object that contains a record for Mary Adam. The PARTY GROUP base object contains the details of the Adam household to which Mary belongs. The relationship between the PARTY and the PARTY GROUP base object is defined in the associated PARTY GROUP REL relationship base object. You need to delete the relationship between Mary and the Adam household that is effective from October 6, 2006.

You want to delete a version of Mary's relationship record that is incorrect. When you delete Mary's relationship record, the MDM Hub adds a record in the deleted state to the cross-reference table associated with the PARTY GROUP REL relationship base object.

The following changes occur in the cross-reference table, RL PARTY GROUP CROSS-REFERENCE, that is associated with the PARTY GROUP REL relationship base object:

- A record is inserted for the relationship between Mary and the Adam household without any change to the effective period.
- The deleted indicator of the new record is -999999999.
- The hub state indicator of the new record is deleted.
- The original record does not change.

Before you delete the relationship between Mary and the Adam household, the cross-reference table shows that the relationship is active.

Rowid XREF	Rowid Object	Party ID	Deleted Indicator	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63	22	147		Active	28	6 October 2006	31 December 2999

After you delete the relationship between Mary and the Adam household, the cross-reference table shows a new record with the relationship deleted.

Rowid XREF	Rowid Object	Party ID	Deleted Indicator	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63	22	147		Active	28	6 October 2006	31 December 2999
621	22	147	-999999999	Deleted	28	6 October 2006	31 December 2999

The relationship base object shows that the relationship between Mary and the Adam household is deleted.

Rowid	Object	Party ID	Deleted Indicator	Hub State Indicator	Party Group ID
22		147	-999999999	-1	28

Delete All Relationship Periods

If all the relationship periods of a relationship record are incorrect, you can delete all the relationship periods. When you delete all the relationship periods, the MDM Hub changes the hub state indicator of the record versions to the deleted state in the cross-reference table associated with the PARTY GROUP REL relationship base object. Also, the MDM Hub sets the hub state indicator and the deleted indicator of the record in the PARTY GROUP REL relationship base object to deleted.

Delete All Relationship Periods Example

Your organization has the PARTY base object that contains a record for Mary Adam. The PARTY GROUP base object contains the details of the Adam household to which Mary belongs. The relationship between the PARTY and the PARTY GROUP base object is defined in the associated PARTY GROUP REL relationship base object. You need to delete the relationship between Mary and the Adam household.

You want to delete Mary's relationship record because her relationship with the Adam household is incorrect. You must delete all relationship record versions that are effective for different periods. When you delete Mary's relationship records for all effective periods, the MDM Hub changes the hub state of the record versions to the deleted state in the cross-reference table associated with the PARTY GROUP REL relationship base object.

When you delete all relationship periods, the hub state indicator of all the original record versions is set to deleted in the cross-reference table, RL PARTY GROUP CROSS-REFERENCE, that is associated with the PARTY GROUP REL relationship base object.

Before you delete all the relationship periods, the cross-reference table shows that all the effective relationships between Mary and the Adam household are active.

Rowid	XREF	Rowid Object	Party ID	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63		22	147	Active	28	October 6, 2006	December 31, 2015
66		22	147	Active	28	January 1, 2016	December 31, 2999

After you delete all the relationship periods, the cross-reference table shows that all relationships between Mary and the Adam household are deleted.

Rowid	XREFRowid	Party ID	Hub State Indicator	Party Group ID	Period Start Date	Period End Date
63	22	147	Deleted	28	October 6, 2006	December 31, 2015
66	22	147	Deleted	28	January 1, 2016	December 31, 2999

The relationship base object shows that the relationship between Mary and the Adam household is deleted.

Rowid	Object	Party ID	Hub State Indicator	Party Group ID
22		147	-1	28

Using Timeline Extract

You can run Timeline Extract by the following methods:

- Run the Extract BVT Versions batch job in the Batch Viewer tool in the MDM Hub Console
- Run the executeBatchExtractBVTVersions SIF API

After you run the Extract BVT Versions batch job, the extracted records are added to the following table:

- E\$_BO_NAME

Run the Extract BVT Versions batch job

For the effected base object, run the Extract BVT Versions batch job in the Batch Viewer tool in the MDM Hub Console.

Note: You cannot specify a limit date when you run the Extract BVT Versions batch job in the Batch Viewer tool. The Extract BVT Versions batch job uses the current application server time for the limit date. To specify a limit date, run the executeBatchExtractBVTVersions SIF API.

For information about the Batch Viewer tool, see [“Running Batch Jobs Using the Batch Viewer Tool” on page 535](#).

Run the executeBatchExtractBVTVersions SIF API

Request

The executeBatchExtractBVTVersions request contains the following parameters:

orsId

The Operational Reference Store name

tableName

The base object name

limitDate

The Extract BVT Version batch job operates on cross-reference records whose last update date is before the limit date.

Response

The executeBatchExtractBVTVersions response returns the following parameters:

Message

Contains a message regarding the status of the request.

RetCode

Contains the return code.

EJB Request Example

The following EJB request runs the Extract BVT Versions batch job:

```
SiperianClient sipClient = SiperianClient.newSiperianClient(new
File( context.getTestPTTStartDir() + "siperian-client.properties" ) );
ExecuteBatchExtractBVTVersionsRequest req = new ExecuteBatchExtractBVTVersionsRequest();
req.setTableName(jobContext.getTableName()); // Pass BO name as a string
req.setLimitDate(jobContext.getLimitDate()); // Pass limit date using java Date type
ExecuteBatchExtractBVTVersionsResponse executed =
(ExecuteBatchExtractBVTVersionsResponse) sipClient.process( req );
String errMessage = executed.getMessage();
int rc = executed.getRetCode();
```

SOAP Request Example

The following SOAP request runs the Extract BVT Versions batch job for the C_TIMELINE base object in the CMX_ORIS10A Operational Reference Store:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:siperian.api">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:executeBatchExtractBVTVersions>
      <urn:username>admin</urn:username>
      <urn:password>
        <urn:password>admin</urn:password>
        <urn:encrypted>false</urn:encrypted>
      </urn:password>
      <urn:orsId>localhost-orcl-CMX_ORIS10A</urn:orsId>
      <urn:tableName>C_TIMELINE</urn:tableName>
      <urn:limitDate>2015-10-29T12:59:14+02:00</urn:limitDate>
    </urn:executeBatchExtractBVTVersions>
  </soapenv:Body>
</soapenv:Envelope>
```

Configure Timeline Extract Properties

If you use the Timeline Extract feature, you can configure optional properties in the `cmxserver.properties` file.

- Add the following optional properties in the `cmxserver.properties` file:

Property	Description
<code>cmx.server.batch.extractbvtversions.removePreviousExtractRecords</code>	Specifies whether to keep the latest snapshot to prevent excessive disk space usage. To prevent excessive disk space usage, set to <code>true</code> . Default is <code>false</code> .
<code>cmx.server.batch.extractbvtversions.dropOutOfSyncBOExtractTable</code>	Specifies whether to prevent base object column changes from causing the existing extract table to be dropped. To prevent the extract table to be dropped, set to <code>false</code> . Default is <code>true</code> .
<code>cmx.server.batch.extractbvtversions.maxJobRejectExtractToKeep</code>	Specifies the number of jobs for which you want to retain the history of rejected records. Default is 10.

CHAPTER 11

State Management and BPM Workflow Tools

This chapter includes the following topics:

- [State Management and BPM Workflow Tools Overview, 169](#)
- [State Management in the MDM Hub, 170](#)
- [BPM Workflow Tools, 174](#)
- [Configure the MDM Hub for ActiveVOS, 174](#)
- [Enabling Match on Pending Records, 179](#)
- [Message Triggers for State Transitions, 179](#)
- [Record Promotion, 180](#)

State Management and BPM Workflow Tools Overview

You can ensure that updated entity data goes through a change-approval workflow before the updated records contribute the Best Version of the Truth (BVT) records. For example, a business process might require that a senior manager review and approve updates to customer data before it becomes master data.

To support a change-approval workflow, the MDM Hub and Data Director (IDD) integrate with the ActiveVOS® Server. Predefined MDM workflows, task types, and roles enable the components to synchronize with one another.

The components support a change-approval workflow in the following ways:

- The MDM Hub manages a state on the records in the base object tables that have state management enabled. Unapproved records have a pending state, while approved records have an active state.
- In IDD applications, authorized business users change entity data and can send updates for approval.
- The ActiveVOS Server runs activities within the MDM workflows and creates tasks for business managers and data stewards.

Example

Your business process requires that a senior manager review and approve updates to customer data before it becomes master data. The IDD application and the MDM environment are configured to use the ActiveVOS Server as the default workflow engine.

The predefined One Step Approval Workflow defines a business process that requires one senior manager to review and approve updates. The following steps summarize what happens when a data steward initiates the One Step Approval Workflow:

1. In an IDD application, a data steward updates a customer entity and sends the update for approval.
2. In the MDM Operational Reference Store, the records enter a pending state.
3. The ActiveVOS Server begins running activities in its One Step Approval Workflow process.
4. When the ActiveVOS Server reaches the Final Review people activity, the server creates a task for senior managers.
5. In the IDD application, all senior managers receive the task in their Task Inboxes. The task links to the updated customer entity.
6. If a senior manager approves the update, the following events occur:
 - In IDD, the task is completed.
 - The ActiveVOS Server marks the Final Review people activity as complete and runs the next activity in the One Step Approval Workflow process.
 - In the MDM Operational Reference Store, the approved records enter an active state.
 - In the MDM Hub, the approved records contribute to the BVT records.

State Management in the MDM Hub

State management is the process of assigning and changing the state associated with a record in the MDM Hub. In the MDM Hub, you can enable state management on base object tables. All records in the state-enabled base object tables and its cross-reference tables are assigned a default state.

The following activities can change the state of a record:

- A business user completes a task in a BPM workflow tool. A completed approval task may trigger a change in the state of associated records.
- A data steward promotes, deletes, or restores a set of records using the tools in the Hub Console. For more information, see the *Multidomain MDM Data Steward Guide*.
- An application uses the SiperianClient API to promote, delete, or restore a set of records. For more information, see the *Multidomain MDM Services Integration Framework Guide*.

Record States

The predefined MDM Hub record states are ACTIVE, PENDING, and DELETED.

Note: By default, only approved records in the ACTIVE state contribute to the Best Version of the Truth (BVT) record. You can modify operations to allow matches to and from ACTIVE and PENDING records.

The following table describes the record states:

Record State	Description
ACTIVE	<p>Default. A record in the ACTIVE state is reviewed and approved. If records must go through a BPM approval workflow, the active records have been through that process and are approved.</p> <p>The following qualities apply to active records:</p> <ul style="list-style-type: none"> - All MDM Hub processes include active records by default. - A base object record is active if at least one of its cross-reference records is in the ACTIVE state. - Only active cross-reference records contribute to the consolidated base object. - In tables, the HUB_STATE_IND system column displays a 1 for records in the ACTIVE state.
PENDING	<p>A record in the PENDING state is awaiting a review. If records must go through a BPM approval workflow, the review tasks associated with the pending records are still open. You cannot edit a record which is participating in a workflow and pending approval.</p> <p>The following qualities apply to active records:</p> <ul style="list-style-type: none"> - Records in the PENDING state are not approved for general use by the MDM Hub. - The MDM Hub processes exclude pending records by default. You can perform most operations on pending records, but you have to request the pending records. - If there are only pending cross-reference records, the MDM Hub determines the BVT on the base object through trust on the PENDING records. - If you delete a record that is in the PENDING state, the record is removed from the table, does not enter the DELETED state, and cannot be restored. - In tables, the HUB_STATE_IND system column displays a 0 for records in the PENDING state.
DELETED	<p>A record in the DELETED state is no longer part of the MDM Hub data.</p> <p>The following qualities apply to active records:</p> <ul style="list-style-type: none"> - The MDM Hub processes exclude deleted records by default. To include deleted records, you have to request them. - You can restore a record in the DELETED state. - In tables, the HUB_STATE_IND system column displays a -1 for records in the DELETED state.

Hub State Indicator

State-enabled base object tables and cross-reference tables have a system column, HUB_STATE_IND, that uses a value to represent the record state for the records in the table.

The following table lists the possible values for the HUB_STATE_IND and maps the values to the record states:

HUB_STATE_IND Value	Record State
1	ACTIVE
0	PENDING
-1	DELETED
-9	Used when cross-reference history is enabled after a hard delete. Records with HUB_STATE_IND of -9 are written to the cross-reference history table (HXRF) when cross-references are deleted. Similarly, if base object records are hard-deleted, then records with HUB_STATE_IND of -9 are added to the history table (HIST) for the deleted base objects.

State Transitions

State transition rules determine whether and when a record can change from one state to another.

The following table describes the common transitions and notes any differences between base object records and cross-reference records:

From State	To State	Notes
ACTIVE	DELETED	This transition is called a soft delete.
PENDING	ACTIVE	This transition is called promote.
DELETED	ACTIVE	This transition is called restore. <i>A cross-reference record can be restored. A base object record can be restored only if cross-reference records are restored.</i>
DELETED	PENDING	You cannot change a deleted record to a pending record. This transition happens indirectly only. If you delete an active base object record and then add a pending cross-reference record for the deleted record, the base object record state goes from ACTIVE to DELETED to PENDING.

Note: The following state transitions are not valid:

- ACTIVE to PENDING. An active record cannot change to a pending record.
- PENDING to DELETED. You can delete a pending record, but it does not transition to the DELETED state. Instead, the MDM Hub removes the record from the database. This is called a hard delete. You cannot restore a hard-deleted record.

Protection of Pending Records

You can protect pending records from updates that are not part of the same process. The Interaction ID controls this setting.

When base object tables or cross-reference tables include Interaction IDs, you can not use the tools in the Hub Console to change the state of a base object record or cross-reference record from PENDING to ACTIVE. When cross-reference records include an Interaction ID, the Interaction ID protects a pending cross-reference record from updates that are not part of the same process as the original cross-reference record. Use one of the state management SIF API requests, instead.

Note: The Interaction ID can be specified through any API. However, it cannot be specified when performing batch processing. For example, records that are protected by an Interaction ID cannot be updated by the Load batch process.

The following table contains an example showing the effect of the Interaction ID field during the match process on existing and incoming records. In the example, the interaction_ID field has the following values: Version A , Version B, and null.

Incoming Record's Interaction ID	Existing Record's Interaction ID		
	Version A	Version B	Null
Version A	OK	Error	OK
Version B	Error	OK	OK
Null	Error	Error	OK

Rules for Loading Data

The load batch process loads records in any state. The state is specified as an input column on the staging table. The input state can be specified in the mapping as a landing table column or it can be derived. If an input state is not specified in the mapping, then the state is assumed to be ACTIVE (for Load inserts). When a record is updated through a Load batch job and the incoming state is null, the existing state of the record to update will remain unchanged.

The following table lists the state of the incoming cross-reference record in the first column, the state of the existing cross-reference record in the top row, and the resulting action in the cells:

XREF States	Existing: ACTIVE	Existing: PENDING	Existing: DELETED	Existing: No XREF (Load by rowid)	Existing: No Base Object Record
Incoming: ACTIVE	Update	Update + Promote	Update + Restore	Insert	Insert
Incoming: PENDING	Pending Update	Pending Update	Pending Update + Restore	Pending Insert	Pending Insert
Incoming: DELETED	Soft Delete	Hard Delete	Soft Delete	Not recommended	Not recommended
Incoming: Undefined	Treat as ACTIVE	Treat as PENDING	Treat as DELETED	Treat as ACTIVE	Treat as ACTIVE

Note: If history is enabled after a hard delete, then records with HUB_STATE_IND of -9 are written to the cross-reference history table (HXRF) when cross-references are deleted. Similarly, if base object records are physically deleted, then records with HUB_STATE_IND of -9 are added to the history table (HIST) for the deleted base objects.

Record States and Base Object Record Value Survivorship

After a merge, put, or load process, a base object record might have multiple cross-reference records with different record states. In this case, the values in the base object record reflect only the values from the active cross-reference records.

In general, when resolving multiple records with different states, the MDM Hub processes apply the following rules:

- Values in active records prevail over values in pending or deleted records.
- Values in pending records prevail over values in deleted records.

Note: For a match and merge operation, the source and target of the records to merge determines the surviving Rowid, not the record state.

BPM Workflow Tools

A business process management (BPM) tool helps you to automate business processes.

The default, predefined workflow engine is the licensed version of the ActiveVOS[®] Server that is included with Multidomain MDM. The installation process integrates this version of the ActiveVOS Server with the MDM Hub and Data Director, and deploys predefined MDM workflows, task types, and roles.

The Informatica ActiveVOS workflow engine supports the following adapters:

- An adapter for tasks that operate on business entities through business services. The adapter name is **BE ActiveVOS**.
- An adapter for tasks that operate on subject areas through SIF APIs. The adapter name is **Informatica ActiveVOS**.

You can also choose to integrate standalone instances of BPM tools:

Informatica ActiveVOS

If you run a standalone instance of Informatica ActiveVOS in your environment, you can manually integrate your instance with the MDM Hub and Data Director. You can deploy the predefined MDM workflows or create custom workflows. For more information, see the *Multidomain MDM Data Director - ActiveVOS Integration Guide*.

Third-party BPM tool

If you run a third-party instance in your environment, you can manually integrate your instance with the MDM Hub and Data Director. You can deploy the predefined MDM workflows or create custom workflows. For more information, see the *Multidomain MDM Business Process Manager Adapter SDK Implementation Guide*.

Important: Informatica recommends that you migrate to the business entity-based ActiveVOS workflow adapter. The Siperian workflow adapter is deprecated. Informatica will continue to support the deprecated adapter, but it will become obsolete and Informatica will drop support in a future release. The MDM Hub supports a primary workflow engine and a secondary workflow engine. You can migrate from the Siperian workflow adapter to the business entity-based ActiveVOS workflow adapter.

Configure the MDM Hub for ActiveVOS

To configure the MDM Hub to use the embedded ActiveVOS Server, you need to perform the following steps:

1. Enable state management on the base object tables in the Operational Reference Store.
2. Map the Operational Reference Store to the ActiveVOS workflow engine.

3. Assign predefined workflow user roles to business managers.

Enabling State Management

For each base object table that contains records that can be updated through an Data Director application, you need to enable state management.

1. In the Model workbench, click **Schema**.
2. Acquire a write lock.
3. In the Schema tool, select a base object table that contains records that can be updated through the application.
4. Click the **Advanced** tab.
5. Select the **Enable State Management** check box.
6. To keep track of when cross-reference records belonging to the base object change state from PENDING to ACTIVE, select the **History of Cross-Reference Promotion** check box.

Adding a Workflow Engine

When you add a workflow engine to the MDM Hub, you associate the workflow engine with a workflow adapter.

When you add a workflow engine, it becomes the primary workflow engine and the existing primary workflow engine becomes the secondary workflow engine. If you have an existing secondary workflow engine, the workflow engine is dropped from the Operational Reference Store and the tasks are removed from the task inbox.

1. In the Configuration workbench, click **Workflow Manager**.
2. Acquire a write lock.
3. Select the **Workflow Engines** tab and click the **Add** button.
4. In the **Add Workflow** dialog box, enter the workflow engine properties.

The following table describes the workflow engine properties:

Field	Description
Workflow Engine	The display name of the workflow engine
Adapter Name	Select BE ActiveVOS for the ActiveVOS workflow adapter based on business entities.
Host	The host name of the Informatica ActiveVOS instance.
Port	The port name of the Informatica ActiveVOS instance.
Username	The user name of the trusted user.
Password	The password of the trusted user.
Protocol	The protocol for communication between the MDM Hub and ActiveVOS. The protocol can be http or https.

5. Click **OK**.

Setting the Primary and Secondary Workflow Adapters

To migrate to the BE ActiveVOS workflow adapter, select the BE ActiveVOS workflow engine as the primary workflow engine. The name of the workflow adapter for business entities is `BE ActiveVOS`. You can process existing tasks with the secondary workflow engine but you cannot create tasks. Do not select the same workflow engine for the primary and secondary workflow engines.

If you have not previously used Multidomain MDM with embedded ActiveVOS, select the `BE ActiveVOS` adapter as the primary workflow adapter. You do not need to select a secondary workflow adapter to process existing tasks.

Note: If your Data Director application uses subject areas, continue to use the `Informatica ActiveVOS` adapter as the primary workflow engine.

When you add a workflow engine, it becomes the primary workflow engine and the existing primary workflow engine becomes the secondary workflow engine. If you have an existing secondary workflow engine, the workflow engine is dropped from the Operational Reference Store and the tasks are removed from the task inbox.

1. In the **Configuration** workbench, click **Workflow Manager**.
2. Acquire a write lock.
3. Select the **Workflow Engine** tab, and ensure that the following BE ActiveVOS workflow adapter information is correct:
 - ActiveVOS Server host
 - ActiveVOS Server port
 - username of trusted user
 - password of trusted user
 - protocol for communication between the MDM Hub and the ActiveVOS Server
4. Select the **Operational Reference Store Workflow Mapping** tab.
The table in the tab contains all the Operational Reference Store databases in the Hub Store.
5. In the **Primary Workflow Engine** column, select the workflow engine for the BE ActiveVOS workflow adapter.
6. In the **Secondary Workflow Engine** column, select a workflow engine.

Configure User Roles

To ensure that MDM can authenticate users with the ActiveVOS Server, the same user roles must exist in the MDM Hub, Data Director, and the ActiveVOS Server. You assign these user roles to users in the MDM Hub, and the Security Access Manager manages user access to all components.

You can assign multiple roles to users. Role privileges are cumulative, so users who have multiple roles receive the combined privileges associated with each role. For example, you might assign a user the Data Steward role, the `abAdmin` role, and the `abadmin` role. The user receives the combined workflow and task administration privileges associated with each role.

The following table lists the required user roles, identifies the workflow actions, and lists task administration actions that are associated with the role:

User Role	Workflow Actions	Task Administration Actions
DataSteward	Update, Merge, Unmerge, Notification	Users can perform the following actions on tasks available to their user role: <ul style="list-style-type: none"> - Claim - Release - Edit - Task actions, such as Accept, Reject, or Disclaim Note: The available task actions depend on your role and ActiveVOS workflows.
Manager	Review No Approve	Users can perform the following actions on tasks available to their user role: <ul style="list-style-type: none"> - Claim - Release - Edit - Task actions, such as Accept, Reject, or Disclaim Note: The available task actions depend on your role and ActiveVOS workflows.
SrManager	Final Review	Users can perform the following actions on tasks available to their user role: <ul style="list-style-type: none"> - Claim - Release - Edit - Task actions, such as Accept, Reject, or Disclaim Note: The available task actions depend on your role and ActiveVOS workflows.
abAdmin	-	Users can perform the following actions on all tasks: <ul style="list-style-type: none"> - Assign - Release - Edit Note: Users can manage tasks for default approval and unmerge ActiveVOS workflows.
abadmin	-	Users can perform the following actions on all tasks: <ul style="list-style-type: none"> - Assign - Release - Edit Note: Users can manage tasks for default merge ActiveVOS workflows.

Note: If you use ActiveVOS for business process management, the user roles must not contain spaces.

To enable the task administration privileges for the abAdmin and abadmin roles, you must configure the Task Administrator role in the Provisioning tool. For more information, see the *Multidomain MDM Provisioning Tool Guide*.

Assigning Workflow User Roles

You assign each of the predefined workflow user roles to business managers or data stewards.

Before you begin, ensure that you have MDM Hub user accounts for all the business users who need to participate in workflows. If required, add new user accounts for business users. If you add new users, add the same users to the container in the application server.

1. In the Hub Console, connect to an Operational Reference Store accessed by an Informatica Data Director application.
2. Acquire a write lock.
3. Open the Security Access Manager workbench and click **Users and Groups**.
The Users and Groups tool opens.
4. Click the **Assign Users/Groups to Role** tab.
5. Select a user role for workflows and click **Edit**.
The **Assign Users to Role** dialog box opens.
6. Select the users or user groups who require this role and click **OK**.
7. Repeat steps 5 and 6 to assign the other workflow roles to users.

Enabling Access to ActiveVOS Web Applications

Informatica ActiveVOS includes two web applications, ActiveVOS Console and ActiveVOS Central. You can enable user access to these web applications by defining the users and workflow roles in the application server.

Define the same user roles and user credentials that are defined in the MDM Hub. For more information about how to create roles and users, see your application server documentation.

Configuring the Task Administrator Role

You can use the Provisioning tool to configure the Task Administrator role and specify the MDM Hub role that you want to map to the Task Administrator role.

Before you begin, if you are using the default ActiveVOS workflows, you must create the abAdmin and abadmin MDM Hub roles in the MDM Hub.

Important: To ensure users can perform task administration actions on all tasks for all workflows, you must assign both the abAdmin and abadmin MDM Hub roles to users.

1. Log in to the Provisioning tool.
2. From the **Database** list, select the database to which you want to associate your configurations.
3. Click **Business Entity > Tasks**.
4. In the Tasks panel, select **Task Administrator Role**, and then click **Create**.
5. In the properties panel, enter `TaskAdministrator` in the **Name** field.
6. Select the **Enable Task Administrator role** check box.
7. From the **MDM Hub Role** list, select the MDM Hub role that you want to map to the Task Administrator role.

If you are using the default ActiveVOS workflows, select **abAdmin**. If you are using custom ActiveVOS workflows, select the MDM Hub role that you use for business administration in ActiveVOS.

8. Click **Apply**.

The changes are saved but are not published to the MDM Hub.

9. Publish the changes to the MDM Hub.
 - a. Click **Publish**.

A confirmation dialog box appears that prompts you to publish or review the changes.
 - b. Review the changes or publish without a review.
 - To publish without a review, click **Publish**.
 - To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.

You configured the Task Administrator role. Users assigned the mapped MDM Hub role can perform task administration actions on all tasks in Data Director.

Enabling Match on Pending Records

By default, the match process includes active records and excludes pending records. You can include pending records.

1. Open the Model workbench and click **Schema**.
2. In the Schema tool, select a state-enabled base object.
3. Expand the tree for the base object and select **Match/Merge Setup**.
4. Select the **Enable Match on Pending Records** check box on the Properties tab of Match/Merge Setup Details panel.

Message Triggers for State Transitions

The MDM Hub communicates with external applications using messages in message queues. You can enable message triggers that report when base object records and cross-reference records change state.

When an action occurs for which a rule is defined, a message is placed in the message queue. The message trigger specifies the queue in which messages are placed.

The following message trigger events are available for state changes to records:

Events	Action
Add new pending data	A new pending record is created.
Update existing pending data	A pending base object record is updated.
Pending update only cross-reference record changed	A pending cross-reference record is updated. This event includes the promotion of a record.
Delete base object data	A base object record is soft deleted.
Delete cross-reference data	A cross-reference record is soft deleted.

Events	Action
Delete pending base object data	A base object record is hard deleted.
Delete pending cross-reference data	A cross-reference record is hard deleted.

Enabling Message Triggers for State Transitions

You can enable message triggers that tell external applications when records change state.

The MDM Hub communicates with external applications using message queues. If you have not done so, you need to configure message queues on your application server. Message queues are part of the publish process.

1. Open the **Model** workbench and click **Schema**.
2. Select the **Trigger on Pending Updates** check box for message queues in the Message Queues tool.

Record Promotion

Record promotion is the process of changing the system state of a set of records from the PENDING state to the ACTIVE state. You can set records for promotion manually, using either the Data Steward tools or the Promote batch process.

The Promote batch process also promotes any children associated with the base object that is promoted. If you want to promote children records and grandchildren records, you must run the Promote batch twice. First run the Promote batch job on the parent base object. Then run the Promote batch job again on the grandchild base object.

Note: When you delete records in the MDM Hub, use a similar approach. To delete children records, delete the base object. Then run a second delete job to delete the grandchildren records of the base object.

Promoting Records in the Data Steward Tools

You can immediately promote PENDING base object or cross-reference records to an ACTIVE state through the tools in the Data Steward workbench. You can also use the Data Manager or Merge Manager to flag these records for promotion at a later time. For more information about using the Hub Console to perform these tasks, see the *Multidomain MDM Data Steward Guide*.

Flagging Base Object or Cross-reference Records for Promotion at a Later Time

You can use the Data Manager to flag records for promotion. You promote the flagged records by running a Promote batch job.

1. Open the Data Steward workbench and click **Data Manager**.
2. In the Data Manager tool, click a base object record or cross-reference record.

3. Click **Flag for Promote** on the associated panel.

Note: When the HUB_STATE_IND field is set to read-only for a package, the Set Record State button is disabled in the Data Manager and Merge Manager Hub Console tools for the associated records. However, the Flag for Promote button remains active because it does not directly alter the HUB_STATE_IND column for the records.

Flagging Matched Records for Promotion Using the Merge Manager

You can use the Merge Manager to flag cross-reference records for promotion. You promote the flagged records by running a Promote batch job.

1. Open the Data Steward workbench and click **Merge Manager**.
2. Click a matched record.
3. In the Matched Records panel, click **Flag for Promote**.

Setting Up a Promote Batch Job Using the Batch Viewer

You can set up a batch job to promote records flagged for promotion.

1. Flag pending records for promotion.
2. Open the Utilities workbench and click **Batch Viewer**.
3. In the Batch Viewer, under the Base Object node, click the **Promote** batch job.
4. Select **Promote flagged records abc**.
Where abc represents the associated records that you previously flagged for promotion.
5. To promote the records flagged for promotion, click **Execute Batch**.

Setting Up a Promote Batch Job Using the Batch Group Tool

To promote flagged records, you can add a Promote Batch job by using the Batch Group tool.

1. Flag pending records for promotion.
2. Open the Utilities workbench and click **Batch Group**.
3. Acquire a write lock.
4. In the Batch Group tree, right-click the Batch Groups node and choose **Add Batch Group**.
5. In the Batch Groups tree, right-click any level, and choose the option to add a level to the batch group.
The **Choose Jobs to Add to Batch Group** dialog box opens.
6. Expand the base objects for the jobs that you want to add.
7. Select the **Promote flagged records in [XREF table]** job.
8. Click **OK**.
The Batch Group tool adds the selected jobs to the batch group.
9. Click **Save**.
The set up is complete. You can run the batch group job.

CHAPTER 12

Data Encryption

This chapter includes the following topics:

- [Data Encryption Overview, 182](#)
- [Data Encryption Architecture, 182](#)
- [Data Encryption Restrictions, 183](#)
- [Data Encryption Utilities, 183](#)
- [Configuring Data Encryption, 184](#)
- [Services Integration Framework API Requests and Responses, 186](#)
- [Sample Data Encryption Properties File, 187](#)

Data Encryption Overview

If you want the MDM Hub implementation to store sensitive data or transfer sensitive data over a network, configure the MDM Hub for data encryption.

When you configure the MDM Hub for data encryption, you can store sensitive data in the database in an encrypted form. When the MDM Hub transfers data from the database to the Hub Server, the Process Server, and Data Director with subject areas, sensitive data is transferred in the encrypted form. Data encryption ensures that any sensitive data that you transfer over a network is secure.

Note: You can only encrypt data of VARCHAR datatype.

Data Encryption Architecture

You can store sensitive data in the database in encrypted form and transfer it to the Hub Server, the Process Server, and Data Director in the encrypted form. Data encryption ensures that sensitive data in the MDM Hub implementation is secure.

Encrypted data is decrypted in Data Director or other Services Integration Framework (SIF) clients before it is shown to users. Also, the encrypted data in the database is decrypted before the cleanse process.

The MDM Hub performs each SIF API call through a `SiperianClient` object. Each call comprises a request and a response. The request that goes out must be encrypted and the response that comes in must be decrypted.

The MDM Hub can encrypt the following data exchanges over a network:

- Between the Hub Server and the database
- Between the Hub Server and the Process Server during cleanse operations
- Between Data Director and the Hub Server

Data Encryption Restrictions

When you configure data encryption for base object columns, data is stored in an encrypted form.

Before you configure data encryption, consider the following restrictions:

- When you use the Hub Console tools such as the Data Manager or the Merge Manager to add or edit data to encrypted columns, the data is not encrypted.
- Data encryption is not supported for Data Director with business entities. Data encryption is supported only for Data Director with subject areas.
- Data encryption is not supported for REST APIs. Data encryption is supported only for Services Integration Framework (SIF) APIs.
- If SIF APIs do not use EjbSiperianClient, the data in requests is not encrypted and the responses are not decrypted. You must encrypt any sensitive data before you send a request and decrypt the encrypted data after you receive a response.
- SOAP calls do not encrypt data. If you use SOAP calls to insert data into an encrypted column, the data must be pre-encrypted.
- SOAP calls, such as SearchMatch and SearchQuery cannot search on data present in encrypted columns.
- If a SOAP call, such as Get, retrieves data from an encrypted column, the data is returned in an encrypted form.
- If a match field is a concatenation of multiple columns, the encrypted data cannot contain spaces. Each of these columns must be encrypted even if the columns do not contain data.
- Search queries with wildcard characters and literals do not fetch search results.

Data Encryption Utilities

To configure data encryption for the MDM Hub, you can use the data encryption utilities available in the Resource Kit.

The following data encryption samples and utilities are included in the Resource Kit:

Data Encryption Libraries

The data encryption libraries are required by the MDM Hub to be bundled in the data encryption JAR file. The data encryption libraries contain APIs and common classes. In particular, it includes the DataEncryptor interface that you must implement when you configure data encryption. The DataEncryptor interface defines the `encrypt` and `decrypt` methods that you need to implement for data encryption in the MDM Hub.

Sample of Data Encryption Properties File

The sample data encryption properties file includes parameters that the data encryption implementation requires. The name of the properties file is `dataencryption.properties`. You can customize the sample properties file to specify the options for the data encryption implementation.

Sample DataEncryption Interface Implementation

The sample `DataEncryption` interface implementation uses the `InformaticaDataEncryptor` class to implement the `DataEncryptor` interface. Refer to the sample implementation to create a custom encryption algorithm. The `mainClass` property in the data encryption properties file must refer to the class name that you use in the interface implementation.

Ant Build Script

The Ant build script, `build.xml`, to create the data encryption JAR file.

Configuring Data Encryption

To use data encryption, you need to configure the MDM Hub for data encryption.

1. Implement the `DataEncryptor` Interface.
2. Configure the data encryption properties file.
3. Configure data encryption for the Hub Server.
4. Configure data encryption for the Process Server.

Step 1. Implement DataEncryptor Interface

You must implement the `DataEncryptor` interface. `DataEncryptor` interface defines the `encrypt` and the `decrypt` methods. The implementation of the `encrypt` and the `decrypt` methods must be thread-safe.

1. Create a Java project in a Java Integrated Development Environment.
2. Add the following MDM Hub JAR files to the Java project:
 - `siperian-api.jar`
 - `siperian-common.jar`

The jar files are in the following directory:

On UNIX. `<infamdm_install_dir>/resourcekit/samples/DataEncryption/lib`

On Windows. `<infamdm_install_dir>\resourcekit\samples\DataEncryption\lib`
3. Create a Java class for data encryption that includes the `encrypt` and the `decrypt` methods.
4. Compile the data encryption Java class.
5. To package the class files in a custom data encryption JAR file, run the following command:

```
ant build
```
6. To clean up the generated files, run the following command after the build completes:

```
ant clean
```
7. Implement the `DataEncryptor` interface in the custom data encryption JAR file that you create.

Step 2. Configure the Data Encryption Properties File

Informatica provides a sample data encryption properties file that includes the parameters that the data encryption implementation requires. You can customize the sample properties file to specify the options for the data encryption implementation.

1. Find the `dataencryption.properties` file in the following directory:

On UNIX. `<ResourceKit_install_dir>/DataEncryption/resources`

On Windows. `<ResourceKit_install_dir>\DataEncryption\resources`

2. Create a backup copy of the `dataencryption.properties` file.
3. Use a text editor to open the file and edit the data encryption parameters.
4. Provide a reference to the `DataEncryptor` interface implementation.
5. Configure `EJBSiperianClient` to encrypt and decrypt base object columns that have sensitive data, their associated tables such as history and cross-reference tables, and packages.

Use the following syntax to specify the base object column names, the associated tables such as history and cross-reference tables, and packages:

```
API.<ORS_ID>.BASE_OBJECT.<BASE_OBJECT_NAME>.<COLUMN_NAME>
API.<ORS_ID>.XREF.<CROSS_REFERENCE_TABLE_NAME>.<COLUMN_NAME>
API.<ORS_ID>.HISTORY.<HISTORY_TABLE_NAME>.<COLUMN_NAME>
API.<ORS_ID>.PACKAGE.<PACKAGE_NAME>.<COLUMN_NAME>
API.<ORS_ID>.HM_ENTITY_TYPE.<HIERARCHY_MANAGER_ENTITY_NAME>.<COLUMN_NAME>
```

6. If you want to use fuzzy match and fuzzy search operations, specify the match fields to use.

Use the following syntax to specify a match field name:

```
MATCHFIELD.<ORS_ID>.BASE_OBJECT.<BASE_OBJECT_NAME>.<MATCH_FIELD_NAME>
```

Note: Ensure that the base object that you specify is the base object that you are matching on, which is the root of the match path component. Even where match field is from a base object that is not the root of the match path component, specify the base object that is the root of the match path component.

If a match field is a concatenation of multiple columns, you must encrypt each of these columns even if the columns do not contain data.

7. Configure the input and output ports of cleanse functions that must encrypt and decrypt data.

Use the following syntax to specify the input and output ports of cleanse functions:

```
CLEANSE.<PORT_NAME>=true
```

8. Save the properties file with the same name, `dataencryption.properties`.

Step 3. Configure Data Encryption for the Hub Server

To configure the Hub Server to use data encryption, configure the Hub Server to use the custom data encryption JAR file.

1. Find the `cmxserver.properties` file in the following directory:

On UNIX. `<infadm_install_directory>/hub/server/resources`

On Windows. `<infadm_install_directory>\hub\server\resources`

2. Use a text editor to open the file and specify the path to the data encryption JAR for the `encryption.plugin.jar` property.

```
encryption.plugin.jar=<Path to the data encryption JAR>
```

3. Save the `cmxserver.properties` properties file.

Step 4. Configure Data Encryption for the Process Server

To configure the Process Server to use data encryption, configure the Process Server to use the custom data encryption JAR file.

1. Find the `cmxcleanse.properties` file in the following directory:
On UNIX. `<infamdm_install_directory>/hub/cleanse/resources`
On Windows. `<infamdm_install_directory>\hub\cleanse\resources`
2. Use a text editor to open the file and specify the path to the data encryption JAR for the `encryption.plugin.jar` property.
`encryption.plugin.jar=<Path to the data encryption JAR>`
3. Save the `cmxcleanse.properties` properties file.

Services Integration Framework API Requests and Responses

The MDM Hub uses the data encryption Java class included in the data encryption JAR file to call Services Integration Framework (SIF) APIs.

The following table describes SIF API requests and responses that function with data encryption:

Request/Response	Description
PutRequest	Inserts or updates a single record identified by a key into a base object. Encrypts input records.
MultiMergeRequest	Merges multiple base object records that represent the same object. You can specify the field level overrides for the merged record. Encrypts overriding fields.
GetResponse	Uses a known key to retrieve a single record from a package. Encrypts records that are returned.
SearchHmQueryRequest	Searches Hierarchy Manager entities and relationships. Encrypts input filter parameters. Use <code>SiperianObjectUidField</code> instead of a parameter to enable encryption.
SearchQueryRequest	Retrieves a set of records from a package that satisfies the criteria you specify. Encrypts input filter parameters. Use <code>SiperianObjectUidField</code> instead of a parameter to enable encryption.
SearchMatchRequest	Searches for records in a package based on match columns and rule definitions. Encrypts the input filter parameters and the records against which you want to perform the match. Use <code>SiperianObjectUidField</code> instead of a parameter to enable encryption.
SearchHmQueryResponse	Returns entities and relationships associated with one or more entities. Encrypts returned records.

Request/Response	Description
SearchMatchResponse	Returns a list of records and match tokens. Encrypts returned records.
SearchQueryResponse	Returns a set of records from a package. Encrypts returned records.
AddRelationshipRequest	Adds a relationship between two entities. Encrypts custom fields provided as a record
UpdateRelationshipRequest	Hierarchy Manager request to change characteristics of a relationship. Encrypts custom fields provided as a record
GetOneHopResponse	Returns information about entities directly related to a specified group of entities in a specified hierarchy configuration. Encrypts returned records.
GetEntityGraphResponse	Returns a graph of entities and relationships related to a specified set of entities. Encrypts returned records.
GetXrefForEffectiveDateResponse	Returns multiple cross-reference records for the specified effective date. Encrypts returned records.

Sample Data Encryption Properties File

The Resource Kit includes a sample data encryption properties file, `dataencryption.properties`.

The sample data encryption properties file is in the following directory:

On UNIX. `<infamdm_install_dir>/hub/resourcekit/samples/DataEncryption/resources`

On Windows. `<infamdm_install_dir>\hub\resourcekit\samples\DataEncryption\resources`

The following example shows the contents of the sample data encryption properties file:

```
# This is an example of dataencryption.properties file.
# File contains encryptor configuration and different parts of product which should be
# aware of data encryption

#
# Encryptor implementation
#

# main class which implements DataEncryptor interface. If this option is empty
# encryption is turned off.
mainClass=com.informatica.dataencryption.sample.InformaticaDataEncryptor

#
# Part 1. EjbSiperianClient configuration.
#
# List of API.<ORS_ID>.<BASE_OBJECT_NAME>.<COLUMN_NAME> which should be encrypted before
# sending out
# and decrypted on return back by EjbSiperianClient.
#
API.orcl-MDM_SAMPLE.BASE_OBJECT.C_PARTY.DISPLAY_NAME=true
```

```

API.orcl-MDM_SAMPLE.PACKAGE.PKG_PERSON_IDD_SEARCH.DISPLAY_NAME=true
API.orcl-MDM_SAMPLE.PACKAGE.PKG_ORG_IDD_SEARCH.DISPLAY_NAME=true
API.orcl-MDM_SAMPLE.HISTORY.C_PARTY.DISPLAY_NAME=true
API.orcl-MDM_SAMPLE.XREF.C_PARTY.DISPLAY_NAME=true
API.orcl-MDM_SAMPLE.HM_ENTITY_TYPE.Organization.DISPLAY_NAME=true

#
# Part 2. Cleanse functions.
#
# List of input and output port of cleanse functions which should take and produce
# encrypted data.
#
CLEANSE.firstNameInput=true
CLEANSE.firstNameOutput=true

#
# Part 3. Match Fields
#
# List of MATCHFIELD.<ORS_ID>.<BASE OBJECT NAME>.<MATCH FIELD NAME>
# The server will decrypt this match field automatically before get_ranges, get_keys,
# and get_match.

MATCHFIELD.orcl-MDM_SAMPLE.C_PARTY.ORGANIZATION_NAME=true

```


CHAPTER 13

Hierarchies

This chapter includes the following topics:

- [Hierarchies Overview, 189](#)
- [Hierarchy Example, 190](#)
- [About Configuring Hierarchies, 190](#)
- [Before You Begin, 191](#)
- [Overview of Configuration Steps, 191](#)
- [Preparing Your Data for Hierarchy Manager, 191](#)
- [Use Case Example of How to Prepare Data for Hierarchy Manager, 192](#)
- [Creating the HM Repository Base Objects, 196](#)
- [Uploading Default Entity Icons, 197](#)
- [Configuring Entity Icons, 197](#)
- [Entities, 198](#)
- [Entity Base Objects, 198](#)
- [Entity Types, 202](#)
- [Display Options for Entities, 205](#)
- [Reverting Entity Base Objects to Base Objects, 206](#)
- [Hierarchy Types, 206](#)
- [Relationship Objects, 207](#)
- [Relationship Base Objects, 208](#)
- [Foreign Key Relationship Base Objects, 211](#)
- [Relationship Types, 212](#)
- [Packages, 215](#)
- [About Profiles, 219](#)

Hierarchies Overview

You can configure hierarchies in the MDM Hub Console. Hierarchies show the relationship between individual records, whereas the datamodel show relationships between base objects.

For example, you can create a hierarchy to track which employees belong to each department within an organization. You could also create a hierarchies of products within various product groups.

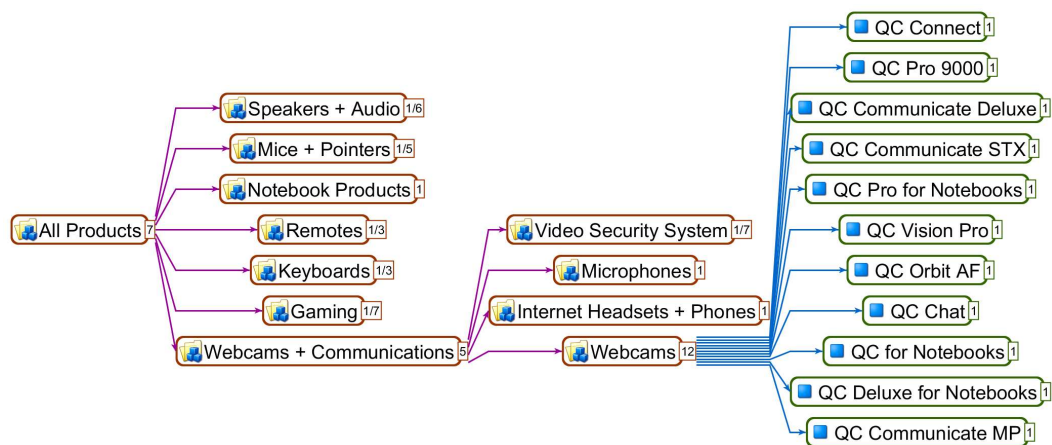
To configure a hierarchy, you need to first create an entity base object. etc.

You must have a Hierarchy Manager license to configure hierarchies.

Hierarchy Example

The sample Operational Reference Store in the Resource Kit contains a schema with preconfigured hierarchies. This chapter uses a portion of the Product hierarchy to illustrate the hierarchy configuration steps.

The following image shows the portion of the hierarchy used in the examples as it appears in the Hierarchy Manager tool in the Hub Console:



About Configuring Hierarchies

MDM Hub administrators use the Hierarchies tool to set up the structures required to view and manipulate data relationships in Hierarchy Manager.

Use the Hierarchies tool to define Hierarchy Manager components, such as entity types, hierarchies, relationships types, packages, and profiles, for your MDM Hub implementation.

When you have finished defining Hierarchy Manager components, you can use the package or query manager tools to update the query criteria.

Note: You must configure packages for use in Hierarchy Manager and validate the profile.

To understand the concepts of hierarchies and their components, you must be familiar with the concepts in the following chapters:

- [Chapter 8, “Building the Schema” on page 75](#)
- [Chapter 9, “Queries and Packages” on page 124](#)
- [Chapter 24, “Configuring the Consolidate Process” on page 483](#)

Before You Begin

Before you begin to configure your Hierarchy Manager (HM) you must complete some tasks.

Complete the following tasks:

- Start with a blank Operational Reference Store or a valid ORS and register the database in CMX_SYSTEM.
- Verify that you have a license for Hierarchy Manager. For details, consult your Informatica MDM Hub administrator.
- Perform data analysis.

Overview of Configuration Steps

To configure Hierarchy Manager, complete the following steps:

1. Start the Hub Console.
2. Launch the Hierarchies tool.
If you have not already created the Repository Base Object (RBO) tables, the Hub Console walks you through the process.
3. Create entity objects and types.
4. Create hierarchies.
5. Create relationship objects and types.
6. Create packages.
7. Configure profiles.
8. Validate the profile.

Note: The same options you see on the right-click menu in the Hierarchy Manager are also available on the Hierarchies menu.

Preparing Your Data for Hierarchy Manager

To make the best use of HM, you should analyze your information and make sure you have done the following:

- Verified that your data sources are valid and trustworthy.
- Created valid schema to work with Informatica MDM Hub and the HM.
- Created hierarchial, foreign key, and one-hop and multi-hop relationships (direct and indirect relationships) between your entities:

All child entities must have a valid parent entity related to them.

Your data cannot have any 'orphan' child entities when it enters HM.

All hierarchies must be validated.

For more information on one-hop and multi-hop relationships, see the *Multidomain MDM Data Steward Guide*.

- Derived HM types.
- Consolidated duplicate entities from multiple source systems.
For example, a group of entities (Source A) might be the same as another group of entities (Source B), but the two groups of entities might have different group names. Once the entities are identified as being identical, the two groups can be consolidated.
- Grouped your entities into logical categories, such as physician's names into the "Physician" category.
- Made sure that your data complies with the rules for referential integrity, invalid data, and data volatility.
For more information on these database concepts, see a database reference text.

Use Case Example of How to Prepare Data for Hierarchy Manager

This section contains an example of how to manipulate your data before it enters Informatica MDM Hub and before it is viewed in Hierarchy Manager.

Typically, a company's data would be much larger than the example given here.

Scenario

John has been tasked with manipulating his company's data so that it can be viewed and used within Hierarchy Manager in the most efficient way.

To simplify the example, we are describing a subset of the data that involves product types and products of the company, which sells computer components.

The company sells three types of products: mice, trackballs, and keyboards. Each of these product types includes several vendors and different levels of products, such as the Gaming keyboard and the TrackMan trackball.

Methodology

This section describes the method of data simplification.

Step 1: Organizing Data into the Hierarchy

In this step you organize the data into the Hierarchy that will then be translated into the HM configuration.

John begins by analyzing the product and product group hierarchy. He organizes the products by their product group and product groups by their parent product group. The sheer volume of data and the relationships contained within the data are difficult to visualize, so John lists the categories and sees if there are relationships between them.

The following table (which contains data from the Marketing department) shows an example of how John might organize his data.

ProdGroup		ProdGroup		ProdGroup		Product	
ProdNumber	Description	ProdNumber	Description	ProdNumber	Description	ProdNumber	Description
ALL	All Products	100	Mice + Pointers	120	Mice	120-0001	Laser Mouse
						120-0002	Nano Cordless Laser Mouse
						120-0003	Cordless Optical Mouse
						120-0004	Nana Cordless Laser Mouse II
						120-0005	Laser Mouse for Notebooks
						120-0006	Revolution
						120-0007	Rechargable Cordless Mouse
						120-0008	Cordless Optical Mouse II
		200	Keyboards	210	Keyboards	-	-
				220	Keyboard and Mice Combos	-	-

Note: Most data sets will have many more items.

The table shows the data that will be stored in the Products BO. This is the BO to convert (or create) in HM. The table shows Entities, such as Mice or Laser Mouse. The relationships are shown by the grouping, that is, there is a relationship between Mice and Laser Mouse. The heading values are the Entity Types: Mice is a Product Group and Laser Mouse is a Product. This Type is stored in a field on the Product table.

Organizing the data in this manner allows John to clearly see how many entities and entity types are part of the data, and what relationships those entities have.

The major category is ProdGroup, which can include both a product group (such as mice and pointers), the category Product, and the products themselves (such as the Trackman Wheel). The relationships between these items can be encapsulated in a relationship object, which John calls Product Rel. In the information for the Product Rel, John has explained the relationships: Product Group is the parent of both Product and Product Group.

Step 2: Creating Relationship Base Object Tables

Having analyzed the data, John comes to the following conclusions:

- Product (the BO) should be converted to an Entity Object.
- Product Group and Product are the Entity Types.
- Product Rel is the Relationship Object to be created.
- The following relationship types (not all shown in the table) need to be created:
 - Product is the parent of Product (not shown)
 - Product Group is the parent of Product (such as with the Mice to Laser Mouse example).
 - Product Group is the parent of Product Group, such as with Mice + Pointers being the parent of Mice).

John begins by accessing the Hierarchy Tool. When he accesses the tool, the system creates the Relationship Base Object Tables (RBO tables). RBO tables are essentially system base objects that are required base objects containing specific columns. They store the HM configuration data, such as the data that you see in the table in Step 1.

For instructions on how to create base objects, see [“Creating Base Objects” on page 104](#). This section describes the choices you would make when you create the example base objects in the Schema tool.

You must create and configure a base object for each entity object and relationship object that you identified in the previous step. In the example, you would create a base object for Product and convert it to an HM Entity Object. The Product Rel BO should be created in HM directly (an easier process) instead of converting. Each new base object is displayed in the Schema panel under the category Base Objects. Repeat this process to create all your base objects.

In the next section, you configure the base objects so that they are optimized for HM use.

Step 3: Configuring Base Objects

You created the two base objects (Product and Product Rel) in the previous section. This section describes how to configure them.











Configuring a base object involves filling in the criteria for the object’s properties, such as the number and type of columns, the content of the staging tables, the name of the cross-reference tables (if any), and so on. You might also enable the history function, set up validation rules and message triggers, create a custom index, and configure the external match table (if any).

Whether or not you choose these options and how you configure them depends on your data model and base object choices.

In the example, John configures his base objects as the following sections explain.

Note: Not all components of the base-object creation are addressed here, only the ones that have specific significance for data that will be used in the HM. For more information on the components not discussed here, see [Chapter 8, “Building the Schema” on page 75](#).










The following figure shows the base object columns:

	Display Name	Physical Name	Nullable	Data Type	Length
	Rowid Object	ROWID_OBJECT	<input type="checkbox"/>	CHAR	14
	Product Number	PRODUCT_NUMBER	<input checked="" type="checkbox"/>	VARCHAR	50
	Product Name	PRODUCT_NAME	<input checked="" type="checkbox"/>	VARCHAR	100
	Product Desc	PRODUCT_DESC	<input checked="" type="checkbox"/>	VARCHAR	255
	Inception Date	INCEPTION_DATE	<input checked="" type="checkbox"/>	DATE	
	Eff Start Date	EFF_START_DATE	<input checked="" type="checkbox"/>	DATE	
	Eff End Date	EFF_END_DATE	<input checked="" type="checkbox"/>	DATE	
	Status Cd	STATUS_CD	<input checked="" type="checkbox"/>	CHAR	2
	Product Type Cd	PRODUCT_TYPE_CD	<input checked="" type="checkbox"/>	VARCHAR	50
	Product Type	PRODUCT_TYPE	<input type="checkbox"/>	VARCHAR	255

This table shows the Product BO after conversion to an HM entity object. In this list, only the Product Type field is an HM field.

Every base object has system columns and user-defined columns. System columns are created automatically, and include the required column: Rowid Object. This is the Primary key for each base object table and contains a unique, Hub-generated value. This value cannot be null because it is the HM lookup for the class code. HM makes a foreign key constraint in the database so a ROWID_OBJECT value is required and cannot be null.

For the user-defined columns, John choose logical names that would effectively include information about the products, such as Product Number, Product Type, and Product Description. These same column and column values must appear in the staging tables as shown in the following figure:

Columns						
The selected columns will be included in this staging table. To include a lookup, use the edit button.						
	Column	Lookup System	Lookup Table	Lookup Column	Allow Null Update	Allow Null Foreign Key
	<input checked="" type="checkbox"/> Product Number				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Product Name				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Product Desc				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Inception Date				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Eff Start Date				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Eff End Date				<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Status Cd		LU Product Status	Status Cd	<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Product Type Cd		LU Product Type	Product Type	<input type="checkbox"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Product Type		Rbo Bo Class	Bo Class Code	<input type="checkbox"/>	<input type="checkbox"/>

John makes sure that all the user-defined columns from the staging tables are added as columns in the base object, as the graphic above shows. The Lookup column shows the HM-added lookup value.

Notice that several columns in the Staging Table (Status Cd, Product Type, and Product Type Cd) have references to lookup tables. You can set these references up when you create the Staging Table. You would use lookups if you do not want to hardcode a value in your staging table, but would rather have the server look up a value in the parent table.

Most of the lookups are unrelated to HM and are part of the data model. The Rbo Bo Class lookup is the exception because it was added by HM. HM adds the lookup on the product Type column.

Note: When you are converting entities to entity base objects (entities that are configured to be used in HM), you must have lookup tables to check the values for the Status Cd, Product Type, and Product Type Cd.

Note: HM Entity objects do not require start and end dates. Any start and end dates would be user defined. However, Rel Objects do use these. Do not create new Rel Objects with different names for start and end dates. These are already provided.

Step 4: Creating Entity Types

You create entity types in the Hierarchy Tool. John creates two entity types: ProdGroup and Product Type.

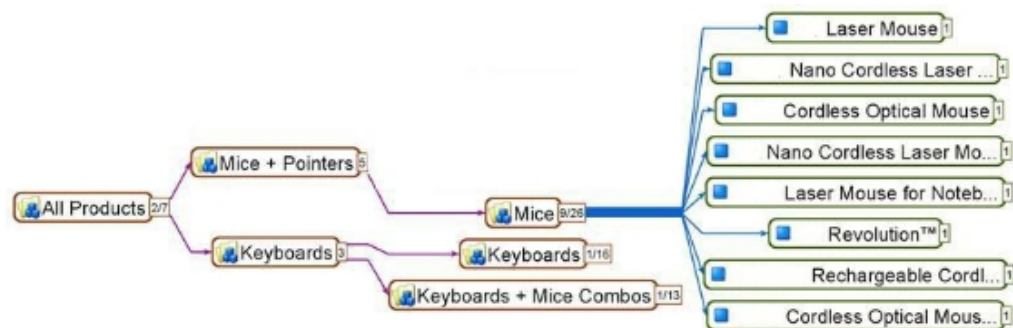
Each entity type has a code that derives from the data analysis and the design. John chooses to use Product as one type, and Product Group as another.

This code must be referenced in the corresponding RBO base object table. In this example, the code Product is referenced in the C_RBO_BO_CLASS table. The value of the BO_CLASS_CODE is 'Product'.

The following table shows the relationship between the HM entity objects and HM relationship objects to the RBO tables:

Object	Table	Relationship
Product Entity Object	BO_CLASS_CODE	many-to-one (from object to table)
Product Relationship Object	C_RBO_BO_CLASS	many-to-one (from object to table)
	C_RBO_HIERARCHY	one-to-one (from table to object)

When John has completed all the steps in this section, he will be ready to create other HM components, such as packages, and to view his data in the HM. For example, the following graphic shows the relationships that John has set up in the Hierarchies Tool, displayed in the Hierarchy Manager. This example shows the hierarchy involving Mice devices fully. For more information on how to use Hierarchy Manager, see the *Multidomain MDM Data Steward Guide*.



Creating the HM Repository Base Objects

To use the Hierarchies tool with an ORS, the system must first create the Repository Base Objects (RBO tables) for the ORS. RBO tables are essentially system base objects.

They are required base objects that must contain specific columns.

Queries and packages (and their associated queries) will also be created for these RBO tables.

Note: Never modify these RBO tables, queries, or packages.

To create the RBOs:

1. Acquire a write lock.
2. Start the Hierarchies tool. Expand the Model workbench and click **Hierarchies**.

Note: Any option that you can select by right-clicking in the navigation panel, you can also choose from the Hierarchies tool menu.

After you start the Hierarchies tool, if an ORS does not have the necessary RBO tables, then the Hierarchies tool walks you through the process of creating them.

The following steps explain what to select in the dialog boxes that the Hierarchies tool displays:

1. Choose **Yes** in the Hub Console dialog to create the metadata (RBO tables) for HM in the ORS.
2. Select the tablespace names in the Create RBO tables dialog, and then click **OK**.

Uploading Default Entity Icons

The Hierarchies tool prompts you to upload default entity icons.

These icons will be useful to you when you are creating entities.

1. Click **Yes**.
2. The Hub Console displays the Hierarchies tool with the default metadata.

Configuring Entity Icons

Using the Hierarchies tool, you can add or configure your own entity icons that you can subsequently use when configuring your entity types.

These entity icons are used to represent your entities in graphic displays within Hierarchy Manager. Entity icons must be stored in a JAR or ZIP file.

Adding Entity Icons

To import your own icons, create a ZIP or JAR file containing your icons. For each icon, create a 16 x 16 icon for the small icon and a 48 x 48 icon for the large icon.

To add new entity icons:

1. Acquire a write lock.
2. Start the Hierarchies tool.
3. Right-click anywhere in the navigation pane and choose **Add Entity Icons**.

Note: You must acquire a lock to display the right-click menu.

A browse files window opens.

4. Browse for the JAR or ZIP file containing your icons.
5. Click **Open** to add the icons.

Modifying Entity Icons

You cannot modify icons directly from the console.

You can download a ZIP or JAR file, modify its contents, and then upload it again into the console.

You can either delete icons groups or make them inactive. If an icon is already associated with an entity, or if you could use a group of icons in the future, you might consider choosing to inactivate them instead of deleting them.

You inactivate a group of icons by marking the icon package **Inactive**. Inactive icons are not displayed in the UI and cannot be assigned to an entity type. To reactivate the icon packet, mark it **Active**.

Note: Informatica MDM Hub does not validate icons assignments before deleting. If you delete an icon that is currently assigned to an Entity Type, you will get an error when you try to save the edit.

Deleting Entity Icons

You cannot delete individual icons from a ZIP or JAR file from the console; you can only delete them as a group or package.

To delete a group of entity icons:

1. Acquire a write lock.
2. Start the Hierarchies tool.
3. Right-click the icon collections in the navigation pane and choose **Delete Entity Icons**.

Entities

In Hierarchy Manager, an entity is any object, person, place, organization, or other thing that has a business meaning and can be acted upon in your database.

Examples include a specific person's name, a specific checking account number, a specific company, a specific address, and so on.

Entity Base Objects

Entity base objects are base objects that contain columns for maintaining hierarchy relationships. You can create a base object as an entity base object, or you can create an entity base object by converting a base object to an entity base object.

When you use the Hierarchies tool to create an entity base object, the Hierarchies tool creates the columns required for Hierarchy Manager. You can also use the options in the Hierarchies tool to convert an existing base object to an entity base object.

Note: You cannot convert a base object into an entity base object if the base object has data.

After you add an entity base object, you use the Schema Manager to view, edit, or delete the entity base object.

When you create an entity base object you select to create one of the following foreign key columns:

ROWID_BO_CLASS

When you select Row ID Object, the Hierarchy tool adds the ROWID_BO_CLASS column.

The ROWID_BO_CLASS field is populated by the row ID value.

BO_CLASS_CODE or an existing column

When you select the base object class code, the hierarchy tool adds the BO_CLASS_CODE column if you create a base object as an entity base object. If you convert an existing base object to an entity base object, you can choose to create the BO_CLASS_CODE column, or you can select an existing column.

The base object class code defines the entity type of the record. The field is populated by a value that you define. Configure your mappings to populate this column.

The row ID is generated and assigned by the system, but the BO Class Code is created by the user, making it easier to remember.

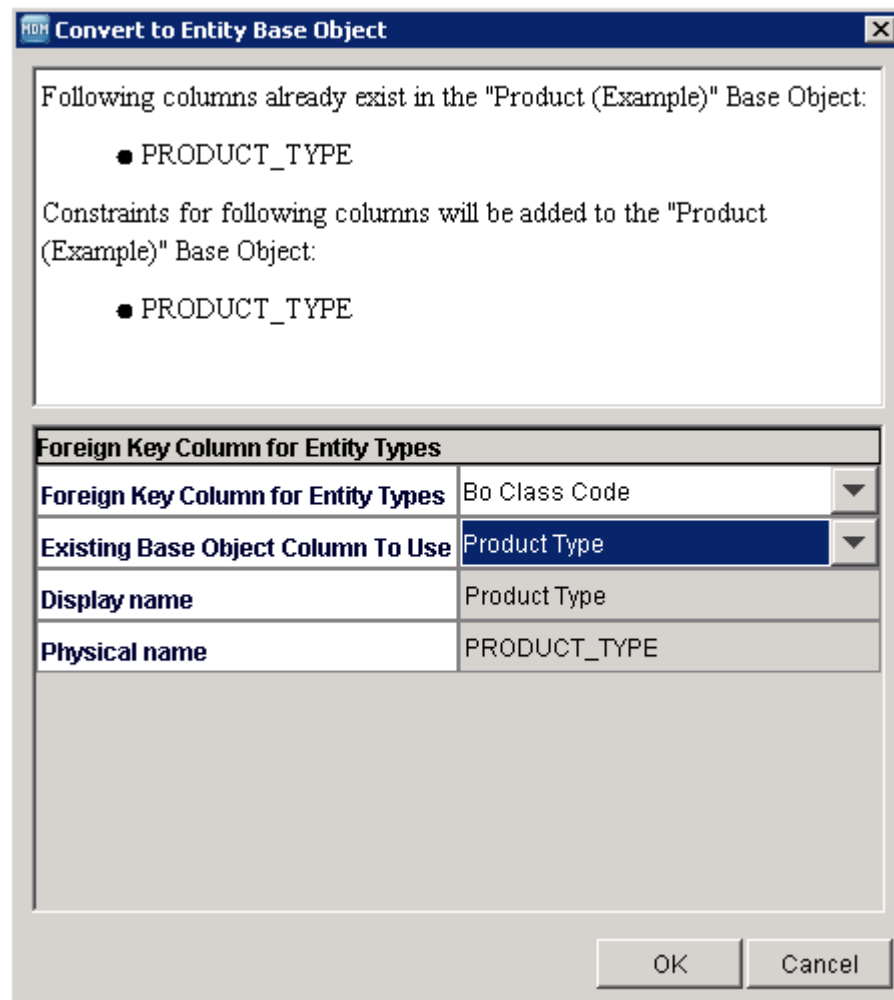
Entity Base Object Example

In the Product hierarchy in our example, we create one entity base object.

When we create the entity base object, we need to decide how we are going to specify the entity type. In our example, we don't want to use a system-generated row ID number, want to use the meaningful codes Product and Product Group to specify the entity type.

We want to use a foreign key entity type column of our own choice, so we first create a base object with a Product_Type column. If you create a new entity base object, you are not able to specify a customized column name. When we convert this base object to an entity base object, we select the Product_Type column as the entity type foreign key. For this hierarchy, we will create two entity types. The value in the Product_Type column determines to which entity type a record belongs.

The following image shows the selections made to convert the sample entity base object:



You can then populate the foreign key column in a mapping.

Create Entity Base Objects

To create a base object that is an entity base object, use the **Hierarchies** tool in the Hub Console.

1. Acquire a write lock.
2. In the **Model** workbench, select the **Hierarchies** tool.
3. Select **Hierarchies > Create New Entity/Relationship Object**.
4. In the **Create New Entity/Relationship Base Object** dialog box, select **Create New Entity Base Object**, and then click **OK**.

5. In the **Create New Entity Base Object** dialog box, specify the following properties for the entity base object:

Field	Description
Item type	Base object. Read-only.
Display name	Name of this base object that the Hub Console displays.
Physical name	Actual name of the table in the database. Informatica MDM Hub will suggest a physical name for the table based on the display name that you enter. The row ID is generated and assigned by the system, but the BO Class Code is created by the user, making it easier to remember.
Data tablespace	Name of the data tablespace. Default is CMX_DATA.
Index tablespace	Name of the index tablespace. Default is CMX_INDX.
Load tablespace	Name of the load tablespace. Default is CMX_DATA.
User Temporary Tablespace	Name of the user temporary tablespace. Default is CMX_USER_TEMP.
Description	Description of the entity base object. Optional.
Secure Resource	Determines the status of the base object in the Secure Resources tool. If enabled, the status of the base object is Secure. If disabled, the status of the base object is Private. Default is enabled.
Foreign Key column for Entity Types	Foreign Key column for the entity type. If Row ID Object, foreign key relationship is based on a Row ID that the MDM Hub generates. If BO Class Object, the foreign key relationship is based on a code that you define. Default is Row ID Object.
Display name	The name of the column that the Hub Console displays.
Physical name	The name of the foreign key column in the entity base object. The Hub Console suggests a physical name for the column based on the display name that you enter.

6. Click **OK**.

The MDM Hub creates the entity base object with the columns that Hierarchy Manager requires. Use the **Schema** tool to add columns to the entity base object. Do not modify the foreign key columns that the Hierarchy Manager uses. If you modify the foreign key columns Hierarchy Manager might not perform as expected and you can lose data.

Converting Base Objects to Entity Base Objects

You must convert base objects to entity base objects before you can use them in the Hierarchy Manager. The base objects must have state management enabled before you can convert them to entity base objects.

Base objects do not have the metadata required by the Hierarchy Manager. To use base objects with the Hierarchy Manager, you must add this metadata using a conversion process. You can use entity base objects with both the MDM Hub and the Hierarchy Manager.

1. In the Hierarchies tool, acquire a write lock.
2. Right-click anywhere in the navigation pane and choose **Convert BO to Entity/Relationship Object**.

Note: The same options you see on the right-click menu are also available on the Hierarchies menu.

3. In the Modify Existing Base Object dialog, select **Convert to Entity Base Object** and click **OK**.

Note: If you do not see any choices in the Modify Base Object field, then there are no non-hierarchy base objects available. You must create one in the Schema tool.

4. Click **OK**.

If the base object already has Hierarchy Manager metadata, the Hierarchies tool will display a message indicating the Hierarchy Manager metadata that exists.

5. In the Foreign Key Column for Entity Types field, select the column to be added: RowId Object or BO Class Code.

The ability to choose a BO Class Code column reduces the complexity by allowing you to define the foreign key relationship based on a predefined code, rather than the MDM Hub-generated ROWID.

6. In the Existing BO Column to use, select an existing column or select the Create New Column option.

If no BO columns exist, only the Create New Column option is available.

7. In the Display Name and Physical Name fields, create display and physical names for the column, and click **OK**.

The base object will now have the columns that Hierarchy Manager requires. To add additional columns, use the Schema Manager.

Important: When you modify the base object using the Schema Manager tool, do not change any of the columns added using the Hierarchies tool. Modifying any of these columns will result in unpredictable behavior and possible data loss.

Entity Types

Entity types classify the entities. Entity types allow you to classify the entities within an entity base object.

Entity types belong to a specific entity. Entity base objects can have more than one entity type, but each individual entity in the base object and only belong to one entity type. For example, a Product entity base object can have a Product entity type or a Product Group entity type. Entities in the Product base object can either have a Product entity type or a Product Group entity type.

Examples include doctors, checking accounts, banks, and so on. An Entity Base Object must have a Foreign Key to the Entity Type table (Rbo BO Class). The foreign key can be defined as either a ROWID or predefined Code value.

Well-defined entity types have the following characteristics:

- Use entity types to organize the records in a way that reflects the real-world organization of the entities. They effectively segment the data in a way that reflects the real-world nature of the entities.
- Taken collectively, they cover the entire set of entities. That is, every entity has one and only one entity type.
- They are granular enough so that you can easily define the types of relationships that each entity type can have. For example, an entity type of “doctor” can have the relationships: “member of” with a medical group, “staff” (or “non-staff with admitting privileges”) with a hospital, and so on.
- A more general entity type, such as “care provider” (which encompasses nurses, nurse practitioners, doctors, and others) is not granular enough. In this case, the types of relationships that such a general entity type will have will depend on something beyond just the entity type. Therefore, you need to need to define more-granular entity types.

When you configure hierarchy types, you select an icon and color for the appearance of the entities of that type in the Hierarchy Manager tool.







You can configure the following properties when you create an entity type:

Field	Description
Code	Unique code name of the Entity Type. Can be used as a foreign key from entity base objects. Each entity type must have a unique code value.
Display name	Name of this entity type that is displayed in the Hierarchies tool.
Description	Description of the entity type. Optional.
Color	Color of the entities associated with this entity type as they appear in the Hierarchy Manager tool in the Hub Console and in the Informatica Data Director Hierarchy view.
Small Icon	Small icon for entities associated with this entity type as they appear in the Hierarchy Manager tool in the Hub Console and in the Informatica Data Director Hierarchy view. The small icon appears when there are a large number of entities to display in the hierarchy.
Large Icon	Large icon for entities associated with this entity type as they appear in the Hierarchy Manager tool in the Hub Console and in the Informatica Data Director Hierarchy view.







Entity Type Example

The entity types in our example are Product and Product Group.

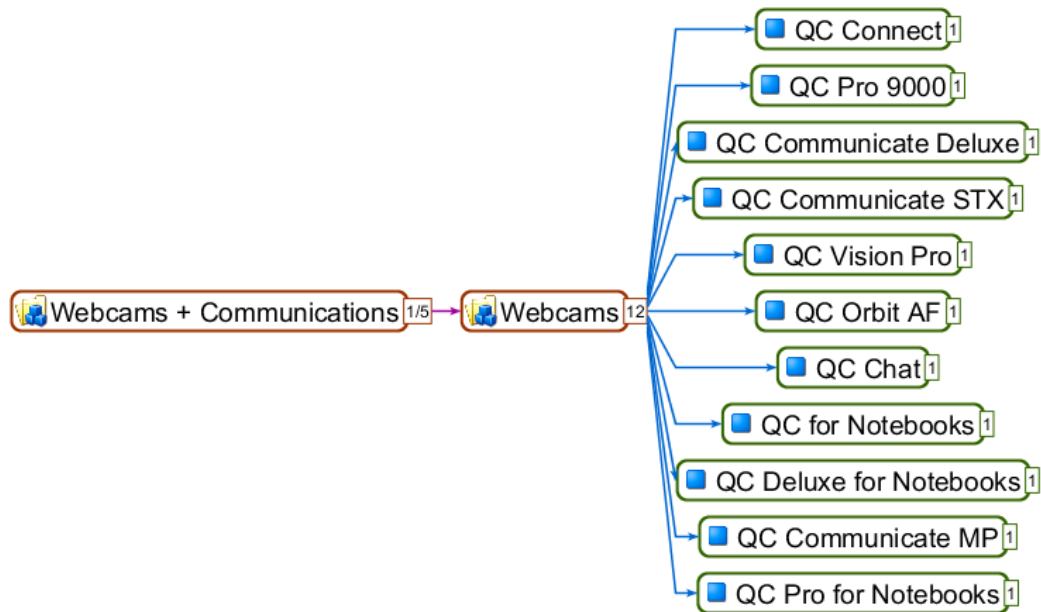
The following image shows the properties for the Product entity type:

Entity Type	
Rowid	4
Code	Product
Display Name	Product
Description	
Color	 0x336600 
Small Icon	 HM_Icons/BulletSquareBlue/bullet_square_blue... 
Large Icon	 HM_Icons/BulletSquareBlue/bullet_square... 

The following image shows the properties for the Product Group entity type:

Entity Type	
Rowid	5
Code	Product Group
Display Name	Product Group
Description	
Color	 0x993300 
Small Icon	 hierarchymanager/Group/Group_Small.png 
Large Icon	 hierarchymanager/Group/Group.png 

The following image shows a hierarchy with the Product and Product Group entity types:



Creating Entity Types

To create an entity type, select an entity object in the **Hierarchies** tool, and then add an entity type.

1. Acquire a write lock.
2. In the **Model** workbench, select the **Hierarchies** tool.

3. In the Hierarchies tool navigation panel, under the **Entity Objects** node, select the entity base object for which you want to create an entity type.
You cannot add an entity type until you first select an entity base object.
4. Select **Hierarchies > Add Entity Type**.
The Hierarchies tool displays an entity type with the name **New Entity Type**.
5. Edit the properties of the entity type, and then click **Save**.

Editing Entity Types

To edit an entity type:

1. In the Hierarchies tool, in the navigation tree, click the entity type to edit.
2. For each field that you want to edit, click the **Edit** button and make the change that you want.
3. When you have finished making changes, click **Save** to save your changes.

Note: If your entity object uses the code column, you probably do not want to modify the entity type code if you already have records for that entity type.

Deleting Entity Types

You can delete any entity type that is not used by any relationship types.

If the entity type is being used by one or more relationship types, attempting to delete it will generate an error.

To delete an entity type:

1. Acquire a write lock.
2. In the Hierarchies tool, in the navigation tree, right-click the entity type that you want to delete, and choose **Delete Entity Type**.
If the entity type is not used by any relationship types, then the Hierarchies tool prompts you to confirm deletion.
3. Choose **Yes**.

The Hierarchies tool removes the selected entity type from the list.

Note: You probably do not want to delete an entity type if you already have entity records that use that type. If your entity object uses the code column instead of the rowid column and you have records in that entity object for the entity type you are trying to delete, you will get an error.

Display Options for Entities

In addition to configuring color and icons for entities, you can also configure the font size and maximum width.

While color and icons can be specified for each entity type, the font size and width apply to entities of all types.

To change the font size in HM, use the HM Font Size and Entity Box Size. The default entity font size (38 pts) and max entity box width (600 pixels) can be overridden by settings in the `cmxserver.properties` file. The settings to use are:

```
sip.hm.entity.font.size=fontSize  
sip.hm.entity.max.width=maxWidth
```

The value for `fontSize` can be from 6 to 100 and the value for `maxWidth` can be from 20 to 5000. If value specified is outside the range, the minimum or maximum values are used. Default values are used if the values specified are not numbers.

Reverting Entity Base Objects to Base Objects

If you inadvertently converted a base object to an entity object, or if you no longer want to work with an entity object in Hierarchy Manager, you can revert the entity object to a base object.

In doing so, you are removing the HM metadata from the object.

To revert an entity base object to a base object:

1. In the Hierarchies tool, acquire a write lock.
2. Right-click on an entity base object and choose **Revert Entity/Relationship Object to BO**.
3. If you are prompted to revert associated relationship objects, click **OK**.

Note: When you revert the entity object, you are also reverting its corresponding relationship objects.

4. In the Revert Entity/Relationship Object dialog box, click **OK**.

A dialog is displayed when the entity is reverted.

Hierarchy Types

When you configure relationship types, you must associate the relationship type with at least one hierarchy type.

These relationship types are not ranked, nor are they necessarily related to each other. They are merely relationship types that are grouped together for ease of classification and identification. The same relationship type can be associated with multiple hierarchies. A hierarchy type is a logical classification of hierarchies.

You configure the following properties when you create a hierarchy type:

Field	Description
Code	Unique code name of the hierarchy. Can be used as a foreign key from Hierarchy Manager relationship base objects. You cannot change the code after you create the hierarchy type.
Display name	Name of this hierarchy as it will be displayed in the Hub Console.
Description	Description of this hierarchy.

When you first create the hierarchy type, the hierarchy type does not have any references. The Hierarchy tool populates the hierarchy type references with entity types, relationship types, and profiles when you associate relationship types with the hierarchy type that you created.

Adding Hierarchies

To add a new hierarchy:

1. In the Hierarchies tool, acquire a write lock.
2. Right-click an entity object in the navigation pane and choose **Add Hierarchy**.
The Hierarchies tool displays a new hierarchy (called New Hierarchy) in the navigation tree under the Hierarchies node. The default properties are displayed in the properties pane.
3. Specify the following properties for this new hierarchy.
4. Click **Save** to save the new hierarchy.

Deleting Hierarchies

You must not delete a hierarchy if you already have relationship records that use the hierarchy.

If your relationship object uses the hierarchy code column instead of the rowid column and you have records in that relationship object for the hierarchy you are trying to delete, you will get an error.

To delete a hierarchy:

1. In the Hierarchies tool, acquire a write lock.
2. In the navigation tree, right-click the hierarchy that you want to delete, and choose **Delete Hierarchy**.
The Hierarchies tool prompts you to confirm deletion.
3. Choose **Yes**.
The Hierarchies tool removes the selected hierarchy from the list.

Note: You are allowed to delete a hierarchy that has relationship types associated with it. There will be a warning with the list of associated relationship types. If you choose to delete the hierarchy, all references to it will automatically be removed.

Relationship Objects

A relationship object describes the affiliation between two specific entities.

A relationship object can be one of the following object:

Relationship Base Object

Maintains a relationship between two entity base objects. Use relationship base objects to establish a many-to-many relationship.

Foreign Key Base Object

A foreign key base object is an entity base object with a foreign key column. Use foreign key base objects to establish a 1-to-1 or 1-to-many relationship. For example, many products might relate to a single product group. An entity base object can have more than one foreign key column. Each foreign key column maintains a relationship.

Hierarchy relationships are defined by specifying the relationship type, hierarchy type, attributes of the relationship, and dates for when the relationship is active.

The Relationship node in the Schema tool display hierarchy relationships and data model relationships.

Relationship Base Objects

A relationship base object is a base object that stores hierarchy relationships. A Foreign Key Relationship Base Object is an Entity Base Object containing a foreign key to another Entity Base Object. A Relationship Base Object is a table that relates the two Entity Base Objects. Foreign key relationship types can only be associated with a single hierarchy.

Only one relationship can exist between two specific entities. If you try to add more relationships between the two entities, the existing relationship is updated instead of a new one being added.

When you modify the base object using the Schema Manager, do not change any of the columns added by Hierarchy Manager. Modifying any of these columns will result in unpredictable behavior and possible data loss.

The following table describes the properties you can configure when you create a relationship base object:

Property	Description
Item Type	Read-only. Already specified.
Display name	Name of this base object as it will be displayed in the Hub Console.
Physical name	Actual name of the table in the database. Informatica MDM Hub will suggest a physical name for the table based on the display name that you enter.
Data tablespace	Name of the data tablespace. For more information, see the <i>Multidomain MDM Installation Guide</i> .
Index tablespace	Name of the index tablespace. For more information, see the <i>Multidomain MDM Installation Guide</i> .
Description	Description of this base object.
Entity Base Object 1	Entity base object to be linked through this relationship base object.
Display name	Name of the column that is a FK to the entity base object 1.
Physical name	Actual name of the column in the database. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.
Entity Base Object 2	Entity base object to be linked through this relationship base object.
Display name	Name of the column that is a FK to the entity base object 2.
Physical name	Actual name of the column in the database. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.

Property	Description
Hierarchy FK Column	Column used as the foreign key for the hierarchy; can be either ROWID or CODE. The ability to choose a BO Class CODE column reduces the complexity by allowing you to define the foreign key relationship based on a predefined code, rather than the Informatica MDM Hub generated ROWID.
Hierarchy FK Display Name	Name of this FK column as it will be displayed in the Hub Console
Hierarchy FK Physical Name	Actual name of the hierarchy foreign key column in the table. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.
Rel Type FK Column	Column used as the foreign key for the relationship; can be either ROWID or CODE.
Rel Type Display Name	Name of the column that is used to store the Rel Type CODE or ROWID.
Rel Type Physical Name	Actual name of the relationship type FK column in the table. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.

Creating Relationship Base Objects

To create a relationship base object, use the **Hierarchies** tool to configure the foreign key constraints.

1. Acquire a write lock.
2. In the **Model** workbench, select the **Hierarchies** tool.
3. Select **Hierarchies > Create New Entity/Relationship Object**.
4. From the **Create New Entity/Relationship Base Object** dialog box, select **Create New Relationship Base Object**. Click **OK**.
5. From the **Create New Relationship Base Object** dialog box, specify the relationship base object properties. Click **OK**.

Converting Base Objects to Relationship Base Objects

Relationship base objects are tables that contain information about two entity base objects.

Base objects do not have the metadata required by the Hierarchy Manager for relationship information. To use base objects, first convert the base objects to entity objects, and then select the entity objects you want to relate in a relationship object.

1. In the Hierarchies tool, acquire a write lock.
2. Right-click in the navigation pane and choose **Convert BO to Entity/Relationship Object**.
3. Click **OK**.

The Convert to Relationship Base Object screen is displayed.

- Specify the following properties for this base object.

Field	Description
Entity Base Object 1	Entity base object to be linked via this relationship base object.
Display name	Name of the column that is a FK to the entity base object 1.
Physical name	Actual name of the column in the database. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.
Entity Base Object 2	Entity base object to be linked via this relationship base object.
Display name	Name of the column that is a FK to the entity base object 2.
Physical name	Actual name of the column in the database. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.
Hierarchy FK Column	Column used as the foreign key for the hierarchy; can be either ROWID or CODE. The ability to choose a BO Class CODE column reduces the complexity by allowing you to define the foreign key relationship based on a predefined code, rather than the Informatica MDM Hub-generated ROWID.
Existing BO Column to Use	Actual column in the existing BO to use.
Hierarchy FK Display Name	Name of this FK column as it will be displayed in the Hub Console
Hierarchy FK Physical Name	Actual name of the hierarchy foreign key column in the table. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.
Rel Type FK Column	Column used as the foreign key for the relationship; can be either ROWID or CODE.
Existing BO Column to Use	Actual column in the existing BO to use.
Rel Type FK Display Name	Name of the FK column that is used to store the Rel Type CODE or ROWID.
Rel Type FK Physical Name	Actual name of the relationship type FK column in the table. Informatica MDM Hub will suggest a physical name for the column based on the display name that you enter.

- Click **OK**.

Note: When you modify the base object using the Schema Manager tool, do not change any of the columns added by the Hierarchy Manager. Modifying these columns will result in unpredictable behavior and possible data loss.

Reverting Relationship Base Objects to Base Objects

Revert relationship base objects to base objects to remove the Hierarchy Manager metadata from the relationship object. The relationship object remains as a base object, but Hierarchy Manager does not display the base object.

If the relationship-type column that you want to revert is in the staging table for a lookup, the staging table column must be empty before you revert the relationship base object.

If you are upgrading Hierarchy Manager relationships, copy relationships in Hierarchy Manager to the Provisioning tool before you revert the relationship base object.

1. In the Hierarchies tool, acquire a write lock.
2. Right-click on the relationship base object and choose **Revert Entity/Relationship Object to BO**.
3. In the **Revert Entity/Relationship Object** dialog box, click **OK**.

A dialog box appears when the entity is reverted.

Foreign Key Relationship Base Objects

A foreign key relationship base object is an entity base object with a foreign key to another entity base object.

You must have an entity base object and add a foreign key column to convert to a foreign key relationship base object.

When you create a foreign key relationship base object, you select the entity base object to add a foreign key column to.

Field	Description
FK Constraint Entity BO 1	Select FK entity base object from the list.
Existing BO Column to Use	Name of existing base object column used for the foreign key, or choose to create a new column.
FK Column Display Name 1	Name of FK column as it will be displayed in the Hub Console.
FK Column Physical Name 1	Actual name of FK column in the database. The Hub Console suggests a physical name for the table based on the display name that you enter.
FK Column Represents	Choose Entity1 or Entity2, depending on what the FK Column represents in the relationship.

To use foreign key relationships, create the entity base objects you want to relate. Use foreign key relationships when you have a one-to-many relationship between entities.

When you create the foreign key relationship, you need to add a foreign key column for each entity you want to relate to.

Creating Foreign Key Relationship Base Objects

To create foreign key relationship base objects, use the **Hierarchies** tool.

1. Acquire a write lock.
2. In the **Model** workbench, select the **Hierarchies** tool.
3. Select **Hierarchies > Create Foreign Key Relationship Object**.
4. From the **Modify Existing Base Object** dialog box, select the entity base object to add the foreign key to, and specify the number of foreign key columns. Click **OK**.
5. Select the foreign key column properties, and then click **OK**.

Relationship Types

A relationship type describes classes of relationships and defines the types of entities that a relationship of this type can include, the direction of the relationship (if any), and how the relationship is displayed in the Hub Console. You can create relationship types for relationship base objects or for foreign key relationships.

Note: Relationship Type is a physical construct and can be configuration heavy, while Hierarchy Type is more of a logical construct and is typically configuration light. Therefore, it is often easier to have many Hierarchy Types than to have many Relationship Types. Be sure to understand your data and hierarchy management requirements prior to defining Hierarchy Types and Relationship Types within the MDM Hub.

A well defined set of hierarchy relationship types has the following characteristics:


- It reflects the real-world relationships between your entity types.
- It supports multiple relationship types for each relationship.



Field	Description
Code	Unique code name of the rel type. Can be used as a foreign key from hierarchy relationship base objects.
Display name	Name of this relationship type as it will be displayed in the Hub Console. Specify a unique, descriptive name.
Description	Description of this relationship type.
Color	Color of the relationships associated with this relationship type as they will be displayed in the Hub Console in the Hierarchy Manager Console and Informatica Data Director.
Entity Type 1	First entity type associated with this new relationship type. Any entities of this type will be able to have relationships of this relationship type.
Entity Type 2	Second entity type associated with this new relationship type. Any entities of this type will be able to have relationships of this relationship type.
Direction	Select a direction for the new relationship type to allow a directed hierarchy. The possible directions are: <ul style="list-style-type: none">- Entity 1 to Entity 2- Entity 2 to Entity 1- Undirected- Bi-Directional- Unknown An example of a directed hierarchy is an organizational chart, with the relationship reports to being directed from employee to supervisor, and so on, up to the head of the organization.
FK Rel Start Date	The start date of the foreign key relationship.
FK Rel End Date	The end date of the foreign key relationship.
Hierarchies	Check the check box next to any hierarchy that you want associated with this new relationship type. Any selected hierarchies can contain relationships of this relationship type.

Relationship Type Example

In our example, we have two types of relationships. In one relationship, the product group entity type is a parent of the product entity type. In the second relationship, the product group entity type is a parent of a product group entity type.

The following figure shows the properties of the Product Group is parent of Product relationship type for the Product Rel relationship object:

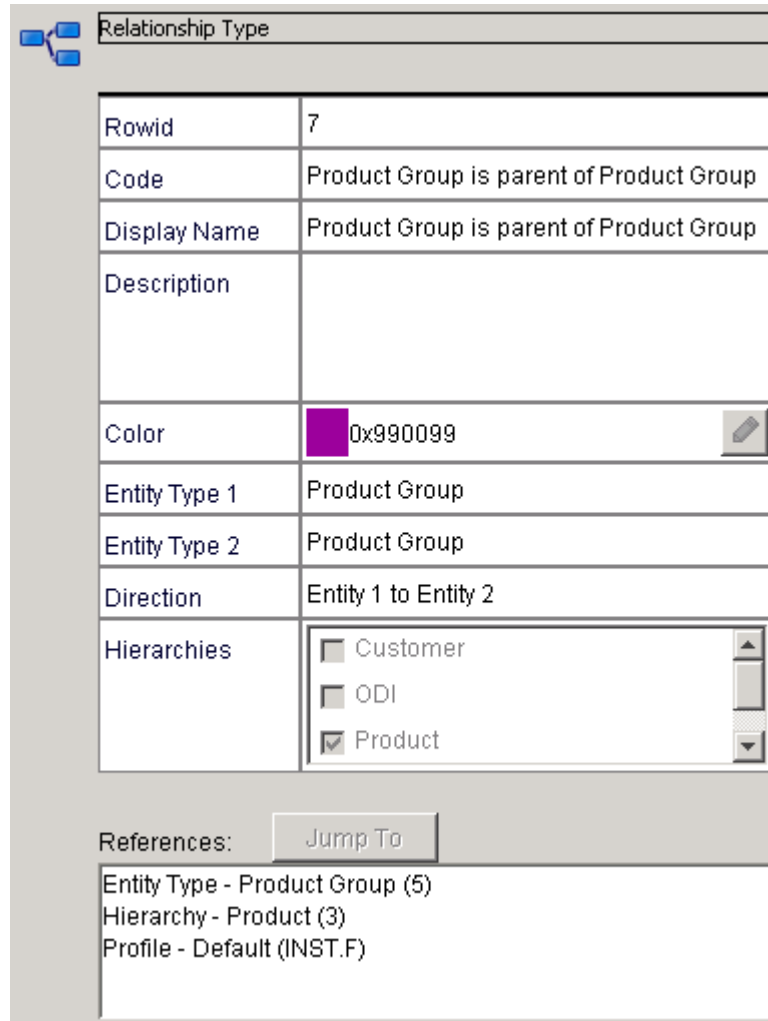



Relationship Type	
Rowid	9
Code	Product Group is parent of Product
Display Name	Product Group is parent of Product
Description	
Color	 0x0066CC 
Entity Type 1	Product Group
Entity Type 2	Product
Direction	Entity 1 to Entity 2
Hierarchies	<input type="checkbox"/> Customer <input type="checkbox"/> ODI <input checked="" type="checkbox"/> Product

References: [Jump To](#)

- Entity Type - Product (4)
- Entity Type - Product Group (5)
- Hierarchy - Product (3)
- Profile - Default (INST.F)

The following figure shows the properties of the Product Group is parent of Product Group relationship type for the Product Rel relationship object:



Rowid	7
Code	Product Group is parent of Product Group
Display Name	Product Group is parent of Product Group
Description	
Color	0x990099 
Entity Type 1	Product Group
Entity Type 2	Product Group
Direction	Entity 1 to Entity 2
Hierarchies	<input type="checkbox"/> Customer <input type="checkbox"/> ODI <input checked="" type="checkbox"/> Product

References: [Jump To](#)

Entity Type - Product Group (5)
Hierarchy - Product (3)
Profile - Default (INST.F)

Creating Relationship Types

To create a relationship type, use the **Hierarchies** tool to add a relationship type to a relationship base object.

1. Acquire a write lock.
2. In the **Model** workbench, select the **Hierarchies** tool.
3. In the **Hierarchies** tool navigation pane, select the relationship base object for which you want to create a relationship type.

You cannot add a relationship type until you first select a relationship base object.

4. Select **Hierarchies > Add Relationship Type**.

The Hierarchies tool displays a relationship type with the name **New Rel Type**.

5. Edit the properties of the relationship type, and then click **Save**.

Editing Relationship Types

To edit a relationship type:

1. In the Hierarchies tool, acquire a write lock.
2. In the navigation tree, click the relationship type that you want to edit.
3. For each field that you want to edit, click **Edit** and make the change that you want.
4. When you have finished making changes, click **Save** to save your changes.

Note: If your relationship object uses the code column, you probably do not want to modify the relationship type code if you already have records for that relationship type.

This warning does not apply to FK relationship types.

Deleting Relationship Types

Note: You probably do not want to delete a relationship type if you already have relationship records that use the relationship type. If your relationship object uses the relationship type code column instead of the rowid column and you have records in that relationship object for the relationship type you are trying to delete, you will get an error.

The above warnings are not applicable to FK relationship types. You can delete relationship types that are associated with hierarchies. The confirmation dialog displays the hierarchies associated with the relationship type being deleted.

To delete a relationship type:

1. In the Hierarchies tool, acquire a write lock.
2. In the navigation tree, right-click the relationship type that you want to delete, and choose **Delete Relationship Type**.

The Hierarchies tool prompts you to confirm deletion.

3. Choose **Yes**.

The Hierarchies tool removes the selected relationship type from the list.

Packages

This section describes how to add packages to your schema using the Hierarchies tool. You can create packages for entity base objects, relationship base objects, and foreign key relationship base objects. If records will be inserted or changed in the package, be sure to enable the Put option.

A package is a public view of one or more underlying tables in Informatica MDM Hub. Packages represent subsets of the columns in those tables, along with any other tables that are joined to the tables. A package is based on a query. The underlying query can select a subset of records from the table or from another package. Packages are used for configuring user views of the underlying data.

You must first create a package, then you must associate it with Entity Types or Relationship Types. In the case of display packages for foreign key relationship objects, you must include REL_START_DATE and REL_END_DATE columns in the package.

Note: You must configure a package and validate the associated profile before you run a load job.

Hierarchy Manager Data Configuration

You can configure what data users have access in the Hierarchy Manager tool.

When you configure hierarchy packages, you configure which entity fields appear in the user interface and in what order. For each field in the package, You can select a number starting from 0. If you assign a field a value of 0, that field will appear in the user interface at the top of the list of fields.

You can configure the which fields and in what order the fields appear for the following Hierarchy Manager user interface elements:

Label

You use the number to specify which entity fields appear in the Hierarchy Manager tool.

Tooltip

The column that you assign the lowest value to is the field which appears in the tooltip when you place your mouse over an entity in the Hierarchy Manager tool.

Common

The fields to appear when entities and relationships of different types appear in the same list. The fields you select must be in all hierarchy manager packages.

Search

The fields that you can use to search for records.

List

The fields that appear in the list of search results.

Detail

The fields that appear when you choose to view the details of a selected entity.

Put

The fields for whose value you can change when you edit a selected entity.

Add

The fields that appear when you add an entity in the Hierarchy Manager.

Creating Entity, Relationship, and FK Relationship Object Packages

To create an HM package:

1. In the Hierarchies tool, right-click anywhere in the navigation pane and choose **Create New Package**.
2. Acquire a write lock.

The Hierarchies tool starts the Create New Package wizard and displays the first dialog box.

3. Specify the following information for this new package.

Field	Description
Type of Package	One of the following types: Entity Object Relationship Object FK Relationship Object
Query Group	Select an existing query group or choose to create a new one. In Informatica MDM Hub, query groups are logical groups of queries. Select the query group under which the query is organized.
Query group name	Name of the new query group - only needed if you chose to create a new group above.
Description	Optional description for the new query group you are creating.

4. Click **Next**.

The Create New Package wizard displays the next dialog box.

5. Specify the following information for this new package.

Field	Description
Query Name	Name of the query. In Informatica MDM Hub, a <i>query</i> is a request to retrieve data from the Hub Store.
Description	Optional description.
Select Primary Table	Primary table for this query.

6. Click **Next**.

The Create New Package wizard displays the next dialog box.

7. Specify the following information for this new package.

Field	Description
Display Name	Display name for this package, which will be used to display this package in the Packages tool.
Physical Name	Physical name for this package. The Hub Console will suggest a physical name based on the display name you entered.
Description	Optional description.

Field	Description
Enable PUT	Select to enable records to be inserted or changed. (optional) If you do not choose this, your package will be read only. If you are creating a foreign key relationship object package, you have additional steps in Step 9 of this procedure. Note: You must have both a PUT and a non-PUT package for every Foreign Key relationship. Both Put and non-Put packages that you create for the same foreign key relationship object must have the same columns.
Secure Resource	Select to create a secure resource. (optional)

8. Click **Next**.

The Create New Package wizard displays a final dialog box. The dialog box you see depends on the type of package you are creating.

If you selected to create either a package for entities or relationships or a PUT package for FK relationships, a dialog box similar to the following dialog box is displayed. The required columns (shown in grey) are automatically selected – you cannot deselect them.

Deselect the columns that are not relevant to your package.

Note: You must have both a PUT and a non-PUT package for every Foreign Key relationship. Both Put and non-Put packages that you create for the same foreign key relationship object must have the same columns.

If you selected to create a non-Put enabled package for foreign key relationships (see Step 7 of this procedure - do not check the Put check box), the following dialog box is displayed.

9. If you are creating a non-Put enabled package for foreign key relationships, specify the following information for this new package.

Field	Description
Hierarchy	Hierarchy associated with this package.
Relationship Type	Relationship type associated with this package.

Note: You must have both a PUT and a non-PUT package for every Foreign Key relationship. Both Put and non-Put packages that you create for the same foreign key relationship object must have the same columns.

10. Select the columns for this new package.
11. Click **Finish** to create the package.

Use the Packages tool to view, edit, or delete this newly-created package.

You should not remove columns that are needed by Hierarchy Manager. These columns are automatically selected (and greyed out) when the user creates packages using the Hierarchies tool.

Assigning Packages to Entity or Relationship Types

After you create a profile, and a package for each of the entity/relationship types in a profile, you must assign the packages to an entity or relationship type. This defines what fields are displayed when an entity is displayed in Hierarchy Manager. You can also assign a package for relationship types and entity types. To assign a package to an entity/relationship type:

1. Acquire a write lock.

2. In the Hierarchies tool, select the Entity/Relationship Type.

The Hierarchy Manager displays the properties for the Package for that type if they exist, or the same properties pane with empty fields. When you make the display and Put package selections, the Hierarchy Manager package column information is displayed in the lower panel.

The numbers in the cells define the sequence in which the attributes are displayed.

3. Configure the package for your entity or relationship type.

Field	Description
Label	Columns used to display the label of the entity/relationship you are viewing in the Hierarchy Manager graphical console. These columns are used to create the Label Pattern in the Hierarchy Manager Console and Informatica Data Director. To edit a label, click the label value to the right of the label. In the Edit Pattern dialog, enter a new label or double-click a column to use it in a pattern.
Tooltip	Columns used to display the description or comment that appears when you scroll over the entity/relationship. Used to create the tooltip pattern in the Hierarchy Manager Console and Informatica Data Director. To edit a tooltip, click the tooltip pattern value to the right of the Tooltip Pattern label. In the Edit Pattern dialog, enter a new tooltip pattern or double-click a column to use it in a pattern.
Common	Columns used when entities/relationships of different types are displayed in the same list. The selected columns must be in packages associated with all Entity/Relationship Types in the Profile.
Search	Columns that can be used with the search tool.
List	Columns to be displayed in a search result.
Detail	Columns used for the detailed view of an entity/relationship displayed at the bottom of the screen.
Put	Columns that are displayed when you want to edit a record.
Add	Columns that are displayed when you want to create a new record.

4. When you have finished making changes, click **Save** to save your changes.

About Profiles

A hierarchy profile defines user access to hierarchy objects

A profile determines what fields and records a user may display, edit, or add in the Hierarchy Manager. For example, one profile can allow full read/write access to all entities and relationships, while another profile can be read-only (no add or edit operations allowed). Once you define a profile, you can configure it as a secure resource.

Adding Profiles

A new profile (called Default) is created automatically for you before you access the HM. The default profile can be maintained, and you can also add additional profiles.

Note: The Informatica Data Director uses the Default Profile to define how Entity Labels as well as Relationship and Entity Tooltips are displayed. Additional Profiles, as well as the additional information defined within Profiles, is only used within the Hierarchy Manager Console and not the Informatica Data Director.

To add a new profile:

1. Acquire a write lock.
2. In the Hierarchy tool, right-click anywhere in the navigation pane and choose **Add Profiles**.

The Hierarchies tool displays a new profile (called **New Profile**) in the navigation tree under the Profiles node. The default properties are displayed in the properties pane.

When you select these relationship types and click Save, the tree below the Profile will be populated with Entity Objects, Entity Types, Rel Objects and Rel Types. When you deselect a Rel type, only the Rel types will be removed from the tree - not the Entity Types.

3. Specify the following information for this new profile.

Field	Description
Name	Unique, descriptive name for this profile.
Description	Description of this profile.
Relationship Types	Select one or more relationship types associated with this profile.

4. Click **Save** to save the new profile.

The Hierarchies tool displays information about the relationship types you selected in the References section of the screen. Entity types are also displayed. This information is derived from the relationship types you selected.

Editing Profiles

To edit a profile:

1. Acquire a write lock.
2. In the Hierarchies tool, in the navigation tree, click the profile that you want to edit.
3. Configure the profile as needed (specifying the appropriate profile name, description, and relationship types and assigning packages).
4. Click **Save** to save your changes.

Validating Profiles

To validate a profile:

1. Acquire a write lock.
2. In the Hierarchies tool, in the navigation pane, select the profile to validate.
3. In the properties pane, click the Validate tab.

Note: Profiles can be successfully validated only after the packages are assigned to Entity Types and Relationship Types.

The Hierarchies tool displays the Validate tab.

4. Select a sandbox to use.
For information about creating and configuring sandboxes, see the *Multidomain MDM Data Steward Guide*.
5. To validate the data, check **Validate Data**. This may take a long time if you have a lot of records.
6. To start the validation process, click **Validate HM Configuration**.
The Hierarchies tool displays a progress window during the validation process. The results of the validation appear in the window below the buttons.
7. When the validation is finished, click **Save**.
8. Choose the directory where the validation report will be saved.
9. Click **Clear** to clear the box containing the description of the validation results.

Copying Profiles

To copy a profile:

1. Acquire a write lock.
2. In the Hierarchies tool, right-click the profile that you want to copy, and then choose **Copy Profile**.
The Hierarchies tool displays a new profile (called New Profile) in the navigation tree under the Profiles node. This new profile is an exact copy (with a different name) of the profile that you selected to copy. The default properties are displayed in the properties pane.
3. Configure the profile as needed (specifying the appropriate profile name, description, relationship types, and assigning packages).
4. Click **Save** to save the new profile.

Deleting Profiles

To delete a profile:

1. Acquire a write lock.
2. In the Hierarchies tool, right-click the profile that you want to delete, and choose **Delete Profile**.
The Hierarchies tool displays a window that warns that packages will be removed when you delete this profile.
3. Click **Yes**.
The Hierarchies tool removes the deleted profile.

Deleting Relationship Types from a Profile

To delete a relationship type:

1. Acquire a write lock.
2. In the Hierarchy tool, right-click the relationship type and choose **Delete Entity Type/Relationship Type From Profile**.

If the profile contains relationship types that use the entity/relationship type that you want to delete, you will not be able to delete it unless you delete the relationship type from the profile first.

Deleting Entity Types from a Profile

To delete an entity type:

1. Acquire a write lock.
2. In the Hierarchy tool, right-click the entity type and choose **Delete Entity Type/Relationship Type From Profile**.

If the profile contains relationship types that use the entity type that you want to delete, you will not be able to delete it unless you delete the relationship type from the profile first.

Assigning Packages to Entity and Relationship Types

After you create a profile, you must:

- Assign packages to the entity types and relationship types associated with the profile.
- Configure the package as a secure resource.

CHAPTER 14

Hierarchy Manager Tutorial

This chapter includes the following topics:

- [Hierarchy Configuration Overview, 223](#)
- [About the Tutorial Example, 225](#)
- [Step 1. Create the Product Entity Base Object, 225](#)
- [Step 2. Create the Entity Types, 226](#)
- [Step 3. Create the Product Relationship Object, 229](#)
- [Step 4. Create the Relationship Types, 230](#)
- [Step 5. Create a Hierarchy Type, 233](#)
- [Step 6. Add the Relationship Types to the Hierarchy Profile, 234](#)
- [Step 7. Create the Packages, 234](#)
- [Step 8. Assign the Packages, 236](#)
- [Step 9. Configure the Display of Data in Hierarchy Manager, 239](#)
- [Hierarchy Management, 257](#)

Hierarchy Configuration Overview

An MDM hierarchy shows the relationships between records in the MDM Hub. The relationships can be between records in the same entity base object or between records in different entity base objects. You must configure the hierarchy before you can populate the hierarchy with data.

You need to be familiar with the data and decide which relationships to establish before you configure the hierarchy. For this tutorial, we have product records that we want to classify into different product groups. This tutorial describes the steps to configure the product hierarchy.

To configure the product hierarchy, perform the following steps:

1. Create the Product entity base object. The Product entity base object stores the data for the Product Group entity type and the Product entity type.
2. Create the entity types:
 - a. Create the Product entity type. The records that contain the data for each product will be a Product entity type.
 - b. Create the Product Group entity type. The product categories in the hierarchy will be a Product Group entity type.

3. Create the Product Rel relationship base object. The Product Rel relationship base object stores the data for the Product Group is parent of Product Group relationship type and the Product Group is parent of Product relationship type.
4. Create the relationship types:
 - a. Create the Product Group is parent of Product Group relationship type.
 - b. Create the Product Group is parent of Product relationship type.
5. Create a hierarchy configuration. The hierarchy configuration associates the entity types, relationship types, and hierarchy profile with a hierarchy.
6. Add the relationship types to the default hierarchy profile. When you add the relationship types to the hierarchy profile, the entity types that are associated with the relationship types are also added to the hierarchy profile.
7. Create the packages. The packages and associated queries define the data that a user can access in the Hierarchy Manager:
 - a. Create the Product entity object package.
 - b. Create the Product Rel relationship object package.
8. Assign the packages:
 - a. Assign the PKG Product package to the Product entity type.
 - b. Assign the PKG Product package to the Product Group entity type.
 - c. Assign the PKG Product Rel package to the Product Group is parent of Product Group relationship type.
 - d. Assign the PKG Product Rel package to the Product Group is parent of Product relationship type.
9. Configure the display of data in the Hierarchy Manager:
 - a. Configure the entity object label for each entity type.
 - b. Configure the entity object tooltip text for each entity type.
 - c. Configure the relationship object tooltip text for each relationship type.
 - d. Configure the entity lists for each entity type.
 - e. Configure the relationship lists for each relationship type.
 - f. Configure the entity search fields for each entity type.
 - g. Configure the relationship search fields for each relationship type.
 - h. Configure the entity search results for each entity type.
 - i. Configure the relationship search results for each entity type.
 - j. Configure the entity details for each entity type.
 - k. Configure the relationship details for each relationship type.
 - l. Configure the editable entity fields.
 - m. Configure the entity creation fields for each entity type.
 - n. Configure the relationship creation fields for each relationship type.

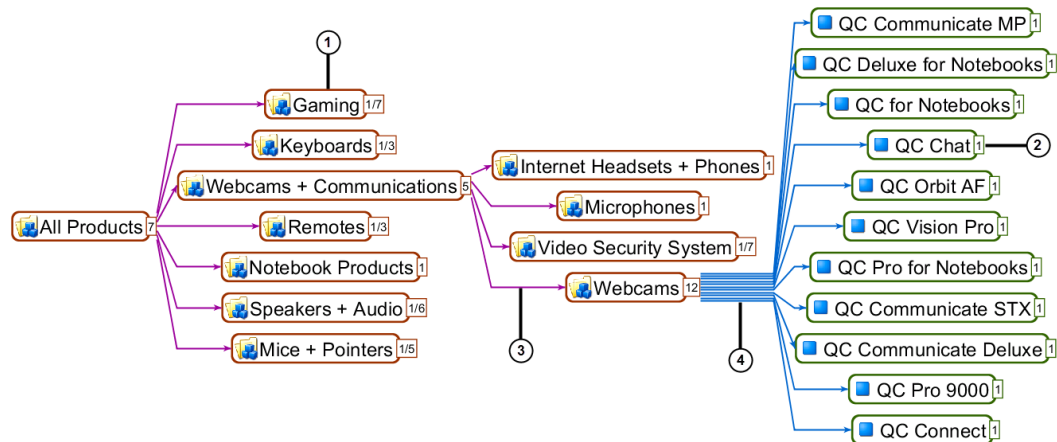
About the Tutorial Example

This tutorial describes the steps to create a product hierarchy. The hierarchy consists of products that are organized into product groups.

The hierarchy that we will configure in the tutorial comes preconfigured and populated with data in the sample Operational Reference Store that is provided in the Resource Kit. Import the sample Operational Reference Store if you want to refer to the completed hierarchy. For more information about the sample Operational Reference Store, see the *Multidomain MDM Sample ORS Guide*.

The hierarchy in the tutorial has two entity types and two relationship types. The entity types are Product Group and Product. The two relationship types are Product Group is parent of Product Group, and Product Group is parent of Product. The data for the entity types are stored in the Product entity base object. The data for the relationship types are stored in the Product Rel relationship base object.

The following image shows a portion of the product hierarchy as it appears in the hierarchy view of the Hub Console Hierarchy Manager:



1. Product Group entity
2. Product entity
3. Product Group is parent of Product Group relationship
4. Product Group is parent of Product relationship

Foreign Key Relationships

A foreign key relationship maintains the relationship between two entity base objects using a foreign key field.

In this tutorial, the relationships in the hierarchy are maintained in the Product Rel relationship base object, so a relationship based on foreign key field is not required.

Step 1. Create the Product Entity Base Object

We need to create a Product entity base object that stores the records of the Product and Product Group entity types. We will assume that we have an empty state management-enabled Product base object that we

can convert to an entity base object. When you convert a base object to an entity base object, the Hub Console adds the required hierarchy columns to the base object.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. Select **Hierarchies > Convert BO to Entity/Relationship Object**.
4. In the Modify Existing Base Object dialog box, select the Product base object from the list. Select **Convert to Entity Base Object** and click **OK**.
5. Click **OK**.
6. Select the following Foreign Key Column for Entity Types field parameters:

Foreign Key Column for Entity Types

Select **Base Object Class Code** to establish the relationship based on a value of your choosing.

Existing Base Object Column to Use

Select **Create New Column**.

Display Name

The display name of the foreign key column. Enter `Product Type`.

Physical Name

The name of the foreign key column. Enter `PRODUCT_TYPE`.

Step 2. Create the Entity Types

We will create a Product entity type and a Product Group entity type. The Product entities will be the product records that contain the product data. The Product Group entities will be used to classify the products into groups. For example, the Webcams product group will contain all the webcam product records. When you create the entity types, you configure the foreign key and how the entity types appear in the Hierarchy Manager.

You can configure the following entity type parameters:

Code

Unique code name of the entity type.

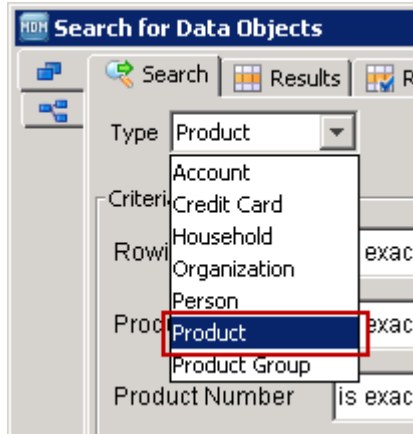
The following image shows the Product foreign key code as it appears in the cells of a webcam product record:

3. Cell data:		
Column Name	...	Base Object Cell
Rowid Object		118
Name		QC Deluxe for Notebooks
Number		960-000043
Description		Stylish design with glass-element lens performance to match.
Product Type Cd		Webcam
Product Type		Product
Hub State Indicator		Active

Display Name

Name of this entity type that is displayed in the Hierarchies tool.

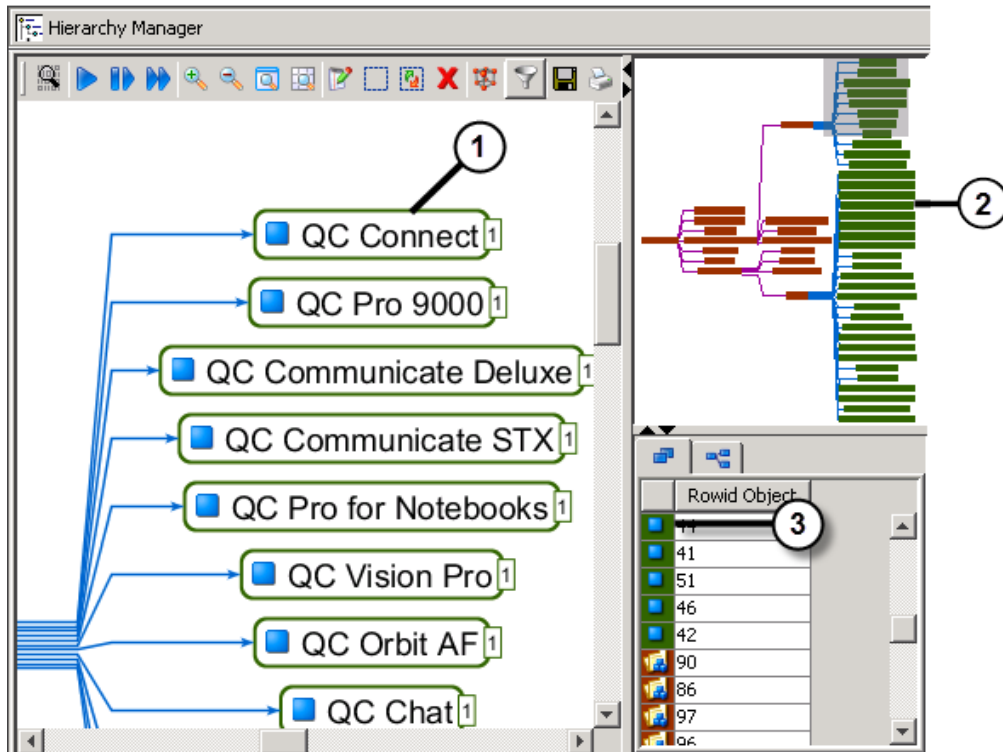
The following image shows the Product display name as it appears in the Hierarchy Manager search window:



Color

Color of the entities associated with this entity type as they appear in the Hierarchy Manager tool.

The following image shows the colors that the entity type color affects in the Hierarchy Manager tool:



1. Outline color of entity in the hierarchy view.
2. Color of the entity in the overview of the hierarchy.
3. Background color of the entity in the entity table.

Small Icon

Icon for entities associated with the entity type as the icons appear in tables in the Hierarchy Manager.

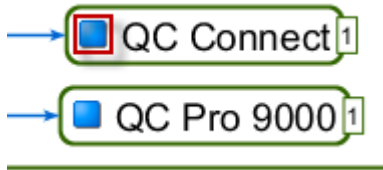
The following image shows the small Product entity type icon as it appears in a Hierarchy Manager table:

Rowid	Object
44	
41	
51	
46	
42	
90	
86	
97	
96	

Large Icon

Icon for entities associated with this entity type as the icons appear in the hierarchy view pane of the Hierarchy Manager.

The following image shows the large Product entity type icon as it appears in the hierarchy view:



Create the Product Entity Type

The product records that contain the product data are of the Product entity type.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the Hierarchies tool navigation panel, under the **Entity Objects** node, select the Product entity base object.
4. Select **Hierarchies > Add Entity Type**.
5. Enter the following Entity Type properties:

Code

Type Product

Display Name

Type Product

Color

From the **Choose Color** dialog, select the **RGB** tab, and then enter Color Code 336600.

Small Icon

Select the small blue square image.

Large Icon

Select the large blue square image.

6. Click **Save**.

Create the Product Group Entity Type

The Product Group entities will be used to classify the products into groups. For example, the Webcams product group will contain all the webcam product records.

1. In the **Model** workbench, click **Hierarchies**.
2. In the Hierarchies tool navigation panel, under the **Entity Objects** node, select the Product entity base object.
3. Select **Hierarchies > Add Entity Type**.
4. Enter the following Entity Type properties:

Code

Type Product Group

Display Name

Type Product Group.

Color

From the **Choose Color** dialog, select the **RGB** tab, and then enter Color Code 993300.

Small Icon

Select the small blue square group image.

Large Icon

Select the large blue square group image.

5. Click **Save**.

Step 3. Create the Product Relationship Object

The Product relationship object maintains the hierarchy relationships associated with the Product entity object. To create the Product relationship object, you can either convert an existing base object, or create the Product relationship object. In this tutorial, we will create the Product relationship object.

1. In the Model workbench, click **Hierarchies**.
2. Select **Hierarchies > Create New Entity/Relationship Object**.
3. Select **Create New Relationship Base Object**. Click **OK**.
4. Enter the following parameters for the relationship object:

Parameter	Parameter Value
Display name	Product Rel
Physical name	C_PRODUCT_REL
Data Tablespace	CMX_DATA
Index Tablespace	CMX_INDX

Parameter	Parameter Value
Description	Optional.
Secure Resource	Enabled
Entity Base Object 1	Product
Display Name	Product ID1
Physical Name	PRODUCT_ID1
Entity Base Object 2	Product
Display Name	Product ID2
Physical Name	PRODUCT_ID2
Hierarchy FK Column	Row ID Object
Hierarchy FK Display Name	Row ID Hierarchy
Hierarchy FK Physical Name	ROWID_HIERARCHY
Rel Type FK Column	Row ID Object
Rel Type FK Display Name	Row ID Rel Type
Rel Type FK Physical Name	ROWID_REL_TYPE

- Click **OK**.

Step 4. Create the Relationship Types

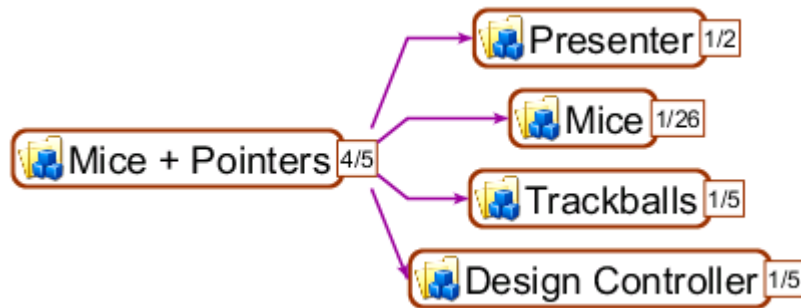
Relationship types define the relationships between entity types and the appearance of the relationships when you view the hierarchy.

We will create the following relationship types:

Product Group is parent of Product Group

A relationship type where a product group entity is the parent of another product group entity. For example, the product group Mice + Pointers is the parent of the Presenter, Mice, Trackballs, and Design Controller product groups.

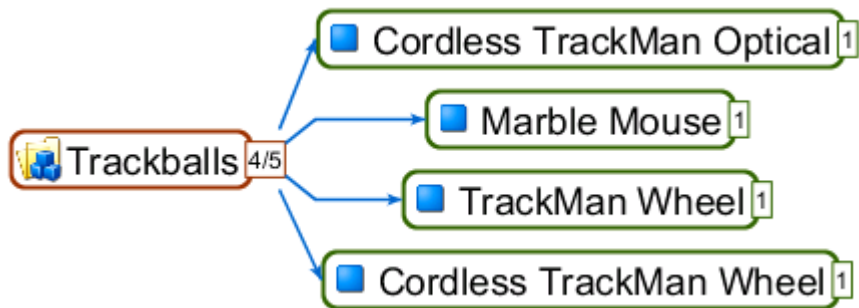
The following image shows a relationship where a product group is the parent of other product groups:



Product Group is parent of Product

A relationship type where a product group entity is the parent of a product entity. For example, the Trackballs product group is the parent of the Cordless TrackMan Optical, Marble Mouse, TrackMan Wheel, and Cordless TrackMan Wheel products.

The following image shows a relationship where a product group is the parent of products:



You can configure the following relationship type parameters:

Code

The code for the relationship in the hierarchy configuration.

Display Name

The relationship name as it appears in the Hierarchies tool.

Description

Optional.

Color

The color of the relationship arrow between the two entities.

Entity Type 1

The first of two entity types that is in the relationship.

Entity Type 2

The other entity type that is in the relationship.

Direction

The direction of the relationship. For example, if you select **Entity 1 to Entity 2**, Entity 1 is the parent of Entity 2.

Hierarchies

The hierarchies to which the relationship belongs.

When you create the relationship types, you configure the relationship, how the relationship type appears in the Hierarchy Manager, and the hierarchies to which the relationship belongs.

Create the Product Group is Parent of Product Group Relationship Type

The Product Group is Parent of Product Group relationship type maintains relationships between product group entities.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the Hierarchies tool navigation panel, under **Relationship Objects**, select the Product Rel relationship object. Select **Hierarchies > Add Relationship**.
4. Enter the following relationship type parameters:

Code

Enter Product Group is parent of Product Group.

Display Name

Enter Product Group is parent of Product Group.

Description

Optional.

Color

Color of the arrow that represents the relationship in the Hierarchy Manager. From the **Choose Color** dialog, select the **RGB** tab, and then enter Color Code 0066CC.

Entity Type 1

Select Product Group is parent of Product Group.

Entity Type 2

Select Product Group is parent of Product Group.

Direction

Select Entity 1 to Entity 2.

Hierarchies

Select the Product hierarchy.

5. Click **Save**.

Create the Product Group is Parent of Product Relationship Type

The Product Group is Parent of Product relationship type maintains relationships between product group entities and product entities.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the Hierarchies tool navigation panel, under **Relationship Objects**, select the Product Rel relationship object. Select **Hierarchies > Add Relationship**.

4. Enter the following relationship type parameters:

Code

Enter `Product Group` is parent of `Product`.

Display Name

Enter `Product Group` is parent of `Product`.

Description

Optional.

Color

Color of the arrow that represents the relationship in the Hierarchy Manager. From the **Choose Color** dialog, select the **RGB** tab, and then enter Color Code `990099`.

Entity Type 1

Select `Product Group`.

Entity Type 2

Select `Product`.

Direction

Select Entity 1 to Entity 2.

Hierarchies

Select the `Product` hierarchy.

5. Click **Save**.

Step 5. Create a Hierarchy Type

Create a hierarchy type to associate the entity types, relationship types, and profiles with a hierarchy. You must create a hierarchy type before you create relationship types.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. Select **Hierarchies > Add Hierarchy**.
4. Enter the following Hierarchy properties:

Code

Unique code name of the hierarchy. Type `Product`.

Display Name

Name of the hierarchy as it appears in the Hub Console. Type `Product`.

Description

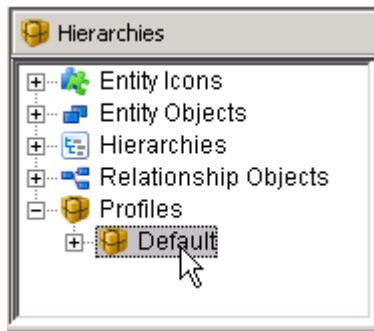
Optional.

5. Click **Save**.

Step 6. Add the Relationship Types to the Hierarchy Profile

When you add the relationship types to the hierarchy profile, the entity types associated with the relationship types are also added to the hierarchy profile. In this tutorial, we will add the relationship types to the default profile.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the default profile.



4. In the Relationship Types section of the General tab in the Profile pane, enable the relationship types **Product Group is parent of Product Group** and **Product Group is parent of Product** relationship types.
5. Click **Save**.

Step 7. Create the Packages

Create packages for the entity object and relationships. The packages define the data that a user can access in the Hierarchy Manager.

Use the Hierarchies tool to create the hierarchy packages and associated queries. When you create the package, you select which columns are available to configure access for. You cannot configure a column to be available in the Hierarchy Manager if the column is not part of the package.

After you create the hierarchy package, the package appears in the Package tool and the query appears in the Queries tool. You cannot use the Package tool or Queries tool to create hierarchy packages or queries.

For more information about queries and packages, see [Chapter 9, "Queries and Packages" on page 124](#).

Create the Product Entity Object Package

Create the PKG Product package so that you can then assign the package to the Product entity object.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. Select **Hierarchies > Create New Package**.
4. In step 1 of the Create New Package dialog box, select **Create Package for Entity Object**.

5. In the Query Group section, select or create a query group to organized the package in the Queries tool. Click **Next**.
6. In step 2 of the Create New Package dialog box, type `Product` in the **Query name** field. In the **Select primary table** field, select the Product entity base object. Click **Next**.
7. In step 3 of the Create New Package dialog box, type `PKG Product`. Enable the **Enable Put** check box so that you can change the values of entity record fields in the Hierarchy Manager tool. Click **Next**.
8. In step 4 of the Create New Package dialog box, select the following columns to use in the query:
 - Rowid Object
 - Product Name
 - Product Number
 - Product Description
 - Product Type Code
 - Product Type
 - Hub State Indicator

Note: You cannot disable Rowid Object, Hub State Indicator, or Product Type.
9. Click **Finish**.

Create the Product Relationship Package

Create the PKG Product Rel package so that you can then assign the package to the Product Rel relationship object.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. Select **Hierarchies > Create New Package**.
4. In step 1 of the Create New Package dialog box, select **Create Package for Relationship Object**.
5. In the Query Group section, select or create a query group to organized the package in the Queries tool, such as RL Queries. Click **Next**.
6. In step 2 of the Create New Package dialog box, type `RL Product` in the **Query name** field. In the **Select primary table** field, select the Product Rel entity base object. Click **Next**.
7. In step 3 of the Create New Package dialog box, type `PKG Product Rel`. Enable the **Enable Put** check box. Click **Next**.
8. In step 4 of the Create New Package dialog box, select the following columns to use in the query:
 - Rowid Object
 - Rowid Hierarchy
 - Rowid Rel Type
 - Product ID1
 - Product ID2
 - Hierarchy Level
 - Hub State Indicator

Note: You cannot disable Rowid Object, Rowid Hierarchy, Rowid Rel Type, Product ID1, Product ID2, or Hub State Indicator.
9. Click **Finish**.

Step 8. Assign the Packages

Assign the packages to the entity types and to the relationship types so that you can configure the data display for the entity types or relationship types.

You can assign the following types of packages:

Display package

A package for read data access. Required.

Put package

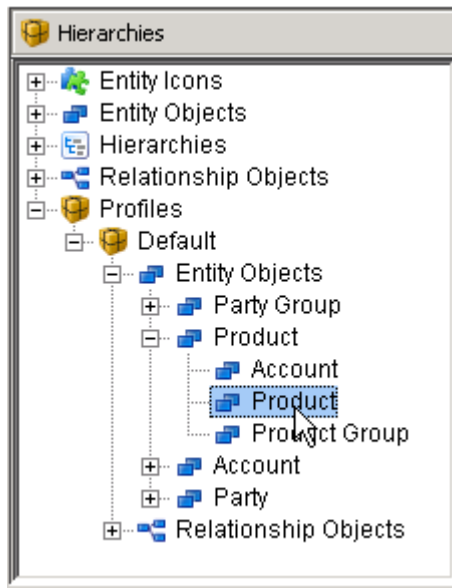
A package for write data access. Optional.

If you do not assign a put package to an entity or relationship type, you cannot create or edit that type. For example, if you do not assign a put package to the Product entity type, you cannot create a new product record or edit an existing product record in the Hierarchy Manager. You can only assign a put package that is put-enabled.

Assign the PKG Product Package to the Product Entity Type


Assign the PKG Product package to the Product entity type so that you can configure how Product entity data appears in the Hierarchy Manager.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.





4. Select the **PKG Product** package from the **MDM Display Package** list.
5. Select the **PKG Product** package from the **MDM Put Package** list.

You can now configure the display of data for the Product entity type in the **HM Packages** section.



HM Packages for an Entity Type

MDM Display Package	PKG Product	▼
MDM Put Package	PKG Product	▼
Label Pattern		
Tooltip Pattern		

NOTES:

1. Any change to the patterns will be reflected in all Profiles that have this Entity Type.
2. Any changes to the Common HM Package will be reflected in all Entity Types in this Profile.
3. Configure columns in Label and Tooltip HM Packages before configuring the Label and Tooltip Patterns.
4. Press the right mouse button in the table below to use auto-fill options.

HM Packages:

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	▼	▼	▼	▼	▼	▼	▼	▼
Product Name	▼	▼	▼	▼	▼	▼	▼	▼
Product Number	▼	▼	▼	▼	▼	▼	▼	▼
Product Desc	▼	▼	▼	▼	▼	▼	▼	▼
Product Type Cd	▼	▼	▼	▼	▼	▼	▼	▼
Product Type	▼	▼	▼	▼	▼	▼	▼	▼
Hub State Indicator	▼	▼	▼	▼	▼	▼	▼	▼

Assign the PKG Product Package to the Product Group Entity Type

Assign the PKG Product package to the Product Group entity type so that you can configure how Product Group entity data appears in the Hierarchy Manager.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group entity type node under the Default hierarchy profile.
4. Select the **PKG Product** package from the **MDM Display Package** list.
5. Select the **PKG Product** package from the **MDM Put Package** list.

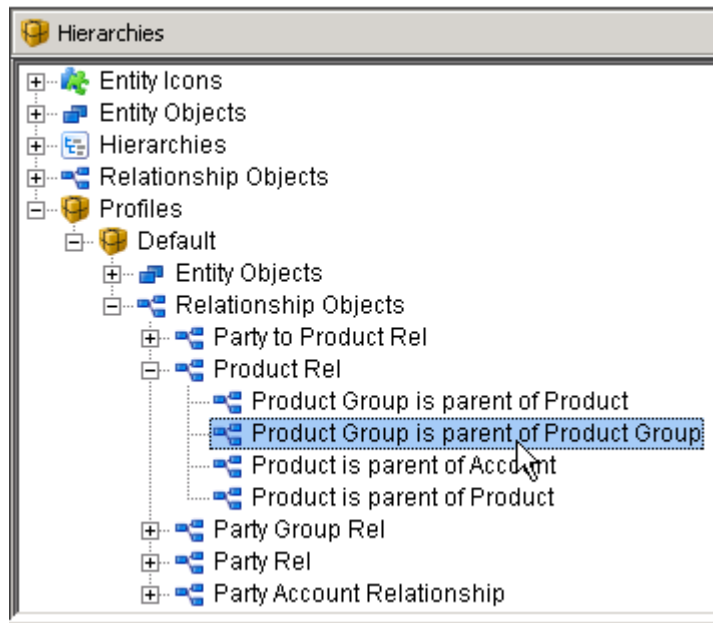
You can now configure the display of data for the Product Group entity type in the **HM Packages** section.

Assign the PKG Product Rel Package to the Product Group is Parent of Product Group Relationship Type

Assign the PKG Product Rel package to the Product Group is parent of Product Group relationship type so that you can configure how the relationship data appears in the Hierarchy Manager.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.

3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product Group relationship type node under the Default hierarchy profile.



4. Select the **PKG Product Rel** package from the **MDM Display Package** list.
5. Select the **PKG Product Rel** package from the **MDM Put Package** list.

You can now configure the display of data for the Product Group is parent of Product Group relationship type in the **HM Packages** section.

HM Packages for a Relationship Type

MDM Display Package	PKG Product Rel	▼
MDM Put Package	PKG Product Rel	▼
Tooltip Pattern		

NOTES:

1. Any change to the pattern will be reflected in all Profiles that have this Relationship Type.
2. Any change to the Common HM Package will be reflected in all Relationship Types in this Profile.
3. Configure columns in Tooltip HM Package before configuring the Tooltip Pattern.
4. Press the right mouse button in the table below to use auto-fill options.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	▼	▼	▼	▼	▼	▼	▼
Rowid Hierarchy	▼	▼	▼	▼	▼	▼	▼
Rowid Rel Type	▼	▼	▼	▼	▼	▼	▼
Product ID1	▼	▼	▼	▼	▼	▼	▼
Product ID2	▼	▼	▼	▼	▼	▼	▼
Hub State Indicator	▼	▼	▼	▼	▼	▼	▼
Hierarchy Level	▼	▼	▼	▼	▼	▼	▼

Assign the PKG Product Rel Package to the Product Group is Parent of Product Relationship

Assign the PKG Product Rel package to the Product Group is parent of Product relationship type so that you can configure how the relationship data appears in the Hierarchy Manager.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product relationship type node under the Default hierarchy profile.
4. Select the **PKG Product Rel** package from the **MDM Display Package** list.
5. Select the **PKG Product Rel** package from the **MDM Put Package** list.

You can now configure the display of data for the Product Group is parent of Product relationship type in the **HM Packages** section.

Step 9. Configure the Display of Data in Hierarchy Manager

You can configure which entity fields and relationship fields appear in the Hierarchy Manager when users search, view, edit, and add records. You can also configure which fields appear in the Hierarchy Manager labels and tooltips.

To configure the display of data in the Hierarchy Manager, configure the Hierarchy Manager packages for each relationship type and entity type. If you have multiple hierarchy profiles, you must configure the display of data for each profile.

You can configure which fields appear and the order in which they appear for the following Hierarchy Manager user interface elements:

Label

The label text for the entity in the Hierarchy Manager tool.

Tooltip

The text that appears when you pause on an entity or relationship in the Hierarchy Manager tool.

Common

The fields to appear when entities and relationships of different types appear in the same list. The fields you select must be in all hierarchy manager packages.

Search

The fields that you can use to search for an entity record or relationship record.

List

The fields that appear in the list of search results for an entity record or relationship record.

Detail

The fields that appear when you choose to view the details of a selected entity or relationship.

Put

The fields for whose value you can change when you edit a selected entity or relationship.

Add

The fields that appear when you add an entity or relationship in the Hierarchy Manager.

Configure the Product Entity Type Label

In this tutorial, we will configure the product name to appear as the label text for Product entities.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
4. In the **Label** column in the **HM packages** section, ensure a number is assigned to the Product Name row.

HM Packages:

Column Name	Label
Rowid Object	3
Product Name	1
Product Number	4
Product Desc	0
Product Type Cd	5
Product Type	
Hub State Indicator	2

5. In the **HM Packages for an Entity Type** section, click the edit button for the **Label Pattern** field.
6. In the **Edit Pattern** dialog box, double-click **{#} ==> {Product Name}**.

HM Edit Pattern

Raw Pattern : {1}

Pattern Preview : {Product Name}
SampleString

Double-click a column name in the list below to use it in the pattern:

- {0} ==> {Product Desc}
- {1} ==> {Product Name}**
- {2} ==> {Hub State Indicator}
- {3} ==> {Rowid Object}
- {4} ==> {Product Number}
- {5} ==> {Product Type Cd}

OK Cancel

In the Raw Pattern field, you can concatenate columns or add custom text. We will use the Product Name column without custom text.

7. Click **Save**.

The following image shows a Product entity with the product name of Media Desktop Laser as it appears in Hierarchy Manager:



Configure the Label for the Product Group Entity Type

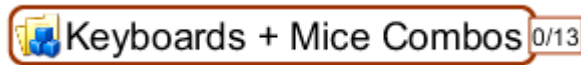
In this tutorial, we will configure the product name to appear as the label text for Product Group entities.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group entity type node under the Default hierarchy profile.
4. In the label column in the **HM packages** section, ensure a number is assigned to the Product Name row.
5. In the **HM Packages for an Entity Type** section, click the edit button for the **Label Pattern** field.
6. In the **Edit Pattern** dialog box, double-click **{#} ==> {Product Name}**.

In the Raw Pattern field, you can concatenate columns or add custom text. We will use the Product Name column without custom text.

7. Click **Save**.

The following image shows a Product Group entity with the product name of Keyboards + Mice Combos as it appears in Hierarchy Manager:



Configure the Product Entity Type Tooltip Text

In this tutorial, we will configure the product description to appear as the tooltip text for the Product entities. When a user pauses over a product entity in the Hierarchy Manager, the product description appears as the tooltip.

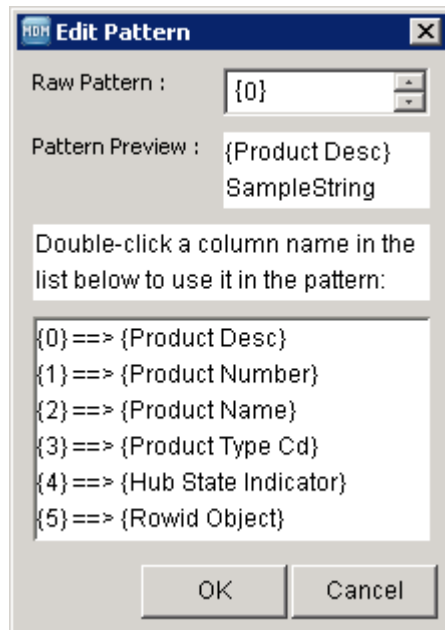
1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
4. In the **Tooltip** column in the **HM packages** section, ensure a number is assigned to the Product Desc row.

HM Packages:

Column Name	Label	Tooltip
Rowid Object	3	5
Product Name	1	2
Product Number	4	1
Product Desc	0	0
Product Type Cd	5	3
Product Type		
Hub State Indicator	2	4

5. In the **HM Packages for an Entity Type** section, click the edit button for the **Tooltip Pattern** field.

6. In the **Edit Pattern** dialog box, double-click **{#}==>{Product Desc}**.



In the **Raw Pattern** field, you can concatenate columns or add custom text. We will use the **Product Desc** column without custom text.

7. Click **Save**.

The following image shows the LS1 Laser Mouse product entity with the product description displayed as the tooltip:



Configure the Product Group Entity Type Tooltip Text

In this tutorial, we will configure the product description to appear as the tooltip text for the Product Group entity type. When a user pauses over a product group entity in the Hierarchy Manager, the product description appears as the tooltip.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group entity type node under the Default hierarchy profile.
4. In the **Tooltip** column in the **HM packages** section, ensure a number is assigned to the Product Desc row.
5. In the **HM Packages for an Entity Type** section, click the edit button for the **Tooltip Pattern** field.
6. In the **Edit Pattern** dialog box, double-click **{#}==>{Product Desc}**.

In the **Raw Pattern** field, you can concatenate columns or add custom text. We will use the **Product Desc** column without custom text.

7. Click **Save**.

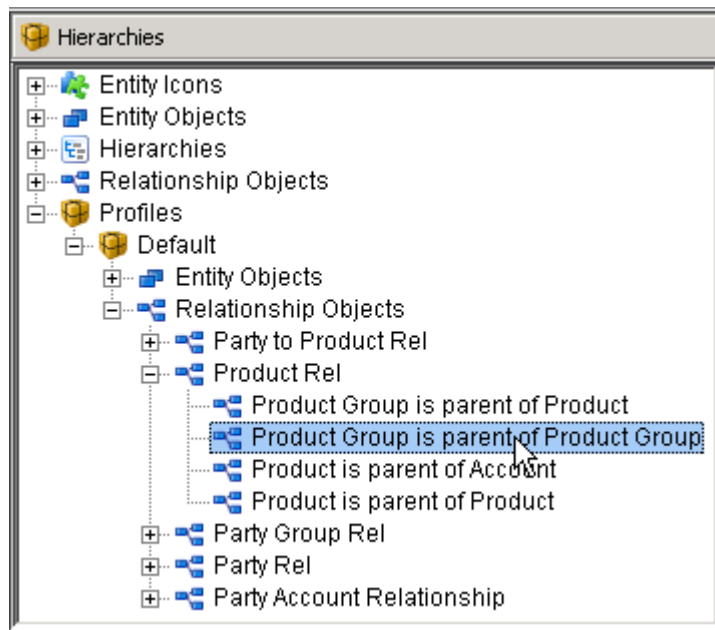
The following image shows the Webcams product group entity with the product description displayed as the tooltip:



Configure the Product Group is Parent of Product Group Relationship Type Tooltip Text

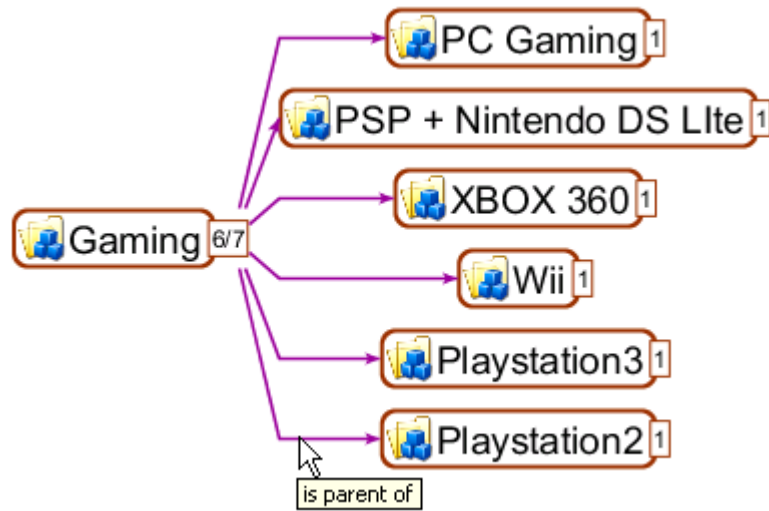
In this tutorial, we will configure the text **is parent of** to appear as the tooltip text for the Product Group is Parent of Product Group relationships. When a user pauses over the arrow that represents the relationship in the Hierarchy Manager, the tooltip text appears.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group is Parent of Product Group relationship type node under the Default hierarchy profile.



4. In the **HM Packages for an Entity Type** section, click the edit button for the **Tooltip Pattern** field.
5. In the Raw Pattern field of the **Edit Pattern** dialog box, type `is parent of`.
6. Click **Save**.

The following image shows the tooltip for the relationship between the Gaming product group and the Playstation 2 product group:

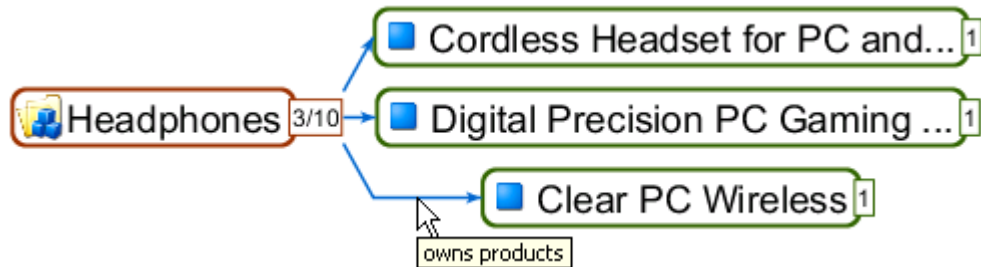


Configure the Product Group is Parent of Product Relationship Type Tooltip Text

In this tutorial, we will configure the text **owns products** to appear as the tooltip text for the Product Group is Parent of Product relationships. When a user pauses over the arrow that represents the relationship in the Hierarchy Manager, the tooltip text appears.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group is Parent of Product relationship type node under the Default hierarchy profile.
4. In the **HM Packages for an Entity Type** section, click the edit button for the **Tooltip Pattern** field.
5. In the Raw Pattern field of the **Edit Pattern** dialog box, type `owns products`.
6. Click **Save**.

The following image shows the tooltip for the relationship between the Headphones product group and the Clear PC Wireless product:



Configure the List of Entities

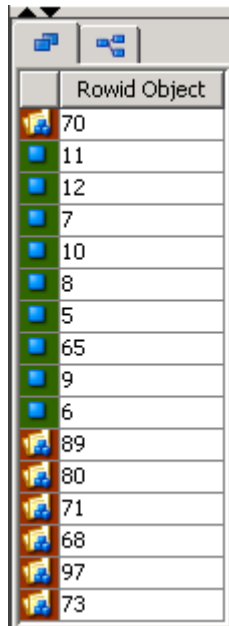
In the tutorial, we will configure the Row ID Object field to appear in the list of entities that appear in the Hierarchy Manager. The list of entities can contain entities of different entity types. You can only select fields that you include in every query for every entity type package.

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
4. In the **Common** column in the **HM packages** section, assign a 0 to the Rowid Object row.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	4		
Product Name	1	1		0	1	0	0	0
Product Number	2	2		1	4	2	1	1
Product Desc	3	3		2	3	1		2
Product Type Cd	4	4		3	2	3		3
Product Type					5	6		
Hub State Ind	5	5				5		

5. Click **Save**.
6. Repeat the steps for the Product Group entity type.

The following image shows a list that contains Product and Product Group entities as it appears in Hierarchy Manager:



Configure the List of Relationships

In the tutorial, we will configure the Row ID Object field to appear in the list of relationships that appear in the Hierarchy Manager. The list of relationships can contain relationships of different entity types. You can only select fields that you include in every query for every relationship type package.

1. Acquire a write lock.

2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product Group relationship type node under the Default hierarchy profile.
4. In the **Common** column in the **HM packages** section, assign a 0 to the Rowid Object row.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	U	U	U	U	U	U	U
Rowid Hierarchy			1	1	1	1	1
Rowid Rel Type							
Product ID1			2	2	2	2	2
Product ID2			3	3	3	3	3
Hub State Indicator			4	4	4	4	4
Hierarchy Level							

5. Click **Save**.
6. Repeat the steps for the Product Group is parent of Product relationship type.

The following image shows a list that contains the relationships as it appears in Hierarchy Manager:

Rowid Object
24
25
30
32
29
28
105
26
31
27
103
106
102
104
112

Configure the Entity Search Fields

In this tutorial, we will configure the following fields to appear in descending order in the **Search** tab of the **Search for Data Objects** dialog box when a user searches for Product entities:

- Product Name
- Product Number
- Product Description
- Product Type Code
- Row ID Object

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.

3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
4. In the **Search** column in the **HM packages** section, assign numbers to the columns that you want to appear as search fields. The column that you assign the lowest value appears at the top of the list of fields.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	0	0	0
Product Name	1	1		0	1	1	1	1
Product Number	2	2		1	4	2	2	2
Product Desc	3	3		2	3	3		3
Product Type Cd	4	4		3	2	4		4
Product Type					5			5
Hub State Ind	5	5				5		6

5. Click **Save**.
6. Repeat the steps for the Product Group entity type.

In this example, the Product Type field and Hub State Ind field are not assigned a number, so those fields do not appear in the **Search for Data Objects** dialog box. The Product Name field appears at the top of the list of fields because we assigned the lowest value to Product Name. The Rowid Object field appears at the bottom of the list because we assigned the highest value to Rowid Object.

The following image shows the **Search** tab as it appears in the Hierarchy Manager:

Search for Data Objects

Search Results Recent Bookmarks

Type: Product

Criteria

Product Name is exactly

Product Number is exactly

Product Desc is exactly

Product Type Cd is exactly

Rowid Object is exactly

Basic Search Select Cancel

Configure the Relationship Search Fields

In this tutorial, we will configure the following fields to appear in descending order in the **Search** tab of the **Search for Data Objects** dialog box when a user searches for relationships:

- Rowid Object

- Rowid Hierarchy
- Product ID1
- Product ID2
- Hub State Indicator

1. Acquire a write lock.
2. In the Model workbench, click **Hierarchies**.
3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product relationship type node under the Default hierarchy profile.
4. In the **Search** column in the **HM packages** section, assign numbers to the columns that you want to appear as search fields. The column that you assign the lowest value appears at the top of the list of fields.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	U	U	U	U	U	U	U
Rowid Hierarchy			1	1	1	1	1
Rowid Rel Type							
Product ID1			2	2	2	2	2
Product ID2			3	3	3	3	3
Hub State Indicator			4	4	4	4	4
Hierarchy Level							

5. Click **Save**.
6. Repeat the steps for the Product Group is parent of Product Group relationship type.

The following image shows the **Search** tab as it appears in the Hierarchy Manager:

Search for Relationships

Search Results Recent Bookmarks

Type: Product Group is parent of Product

Criteria

Rowid Object	is exactly	
Rowid Hierarchy	is exactly	
Product ID1	is exactly	
Product ID2	is exactly	
Hub State Indicator	is exactly	

Basic Search Select Cancel

Configure the Entity Search Results

In this tutorial, we will configure the following Product fields to appear in order from left to right in the **Results** tab of the **Search for Data Objects** dialog box when a user searches for entities:

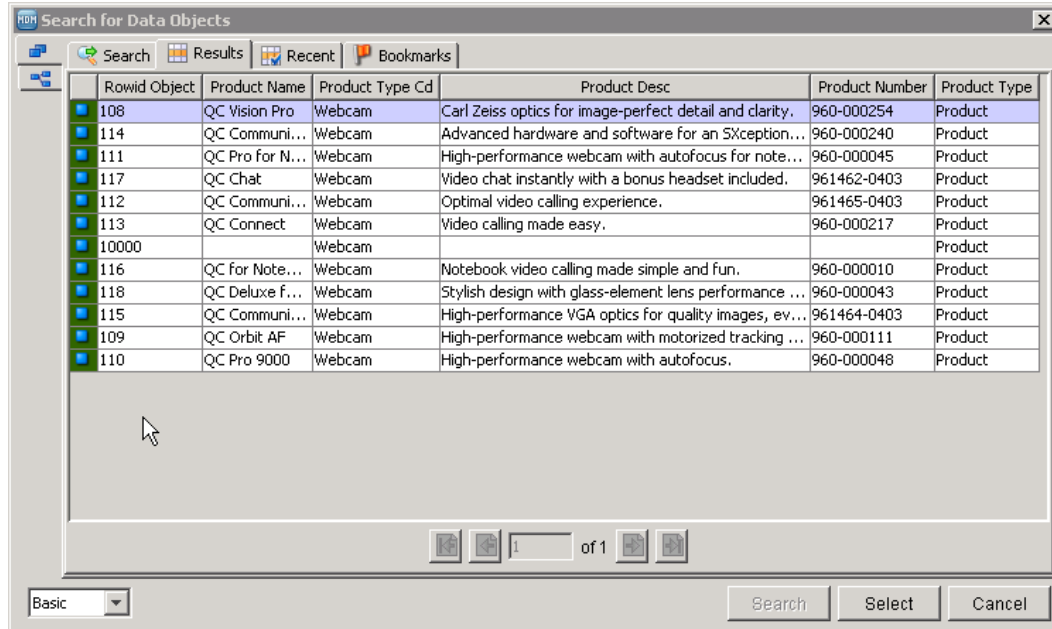
- Row ID Object
 - Product Name
 - Product Type Code
 - Product Description
 - Product Number
 - Product Type
1. Acquire a write lock.
 2. In the Model workbench, click **Hierarchies**.
 3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
 4. In the **List** column in the **HM packages** section, assign numbers to the columns that you want to appear in the search results. The column that you assign the lowest value appears at the top of the list of fields.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	0	0	0
Product Name	1	1		0	1	1	1	1
Product Number	2	2		1	4	2	2	2
Product Desc	3	3		2	3	3		3
Product Type Cd	4	4		3	2	4		4
Product Type					5			5
Hub State Ind	5	5				5		6

5. Click **Save**.
6. Repeat the steps for the Product Group entity type.

In this example, the Hub State Ind field is not assigned a number, so the field does not appear in the **Results** tab. The Rowid Object field appears on the right because we assigned the lowest value to Rowid Object. The Product Type field appears on the left because we assigned the highest value to Product Type.

The following image shows the **Results** tab as it appears when you use the configuration settings in this example:



Rowid Object	Product Name	Product Type Cd	Product Desc	Product Number	Product Type
108	QC Vision Pro	Webcam	Carl Zeiss optics for image-perfect detail and clarity.	960-000254	Product
114	QC Communi...	Webcam	Advanced hardware and software for an SXception...	960-000240	Product
111	QC Pro for N...	Webcam	High-performance webcam with autofocus for note...	960-000045	Product
117	QC Chat	Webcam	Video chat instantly with a bonus headset included.	961462-0403	Product
112	QC Communi...	Webcam	Optimal video calling experience.	961465-0403	Product
113	QC Connect	Webcam	Video calling made easy.	960-000217	Product
10000		Webcam			Product
116	QC for Note...	Webcam	Notebook video calling made simple and fun.	960-000010	Product
118	QC Deluxe f...	Webcam	Stylish design with glass-element lens performance ...	960-000043	Product
115	QC Communi...	Webcam	High-performance VGA optics for quality images, ev...	961464-0403	Product
109	QC Orbit AF	Webcam	High-performance webcam with motorized tracking ...	960-000111	Product
110	QC Pro 9000	Webcam	High-performance webcam with autofocus.	960-000048	Product

Configure the Relationship Search Results

In this tutorial, we will configure the following fields for the Product Group is parent of Product relationship type to appear in order from left to right in the **Results** tab of the **Search for Data Objects** dialog box when a user searches for entities:

- Rowid Object
 - Rowid Hierarchy
 - Product ID1
 - Product ID2
 - Hub State Indicator
1. Acquire a write lock.
 2. In the Model workbench, click **Hierarchies**.
 3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product relationship type node under the Default hierarchy profile.

4. In the **List** column in the **HM packages** section, assign numbers to the columns that you want to appear in the search results. The column that you assign the lowest value appears on the left in the search results.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	U	U	U	U	U	U	U
Rowid Hierarchy			1	1	1	1	1
Rowid Rel Type							
Product ID1			2	2	2	2	2
Product ID2			3	3	3	3	3
Hub State Indicator			4	4	4	4	4
Hierarchy Level							

5. Click **Save**.
6. Repeat the steps for the Product Group is parent of Product Group relationship type.

The following image shows the **Results** tab as it appears when you use the configuration settings in this example:

Rowid Object	Rowid Hierarchy	Product ID1	Product ID2	Hub State Indicator
42	3	74	29	1
43	3	74	39	1
44	3	74	38	1
45	3	76	22	1
46	3	76	24	1
47	3	76	26	1
48	3	76	23	1
49	3	76	19	1
51	3	76	14	1
52	3	76	21	1
63	3	77	51	1
64	3	77	53	1
65	3	77	64	1

Configure the Entity Details in Hub Console Hierarchy Manager

In this tutorial, we will configure the following fields to appear in descending order in the **Details** dialog box when a user views the entity details in the Hub Console Hierarchy Manager:

- Product Name
- Product Description
- Product Number
- Product Type Code

- Row ID Object
 - Hub State Indicator
1. Acquire a write lock.
 2. In the Model workbench, click **Hierarchies**.
 3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
 4. In the **Detail** column in the **HM packages** section, assign numbers to the columns that you want to appear in the Details dialog box. The column that you assign the lowest value appears at the top of the list of fields.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	4	0	0
Product Name	1	1		0	1	0	1	1
Product Number	2	2		1	4	2	2	2
Product Desc	3	3		2	3	1		3
Product Type Cd	4	4		3	2	3		4
Product Type					5			5
Hub State Ind	5	5				5		6

5. Click **Save**.
6. Repeat the steps for the Product Group entity type.

The following image shows the **Details** dialog box as it appears in the Hierarchy Manager:

Name	Value
Product Name	QC for Notebooks
Product Desc	Notebook video calling made simple and fun.
Product Number	960-000010
Product Type Cd	Webcam
Rowid Object	116
Hub State Ind	1

Configure the Entity Details for IDD Hierarchy Manager

In this tutorial, we will configure the customer name to appear when a user views the details of a Person object in the IDD Hierarchy Manager.

1. Log in to the Informatica Data Director Configuration Manager.
`http://<host>:<port>/bdd/config`
2. Select the IDD application, and then click **Edit**.
3. In the Subject Areas tab of the Edit Application screen, select **Subject Area Groups > Customer > Person > Names**.
4. Click **Edit Subject Area Child**.
5. In the Layout tab of the Subject Area Child dialog box, select the Name column name from the table, and then click **Edit Layout**.

6. Enable **Show in HM**. Click **OK**.
7. Click **OK**, and then click **Save**.

Configure the Relationship Details

In this tutorial, we will configure the following fields to appear in descending order in the **Details** dialog box when a user views the relationship details:

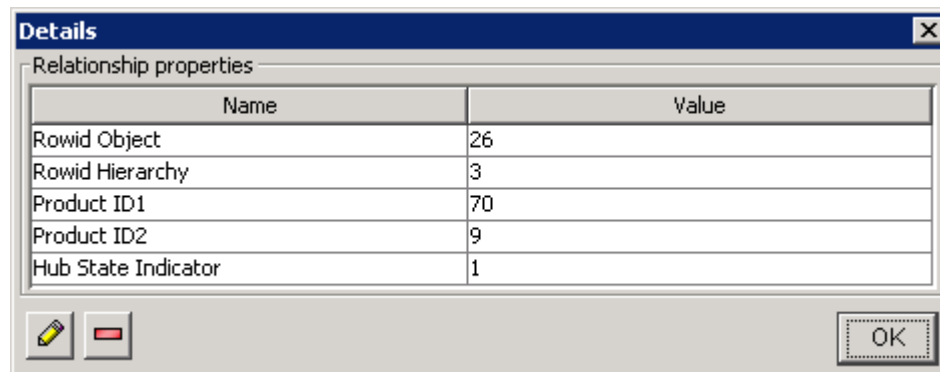
- Rowid Object
 - Rowid Hierarchy
 - Product ID1
 - Product ID2
 - Hub State Indicator
1. Acquire a write lock.
 2. In the Model workbench, click **Hierarchies**.
 3. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product relationship type node under the Default hierarchy profile.
 4. In the **Detail** column in the **HM packages** section, assign numbers to the columns that you want to appear in the Details dialog box. The column that you assign the lowest value appears at the top of the list of fields.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	U	U	U	U	U	U	U
Rowid Hierarchy			1	1	1	1	1
Rowid Rel Type							
Product ID1			2	2	2	2	2
Product ID2			3	3	3	3	3
Hub State Indicator			4	4	4	4	4
Hierarchy Level							

5. Click **Save**.
6. Repeat the steps for the Product Group is parent of Product Group relationship type.

The following image shows the **Details** dialog box as it appears in the Hierarchy Manager:



Configure the Editable Entity Fields

To configure the entity fields that users can edit, you must have assigned a put package to the entity type. In this tutorial, we will configure the following fields to appear in descending order in the **Entity Record Editor** dialog box when a user edits an entity:

- Product Name
 - Product Number
1. Acquire a write lock.
 2. In the Model workbench, click **Hierarchies**.
 3. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
 4. In the **Put** column in the **HM packages** section, assign numbers to the columns that you want to appear as editable fields. The column that you assign the lowest value appears at the top of the list of fields.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	4		
Product Name	1	1		0	1	0	0	0
Product Number	2	2		1	4	2	1	1
Product Desc	3	3		2	3	1		2
Product Type Cd	4	4		3	2	3		3
Product Type					5			4
Hub State Ind	5	5				5		

5. Click **Save**.
6. Repeat the steps for the Product Group entity type.

The following image shows the **Entity Record Editor** dialog box as it appears in the Hierarchy Manager:

The screenshot shows the 'Entity Record Editor' dialog box. The 'Type' dropdown menu is set to 'Product'. Below it, the 'Product Name' text box contains 'QC Pro for Notebooks' and the 'Product Number' text box contains '960-000045'. At the bottom left, there is a checkbox labeled 'Show only editable cells' which is currently unchecked. At the bottom right, there are three buttons: 'OK', 'Cancel', and 'Set Default Tru...'.

Relationship Field Editability

The relationship fields in the PKG Product Rel relationship package are not editable in the Hierarchy Manager. We do not need to configure the Put column for the relationship package. Users cannot edit the fields in the relationship package, even if you assign numbers to the columns that you want to edit.

For more information about the relationship record editor in the Hierarchy Manager, see the *Multidomain MDM Data Steward Guide*.

Configure the Entity Creation Fields

In this tutorial, we will configure the following fields to appear in descending order in the **Entity Record Editor** dialog box when a user creates an entity in the Hierarchy Manager:

- Product Name
- Product Number
- Product Description
- Product Type Code

1. Acquire a write lock.
2. In the navigation pane of the Hierarchies tool, select the Product entity type node under the Default hierarchy profile.
3. In the **Add** column in the **HM packages** section, assign numbers to the columns that you want to appear as fields in the **Entity Record Editor** dialog box. The column that you assign the lowest value appears at the top of the list of fields.

Column Name	Label	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	0	0	0	4	0	4		
Product Name	1	1		0	1	0	0	0
Product Number	2	2		1	4	2	1	1
Product Desc	3	3		2	3	1		2
Product Type Cd	4	4		3	2	3		3
Product Type					5			
Hub State Ind	5	5				5		

4. Click **Save**.
5. Repeat the steps for the Product Group entity type.

The following image shows the **Entity Record Editor** dialog box when a user creates a Product entity in the Hierarchy Manager:

The screenshot shows the 'Entity Record Editor' dialog box. The 'Type' dropdown menu is set to 'Product'. Below this, there are four input fields: 'Product Name', 'Product Number', 'Product Desc', and 'Product Type Cd'. At the bottom of the dialog, there is a checkbox labeled 'Show only editable cells' which is currently unchecked. Below the checkbox are three buttons: 'OK', 'Cancel', and 'Set Default Trust ...'.

Configure the Relationship Creation Fields

In this tutorial, we will configure the following fields to appear in descending order in the **Relationship Record Editor** dialog box when a user creates a relationship between two entities in the Hierarchy Manager:

- Hub State Indicator
 - Hierarchy Level
1. Acquire a write lock.
 2. In the navigation pane of the Hierarchies tool, select the Product Group is parent of Product relationship type node under the Default hierarchy profile.
 3. In the **Add** column in the **HM packages** section, assign numbers to the columns that you want to appear as fields in the **Relationship Record Editor** dialog box. The column that you assign the lowest value appears at the top of the list of fields.

HM Packages:

Column Name	Tooltip	Common	Search	List	Detail	Put	Add
Rowid Object	U	U	U	U	U	U	
Rowid Hierarchy			1	1	1	1	
Rowid Rel Type						5	
Product ID1			2	2	2	2	
Product ID2			3	3	3	3	
Hub State Indicator			4	4	4	4	U
Hierarchy Level						6	1

4. Click **Save**.
5. Repeat the steps for the Product Group is parent of Product Group relationship type.

The following image shows the **Relationship Record Editor** dialog box when a user creates a relationship between two entities in the Hierarchy Manager:

77 Mice

61 S150 Laser Mouse for Notebooks

Hierarchy: Product

Relationship Type: Product Group is parent of Pr...

Start Date: [Red X] [Date 31]

End Date: [Red X] [Date 31]

Hub State Indicator: [Dropdown]

Hierarchy Level: [Text Box]

☐ Show only editable cells

OK Cancel Set Default ...

Hierarchy Management

After you configure the hierarchy, you can add records and relationships between records to the hierarchy in the Hierarchy Manager tool in the MDM Hub Console or the Hierarchy Manager tool in Informatica Data Director. This tutorial does not describe how to use the Hierarchy Manager tool to populate the configured hierarchy with data.

For more information about the Hierarchy Manager tool in the MDM Hub Console, see the *Multidomain MDM Data Steward Guide*.

For more information about the Hierarchy Manager tool in Informatica Data Director, see the *Multidomain MDM Data Director User Guide*.

Part IV: Configuring the Data Flow

This part contains the following chapters:

- [MDM Hub Processes, 259](#)
- [Configuring the Land Process, 287](#)
- [MDM Hub Staging, 294](#)
- [Hard Delete Detection, 316](#)
- [Data Cleansing Configuration, 333](#)
- [Configuring the Load Process , 354](#)
- [Configuring the Match Process, 375](#)
- [Match Rule Configuration Example, 435](#)
- [Search with Elasticsearch, 459](#)
- [Configuring the Consolidate Process, 483](#)
- [Pending Control Table, 489](#)
- [Configuring the Publish Process, 490](#)

CHAPTER 15

MDM Hub Processes

This chapter includes the following topics:

- [MDM Hub Processes Overview, 259](#)
- [About Informatica MDM Hub Processes, 259](#)
- [Land Process, 262](#)
- [Stage Process, 264](#)
- [Load Process, 265](#)
- [Tokenize Process, 271](#)
- [Match Process, 277](#)
- [Consolidate Process, 282](#)
- [Publish Process, 284](#)

MDM Hub Processes Overview

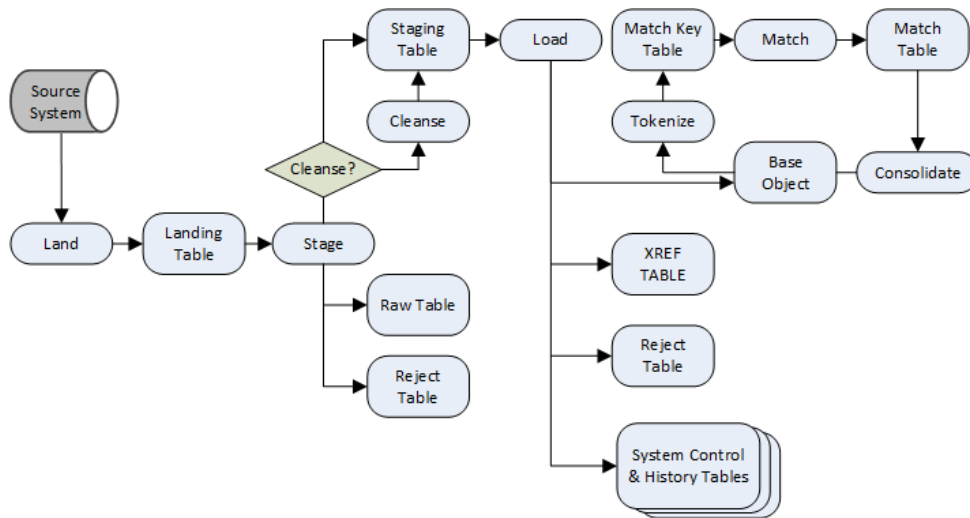
You can load data into the MDM Hub and process them through multiple batch processes. You can move data into the MDM Hub through the land process. You can then use the stage process to cleanse and prepare data for the load process that loads data into base objects. After the load process, you can run the tokenize and the match process to identify duplicate records and consolidate them through the consolidate process.

About Informatica MDM Hub Processes

With batch processing in Informatica MDM Hub, data flows through Informatica MDM Hub in a sequence of individual processes.

Overall Data Flow for Batch Processes

The following figure provides a detailed perspective on the overall flow of data through the Informatica MDM Hub using batch processes, including individual processes, source systems, base objects, and support tables.



Note: The publish process is not shown in this figure because it is not a batch process.

Consolidation Status for Base Object Records

This section describes the consolidation status of records in a base object.

Consolidation Indicator

All base objects have a system column named CONSOLIDATION_IND.

The *consolidation indicator* represents the consolidation status of individual records as they progress through multiple processes in MDM Hub.

The following table describes the consolidation indicator values:

Value	State Name	Description
1	CONSOLIDATED	Indicates that the record is consolidated, determined to be unique, and represents the best version of the truth.
2	QUEUED_FOR_MERGE	Indicates that the record might have gone through the match process and is ready for a merge. Also, the record might have gone through the merge process and is ready for another merge.
3	Not MERGED or MATCHED	This record has gone through the match process and is ready to be consolidated. Also, if a record has gone through the manual merge process, where the matching is done by a data steward, the consolidation indicator is 2.
4	NEWLY_LOADED	Indicates that the record is a new insert or that the record is flagged to undergo the match process.
9	ON_HOLD	Indicates that the data steward has put the record on hold until further notice. You can put any record on hold, regardless of its consolidation indicator value. The match and consolidate processes ignore records that are on hold.

How the Consolidation Indicator Changes

Informatica MDM Hub updates the consolidation indicator for base object records in the following sequence.

1. During the load process, when a new record is loaded into a base object, Informatica MDM Hub assigns the record a consolidation indicator of 4, indicating that the record needs to be matched.
2. Near the start of the match process, when a record is selected as a match candidate, the match process changes its consolidation indicator to 3.

Note: Any change to the match or merge configuration settings will trigger a reset match dialog, asking whether you want to reset the records in the base object (change the consolidation indicator to 4, ready for match).

3. Before completing, the match process changes the consolidation indicator of match candidate records to 2 (ready for consolidation). If you perform a manual merge, the data steward performs a manual match, and the consolidation indicator of the merged record is 2.

Note: The match process may or may not have found matches for the record.

A record with a consolidation indicator of 2 is visible in Merge Manager.

4. If Accept All Unmatched Rows as Unique is enabled, and a record has undergone the match process but no matches were found, then Informatica MDM Hub automatically changes its consolidation indicator to 1 (unique).
5. If Accept All Unmatched Rows as Unique is enabled, after the record has undergone the consolidate process, and once a record has no more duplicates to merge with, Informatica MDM Hub changes its consolidation indicator to 1, meaning that this record is unique in the base object, and that it represents the master record (best version of the truth) for that entity in the base object.

Note: Once a record has its consolidation indicator set to 1, Informatica MDM Hub will never directly match it against any other record. New or updated records (with a consolidation indicator of 4) can be matched against consolidated records.

Cell Data Survivorship and Order of Precedence

During the evaluation of cells to merge from two records, the MDM Hub determines which cell data survives and which data to discard. The MDM Hub considers the surviving cell data, or winning cell, to represent the better version of the truth between the two cells. Ultimately, a single, consolidated record contains the best surviving cell data and represents the best version of the truth.

Survivorship applies to both trust-enabled columns and columns that are not trust enabled. When the MDM Hub compares cells from two different records, it determines survivorship based on the following factors, in order of precedence:

1. By trust score. The trust score applies only if a column is trust-enabled. The data with the highest ROWID_OBJECT wins. If the trust scores are equal, or if trust is not enabled for a column, the MDM Hub proceeds to the next comparison.
2. By SRC_LUD in the cross-reference record. The data with the more recent cross-reference SRC_LUD value wins. If the SRC_LUD values are equal, the MDM Hub proceeds to the next comparison.
3. By ROWID_OBJECT in the base object. ROWID_OBJECT values are evaluated in numeric descending order. If the ROWID_OBJECT values are equal, the MDM Hub proceeds to the next comparison.
4. By ROWID_XREF in the cross-reference. ROWID_XREF values are evaluated in numeric descending order. The data with the highest ROWID_XREF wins.

ROWID_OBJECT Survivorship

When records are merged, the MDM Hub determines which ROWID_OBJECT survives to become the ROWID_OBJECT of the merged record. Survivorship of the ROWID_OBJECT depends on how and where the records are merged. For example, how records survive after a batch merge job is different from how records survive in Informatica Data Director (IDD).

The MDM Hub handles survivorship of the ROWID_OBJECT differently in each of the following scenarios:

Batch Merge Job

During a batch merge job, the MDM Hub considers the consolidation indicator for all records. When a new record (CONSOLIDATION_IND = 4) is matched and merged with a consolidated record (CONSOLIDATION_IND = 1), the consolidated ROWID_OBJECT wins. When two new records with (CONSOLIDATION_IND = 4) are matched, the record with the lowest ROWID_OBJECT wins.

Merge SIF API

When you use the Merge SIF API to merge records, the target ROWID_OBJECT wins. The target is the first record listed in the SIF request.

Informatica Data Director

When you merge records in real time in IDD, the target ROWID_OBJECT wins. The target is the record that displays in the Merge Preview column in the Match Merge Comparison view in IDD.

When you queue records for a merge, the ROWID_OBJECT of the current record that displays in IDD wins. However, survivorship in IDD also depends on the consolidation indicator for base object records. For example, if a record (CONSOLIDATION_IND = 4) is matched and merged with a consolidated record (CONSOLIDATION_IND = 1), the consolidated ROWID_OBJECT wins.

Note: When records are queued for a merge, IDD uses entries from the underlying _MTCH table. So, to know exactly which ROWID_OBJECT survives, view the _MTCH table. The ROWID_OBJECT_MATCHED column lists the surviving ROWID_OBJECT.

Land Process

This section describes concepts and tasks associated with the land process in Informatica MDM Hub.

About the Land Process

Landing data is the initial step for loading data into Informatica MDM Hub.

Source Systems and Landing Tables

Landing data involves the transfer of data from one or more source systems to Informatica MDM Hub landing tables.

- A source system is an external system that provides data to Informatica MDM Hub. Source systems can be applications, data stores, and other systems that are internal to your organization, or obtained or purchased from external sources.
- A landing table is a table in the Hub Store that contains the data that is initially loaded from a source system.

Land Process is External to Informatica MDM Hub

The land process is external to Informatica MDM Hub and is executed using an external batch process or an external application that directly populates landing tables in the Hub Store. Subsequent processes for managing data are internal to Informatica MDM Hub.

Ways to Populate Landing Tables

Landing tables can be populated in the following ways:

Load Method	Description
external batch process	An ETL (Extract-Transform-Load) tool or other external process copies data from a source system to Informatica MDM Hub. Batch loads are external to Informatica MDM Hub. Only the results of the batch load are visible to Informatica MDM Hub in the form of populated landing tables. Note: This process is handled by a separate ETL tool of your choice. This ETL tool is not part of the Informatica MDM Hub suite of products.
real-time processing	External applications can populate landing tables in on-line, real-time mode. Such applications are not part of the Informatica MDM Hub suite of products.

For any given source system, the approach used depends on whether it is the most efficient—or perhaps the only—way to data from a particular source system. In addition, batch processing is often used for the initial data load (the first time that business data is loaded into the Hub Store), as it can be the most efficient way to populate the landing table with a large number of records.

Note: Data in the landing tables cannot be deleted until after the load process for the base object has been executed and completed successfully.

Managing the Land Process

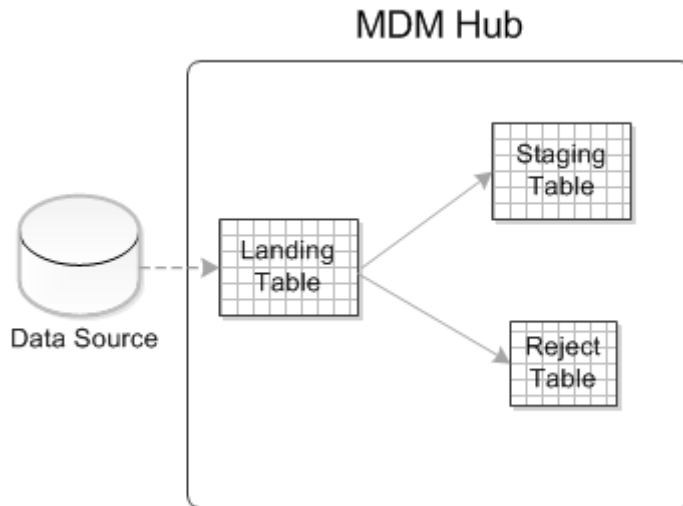
To manage the land process, see the following topics in this documentation:

Task	Topics
Configuration	Chapter 16, "Configuring the Land Process" on page 287 : <ul style="list-style-type: none">- "Configuring Source Systems" on page 287- "Configuring Landing Tables" on page 290
Execution	Execution of the land process is external to Informatica MDM Hub and depends on the approach you are using to populate landing tables, as described in "Ways to Populate Landing Tables" on page 263 .
Application Development	If you are using external application(s) to populate landing tables, see the developer documentation for the API used by your applications.

Stage Process

The MDM Hub stage process transfers source data from a landing table to the staging table associated with a particular base object. The complete stage process occurs in the MDM Hub.

The following image shows the MDM Hub stage process where source data is transferred from a landing table to a staging table and reject table:



Before you perform the MDM Hub stage process, load data from the external data source to the landing tables. You define mappings between the landing and staging tables. Mappings link a source column in the landing table with a target column in the staging table. If you need to cleanse data before the MDM Hub moves the data to the staging table, configure data cleansing in the mapping. When you run the stage job, the MDM Hub transfers data based on the mappings from the columns in a landing table to the columns in a staging table.

During the stage process, the MDM Hub processes one block of 250 records at a time. If there is a problem with a record in the block, the MDM Hub moves the record to the reject table. A record could be rejected because the value of a cell is too long, or because the record's update date is later than the current date. After the MDM Hub moves the rejected record, the MDM Hub stops processing the remaining records in the block, and moves on to other blocks. When the stage process completes, run the job again. The records that were not processed are picked up again and processed.

You can retain the history of data in landing tables. When you enable audit trail for a staging table, the landing table data is archived in a raw table. The MDM Hub retains the landing table data in the raw table for the number of stage job runs or the retention period that you configure. After the MDM Hub reaches the number of stage job runs or the retention period that you specify, it retains one record for each primary key of the source object in the raw table.

You can configure the MDM Hub to identify new and updated records in the landing table. If you enable delta detection for a staging table, the MDM Hub processes new and updated records and ignores unchanged records.

The MDM Hub can transfer data from one landing table to multiple staging tables. However, each staging table receives data from only one landing table.

The stage process prepares data for the load process, which loads data from the staging table into a target base object.

Load Process

This section describes concepts and tasks associated with the load process in Informatica MDM Hub.

About the Load Process

In Informatica MDM Hub, the load process moves data from a staging table to the corresponding target table (the base object) in the Hub Store.

The load process determines what to do with the data in the staging table based on:

- whether a corresponding record already exists in the target table and, if so, whether the record in the staging table has been updated since the load process was last run
- whether trust is enabled for certain columns (base objects only); if so, the load process calculates trust scores for the cell data
- whether the data is valid to load; if not, the load process rejects the record instead
- other configuration settings

Tables Associated with the Load Process

In addition to base objects, the staging table, cross-reference table, history tables, and reject table in the Hub Store are associated with the load process.

The following tables are associated with the load process:

staging table

Contains the data that was accepted and copied from the landing table during the stage process.

cross-reference table

Used for tracking the lineage of data—the source system for each record in the base object. For each source system record that is loaded into the base object, Informatica MDM Hub maintains a record in the cross-reference table that includes:

- an identifier for the system that provided the record
- the primary key value of that record in the source system
- the most recent cell values provided by that system

Each base object record will have one or more cross-reference records.

history tables

If history is enabled for the base object, and records are updated or inserted, then the load process writes to this information into two tables:

- base object history table
- cross-reference history table

reject table

Contains records from the staging table that the load process has rejected for a specific reason. Rejected records will not be loaded into base objects. The reject table is associated with the staging table (called stagingTableName_REJ). Rejected records can be inspected after running Load jobs.

Informatica MDM Hub increases performance by rejecting a record when it first encounters a reason to reject the record. The reject table describes one reason why Informatica MDM Hub rejected a record. If

there is more than one reason for Informatica MDM Hub to reject a record, the reject table describes the first reason that Informatica MDM Hub encounters.

Initial Data Loads and Incremental Loads

The initial data load (IDL) is the very first time that data is loaded into a newly-created, empty base object.

During the initial data load, all records in the staging table are inserted into the base object as new records.

Once the initial data load has occurred for a base object, any subsequent load processes are called incremental loads because only new or updated data is loaded into the base object.

Duplicate data is ignored.

Trust Settings and Validation Rules

Informatica MDM Hub uses trust and validation rules to help determine the most reliable data.

Trust Settings

If a column in a base object derives its data from multiple source systems, Informatica MDM Hub uses trust to help with comparing the relative reliability of column data from different source systems. For example, the Orders system might be a more reliable source of billing addresses than the Direct Marketing system.

Trust is enabled and configured at the column level. For example, you can specify a higher trust level for Customer Name in the Orders system and for Phone Number in the Billing system.

The following table shows two base object records to consolidate:

ROWID_OBJECT	Name	Phone
100	Doug McDougal Grp	1-555-901-4670
200	The Doug McDougal Group	201-10810

The following table shows the calculated trust scores for each column:

ROWID_OBJECT	Name	Phone
100	62	56 (the winning trust score for 'Phone' column)
200	71 (the winning trust score for 'Name' column)	37

The data with the highest trust score survives in the consolidated record. The following table shows the consolidated record:

ROWID_OBJECT	Name	Phone
100	The Doug McDougal Group	1-555-901-4670

Trust provides a mechanism for measuring the relative confidence factor associated with each cell based on its source system, change history, and other business rules. Trust takes into account the quality and age of the cell data, and how its reliability decays (decreases) over time. Trust is used to determine survivorship

(when two records are consolidated) and whether updates from a source system are sufficiently reliable to update the master record.

Data stewards can manually override a calculated trust setting if they have direct knowledge that a particular value is correct. Data stewards can also enter a value directly into a record in a base object. For more information, see the *Multidomain MDM Data Steward Guide*.

Validation Rules

Trust is often used in conjunction with validation rules, which might downgrade (reduce) trust scores according to configured conditions and actions.

When data meets the criterion specified by the validation rule, then the trust value for that data is downgraded by the percentage specified in the validation rule. For example:

```
Downgrade trust on First_Name by 50% if Length < 3
Downgrade trust on Address Line 1, City, State, Zip and Valid_address_ind if
Valid_address_ind= 'False'
```

If the Reserve Minimum Trust flag is enabled (checked) for a column, then the trust cannot be downgraded below the column's minimum trust setting.

Run-time Execution Flow of the Load Process

This section provides a detailed explanation of what can occur during the load process based on configured settings as well as characteristics of the data being processed. This section describes the default behavior of the Informatica MDM Hub load process. Alternatively, for incremental loads, you can streamline load, match, and merge processing by loading by RowID.

Determining Whether Records Already Exist

During the load process, Informatica MDM Hub first checks to see whether the record has the same primary key as an existing record from the same source system. It compares each record in the staging table with records in the target table to determine whether it already exists in the target table.

What occurs next depends on the results of this comparison.

Load Operation	Description
load insert	If a record in the staging table does not already exist in the target table, then Informatica MDM Hub <i>inserts</i> that new record in the target table.
load update	<p>If a record in the staging table already exists in the target table, then Informatica MDM Hub takes the appropriate action. A load update occurs if the target base object gets updated with data in a record from the staging table. The load process updates a record only if it has changed since the record was last supplied by the source system.</p> <p>Load updates are governed by current Informatica MDM Hub configuration settings and characteristics of the data in each record in the staging table. For example, if Force Update is enabled, the records will be updated regardless of whether they have already been loaded.</p>

During the load process, load updates are executed first, followed by load inserts.

Load Inserts

The MDM Hub performs the following steps during the Load insert process:

1. Calculate trust for cell data.
2. Run validation rules.
3. Perform foreign key lookups.
4. Create a ROWID_OBJECT value.
5. Insert the record into the target base object and other tables.

What happens during a load insert depends on the target base object and other factors.

Load Inserts and Target Base Objects

To perform a load insert for a record in the staging table:

- The load process generates a unique ROWID_OBJECT value for the new record.
- The load process performs foreign key lookups and substitutes any foreign key value(s) required to maintain referential integrity.
- The load process inserts the record into the base object, and copies into this new record the generated ROWID_OBJECT value (as the primary key for this record in the base object), any foreign key lookup values, and all of the column data from the staging table (except PKEY_SRC_OBJECT)—including null values.
The base object may have multiple records for the same object (for example, one record from source system A and another from source system B). Informatica MDM Hub flags both new records as new.
- For each new record in the base object, the load process inserts the ROWID_OBJECT value into the associated dirty table so that match keys can be regenerated during the tokenization process.
- For each new record in the base object, the load process sets its CONSOLIDATION_IND to 4 (ready for match) so that the new record can be matched to other records in the base object.
- The load process inserts a record into the cross-reference table associated with the base object. The load process generates a primary key value for the cross-reference table, then copies into this new record the generated key, an identifier for the source system, and the columns in the staging table (including PKEY_SRC_OBJECT).
Note: The base object does not contain the primary key value from the source system. Instead, the base object's primary key is the generated ROWID_OBJECT value. The primary key from the source system (PKEY_SRC_OBJECT) is stored in the cross-reference table instead.
- If history is enabled for the base object, then the load process inserts a record into its history and cross-reference history tables.
- If trust is enabled for one or more columns in the base object, then the load process also inserts records into control tables that support the trust algorithms, populating the elements of trust and validation rules for each trusted cell with the values used for trust calculations. This information can be used subsequently to calculate trust when needed.
- If Generate Match Tokens on Load is enabled for a base object, then the tokenize process is automatically started after the load process completes.

Load Updates

The MDM Hub performs the following steps during the load update process:

1. Check whether the record has changed.

2. Calculate trust for cell data.
3. Run validation rules.
4. Perform foreign key lookups.
5. Update the target record in the target base object and other tables.

What happens during a load update depends on the target base object and other factors.

Load Updates and Target Base Objects

The following changes occur during a load update on a target base object:

1. By default, for each record in the staging table, the load process compares the value in the LAST_UPDATE_DATE column with the source last update date (SRC_LUD) in the associated cross-reference table.
 - If the record in the staging table has been updated since the last time the record was supplied by the source system, then the load process proceeds with the load update.
 - If the record in the staging table is unchanged since the last time the record was supplied by the source system, then the load process ignores the record (no action is taken) if the dates are the same and trust is not enabled, or rejects the record if it is a duplicate.
Administrators can change the default behavior so that the load process bypasses this LAST_UPDATE_DATE check and forces an update of the records regardless of whether the records might have already been loaded.
2. The load process performs foreign key lookups and substitutes any foreign key value(s) required to maintain referential integrity.
3. If the target base object has trust-enabled columns, then the load process:
 - Calculates the trust score for each trust-enabled column in the record to be updated, based on the configured trust settings for this trusted column.
 - Applies validation rules, if defined, to downgrade trust scores where applicable.
4. The load process updates a record in the base object, and updates the associated record in the cross-reference table, history tables, and other control tables as applicable. Also, the load process inserts the ROWID_OBJECT value of the record into the dirty table associated with the base object so that the tokenize process can regenerate match keys. The base object retains the consolidation indicator value. The load process updates the target record in the base object according to the following rules:
 - If the trust score for the cell in the staging table record is higher than the trust score in the corresponding cell in the target base object record, then the load process updates the cell in the target record.
 - If the trust score for the cell in the staging table record is *lower* than the trust score in the corresponding cell in the target base object record, then the load process does not update the cell in the target record.
 - If the trust score for the cell in the staging table record is the same as the trust score in the corresponding cell in the target base object record, or if trust is not enabled for the column, then the cell value in the record with the most recent LAST_UPDATE_DATE wins.
 - If the staging table record has a more recent LAST_UPDATE_DATE, then the corresponding cell in the target base object record is updated.
 - If the target record in the base object has a more recent LAST_UPDATE_DATE, then the cell is not updated.
5. If Generate Match Tokens on Load is enabled for a base object, the tokenize process is automatically started after the load process completes.

Foreign Key Lookups

When a batch load or Put API inserts or updates a record, Informatica MDM Hub uses the staging table lookup configuration to translate source system foreign keys into Informatica MDM Hub foreign keys.

Disabling Referential Integrity Constraints

During the initial load/updates, or if there is no real-time, concurrent access, you can disable the referential integrity constraints on the base object to increase performance.

Undefined Lookups

If you do not populate the lookup table and lookup column to define a lookup on a child object, you must repeat the stage process for the child object prior to executing the load process before you can successfully load data.

Allowing Null Foreign Keys

When you configure columns for a staging table, you can specify whether to allow NULL foreign keys for target base objects. The Allow Null Foreign Key staging table column property determines whether null foreign key values are permitted.

Note: In the MDM Hub, an empty string is the equivalent of a null value regardless of the database type that contributes the empty string.

By default, the Allow Null Foreign Key check box is unchecked, which means that NULL foreign keys are not allowed. The load process:

- accepts records valid lookup values
- rejects records with NULL foreign keys
- rejects records with invalid foreign key values

If Allow Null Foreign Key is enabled (selected), then the load process:

- accepts records with valid lookup values
- accepts records with NULL foreign keys (and permits load inserts and load updates for these records)
- rejects records with invalid foreign key values

The load process permits load inserts and load updates for accepted records only. Rejected records are inserted into the reject table rather than being loaded into the target table.

Note: During the initial data load only, when the target base object is empty, the load process allows null foreign keys.

Rejected Records in Load Jobs

During the load process, records in the staging table might be rejected for the following reasons:

- future date or NULL date in the LAST_UPDATE_DATE column
- LAST_UPDATE_DATE less than 1900
- NULL value mapped to the PKEY_SRC_OBJECT of the staging table
- duplicates found in PKEY_SRC_OBJECT
- invalid value in the HUB_STATE_IND field (for state-enabled base objects only)
- invalid or NULL foreign keys.

Rejected records will not be loaded into base objects. Rejected records can be inspected after running Load jobs.

Note: To reject records, the load process requires traceability back to the landing table. If you are loading a record from a staging table and its corresponding record in the associated landing table has been deleted, then the load process does not insert it into the reject table.

Other Considerations for the Load Process

This section describes other considerations for the load process.

How the Load Process Handles Parent-Child Records

If the child table contains generated keys from the parent table, the load process copies the appropriate primary key value from the parent table into the child table. For example, suppose you had the following data.

PARENT TABLE:

PARENT_ID	FNAME	LNAME
101	Joe	Smith
102	Jane	Smith

CHILD TABLE: has a relationship to the PARENTS PKEY_SRC_OBJECT

ADDRESS	CITY	STATE	FKEY_PARENT
1893	my city	CA	101
1893	my city	CA	102

In this example, you can have a relationship pointing to the ROWID_OBJECT, to PKEY_SRC_OBJECT, or to a unique column for table lookup.

Loading State-Enabled Base Objects

The load process has special considerations when processing records for state-enabled base objects.

Note: The load process rejects any record from the staging table that has an invalid value in the HUB_STATE_IND column.

Generating Match Tokens (Optional)

Generating match tokens is required before running the match process. In the Schema Manager, when configuring a base object, you can specify whether to generate match tokens immediately after the Load job completes, or to delay tokenizing data until the Match job runs. The setting of the Generate Match Tokens on Load check box determines when the tokenize process occurs.

Tokenize Process

The tokenize process generates match tokens and stores them in a match key table associated with the base object. Match tokens are used subsequently by the match process to identify candidates for matching.

Match Tokens and Match Keys

Match tokens are encoded and non-encoded representations of the data in base object records. Match tokens include:

- Match keys, which are fixed-length, compressed strings consisting of encoded values built from all of the columns in the Fuzzy Match Key of a fuzzy-match base object. Match keys contain a combination of the words and numbers in a name or address such that relevant variations have the same match key value.
- Non-encoded strings consisting of flattened data from the match columns (Fuzzy Match Key as well as all fuzzy-match columns and exact-match columns).

Match Key Tables

The match key table contains the match tokens and match keys that the MDM Hub generates for base object records. A match key table is associated with each base object.

The format of match key table names is `C_<base object name>_STRP`. For example, if the name of the base object is `PARTY`, the name of the associated match key table is `C_PARTY_STRP`. The tokenize process generates one or more records in the match key table for each record in the base object.

After a match and merge process is complete, the match tokens that are not valid needs to be deleted from the match key table. Also, during retokenization, the match tokens that are not valid are deleted and replaced with valid ones. The operation to delete match tokens that are not valid can affect the performance of the MDM Hub. To improve performance in IBM DB2 environments, configure the `cmx.server.stripDML.useUpdate=true` property that marks the match tokens as not valid instead of deleting them.

The following table describes the important columns in the match key table:

Column Name	Data Type (Size)	Description
ROWID_OBJECT	CHAR (14)	Identifies the record for which the match token was generated.
SSA_KEY	CHAR (8)	Match key for the record. Encoded representation of the value in the fuzzy match key column, such as name, address, or organization name, for the associated base object record. String consists of fixed-length, compressed, and encoded values built from a combination of the words and numbers in a name or address.
SSA_DATA	VARCHAR2 (500)	Non-encoded, plain text, string representation of the concatenated match columns defined in the associated base object record. Concatenated match columns include fuzzy match key as well as all fuzzy match columns and exact match columns.

Each record in the match key table contains a match token, which is the data in both `SSA_KEY` and `SSA_DATA`.

Example Match Keys

The match keys that are generated depend on your configured match settings and characteristics of the data in the base object.

The following example shows match keys generated from strings using a fuzzy match / search strategy:

String in Record	Generated Match Keys
BETH O'BRIEN	MMU\$?/\$-
BETH O'BRIEN	PCOG\$\$\$\$
BETH O'BRIEN	VL/IEFLM
LIZ O'BRIEN	PCOG\$\$\$\$
LIZ O'BRIEN	SXOG\$\$\$\$-
LIZ O'BRIEN	VL/IEFLM

In this example, the strings BETH O'BRIEN and LIZ O'BRIEN have the same match key values (PCOG\$\$\$\$). The match process would consider these to be match candidates while searching for match candidates during the match process.

Tokenize Process Applies to Fuzzy-match Base Objects Only

The tokenize process applies to fuzzy-match base objects only—it does not apply to exact-match base objects. For fuzzy-match base objects, the tokenize process allows Informatica MDM Hub to match rows with a degree of fuzziness—the match need not be identical—just sufficiently similar to be considered a match.

Key Concepts for the Tokenize Process

This section describes key concepts that apply to the tokenize process.

Match Token Generation

The MDM Hub generates or updates match tokens when batch jobs or SIF APIs are run. The match tokens are stored in the match key table and must be up-to-date for the match process. The MDM Hub maintains the match tokens independent of the match process.

When batch jobs or SIF APIs are run, base object records might be marked dirty. The MDM Hub generates or updates match tokens for base object records that are marked dirty.

Base object records are marked dirty if all of the following conditions are met:

- An update affects match columns in a base object.
- The best version of the truth (BVT) of a match column after the update is different from the old value.

Dirty Table

All base objects have an associated dirty table named *<base object name>_DRTY*, which is a system table. The dirty table has a ROWID_OBJECT column, which identifies the base object records for which match tokens need to be generated. The MDM Hub stores the match tokens in the match key table.

For each unique ROWID_OBJECT in the dirty table, the tokenize process generates match tokens, and then cleans up the dirty table.

The MDM Hub updates the dirty table during the following sequence of batch processes:

1. **Load.** The MDM Hub loads new records or updates existing records. The MDM Hub populates the dirty table with the ROWID_OBJECT values of the new records or updated records for which match column values change.
2. **Tokenize.** The MDM Hub generates the match keys. The MDM Hub removes the ROWID_OBJECT values of records that are tokenized from the dirty table.
3. **Match.** The MDM Hub identifies matches. The dirty table remains unchanged.
4. **Consolidate.** The MDM Hub consolidates the matched records. The MDM Hub populates the dirty table with the ROWID_OBJECT values of the new records or updated records for which match column values change.
5. **Tokenize.** The MDM Hub generates the match keys. The MDM Hub removes the ROWID_OBJECT values of records that are tokenized from the dirty table.

Key Types and Key Widths in Fuzzy-Match Base Objects

For fuzzy-match base objects, match keys are generated based on the following settings:

Property	Description
key type	Identifies the primary type of information being tokenized (Person_Name, Organization_Name, or Address_Part1) for this base object. The match process uses its intelligence about name and address characteristics to generate match keys and conduct searches. Available key types depend on the population set being used.
key width	Determines the thoroughness of the analysis of the fuzzy match key, the number of possible match candidates returned, and how much disk space the keys consume. Available key widths are Limited, Standard, Extended, and Preferred.

Because match keys must be able to overcome errors, variations, and word transpositions in the data, Informatica MDM Hub generates multiple match tokens for each name, address, or organization. The number of keys generated per base object record varies, depending on your data and the match key width.

Match Key Distribution and Hot Spots

The Match Keys Distribution tab in the Match / Merge Setup Details pane of the Schema Manager allows you to investigate the distribution of match keys in the match key table. This tool can assist you with identifying potential hot spots in your data—high concentrations of match keys that could result in overmatching—where the match process generates too many matches, including matches that are not relevant.

Tokenize Ratio

You can configure the match process to repeat the tokenize process whenever the percentage of changed records exceeds the specified ratio, which is configured as an advanced property in the base object.

Optimizing the Tokenize and Merge Process Performance

To optimize the performance of the tokenize and merge process for the MDM Hub environment, add the optimization properties to the `cmxcleanse.properties` file.

1. Open the `cmxcleanse.properties` file in the following directory:

On UNIX. `<infamdm installation directory>/hub/cleanse/resources`

On Windows. `<infamdm installation directory>\hub\cleanse\resources`

2. To optimize the performance of the tokenize and merge process, add the optimization properties to the `cmxcleanse.properties` file.

The following table describes the optimization properties that you can configure:

Property	Description
<code>cmx.server.stripDML.useUpdate</code>	IBM DB2 only. When set to <code>true</code> , configures the MDM Hub to update the status of the match tokens that are not required as not valid instead of deleting them from the match key table. Default is <code>false</code> . The update to the status of the match tokens that are not valid is more efficient than the delete operation that removes the match tokens that are not valid. Note: When you set the property to <code>true</code> , to remove records that are not valid, ensure that you periodically clean up the match key tables.
<code>cmx.server.stripDML.blockSize</code>	Number of records that the MDM Hub processes in each block. Default is <code>100</code> .
<code>cmx.server.stripDML.noOfThreadsForInsert</code>	Number of threads that the MDM Hub uses to insert records into the match key tables. Default is <code>50</code> .
<code>cmx.server.stripDML.noOfThreadsForUpdate</code>	Number of threads that the MDM Hub uses to update records in the match key tables. Default is <code>30</code> .
<code>cmx.server.stripDML.noOfThreadsForDelete</code>	Number of threads that the MDM Hub uses to delete records from the match key tables. Default is <code>30</code> .

Note: Configure the block size and the thread counts for the various operations based on the requirements of the MDM Hub environment.

Clean Up Match Key Tables

You need to clean up match key tables if the match key tables contain match tokens that are not valid.

To clean up match key tables, perform one of the following tasks:

- Regenerate all match tokens.
- Run the CleanStrp batch job.
- Run a background cleanup process.

Regenerate Match Tokens

To delete all the match tokens that are not valid from the match key tables, you can regenerate all the match tokens for the base object.

1. In the Hub Console, start the Batch Viewer tool.
2. Expand the base object for which you want to regenerate all the match tokens.
3. Expand **Generate Match Tokens**.

The batch jobs associated with the base object that you can use to generate match tokens appear.

4. Select the batch job that you want to use to generate match tokens.
The properties for the Generate Match Tokens batch job appear.
5. Enable the **Re-generate All Match Tokens** option.
6. Click **Execute Batch**.

The MDM Hub regenerates match tokens for the entire base object.

Run the CleanStrp Batch Job

You can run the CleanStrp batch job to delete match tokens that are not valid from the match key tables. The CleanStrp batch job identifies match token that have `invalid_ind=1` and deletes these match tokens. Also, after the deletion of match tokens with status `invalid_ind=1`, the batch job rebuilds the indexes in the match key table.

1. In the Hub Console, start the Batch Viewer tool.
2. Expand the base object for which you want to clean up all the match tokens that are not valid.
3. Expand **CleanStrp**.

The batch jobs that you can use to clean up match tokens that are not valid from the match key tables appear.

4. Select the batch job that you want to use to clean up match tokens.
The properties for the CleanStrp batch job appear.
5. Click **Execute Batch**.

The MDM Hub cleans up match tokens that are not valid from the match key table associated with the base object.

Run the Background Cleanup Process

You can run the background cleanup process to delete match tokens that are not valid from the match key tables. The background cleanup batch process identifies match tokens that have `invalid_ind=1` and deletes these match tokens. The background cleanup process does not rebuild the indexes in the match key table after the deletion of match tokens.

- To enable the background cleanup process to run, add the related Hub Server properties to the properties file.
 - a. Open the `cmxserver.properties` file in the following directory:
On UNIX. `<infamdm install directory>/hub/server/resources`
On Windows. `<infamdm install directory>\hub\server\resources`

- b. Add the following properties in the `cmxserver.properties` file:

Property	Description
<code>cmx.server.strp_clean.execution_mode</code>	Configures the scope of operation of the background cleanup process on the match key table. Specify one of the following values for the scope of operation: <ul style="list-style-type: none">- ALL. Deletes match tokens that have <code>invalid_ind=1</code> from all the match key tables on all the registered Operational Reference Stores.- CONFIGURED_ORs. Deletes match tokens that have <code>invalid_ind=1</code> from all the match key tables on the Operational Reference Stores that you specify. If you set the scope of operation to <code>CONFIGURED_ORs</code>, add the <code>cmx.server.strp_clean.ors</code> property to the <code>cmxserver.properties</code> file.- CONFIGURED_STRP. Deletes match tokens that have <code>invalid_ind=1</code> from the match key tables of specific base objects in specific Operational Reference Stores. If you set the scope of operation to <code>CONFIGURED_STRP</code>, add the <code>cmx.server.strp_clean.strp</code> property to the <code>cmxserver.properties</code> file.
<code>cmx.server.strp_clean.ors</code>	Specifies the names of the Operational Reference Stores on which the background cleanup process must run to delete match tokens that are not valid. For example, to delete match tokens that have <code>invalid_ind=1</code> from all the match key tables in <code>cmx_ors1</code> and <code>cmx_ors2</code> , add <code>cmx.server.strp_clean.ors=cmx_ors1,cmx_ors2</code> .
<code>cmx.server.strp_clean.strp</code>	Specifies the Operational Reference Store and base object combinations for which the background cleanup process must run to clean up match key tables. For example, to delete match tokens that have <code>invalid_ind=1</code> from the match key tables for B01 in <code>cmx_ors1</code> and B02 in <code>cmx_ors2</code> , add <code>cmx.server.strp_clean.strp=cmx_ors1.C_B01,cmx_ors2.C_B02</code> .
<code>cmx.server.strp_clean.delete_records_count</code>	Specifies the number of records to clean up from the match key table.
<code>cmx.server.strp_clean.retry_sec</code>	Specifies the time duration in seconds for which you want the MDM Hub to search for records with match tokens that are not valid in the match key table. Default is 60.
<code>cmx.server.strp_clean.threads_count</code>	Specifies the number of threads that the MDM Hub uses when it searches for records with match tokens that are not valid in the match key table. Default is 20.

Match Process

Before records in a base object can be consolidated, Informatica MDM Hub must determine which records are likely duplicates, that is matches, of each other.

The match process uses match rules to perform the following tasks:

- Identify which records in the base object are likely duplicates (identical or similar)
- Determine which records are sufficiently similar to be consolidated automatically, and which records should be reviewed manually by a data steward prior to consolidation

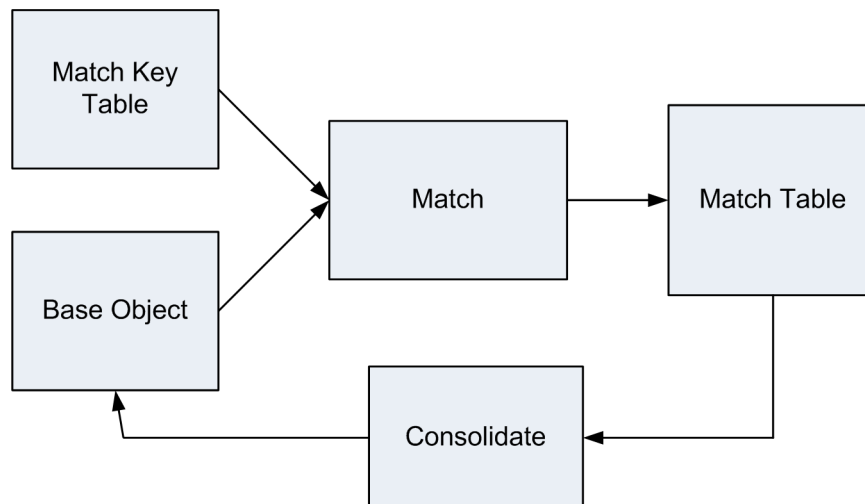
In Informatica MDM Hub, the match process provides you with two main ways in which to compare records and determine duplicates:

- Fuzzy matching is the most common means used in Informatica MDM Hub to match records in base objects. Fuzzy matching looks for sufficient points of similarity between records and makes probabilistic match determinations that consider likely variations in data patterns, such as misspellings, transpositions, the combining or splitting of words, omissions, truncation, phonetic variations, and so on.
- Exact matching is less commonly-used because it matches records with identical values in the match column(s). An exact strategy is faster, but an exact match might miss some matches if the data is imperfect.

The best option to choose depends on the characteristics of the data, your knowledge of the data, and your particular match and consolidation requirements.

During the match process, Informatica MDM Hub compares records in the base object for points of similarity. If the match process finds sufficient points of similarity (identical or similar matches) between two records, indicating that the two records probably are duplicates of each other, then the match process:

- populates a match table with ROWID_OBJECT references to matched record pairs, along with the match rule that identified the match, and whether the matched records qualify for automatic consolidation.



- flags those records for consolidation by changing their consolidation indicator to 2 (ready for consolidation).

Match Rules

A match rule defines the criteria by which Informatica MDM Hub determines whether two records in the base object might be duplicates. Match rules can be based on match columns or primary keys.

The following table describes the types of match rules that you can create:

Rule Type	Description
Exact match	Values must match exactly, or the special case must match exactly, such as null matches null. The processing for an exact-match rule uses SQL statements on the database.
Fuzzy match	Values are not an exact match, but the values are similar to the value being matched against. Matches are determined by the match tokens that are shared by some values and this depends on the population. For example, Robert, Rob, and Bob in English speaking populations, for the match purpose of Name, may have the same match token value. A fuzzy match uses the fuzzy match key, which is an index with ranges for all match tokens.
Filtered match	Identifies match candidates by using the fuzzy match key, and then runs an exact match on the candidates. Note: If you are constrained by performance issues related to the database server, consider using filtered match rules instead of exact match rules.

The following table describes match rules that are based on match columns and primary keys:

Match Rule Basis	Description
Match columns	Used to match base object records based on the values in the columns that you defined as match columns, such as last name, first name, address1, and address2. Match columns are the most commonly used method for identifying matches.
Primary keys	Used to match records from two systems that use the same primary keys for records. Primary key matches are quick and very accurate. However, it is uncommon for two source systems to use identical primary keys.

You can use both match columns and primary keys on the same base object.

Exact-match and Fuzzy-match Base Objects

A base object is configured to use one of the following types of matching:

Type of Base Object	Description
exact-match base object	Can have only exact match columns.
fuzzy-match base object	Can have both fuzzy match and exact match columns: <ul style="list-style-type: none">- fuzzy match only- exact match only, or- some combination of fuzzy and exact match

The type of base object determines the type of match and the type of match columns you can define. The base object type is determined by the selected match / search strategy for the base object.

Support Tables Used in the Match Process

The match process uses the following support tables:

Table	Description
match key table	Contains the match keys that were generated for all base object records. A match key table uses the following naming convention: <i>C_baseObjectName_STRP</i> where <i>baseObjectName</i> is the root name of the base object. Example: C_PARTY_STRP.
match table	Contains the pairs of matched records in the base object resulting from the execution of the match process on this base object. Match tables use the following naming convention: <i>C_baseObjectName_MTCH</i> where <i>baseObjectName</i> is the root name of the base object. Example: C_PARTY_MTCH.
match flag audit table	Contains the userID of the user who, in Merge Manager, queued a manual match record for automerging. Match flag audit tables use the following naming convention: <i>C_baseObjectName_FHMA</i> where <i>baseObjectName</i> is the root name of the base object. Used only if Match Flag Audit Table is enabled for this base object.

Population Sets

For base objects with the fuzzy match/search strategy, the match process uses standard population sets to account for national, regional, and language differences. The population set affects how the match process handles tokenization, the match / search strategy, and match purposes.

A population set encapsulates intelligence about name, address, and other identification information that is typical for a given population. For example, different countries use different address formats, such as the placement of street numbers and street names, location of postal codes, and so on. Similarly, different regions have different distributions for surnames—the surname “Smith” is quite common in the United States population, for example, but not so common for other parts of the world.

Population sets improve match accuracy by accommodating for the variations and errors that are likely to appear in data for a particular population.

Matching for Duplicate Data

The match for duplicate data functionality is used to generate matches for duplicates of all non-system base object columns. These matches are generated when there are more than a set number of occurrences of complete duplicates on the base object columns. For most data, the optimal value is 2.

Although the matches are generated, the consolidation indicator remains at 4 (unconsolidated) for those records, so that they can be later matched using the standard match rules.

Note: The Match for Duplicate Data job is visible in the Batch Viewer if the threshold is set above 1 and there are no NON_EQUAL match rules defined on the corresponding base object.

Build Match Groups and Transitive Matches

The Build Match Group (BMG) process removes redundant matching in advance of the consolidate process. For example, suppose a base object had the following match pairs:

- record 1 matches to record 2
- record 2 matches to record 3
- record 3 matches to record 4

After running the match process and creating build match groups, and before the running consolidation process, you might see the following records:

- record 2 matches to record 1
- record 3 matches to record 1
- record 4 matches to record 1

In this example, there was no explicit rule that matched record 4 to record 1. Instead, the match was made indirectly due to the behavior of other matches (record 1 matched to 2, 2 matched to 3, and 3 matched to 4). An indirect matching is also known as a *transitive match*. In the Merge Manager and Data Manager, you can display the complete match history to expose the details of transitive matches.

If you use the Build Match Group (BMG) process, enable **Accept All unmatched Rows as Unique** in the match properties for the base object. If you do not enable **Accept All unmatched Rows as Unique**, the wrong value for rowid_match_rule appears in the HMRG table during transitive matching.

Maximum Matches for Manual Consolidation

You can configure the maximum number of manual matches to process during batch jobs.

Setting a limit helps prevent data stewards from being overwhelmed with thousands of manual consolidations to process. Once this limit is reached, the match process stops running until the number of records ready for manual consolidation has been reduced.

External Match Jobs

Informatica MDM Hub provides a way to match new data with an existing base object without actually loading the data into the base object. Rather than run an entire Match job, you can run the External Match job instead to test for matches and inspect the results. External Match jobs can process both fuzzy-match and exact-match rules, and can be used with fuzzy-match and exact-match base objects.

Distributed Process Server

For your Informatica MDM Hub implementation, you can increase the throughput of the match process by running multiple Process Servers in parallel.

Handling Application Server or Database Server Failures

When running very large Match jobs with large match batch sizes, if there is a failure of the application server or the database, you must re-run the entire batch. Match batches are a unit. There are no incremental checkpoints. To address this, if you think there might be a database or application server failure, set your match batch sizes smaller to reduce the amount of time that will be spent re-running your match batches.

Consolidate Process

This section describes concepts and tasks associated with the consolidate process in Informatica MDM Hub.

About the Consolidate Process

After match pairs have been identified in the match process, consolidation is the process of consolidating data from matched records into a single, master record.

The following figure shows cell data in records from three different source systems being consolidated into a single master record.

Informatica MDM Hub	Master ID	First Name	MN	Last Name	Address	City	State	Zip
	M-0001	Abel	Noel	Willan	161 Washington Ave.	Buffalo	NY	14263
Sales	SFA_ID	First Name	MN	Last Name	Address	City	State	Zip
	12345	Abel		Willan	161 Washington Ave.	Buffalo	NY	14263
Accounts	Cust_ID	First Name	MN	Last Name	Address	City	State	Zip
	502068	Abel	Noel	Willan	161 Washington Ave.	Buffalo	NY	14263
Marketing	Target_ID	First Name	MN	Last Name	Address	City	State	Zip
	willan05	Abel	N	Willan	Elm & Carlston Streets	Buffalo	NY	14263

Consolidating Records Automatically or Manually

Match rules set the AUTOMERGE_IND column in the match table to specify how matched records are consolidated: automatically or manually.

- Records flagged for manual consolidation are reviewed by a data steward using the Merge Manager tool. For more information, see the *Multidomain MDM Data Steward Guide*.
- Records flagged for automatic consolidation are automatically merged. Alternately, you can run the automatch-and-merge job for a base object, which calls the match and then automerge jobs repeatedly, until either all records in the base object have been checked for matches, or the maximum number of records for manual consolidation is reached.

Traceability

The goal in Informatica MDM Hub is to identify and eliminate all duplicate data and to merge or link them together into a single, consolidated record while maintaining full traceability.

Traceability is Informatica MDM Hub functionality that maintains knowledge about which systems—and which records from those systems—contributed to consolidated records. Informatica MDM Hub maintains traceability using cross-reference and history tables.

Key Configuration Settings for the Consolidate Process

The following configurable settings affect the consolidate process.

Option	Description
base object style	Determines whether the consolidate process using merging or linking.
immutable sources	Allows you to specify source systems as immutable, meaning that records from that source system will be accepted as unique and, once a record from that source has been fully consolidated, it will not be changed subsequently.
distinct systems	Allows you to specify source systems as distinct, meaning that the data from that system gets inserted into the base object without being consolidated.
cascade unmerge for child base objects	Allows you to enable cascade unmerging for child base objects and to specify what happens if records in the parent base object are unmerged.
child base object records on parent merge	For two base objects in a parent-child relationship, if enabled on the child base object, child records are resubmitted for the match process if parent records are consolidated.

Consolidation Options

You can consolidated matched records by merging. Merging (physical consolidation) combines the matched records and updates the base object. Merging occurs for merge-style base objects.

By default, base object consolidation is physically saved, so merging is the default behavior.

Merging combines two or more records in a base object table. Depending on the degree of similarity between the two records, merging is done automatically or manually.

- Records that are definite matches are automatically merged (automerger process).
- Records that are close but not definite matches are queued for manual review (manual merge process) by a data steward in the Merge Manager tool. The data steward inspects the candidate matches and selectively chooses matches that should be merged. Manual merge match rules are configured to identify close matches.
- Informatica MDM Hub queues all other records for manual review by a data steward in the Merge Manager tool.

Match rules are configured to identify definite matches for automerging and close matches for manual merging.

To allow Informatica MDM Hub to automatically change the state of such records to Consolidated (thus removing them from the Data Steward's queue), you can check (select) the **Accept all other unmatched rows as unique** check box.

Best Version of the Truth

For a base object, the best version of the truth (sometimes abbreviated as BVT) is a record that has been consolidated with the best cells of data from the source records.

The base object record is the BVT record, and is built by consolidating with the most-trustworthy cell values from the corresponding source records.

Consolidation and Workflow Integration

For state-enabled base objects, consolidation behavior is affected by the current system state of records in the base object. For example, only ACTIVE records can be automatically consolidated—records with a PENDING or DELETED system state cannot be. To understand the implications of system states during consolidation, refer to the following topics:

- [Chapter 11, “State Management and BPM Workflow Tools” on page 169](#), especially [“State Transitions” on page 172](#) and [“Record States and Base Object Record Value Survivorship” on page 173](#)
- “Consolidating Data” in the *Multidomain MDM Data Steward Guide*.

Publish Process

Informatica MDM Hub integrates with external systems by generating XML messages about data changes in the Hub Store and publishing these messages to an outbound Java Messaging System (JMS) message queue.

Informatica MDM Hub implementations use the publish process in support of stated business and technical requirements. Other external systems, processes, or applications can listen on the JMS message queue, retrieve the XML messages, and process them accordingly.

Not all organizations take advantage of this functionality and its use in Informatica MDM Hub implementations is optional.

JMS Models Supported by Informatica MDM Hub

Informatica MDM Hub supports the following JMS models:

Point-to-point

Specific destination for a target external system.

Publish/Subscribe

Point-to-point to an Enterprise Service Bus (ESB), then publish/subscribe from the ESB to other systems.

Publish Process for Outbound Distribution of Reconciled Data

The publish process is the main outbound flow for Informatica MDM Hub. The land, stage, load, match, and consolidate processes are all associated with reconciliation, which is the main inbound flow for Informatica MDM Hub.

The publish process belongs to the main Informatica MDM Hub outbound flow for distribution. After reconciliation, Informatica MDM Hub can distribute the master record data to other applications or other databases.

Publish Process Message Triggers

Informatica MDM Hub message triggers launch the publish process.

The Informatica MDM Hub generates a message trigger when data changes in the MDM Hub Store. The message trigger creates an XML message that the Informatica MDM Hub publishes on a Java Message Service message queue. The publish process runs when it receives the XML message.

Outbound JMS Message Queues

Informatica MDM Hub use an outbound message queue as a communication channel to feed data changes back to external systems.

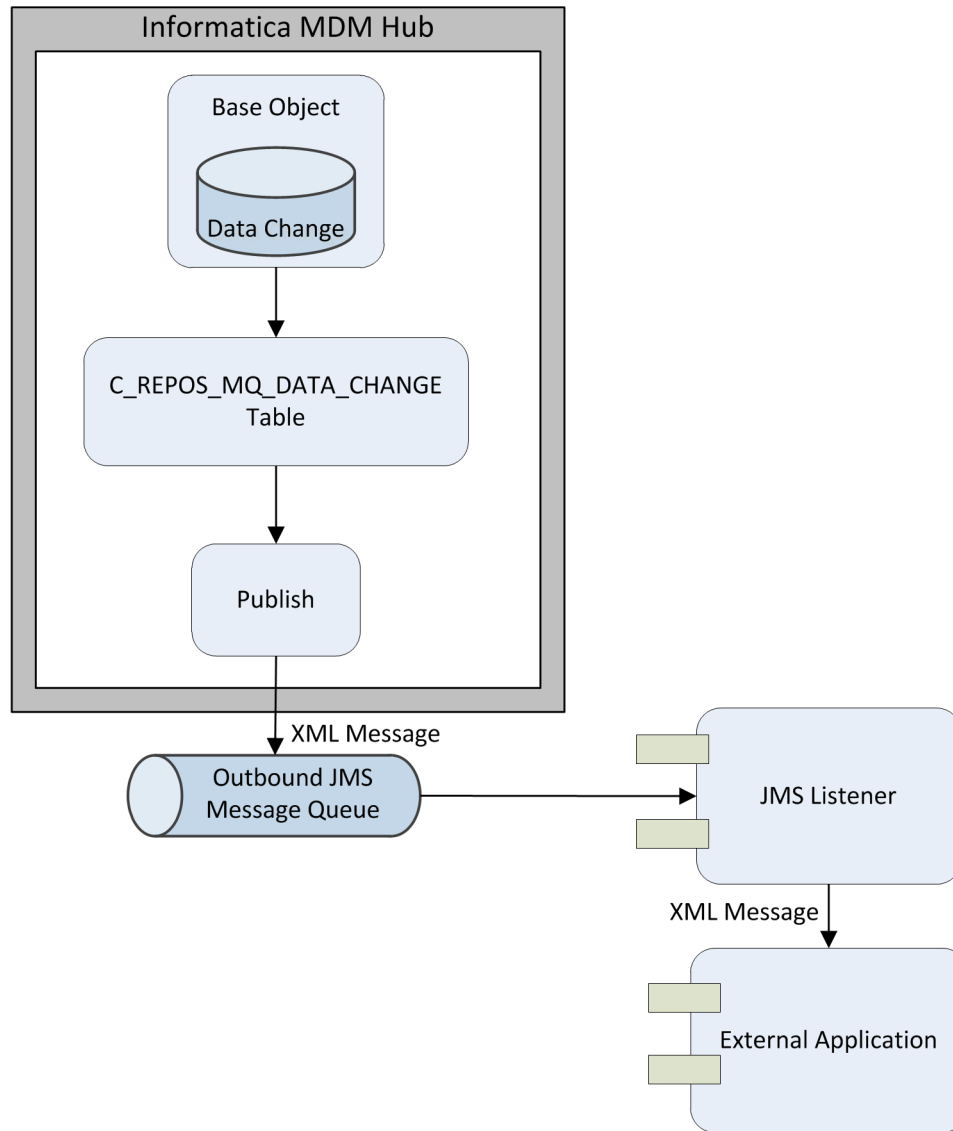
Informatica supports embedded message queues, which uses the JMS providers that come with application servers. An embedded message queue uses the JNDI name of ConnectionFactory and the name of the JMS queue to connect with. It requires those JNDI names that have been set up by the application server. The Hub Console allows you to register message queue servers and message queues that have already been configured in the application server environment.

ORS-specific XML Message Schemas

XML messages are created using an ORS-specific schema file (<ors-name>-siperian-mrm-event.xsd) that is based on a common XML schema (siperian-mrm-events.xsd). You use the JMS Event Schema Manager to generate this ORS-specific schema. This is a required task for setting up the publish process.

Run-time Flow of the Publish Process

The following figure shows the run-time flow of the publish process:



In this scenario:

1. A batch load or a real-time SIF API request (SIF put or cleanse_put request) may result in an insert or update on a base object.
You can configure a message rule to control data going to the C_REPOS_MQ_DATA_CHANGE table.
2. Hub Server polls data from C_REPOS_MQ_DATA_CHANGE table at regular intervals.
3. For data that has not been sent, Hub Server constructs an XML message based on the data and sends it to the outbound queue configured for the message queue.
4. It is the external application's responsibility to retrieve the message from the outbound queue and process it.

CHAPTER 16

Configuring the Land Process

This chapter includes the following topics:

- [Configuring the Land Process Overview, 287](#)
- [Configuring Source Systems, 287](#)
- [Configuring Landing Tables, 290](#)

Configuring the Land Process Overview

To configure the land process, perform the following tasks in the Hub Console:

- [“Configuring Source Systems” on page 287](#)
- [“Configuring Landing Tables” on page 290](#)

Configuring Source Systems

This section describes how to define source systems for your Informatica MDM Hub implementation.

About Source Systems

Source systems are external applications or systems that provide data to Informatica MDM Hub. To manage input from various source systems, Informatica MDM Hub requires a unique internal name for each source system. You use the Systems and Trust tool in the Model workbench to define source systems for your Informatica MDM Hub implementation.

Configuring Trust for Source Systems

If multiple source systems contribute data for the same column in a base object, you can configure *trust* on a column-by-column basis to specify which source system(s) are more reliable providers of data (relative to other source systems) for that column. Trust is used to determine survivorship when two records are consolidated, and whether updates from a source system are sufficiently reliable to update the “best version of the truth” record.

Administration Source System

Informatica MDM Hub uses an administration source system for manual trust overrides and data edits from the Data Manager or Merge Manager tools, which are described in the *Multidomain MDM Data Steward Guide*.

This administration source system can contribute data to any trust-enabled column. The administration source system is named Admin by default, but you can optionally change its name.

State Management Override System

The state management override system is a source system that can override the state of records from all other source systems and mark the state of records as deleted. You can specify a record as deleted, even if some cross-references indicate that the record is in the active state.

If multiple source systems contribute data to a base object record, and if at least one contributing record is in the active state, then the MDM Hub inserts the record with the deleted state from the state management override system. The overall state of the record is set to deleted.

You can set only one source system as the state management override system. Use the Systems and Trust tool of the Model workbench to enable a source system as a state management override system.

Note: The state management override system is not applicable to batch jobs.

Informatica System Repository Table

The source systems that you define in the Systems and Trust tool are stored in a special public Informatica MDM Hub repository table (C_REPOS_SYSTEM, with a display name of MDM System). This table is visible in the Schema Manager if the Show System Tables option is selected. C_REPOS_SYSTEM can also be used in packages.

Note: The C_REPOS_SYSTEM table contains Informatica MDM Hub metadata. As with any Informatica MDM Hub systems tables, you should never alter the structure of, or data in, the C_REPOS_SYSTEM table. Doing so causes Informatica MDM Hub to behave unpredictably and can result in data loss.

Starting the Systems and Trust Tool

To start the Systems and Trust tool:

- In the Hub Console, expand the Model workbench, and then click **Systems and Trust**.

The Hub Console displays the Systems and Trust tool.

The Systems and Trust tool displays the following panes:

Pane	Description
Navigation	Systems List of every source system that contributes data to Informatica MDM Hub, including the administration source system. Trust Expand the tree to display: <ul style="list-style-type: none">- base objects containing one or more trust-enabled columns- trust-enabled columns (only)
Properties	Properties for the selected source system. Trust settings for the base object column if the base object column is selected.

Source System Properties

You define the source system that must contribute to the MDM Hub. A source system definition is external to the MDM Hub. You define source systems for the MDM Hub, in the Systems and Trust tool of the Model workbench.

The following table describes the properties of a source system definition in the MDM Hub:

Property	Description
Name	Unique, descriptive name for the source system.
Primary Key	A unique identifier for the source system that the MDM Hub adds as a prefix to the primary key value for the source system. The value is read only.
State Management override system	Specifies whether to override the record state of all other source systems that contribute to the MDM Hub. Enable the property to override the record state of all other source systems. Disable the property if you do not want to override the record state of all other source systems that contribute to the MDM Hub. Default is disabled.
Description	Optional. Description for the source system.

Adding Source Systems

Use the Systems and Trust tool to define each source system that contributes data to your Informatica MDM Hub implementation.

Note: The primary key for a source system is the first 14 characters of the source system name in uppercase. The primary key is stored in the ROWID_SYSTEM field of the C_REPOS_SYSTEM repository table.

1. Start the Systems and Trust tool.
2. Acquire a write lock.
3. Right-click the list of source systems and choose **Add System**.
The Systems and Trust tool displays the New System dialog box.
4. Specify the source system properties.

5. Click **OK**.

The Systems and Trust tool displays the source system in the list.

Editing Source System Properties

You can rename any source system, including the administration system. When you rename a source system, the name is changed within the context of the Hub Console.

Note: If this source system has contributed data to your Informatica MDM Hub implementation, Informatica MDM Hub continues to track the lineage for the data from this source system.

1. Start the Systems and Trust tool.
2. Acquire a write lock.
3. In the list of source systems, select the source system.

The screen refreshes and displays the source system properties. The **Edit** icon appears beside the editable fields.

4. To edit a property, click the **Edit** icon and make the edit.
5. Optionally, edit the trust settings.
6. Click **Save**.

Removing Source Systems

You can remove a source system before it contributes data to a staging table. When you remove a source system, the source system definition and any associated metadata is deleted. There is no effect outside of Informatica MDM Hub.

Note: You cannot remove the following source systems:

- The Administration system.
- Any source system that is configured as a source for a base object. The staging table that is associated with a base object points to the source system.
- Any source system that has contributed data to a staging table, that is, the staging process has populated the staging table with data.

1. Start the Systems and Trust tool.
2. Acquire a write lock.
3. In the list of source systems, right-click the source system and select **Remove System**.
4. When prompted to confirm the action, click **Yes**.

The Systems and Trust tool removes the source system from the list.

Configuring Landing Tables

This section describes how to configure landing tables in your Informatica MDM Hub implementation.

About Landing Tables

A landing table provides intermediate storage in the flow of data from source systems into Informatica MDM Hub. In effect, landing tables are “where data lands” from source systems into the Hub Store. You use the Schema Manager in the Model workbench to define landing tables.

The manner in which source systems populate landing tables with data is entirely external to Informatica MDM Hub. The data model you use for collecting data in landing tables from various source systems is also external to Informatica MDM Hub. One source system could populate multiple landing tables. A single landing table could receive data from different source systems. The data model you use is entirely up to your particular implementation requirements.

Inside Informatica MDM Hub, however, landing tables are mapped to staging tables. It is in the staging table—mapped to a landing table—where the source system supplying the data to the base object is identified. During the load process, Informatica MDM Hub copies data from a landing table to a target staging table, tags the data with the source system identification, and optionally cleanses data in the process. A landing table can be mapped to one or more staging tables. A staging table is mapped to only one landing table.

Landing tables are populated using batch or real-time approaches that are external to Informatica MDM Hub. After a landing table is populated, the stage process pulls data from the landing tables, further cleanses the data if appropriate, and then populates the appropriate staging tables.

Landing Table Columns

Landing tables have user-defined columns, which are columns that are added by users. Also, landing tables have the SRC_ROWID system column, which uniquely identifies the landing table records.

The SRC_ROWID values help trace if a record has been loaded by the stage process. The values of the SRC_ROWID column must be unique. If duplicate values exist in the SRC_ROWID column, the stage job fails. Contact Informatica Global Customer Support if duplicate values exist in the SRC_ROWID column.

Note: If the source system table has a multiple-column key, concatenate these columns to produce a single unique VARCHAR value for the primary key column.

Landing Table Properties

Landing tables have the following properties.

Property	Description
Item Type	Type of table that you are adding. Select Landing Table .
Display Name	Name of this landing table as it will be displayed in the Hub Console.
Physical Name	Actual name of the landing table in the database. Informatica MDM Hub will suggest a physical name for the landing table based on the display name that you enter.
Data Tablespace	Name of the data tablespace for this landing table. For more information, see the <i>Multidomain MDM Installation Guide</i> .
Index Tablespace	Name of the index tablespace for this landing table. For more information, see the <i>Multidomain MDM Installation Guide</i> .
Description	Description of this landing table.

Property	Description
Create Date	Date and time when this landing table was created.
Contains Full Data Set	<p>Specifies whether this landing table contains the full data set from the source system, or only updates.</p> <ul style="list-style-type: none"> - If selected (default), indicates that this landing table contains the full set of data from the source system (such as for the initial data load). When this check box is enabled, you can configure Informatica MDM Hub's delta detection feature so that, during the stage process, only changed records are copied to the staging table. - If not selected, indicates that this landing table contains only changed data from the source system (such as for incremental loads). In this case, Informatica MDM Hub assumes that you filtered out unchanged records <i>before</i> populating the landing table. Therefore, the stage process inserts <i>all</i> records from the landing table directly into the staging table. When this check box is disabled, Informatica MDM Hub's delta detection feature is not available. <p>Note: You can change this property only when editing the source system properties.</p>

Adding Landing Tables

You can add a landing table.

1. Start the Schema Manager.
2. Acquire a write lock.
3. Select the **Landing Tables** node.
4. Right-click the Landing Tables node and choose **Add Item**.

The Schema Manager displays Add Table dialog box.

5. Specify the properties for this new landing table.
6. Click **OK**.

The Schema Manager creates the new landing table in the Operational Reference Store (Operational Reference Store), along with support tables, and then adds the new landing table to the schema tree.

7. Configure the columns for your landing table.
8. If you want to configure this landing table to contain only changed data from the source system (Contains Full Data Set), edit the landing table properties.

Editing Landing Table Properties

To edit properties in a landing table:

1. Start the Schema Manager.
2. Acquire a write lock.
3. Select the landing table that you want to edit.

The Schema Manager displays the Landing Table Identity pane for the selected table.

4. Change the landing table properties you want.
5. Click the **Save** button to save your changes.
6. Change the column configuration for your landing table, if you want.

Removing Landing Tables

To remove a landing table:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Landing Tables** node.
4. Right-click the landing table that you want to remove, and choose **Remove**.
The Schema Manager prompts you to confirm deletion.
5. Choose **Yes**.

The Schema Manager drops the landing table from the database, deletes any mappings between this landing table and any staging table (but does not delete the staging table), and removes the deleted landing table from the schema tree.

CHAPTER 17

MDM Hub Staging

This chapter includes the following topics:

- [MDM Hub Staging Overview, 294](#)
- [MDM Hub Staging Tables, 295](#)
- [MDM Hub Staging Prerequisites, 297](#)
- [Add Staging Tables, 297](#)
- [Map Columns Between a Landing Table and Staging Tables, 300](#)
- [Configure Audit Trail and Delta Detection, 309](#)
- [Staging Table Management, 313](#)
- [Mappings Management, 314](#)

MDM Hub Staging Overview

MDM Hub staging moves source data from landing tables to the staging tables associated with a base object. To perform MDM Hub staging, you must create landing tables and staging tables. Before you perform MDM Hub staging, you must load data into the landing tables.

Note: The MDM Hub can move data from a single landing table to multiple staging tables. However, each staging table receives data from only one landing table.

Before you run a stage job, define mappings between the landing and staging tables. Mappings link a source column in the landing table with a target column in the staging table. When you run the stage job, the MDM Hub moves data based on the mappings from the landing table columns to the staging table columns.

During the stage process, the MDM Hub processes one block of 250 records at a time. If there is a problem with a record in the block, the MDM Hub moves the record to the reject table. A record could be rejected because the value of a cell is too long, or because the record's update date is later than the current date. After the MDM Hub moves the rejected record, the MDM Hub stops processing the remaining records in the block, and moves on to other blocks. When the stage process completes, run the job again. The records that were not processed are picked up again and processed.

If you want to perform cleanse operations during the stage process, use the MDM Hub cleanse functions. Configure data cleansing in the mapping. When you perform MDM Hub staging, you can set up delta detection and audit trails. The stage process prepares data for the load process.

MDM Hub Staging Tables

When you run the MDM Hub stage job, the MDM Hub loads data from the landing tables into the MDM Hub staging tables. Base the structure of a staging table on the structure of the target base object that will contain the consolidated data. Use the Schema tool in the Model workbench to configure staging tables.

Perform the following tasks to configure the MDM Hub staging tables:

1. Complete the MDM Hub staging prerequisites.
2. Add staging tables.
3. Map columns between landing and staging tables.
4. Configure audit trail and delta detection.

MDM Hub Stage Process Tables

The MDM Hub stage process uses the landing table, staging table, raw table, and reject table to move data from the source system to the MDM Hub.

The MDM Hub stage process uses the following MDM Hub tables:

Landing table

An MDM Hub table that contains data that is initially loaded from a source system. The MDM Hub uses data in a landing table for the MDM Hub stage process.

Staging table

An MDM Hub table that contains data that the MDM Hub accepts during the stage process. During the An MDM Hub stage process data moves from the landing tables to the staging tables.

Raw table

An MDM Hub table that contains raw data that the MDM Hub archives from the landing tables. Each raw table is associated with a staging table and is named <staging table name>_RAW. You can configure the MDM Hub to archive raw data after a certain number of data loads or after a certain time interval. The MDM Hub inserts null primary keys even when the landing table has not changed.

Reject table

An MDM Hub table that contains records that the MDM Hub rejects and a description of the reason for each reject. Each reject table is associated with a staging table and is named <staging table name>_REJ. The MDM Hub generates a reject table for each staging table that you create. The MDM Hub inserts null primary keys even when the landing table has not changed.

The MDM Hub rejects records during the stage process for the following reasons:

- The LAST_UPDATE_DATE column contains a future date or null date.
- The LAST_UPDATE_DATE column value is less than 1900.
- A null value is mapped to the PKEY_SRC_OBJECT column of the staging table.
- The PKEY_SRC_OBJECT column contains duplicates. If the MDM Hub finds multiple records with the same PKEY_SRC_OBJECT value, the MDM Hub loads the record with the highest SRC_ROWID value. The MDM Hub moves the other records to the reject table.
- The HUB_STATE_IND field contains a value that is not valid for a base object for which state management is enabled.
- A unique column contains duplicates.

Staging Table Properties

You can create and manage staging tables through the Hub Console. You configure some staging table properties when you create a staging table.

The following table describes the staging table properties that you configure when you create a staging table:

Property	Description
Display Name	Name of the staging table as it appears in the Hub Console.
Physical Name	Name of the staging table in the database. The MDM Hub suggests a physical name for the staging table based on the display name that you enter.
Data Tablespace	Name of the data tablespace for the staging table.
Index Tablespace	Name of the index tablespace for the staging table.
Description	Description of the staging table.
Table type	Type of table. Default is <i>Staging</i> .
Create Date	Date the staging table was created.
System	Source system of the staging table data.
Preserve Source System Keys	<p>Specifies whether the MDM Hub must use key values from the source system or use the key values that the MDM Hub generates. Enable to use key values from the source system. Disable to use the key values that the MDM Hub generates. Default is disabled.</p> <p>Note: During the stage process, if multiple records contain the same PKEY_SRC_OBJECT, the surviving record is the one with the most recent LAST_UPDATE_DATE. The other records are sent to the reject table.</p>
Fill on gap	<p>Specifies whether contiguity between the effective dates of record versions is maintained when you add new record versions. If enabled, when you can add a new record version to the base object, the MDM Hub maintains the contiguity between effective periods of record versions. If not enabled, the MDM Hub rejects any addition of record version that breaks the contiguity between effective periods of record versions. Default is disabled.</p>
Highest Reserved Key	<p>Number by which the key must increase after the first load. The property appears if you enable the Preserve Source System Keys check box.</p>
Cell Update	<p>Enables the MDM Hub to update the cell in the target table if the value in the incoming record from the staging table is the same.</p>
Columns	Columns in the staging table.

Mapping Properties

When you create a mapping between the landing and staging tables, you configure properties for the mappings.

The following table describes mappings properties.

Field	Description
Name	Name of the mapping as it will be displayed in the Hub Console.
Description	Description of the mapping.
Landing Table	Select the landing table that will be the <i>source</i> of the mapping.
Staging Table	Select the staging table that will be the <i>target</i> of the mapping.
Secure Resource	Enable to make the mapping a secure resource, which allows you to control access to the mapping. After a mapping is designated as a secure resource, you can assign privileges to it in the Secure Resources tool.

MDM Hub Staging Prerequisites

Before you begin, perform the following installation and configuration tasks:

1. Configure the external cleanse engine that you need to cleanse data.
2. Complete the land process to load data into the landing tables.

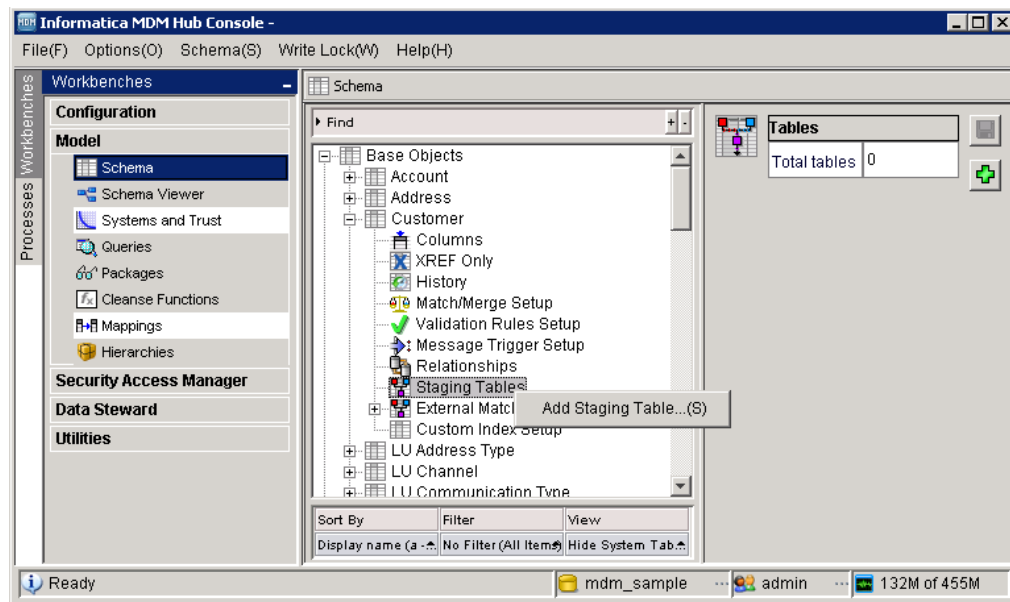
Add Staging Tables

Add a staging table to which you want to move data from a landing table.

1. In the Hub Console, start the Schema tool.
2. Acquire a write lock.
3. In the navigation tree, expand the node of the base object that you want to add a staging table to.
4. Right-click the Staging Tables node.

The **Add Staging Table** option appears.

The following image shows the **Add Staging Table** option to add a staging table to the Customer base object:



5. Click **Add Staging Table**.
The **Add Staging to Base Object** dialog box appears.
6. Specify the staging table properties.

The following image shows the **Add Staging to Base Object Customer** dialog box with the **Display Name** field set to S_CRM_CUST:

Table Identity	
Display name	S_CRM_CUST
Physical name	C_S_CRM_CUST
System	SFA
Preserve Source System Keys	<input type="checkbox"/>
Data Tablespace	CMX_DATA
Index Tablespace	CMX_INDX
Description	

Columns							
		Column	Lookup Sy...	Lookup Table	Lookup Col...	Allow Null ...	Allow Null ...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	First Name				<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Last Name				<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Gender Cd				<input type="checkbox"/>	<input type="checkbox"/>

☐ Cell update

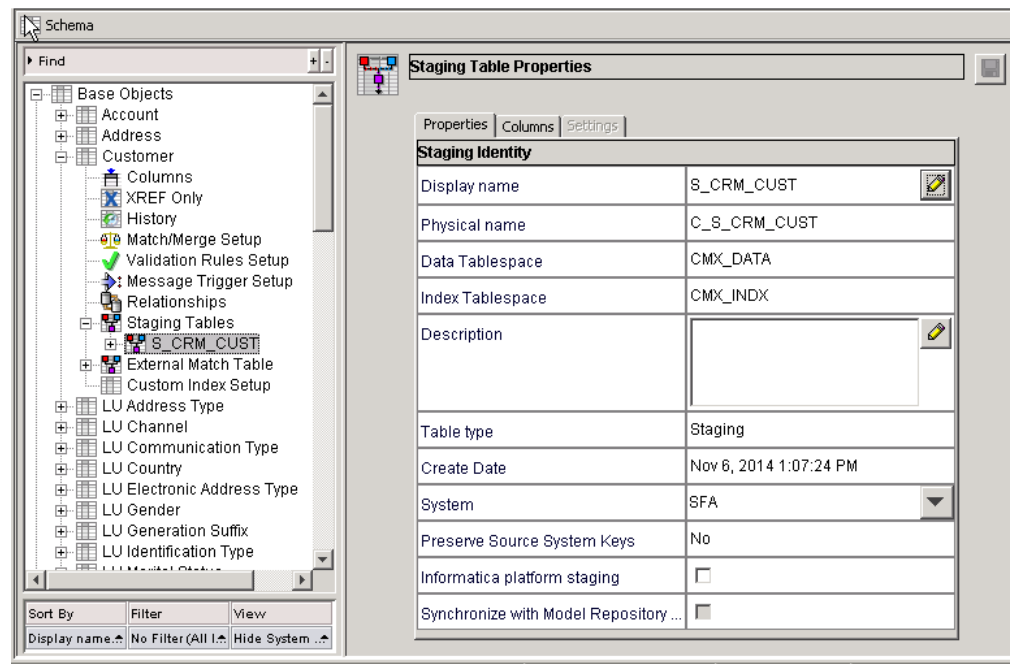
OK Cancel

7. From the list of the columns in the base object, select the columns that the source system provides. You can click **Select all columns** to select all the base object columns.
8. Specify column properties.

9. Click **OK**.

The Schema tool creates the staging table in the Operational Reference Store along with any support tables, and then adds the staging table to the navigation tree.

The following image shows the S_CRM_CUST staging table in the navigation tree:



Map Columns Between a Landing Table and Staging Tables

You map columns from one landing table to one or more staging tables. The column mapping defines how to transfer data from the columns in the landing table to the columns in staging tables. You must map at least a unique primary key to ensure traceability from the source data to the master data. You can also map other source data to master data.

Each column mapping has inputs and outputs. The inputs are columns from the landing table. The outputs are columns in a staging table. You can pass the inputs through a cleanse function that acts on the data. For example, you can standardize data, verify data, or aggregate data. You can also define whether the column mapping as a secure resource or a private resource.

After you configure the column mappings, you run a stage job to populate the staging tables with the source data.

Data is Either Cleansed or Passed Through Unchanged

For each column of data in the staging table, the data comes from the landing column in one of two ways:

Copy Method	Description
passed through	Informatica MDM Hub copies the data as is, without making any changes to it. Data comes directly from a column in the landing table.
cleansed	Informatica MDM Hub standardizes and verifies data using cleanse functions. The output of the cleanse function becomes the input to the target column in the staging table.

Note: A staging table does not need to use every column in the landing table or every output string from a cleanse function. The same landing table can provide input to multiple staging tables, and the same cleanse function can be reused for multiple columns in multiple landing tables.

Decomposition and Aggregation

Cleanse functions can also decompose data and aggregate data.

Cleanse Functions that Decompose Data

A decomposition cleanse function takes data from one field, breaks up the data into smaller pieces, and assigns the pieces to different columns in the staging table.

For example, you have a salutation field that you want to break up into five separate name-type fields. The decomposition cleanse function has one input string and five output strings. In the mapping, you map the input string to the cleanse function, and you map each output string to the target columns in the staging table.

Cleanse Functions that Aggregate Data

An aggregation cleanse function takes data from multiple fields, aggregates the data, and assigns the aggregated data to a single column in the staging table.

For example, you have five name-type fields that you want to aggregate into a single salutation field. The aggregation cleanse function has five input strings and one output string. In the mapping, you map the input strings to the cleanse function, and you map the output string to the target column in the staging table.

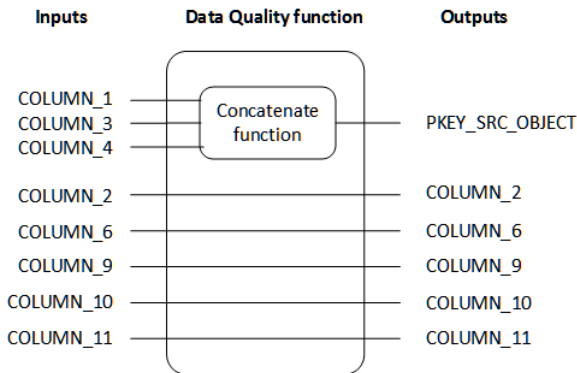
Informatica Data Quality Batch Jobs and Column Mappings

You can use Informatica Data Quality batch jobs to process Informatica MDM master records. To use Informatica Data Quality batch jobs, all the columns that you want to map from the landing table must go through the same Data Quality function. The columns can be cleansed within the function or pass through the function unchanged.

Note: If the mapping does not pass all inputs through a single Data Quality function, you cannot use Informatica Data Quality batch jobs. Instead, Informatica Data Quality processes the records one at a time.

For example, let's say that you want to construct a primary key from multiple columns in the landing table. You create a mapping, add a Data Quality function, and then add all the columns that you want to map from the landing table to the staging table. Within the Data Quality function, you add a function to concatenate the columns for the primary key, and map the output of the concatenate function to the PKEY_SRC_OBJECT column. For the remaining columns, you can use other functions or pass through the columns unchanged.

The following diagram shows the Data Quality function with all the required inputs and the mapped outputs:



If you enable hard delete detection on the staging table, specify the columns that you used to construct the primary key in the hard delete detection table.

Starting the Mappings Tool

To start the Mappings tool, in the Hub Console, expand the Model workbench, and then click **Mappings**.

The Hub Console displays the Mappings tool with the following panels:

Column	Description
Mappings List	List of every defined landing-to-staging mapping.
Properties	Properties for the selected mapping.

When you select a mapping in the mappings list, its properties are displayed.

Tabs in the Mappings Tool

When a mapping is selected, the Mappings tool displays the following tabs:

Tab	Description
General	General properties for this mapping.
Diagram	Interactive diagram that lets you define mappings between columns in the landing and staging tables.
Query Parameters	Allows you to specify query parameters for this mapping.
Test	Allows you to test the mapping.

Mapping Diagrams

When you click the Diagram tab for a mapping, the Mappings tool displays the current column mappings.

Mapping lines show the mapping from source columns in the landing table to target columns in the staging table. Colors in the circles at either end of the mapping lines indicate data types.

Creating a Mapping

Create a mapping between a landing table and a staging table.

1. In the **Model** workbench, click **Mappings**.
2. Acquire a write lock.
3. Right-click in the mappings area and select **Add Mapping**.
The Mappings tool displays the Mapping dialog box.
4. Specify the mapping properties.

Field	Description
Name	Name of the mapping as it will be displayed in the Hub Console.
Description	Description of the mapping.
Landing Table	Select the landing table that will be the <i>source</i> of the mapping.
Staging Table	Select the staging table that will be the <i>target</i> of the mapping.
Secure Resource	Enable to make the mapping a secure resource, which allows you to control access to the mapping. After a mapping is designated as a secure resource, you can assign privileges to it in the Secure Resources tool.

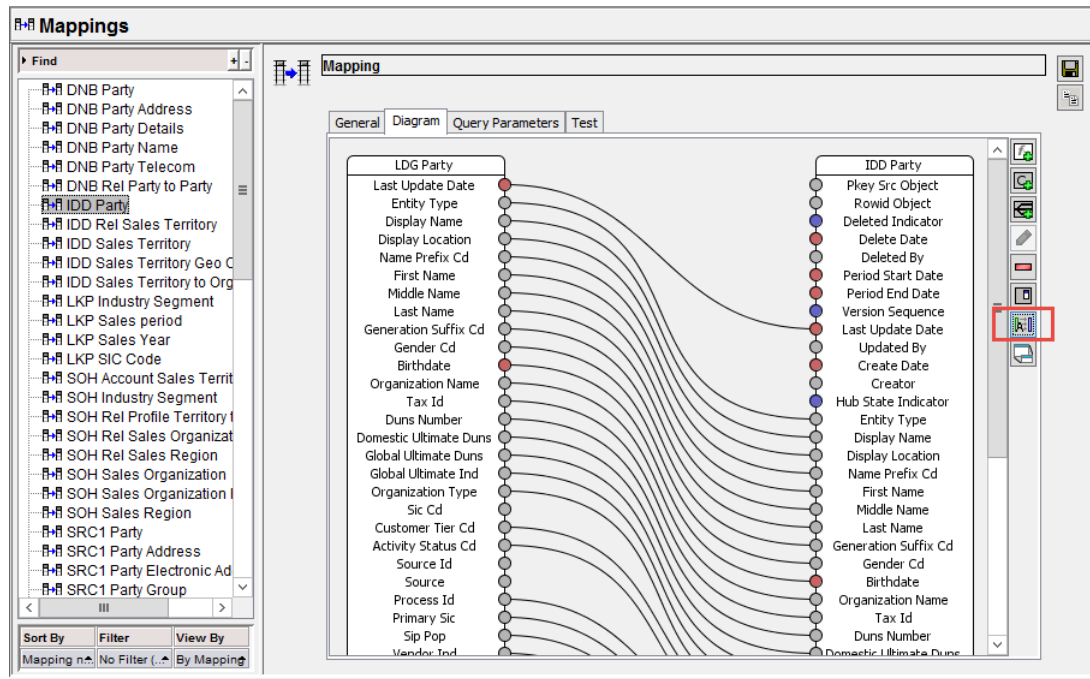
The following image shows a sample mapping:

Mapping	
Name	IDD Party
Description	Example mapping
Landing table	LDG Party
Staging table	IDD Party
Secure Resource	<input checked="" type="checkbox"/>

OK Cancel

5. Click **OK**.
6. Click the **Diagram** tab.
The Mappings tool displays the landing table and staging table on the workspace.
7. To map columns with the same name from the landing table to the staging table, click the **Auto mapping** button.

The following image shows the results of automapping on the sample mapping. You can change the connections:



8. Click **Save**.

You have a basic mapping. Next, you map the primary key.

Mapping the Primary Key

To identify a record in the source system, the MDM Hub requires a unique key, called a primary key. If the source system contains a column with unique keys, map that column from the landing table to the

PKEY_SRC_OBJECT column in the staging table. If the source system does not contain unique keys, you can create unique keys by concatenating the values from multiple columns.

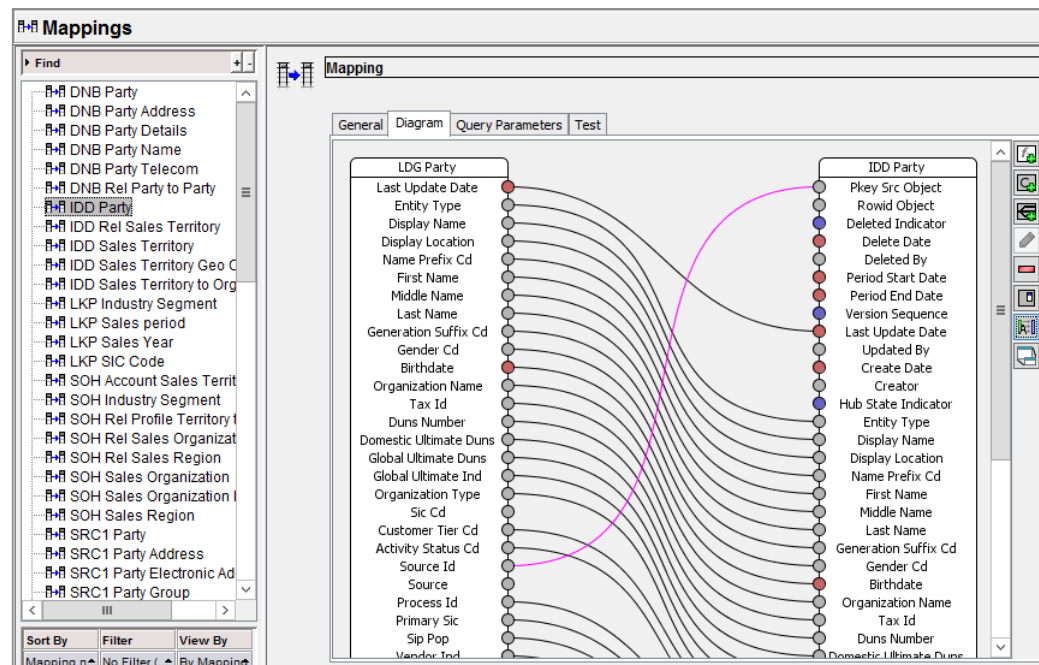
Using a Single Column as the Primary Key

If the source system contains a column with unique keys, map that column from the landing table to the PKEY_SRC_OBJECT column in the staging table.

This step assumes that you have the mapping open in the Diagram workspace.

1. In the **Diagram** workspace, find the unique column name in the landing table. Drag its output connector to the input connector on the PKEY_SRC_OBJECT column in the staging table.

A line connects the columns.



2. Click **Save**.

Using Multiple Columns as the Primary Key

If the source data does not have a column with unique values, concatenate the values from multiple columns to form unique primary keys. Use a string concatenation function from MDM Hub or Informatica Data Quality, or create a custom function.

Ensure that you have the mapping open in the **Diagram** workspace.

Note: Although more than three columns from the landing table can be mapped to the PKEY_SRC_OBJECT column in the staging table, index creation is restricted to a maximum of three columns.

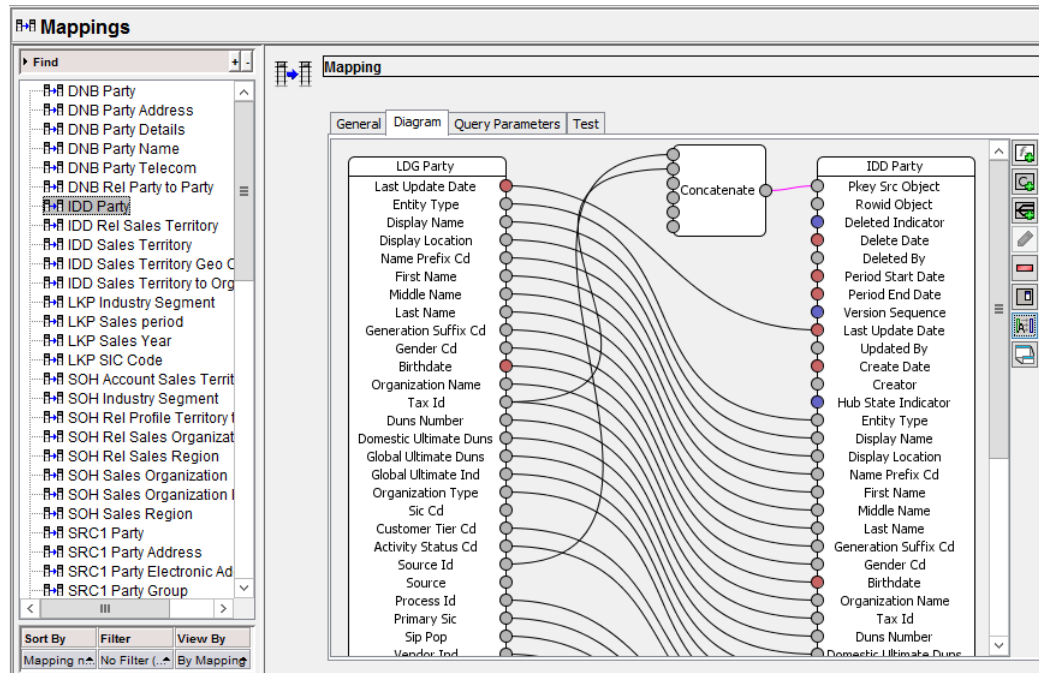
1. Identify the columns that you want to concatenate for the primary key.
2. Add a concatenate function.
 - a. In the **Diagram** workspace, right-click the workspace and select **Add Function**.
 - b. Expand **String Functions**.
 - c. Click **Concatenate**.

d. Click **OK**.

The concatenate function appears in the workspace.

3. From the landing table, for each column that you want to concatenate, drag its output connector to an input connector on the concatenate function.
4. From the function, drag the output connector to the input connector on the PKEY_SRC_OBJECT column in the staging table.

The primary key connection is complete.



5. Click **Save**.

Mapping Columns

For a direct connection, connect columns from the landing table to columns in the staging table. To cleanse the data coming from the landing table, add a function. The inputs to the function come from columns in the landing table. The output from the function goes to columns in the staging table.

This step assumes that you have the mapping open in the Diagram tab. If you used the Auto Mapping process, the columns with the same names are connected directly.

1. In the **Diagram** workspace, review the connections that were created automatically. If any of the automatic connections are incorrect, delete the connections.
 - a. Click the connector line.
 - b. Right-click and select **Delete Link**.
2. Decide if any of the unconnected columns in the landing table must be connected to columns in the staging table.

When planning a connection, consider the following requirements:

- Data types. The columns must have either the same data type or a data type that can be implicitly converted.

- Cleansing. Do you want to cleanse the data or transform the data before loading it into the staging table? If so, you need to decide which function you want to use.
 - String length. In the staging table, columns with the CHAR or VARCHAR data types must have a length that is equal to or greater than the length for the column in the landing table.
Warning: At load time, if a string is too long for the column in the staging table, the load process sends the entire record to the reject table.
3. If data does not require cleansing, connect the columns directly. To connect the columns, from the landing table column, drag the output connector to the input connector on staging table column.
 4. If data requires cleansing, add a function to the workspace and then connect the columns.
 - a. In the workspace, right-click and select **Add Function**.
 - b. Select the function and click **OK**.
The function appears in the Diagram workspace.
 - c. Find the column in the landing table. Drag its output connector to the input connector on the function. Repeat to add additional columns as inputs to the function.
 - d. From the function, drag its output connector to the input connector of the column in the staging table.
 5. After you finish adding connections, click **Save**.

Filtering Records in Mappings

By default, all records are retrieved from the landing table.

Optionally, you can configure a mapping that filters records in the landing table. There are two types of filters: distinct and conditional. You configure these settings on the Query Parameters tab in the Mappings tool.

Distinct Mapping

If you click the Enable Distinct check box on the Query Parameters tab, the Stage job selects only the distinct records from the landing table. Informatica MDM Hub populates the staging table using the following SELECT statement:

```
select distinct * from landing_table
```

Using distinct mapping is useful in situations in which you have a single landing table feeding multiple staging tables and the landing table is denormalized (for example, it contains both customer and address data). A single customer could have three addresses. In this case, using distinct mapping prevents the two extra customer records from being written to the rejects table.

In another example, suppose a landing table contained the following data:

LUD	CUST_ID	NAME	ADDR_ID	ADDR
7/24	1	JOHN	1	1 MAIN ST
7/24	1	JOHN	2	1 MAPLE ST

In the mapping to the customer table, check (select) Enable Distinct to avoid having duplicate records because only LUD, CUST_ID, and NAME are mapped to the Customer staging table. With Distinct enabled, only one record would populate your customer table and no rejects would occur.

Alternatively, for the address mapping, you map ADDR_ID and ADDR with Distinct disabled so that you get two records and no rejects.

Conditional Mapping

If you select the Enable Condition check box, you can apply a SQL WHERE clause to unload the data in cleanse. For example, suppose the data in your landing table is from all states in the US. You can use the WHERE clause to filter the data that is written to the staging tables to include only data from one state, such as California. To do this, type in a WHERE clause (but omit the WHERE keyword): STATE = 'CA'. When the cleanse job is run, it unloads and processes records as SELECT * FROM LANDING WHERE STATE = 'CA'. If you specify conditional mapping, click the Validate button to validate the SQL statement.

Configuring Query Parameters for Mappings

To configure query parameters for a mapping:

1. Start the Mappings tool.
2. Acquire a write lock.
3. Select the mapping that you want to configure.
4. Click the Query Parameters tab.
The Mappings tool displays the Query Parameters tab for this mapping.
5. If you want, check or uncheck the **Enable Distinct** check box, as appropriate, to configure distinct mapping.
6. If you want, check or uncheck the **Enable Condition** check box, as appropriate, to configure conditional mapping.
If enabled, type the SQL WHERE clause (omitting the WHERE keyword), and then click **Validate** to validate the clause.
7. Click the **Save** button to save your changes.

Maintaining Mappings that Use Cleanse Functions

The mappings that contain cleanse functions can be affected by the state of the cleanse engine.

Under the following circumstances, the MDM Hub removes functions from mappings:

- You change cleanse engines in your environment and the functions that you use in the mappings are not available in the new cleanse engine.
- You restart the application server for the Process Server and the cleanse engine fails to initialize.

To restore the functions to the mappings, edit the mappings and add functions that are available in the active cleanse engine.

Loading by Row ID

You can configure load by row ID for the Informatica MDM Hub to streamline load, match, and merge processing.

When you configure the stage process, you need to map the landing table columns to the staging table columns. To configure load by row ID, map the landing table column that contains ROWID_OBJECT values to the ROWID_OBJECT column in a staging table. For example, you can provide ROWID_OBJECT values to the MDM Hub by mapping the INPUT_ROWID_OBJECT column in a landing table to the ROWID_OBJECT column in a staging table.

The load process compares the values from the ROWID_OBJECT column in a staging table with the ORIG_ROWID_OBJECT column values of the cross-reference table that is associated with a base object. If the values match, the load process updates the records in the cross-reference table.

The load process updates the existing cross-reference record with the incoming record if the values of the PKEY_SRC_OBJECT and ROWID_SYSTEM columns match. Otherwise, the load process inserts a new cross-reference record that has the values of the incoming record, and merges with the existing cross-reference record. The load process updates the base object record with the winning cell values from the cross-reference records.

When you load a HUB_STATE_IND value of -1 for a specified source and specified row ID, the HUB_STATE_IND value is set to -1 for all cross-reference records of the specified row ID, regardless of the source. The MDM Hub considers the base object record to be deleted since all cross-reference records have a HUB_STATE_IND value of -1.

The initial data load for a base object inserts all records into the target base object. Therefore, enable loading by row ID for incremental loads that occur after the initial data load.

Configure Audit Trail and Delta Detection

After you finish mapping columns to the staging tables, you can configure audit trail and delta detection for the staging table data.

If you enable audit trail, the MDM Hub retains records in the raw table. The MDM Hub retains records for either the number of stage job runs or the retention period that you configure. All duplicate records in a landing table are present in the corresponding raw table. After the MDM Hub reaches the number of stage job runs or the retention period, for each stage job run, the MDM Hub preserves one record for each primary key of the source object. For example, if the retention that you configure is 2 stage job runs, in the first two stage job runs, the MDM Hub preserves all the duplicates for a primary key of the source object in the raw table. After the two stage job runs, the raw table retains one record for each primary key of the source object for each stage job run.

If you enable delta detection, the MDM Hub detects the records in the landing table that are new or updated and copies the records, unchanged, to the corresponding raw table. Otherwise, the MDM Hub copies the records to the target table.

To configure audit trail and delta detection, click the **Settings** tab.

Configuring the Audit Trail for a Staging Table

Informatica MDM Hub allows you to configure an audit trail that retains the history of the data in the RAW table based on the number of Loads and timestamps.

This audit trail is useful, for example, when using HDD (Hard Delete Detection). By default, audit trails are not enabled, and the RAW table is empty. If enabled, then records are kept in the RAW table for either the configured number of stage job executions or the specified retention period.

Note: The Audit Trail has very different functionality from—and is not to be confused with—the Audit Manager tool.

To configure the audit trail for a staging table:

1. Start the Schema Manager.
2. Acquire a write lock.
3. If you have not already done so, add a mapping for the staging table.
4. Select the staging table that you want to configure.

- At the bottom of the properties panel, click **Preserve an audit trail in the raw table** to enable the raw data audit trail.

The Schema Manager prompts you to select the retention period for the audit table.

- Selecting one of the following options for audit retention period:

Option	Description
Loads	Number of batch loads for which to retain data.
Time Period	Period of time for which to retain data.

- Click **Save** to save your changes.

Once configured, the audit trail keeps data for the retention period that you specified. For example, suppose you configured the audit trail for two loads (Stage job executions). In this case, the audit trail will retain data for the two most recent loads to the staging table. If there were ten records in each load in the landing table, then the total number of records in the RAW table would be 20.

If the Stage job is run multiple times, then the data in the RAW table will be retained for the most recent two sets based on the ROWID_JOB. Data for older ROWID_JOBS will be deleted. For example, suppose the value of the ROWID_JOB for the first Stage job is 1, for the second Stage job is 2, and so on. When you run the Stage job a third time, then the records in which ROWID_JOB=1 will be discarded.

Note: Using the Clear History button in the Batch Viewer after the first run of the process: If the audit trail is enabled for a staging table and you choose the Clear History button in the Batch Viewer while the associated stage job is selected, the records in the RAW and REJ tables will be cleared the next time the stage job is run.

Configuring Delta Detection for a Staging Table

If you enable delta detection for a staging table, Informatica MDM Hub processes only new or changed records and ignores unchanged records. Records with null values or future dates in the columns that you select for delta detection are rejected by the staging process. In case of timeline-enabled base objects, if you run delta detection based on PERIOD_START_DATE or PERIOD_END_DATE columns, records with null values or future dates in the PERIOD_START_DATE or PERIOD_END_DATE columns are rejected.

Enabling Delta Detection on Specific Columns

To enable delta detection for specific columns:

- Start the Schema Manager.
- Acquire a write lock.
- Select the staging table that you want to configure.
The Staging Table Properties pane is displayed.
- Select the **Enable delta detection** check box.
- Select the **Detect deltas using specific columns** option.
A list of available columns is displayed.
- Choose the columns that you want to use for delta detection and click **Add >**, or click **Add All >>** to add all the columns for delta detection.

When loading data to the staging table, if any column from the defined set has values different from the available previous load values, the row is treated as changed. If all columns from defined set are the same, the row is treated as unchanged. Columns that are not mapped are ignored.

Enabling Delta Detection for a Staging Table

To enable delta detection for a staging table:

1. Start the Schema Manager.
2. Select the staging table that you want to configure.
3. Acquire a write lock.
4. Select (check) the **Enable delta detection** check box to enable delta detection for the table. You might need to scroll down to see this option.
5. Specify the manner in which you want to have deltas detected.

Option	Description
Detect deltas by comparing all columns in mapping	All columns are selected for delta comparison, including the Last Update Date.
Detect deltas via a date column	If your schema has an applicable date column, choose this option and select the date column you want to use for delta comparison. This is the preferred option in cases where you have an applicable date column.

6. Specify whether to allow staging if a prior duplicate was rejected during the stage process or load process.
 - Select (check) this option to *allow* the duplicate record being staged, during this next stage process execution, to bypass delta detection if its previously-staged duplicate was rejected.
Note: If this option is enabled, and a user in the Batch Viewer clicks the **Clear History** button while the associated stage job is selected, then the history of the prior rejection (that this feature relies on) will be discarded because the records in the REJ table will be cleared the next time the stage job is run.
 - Clear (uncheck) this option (the default) to prevent the duplicate record being staged, during this next stage process execution, from bypassing delta detection if its previously-staged duplicate was rejected. Delta detection will filter out any corresponding duplicate landing record that is subsequently processed in the next stage process execution.

How Informatica MDM Hub Handles Delta Detection

If delta detection is enabled, then the Stage job compares the contents of the landing table—which is mapped to the selected staging table—against the data set processed in the previous run of the stage job.

This comparison is done to determine whether the data has changed since the previous run. Changed, new records, and rejected records will be put into the staging table. Duplicate records are ignored.

Note: Reject records move from cleanse to load after the second stage run.

Considerations for Using Delta Detection

When you use delta detection, you must consider certain issues.

Consider the following issues:

- Delta detection can be done either by comparing entire records or via a date column. Delta detection on last update date is the most efficient, as Informatica MDM Hub can simply compare the last update date columns for each incoming record against the record's previous last update date.
- With Delta detection you have the option to not include the column in landing table that is mapped to the last_update_date column in the staging table for delta detection.
Note: If you do include the last_update_date column when you set up Delta Detection, and the only thing that changes is the last_update_date column, the MDM Hub will do lots unnecessary work in delta detection and load work.
- When processing records by last update date, do not use the Now cleanse function to compare last update values (for example, testing whether the last update date in a source record occurred before the current system date). Using Now in this way can produce unpredictable results.
- Perform delta detection only on columns for those sources where the Last Update Date is not a true indicator of change. The Informatica MDM Hub stage job will compare the entire source record against the most recent corresponding record in the PRL (previous load) table. If any cell is different, then the record is passed on to the staging table. Delta detection is being done from the PRL table.
- If the delta detection is based on a date column (last_update_date, system-defined date column, or custom-defined date column, such as DOB), then only the record with the later date value in the date column (in comparison to the corresponding record in the PRL table, not the max date in the RAW table) will be inserted into the staging table.
- If the delta detection is based on specific columns, including date columns (such as DOB and last_update_date), then records with any change to the date values in the specified columns (in comparison to the corresponding record in the PRL table, not the max date in the RAW table) will be inserted into the staging table.
- During delta detection, when you are checking for deltas on all columns, only records that have null primary keys are rejected. This is expected behavior. Any other records that fail the delta process are rejected on subsequent stage processes.
- When delta detection is based on the Last Update Date, any changes to the last update date or the primary key will be detected. Updates to any values that are not the last update date or part of the concatenated primary key will not be detected.
- Duplicate primary keys are not considered during subsequent stage processes when using delta detection by mapped columns.
- Reject handling allows you to:
 - View all reject records for a given staging table regarding of the batch job
 - View all reject records by day across all staging tables
 - Query reject tables based on query filters

Staging Table Management

After you configure the MDM Hub staging tables, you might want to change the staging table properties. Also, you might want to view the source system associated with a staging table. You can also remove a staging table that you do not need.

Changing Properties in Staging Tables

You can change the staging table properties when you need.

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Base Objects** node, and then expand the node for the base object associated with this staging table.
If the staging table is associated with the base object, then expand the **Staging Tables** node to display it.
4. Select the staging table that you want to configure.
The Schema Manager displays the properties for the selected table.
5. Specify the staging table properties.
For each property that you want to edit, click the **Edit** button next to it, and specify the new value.
Note: You can change the source system if the staging table and its related support tables such as the raw and primary landing tables are empty.
Do not change the source system if the staging table or its related tables contain data.
6. From the list of the columns in the base object, change the columns that the source system will provide.
 - Click the **Select All** button to select all of the columns without needing to click each column individually.
 - Click the **Clear All** button to uncheck all selected columns.**Note:** The Rowid Object and the Last Update Date are automatically selected. You cannot uncheck these columns or change their properties.
7. If you want, change column properties.
8. If you want, change lookups for foreign key columns. Select the column and click the **Edit** button to configure the lookup column.
9. If you want to change cell updating, click the **Cell update** check box.
10. Change the column configuration for your staging table, if you want.
11. If you want, configure an Audit Trail and Delta Detection for this staging table.
12. Click the **Save** button to save your changes.

Jumping to the Source System for a Staging Table

To view the source system associated with a staging table, you must right-click the staging table and choose **Jump to Source System**.

The Hub Console launches the Systems and Trust tool and displays the source system associated with this staging table.

Removing Staging Tables

To remove a staging table:

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Base Objects** node, and then expand the node for the base object associated with this staging table.

4. Right-click the staging table that you want to remove, and then choose **Remove**.

The Schema Manager prompts you to confirm deletion.

5. Choose **Yes**.

The Schema Manager drops the staging table from the Operational Reference Store (Operational Reference Store), deletes associated control tables, and removes the deleted staging table from the schema tree.

Mappings Management

You might want to change the properties of a mapping, remove a mapping, or view the schema associated with a mapping. Also, you might want to test a mapping before you run staging.

Editing Mapping Properties

To create a new mapping by copying an existing one:

1. Start the Mappings tool.
2. Acquire a write lock.
3. Select the mapping that you want to edit.
4. Edit the mapping properties, diagram, and mapping settings as needed.
5. Click the **Save** button to save your changes.

Copying Mappings

To create a new mapping by copying an existing one:

1. Start the Mappings tool.
2. Acquire a write lock.
3. Right-click the mapping that you want to copy, and then choose **Copy Mapping**.

The Mappings tool displays the Mapping dialog.

4. Specify the mapping properties.
5. Click **OK**.
6. Click the **Save** button to save your changes.

Jumping to a Schema

The Mappings tool allows you to quickly launch the Schema Manager and display the schema associated with the selected mapping.

Note: The Jump to Schema command is available only in the Workbenches view, not the Processes view.

To jump to the schema for a mapping:

1. Start the Mappings tool.
2. Select the mapping whose schema you want to view.
3. In the View By list at the bottom of the navigation pane, choose one of the following options:
 - By Staging Table
 - By Landing Table
 - By Mapping
4. Right-click anywhere in the navigation pane, and then choose **Jump to Schema**.

The Mappings tool displays the schema for the selected mapping.

Testing Mappings

To test a mapping that you have configured:

1. Start the Mappings tool.
2. Acquire a write lock.
3. Select the mapping that you want to configure.
4. Click the **Test** tab.

The Mappings tool displays the Test tab for this mapping.
5. Specify input values for the columns under Input Name.
6. Click **Test**.

The Mappings tool tests the mapping and populates the columns under Output Name with the results.

Removing Mappings

To remove a mapping:

1. Start the Mappings tool.
2. Acquire a write lock.
3. Right-click the mapping that you want to remove, and choose **Delete Mapping**.

The Mappings tool prompts you to confirm deletion.
4. Click **Yes**.

The Mappings tool drops supporting tables, removes the mapping from the metadata, and updates the list of mappings.

CHAPTER 18

Hard Delete Detection

This chapter includes the following topics:

- [Hard Delete Detection Overview, 316](#)
- [Hard Delete Detection Types, 317](#)
- [Delete Flag Values of Base Objects, 317](#)
- [Trust and Validation Rules, 317](#)
- [Hard Delete Detection Table, 318](#)
- [Configuring Hard Delete Detection, 319](#)
- [Specifying the Primary Key Columns for Hard Delete Detection , 322](#)
- [Direct Delete, 323](#)
- [Consensus Delete, 326](#)
- [Using Hard Delete Detection within User Exits, 331](#)

Hard Delete Detection Overview

Source systems that populate base objects are continually updated, and records might be hard-deleted from these source systems. Hard-deleted records are records that are removed from the source system. The MDM Hub can detect the records that are hard-deleted in source systems, and it can reflect the change in the associated base objects.

Note: The MDM Hub in the Oracle and Microsoft SQL Server environments can detect records that are hard-deleted in source systems.

The MDM Hub stage job compares all the records in the latest source system file against all the records in the previous landing table. The MDM Hub detects the records that are missing in the latest source system and flags them as hard deletes for a full load. The MDM Hub then reinserts the column values for the deleted records into the landing table along with a delete flag and optionally the end date. Finally the MDM Hub updates the associated base object.

The MDM Hub does not detect the records that are missing in the latest source system file for incremental or transactional load. You must create a hard delete detection table in the repository table to configure hard deletes. When the MDM Hub detects hard deletes, it inserts the number of records flagged as deleted, into the job metric table. You implement hard delete detection as Java user exits that are part of the stage process.

Hard Delete Detection Types

The type of Hard Delete Detection that you choose depends on the source systems and the cross-reference records associated with the base object records.

The MDM Hub flags a base object record for direct delete if a single source system is responsible to determine the delete state of a record. If multiple source systems are responsible to determine the delete state of a record, the MDM Hub performs a consensus delete. The MDM Hub checks if records in all the source systems are in the deleted state and assigns a delete flag to the base object record.

Delete Flag Values of Base Objects

When you define how to consolidate data in base objects, you need to identify the delete flag and determine the delete flag values.

The MDM Hub has default delete flag values. You can configure the MDM Hub to provide an end date value along with a delete flag for base object records.

Base objects have the following default delete flag values:

Active

Indicates that the record is active in all source systems. To change the value, set the value in the DELETE_FLAG_VAL_ACTIVE column to some other value. Default is A.

Inactive or fully deleted

Indicates that the record is hard-deleted in all source systems. To change the value, set the value in the DELETE_FLAG_VAL_INACTIVE column to some other value. Default is I.

Partially deleted

Indicates that the record is deleted in some source systems. To change the value, set the value in the DELETE_FLAG_VAL_PARTIAL column to some other value. Default is P.

Determine the base objects that need delete flags and possibly end dates. Ensure that you include the appropriate columns in the landing tables and staging tables associated with the base objects. Also, verify that you configured the hard delete detection table in the repository to detect hard deletes from the staging table.

Trust and Validation Rules

After the MDM Hub flags the deleted records from a source system, trust and validation rules govern the behavior of the delete flags and end dates. The trust and validation rules help determine the most reliable data.

When the MDM Hub detects that the column values in a base object record are hard-deleted in the source, it uses trust and validation rules. The trust and validation rules determine when these base object columns must be overwritten by values from another source system. The MDM Hub uses a validation rule to downgrade the trust percentage on all the trusted columns for a deleted source record. The values in the base object record that were contributed by the deleted source record remains in the base object record. The MDM Hub replaces the deleted values after another source system provides an update with a higher trust to overwrite the values from the deleted source record.

Hard Delete Detection Table

The hard delete detection table is a repository table that you create to store metadata that the MDM Hub requires to detect hard deletes in source systems. The name of the hard delete detection table is C_REPOS_EXT_HARD_DEL_DETECT.

The following table describes the columns of the hard delete detection table:

Column Name	Datatype and Size in Bytes	Description
ROWID_TABLE	CHAR (14)	Contains values of the ROWID_OBJECT column of the staging table for which hard deleted records need to be detected. The ROWID_TABLE column cannot have a null value.
HDD_ENABLED	INTEGER	Enables or disables the MDM Hub to detect hard deletes for a staging table. The HDD_ENABLED column cannot have a null value. Use one of the following values: <ul style="list-style-type: none">- 0. Enables the MDM Hub to detect hard deletes.- 1. Disables the MDM Hub to detect hard deletes. Default is 0.
HDD_TYPE	VARCHAR2 (14)	Indicates the type of hard delete for the staging table. Use one of the following values for the hard delete type: <ul style="list-style-type: none">- DIRECT. Performs the direct hard delete of the base object record.- CONSENSUS. Performs the consensus based hard delete of the base object record.
DELETE_FLAG_COLUMN_NAME	VARCHAR2 (30)	Name of the staging table column that contains the value of the delete flag for any hard-deleted records that the MDM Hub detects.
DELETE_FLAG_VAL_ACTIVE	VARCHAR2 (14)	Value of the delete flag column for active records.
DELETE_FLAG_VAL_INACTIVE	VARCHAR2 (14)	Value of the delete flag column for inactive records.
DELETE_FLAG_VAL_PARTIAL	VARCHAR2 (14)	Value of the delete flag column for the partial delete in a consensus delete scenario.
HAS_END_DATE	INTEGER	Indicates whether the staging table has an end date column. The HAS_END_DATE column cannot have a null value. Use one of the following values: <ul style="list-style-type: none">- 0. Staging table does not have an end date column.- 1. Staging table has an end date column. Default is 0.
END_DATE_COLUMN_NAME	VARCHAR2 (30)	Name of the end date column. The MDM Hub uses the column when HAS_END_DATE=1.

Column Name	Datatype and Size in Bytes	Description
END_DATE_VAL	DATE	Value to use for the end date. If the implementation requires that you hard-code an end date value, then that value is stored in the END_DATE_VAL column in the MM/DD/YYYY format. Default is SYSDATE.
TRAN_HDD_ENABLED	INTEGER	Indicates whether the process to detect hard deletes is transactional. The TRAN_HDD_ENABLED column cannot have a null value. Use one of the following values: - 0. The process to detect hard delete is not transactional. - 1. The process to detect hard deletes is transactional and does not compare the full load against the previous landing and landing table. Default is 0.

Configuring Hard Delete Detection

You can configure the MDM Hub to detect hard deletes in the source systems.

To configure the MDM Hub to detect hard deletes, you need to create the hard delete detection table and register the job metric type. You must also set up landing and staging tables, and create mappings. After you set up the landing and staging tables, populate the hard delete detection table with the staging table information. Finally, implement a user exit that calls hard delete detection functionality to detect hard deleted records.

Note: You can combine all the SQL statements that are provided to configure the hard delete detection table and run them as a single set of SQL statements.

1. Configure the hard delete detection table.
 - a. To create the hard delete detection table, run the following SQL statement:

On Oracle.

```
CREATE TABLE C_REPOS_EXT_HARD_DEL_DETECT
(
  ROWID_TABLE           CHAR(14 BYTE),
  HDD_ENABLED           INTEGER           DEFAULT 0           NOT NULL,
  HDD_TYPE              VARCHAR2(14 BYTE),
  DELETE_FLAG_COLUMN_NAME VARCHAR2(30 BYTE),
  DELETE_FLAG_VAL_ACTIVE VARCHAR2(14 BYTE),
  DELETE_FLAG_VAL_INACTIVE VARCHAR2(14 BYTE),
  DELETE_FLAG_VAL_PARTIAL VARCHAR2(14 BYTE),
  HAS_END_DATE          INTEGER           DEFAULT 0,
  END_DATE_COLUMN_NAME  VARCHAR2(30 BYTE),
  END_DATE_VAL          DATE              DEFAULT
'SYSDATE',
  TRAN_HDD_ENABLED      INTEGER           DEFAULT 0
  HDD_LANDING_PKEY_COLUMNS VARCHAR2(<TBD> BYTE),
  EMPTY_LANDING_TABLE_CHECK_OFF INTEGER   DEFAULT 0
);
```

On Microsoft SQL Server.

```
CREATE TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT]
(
    [ROWID_TABLE] [nvarchar] NOT NULL,
    [HDD_ENABLED] [bigint] NOT NULL,
    [HDD_TYPE] [nvarchar] (14) NULL,
    [DELETE_FLAG_COLUMN_NAME] [nvarchar] (30) NULL,
    [DELETE_FLAG_VAL_ACTIVE] [nvarchar] (14) NULL,
    [DELETE_FLAG_VAL_INACTIVE] [nvarchar] (14) NULL,
    [DELETE_FLAG_VAL_PARTIAL] [nvarchar] (14) NULL,
    [HAS_END_DATE] [bigint] NULL,
    [END_DATE_COLUMN_NAME] [nvarchar] (30) NULL,
    [END_DATE_VAL] [datetime2] (7) NULL,
    [TRAN_HDD_ENABLED] [bigint] NULL,
    [HDD_LANDING_PKEY_COLUMNS] [nvarchar] (<TBD>) NULL,
    [EMPTY_LANDING_TABLE_CHECK_OFF] [bigint] NULL
)
ON [CMX_DATA]
```

- b. To add constraints to the hard delete detection table, run the following SQL statements:

On Oracle.

```
ALTER TABLE C_REPOS_EXT_HARD_DEL_DETECT ADD (
    CHECK (HDD_TYPE IN ('DIRECT','CONSENSUS')));

ALTER TABLE C_REPOS_EXT_HARD_DEL_DETECT ADD (
    CHECK (HDD_ENABLED IN (0,1)));

ALTER TABLE C_REPOS_EXT_HARD_DEL_DETECT ADD (
    CHECK (HAS_END_DATE IN (0,1)));

ALTER TABLE C_REPOS_EXT_HARD_DEL_DETECT ADD (
    UNIQUE (ROWID_TABLE));
```

On Microsoft SQL Server.

```
ALTER TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT] ADD CONSTRAINT
[CH_C_REPOS_EXT_HARD_DEL_DETECT_HDD_TYPE] CHECK (HDD_TYPE IN
('CONSENSUS','DIRECT'))
GO
ALTER TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT] ADD CONSTRAINT
[CH_C_REPOS_EXT_HARD_DEL_DETECT_HDD_ENABLED] CHECK (HDD_ENABLED IN (0,1))
GO
ALTER TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT] ADD CONSTRAINT
[CH_C_REPOS_EXT_HARD_DEL_DETECT_HAS_END_DATE] CHECK (HAS_END_DATE IN (0,1))
GO
ALTER TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT] ADD CONSTRAINT
[UQ_C_REPOS_EXT_HARD_DEL_DETECT_ROWID_TABLE] UNIQUE (ROWID_TABLE)
GO
```

- c. To create a foreign key constraint on the ROWID_TABLE column of the hard delete detection table to the ROWID_TABLE column of C_REPOS_TABLE, run the following SQL statements:

On Oracle.

```
ALTER TABLE C_REPOS_EXT_HARD_DEL_DETECT ADD (
    CONSTRAINT FK_ROWID_TABLE_FOR_HDD FOREIGN KEY (ROWID_TABLE)
    REFERENCES C_REPOS_TABLE (ROWID_TABLE)
    ON DELETE CASCADE);
```

On Microsoft SQL Server.

```
ALTER TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT] ADD CONSTRAINT
[FK_ROWID_TABLE_FOR_HDD]
FOREIGN KEY (ROWID_TABLE) REFERENCES [dbo].[C_REPOS_TABLE] (ROWID_TABLE) ON
DELETE CASCADE
GO
```

Define the foreign key with the ON DELETE CASCADE clause so that the MDM Hub deletes the appropriate metadata if the parent record is removed from C_REPOS_TABLE.

2. Register job metric type code in the job metric type table.

Run the following SQL statement to register the job metric type code as 100:

On Oracle.

```
INSERT INTO C_REPOS_JOB_METRIC_TYPE
(METRIC_TYPE_CODE, CREATE_DATE, CREATOR, LAST_UPDATE_DATE, UPDATED_BY,
METRIC_TYPE_DESC, SEQ)
VALUES
(100, SYSDATE, 'hdd', SYSDATE, 'hdd', 'Flagged as Deleted', 100);
```

On Microsoft SQL Server.

```
INSERT INTO [dbo].[C_REPOS_JOB_METRIC_TYPE]
([METRIC_TYPE_CODE],[CREATE_DATE],[CREATOR],[LAST_UPDATE_DATE],
[UPDATED_BY],[METRIC_TYPE_DESC],[SEQ])
VALUES
(100,getDate(),'HDD',getDate(),'HDD','Flagged as Deleted',100)
GO
```

3. Configure the landing and staging tables.

- a. Determine the base object records that need delete flags and end dates.
- b. Verify that you included all the landing and staging table columns that provide data to the base objects.
- c. In the **Schema** tool of the Hub Console, verify that the value you specify for DELETE_FLAG_COLUMN_NAME in the hard delete detection table is visible. Also, verify that the default value is A, I, or P.
- d. If the record is active, and if the record has a end date, verify if the end date is null or a system date. If you provide a null value for the end date, enable the **Allow Null Update** check box for the end date column.
- e. In the **Mappings** tool of the Hub Console, map the columns between landing and staging tables.
- f. In the **Schema** tool of the Hub Console, enable delta detection for the staging table for which you need to detect hard deletes.

If you enable the option to detect deltas by using specific columns, include the delete flag column name in the column list to detect deleted records.

4. To define the metadata in the hard delete detection table, run the following SQL statement:

On Oracle.

```
INSERT INTO C_REPOS_EXT_HARD_DEL_DETECT (
ROWID_TABLE, HDD_ENABLED, HDD_TYPE,
DELETE_FLAG_COLUMN_NAME, DELETE_FLAG_VAL_ACTIVE, DELETE_FLAG_VAL_INACTIVE,
DELETE_FLAG_VAL_PARTIAL, HAS_END_DATE, END_DATE_COLUMN_NAME,
END_DATE_VAL, TRAN_HDD_ENABLED, HDD_LANDING_PKEY_COLUMNS)
SELECT
ROWID_TABLE, 1, <Column to enable hard delete detection>,'<hard delete detection
type such as DIRECT or CONSENSUS>',
'<name of the delete flag column>', '<Value of delete flag column for active
records>', '<Value of delete flag column for inactive records>',
'<Value of delete flag column for partially active records>', <Indicator whether
end date is used in hard delete detection or not>,
```

```

        <End date column name if staging table has end date column>, '<End date value as
MM/DD/YYYY>', <Transactional Hard Delete Detection indicator>,
        '<Comma-separated list of column names that contribute to the primary key>',
        FROM C_REPOS_TABLE WHERE table_name = '<Staging table name>'

```

On Microsoft SQL Server.

```

INSERT INTO [dbo].[C_REPOS_EXT_HARD_DEL_DETECT]
    ([ROWID_TABLE]
    ,[HDD_ENABLED]
    ,[HDD_TYPE]
    ,[DELETE_FLAG_COLUMN_NAME]
    ,[DELETE_FLAG_VAL_ACTIVE]
    ,[DELETE_FLAG_VAL_INACTIVE]
    ,[DELETE_FLAG_VAL_PARTIAL]
    ,[HAS_END_DATE]
    ,[END_DATE_COLUMN_NAME]
    ,[END_DATE_VAL]
    ,[TRAN_HDD_ENABLED]
    ,[HDD_LANDING_PKEY_COLUMNS])
SELECT
    [ROWID_TABLE]
    ,1--Column for which hard delete detection must be enabled
    , '<hard delete detection type such as DIRECT or CONSENSUS>'
    , '<name of the delete flag column>'
    , '<Value of delete flag column for active records>'
    , '<Value of delete flag column for inactive records>'
    , '<Value of delete flag column for partially active records>'
    , '<Indicator whether end date is used in hard delete detection or not>'
    , '<End date column name if staging table has end date column>'
    , '<End date value>'
    , '<Transactional Hard Delete Detection indicator>'
    , '<Comma-separated list of column names that contribute to the primary key>'
FROM [dbo].[C_REPOS_TABLE] WHERE table_name = '<Staging table name>'
GO

```

5. Implement user exits for hard delete detection.

- a. Implement Post Landing and Post Stage Java user exit and add logic to the implementation to call the hard delete functionality.

The user exit interfaces to implement are in `mdm-ue.jar`.

- b. Package the implemented Post Landing and Post Stage user exit classes into a JAR file.
- c. Upload the packaged JAR file to the MDM Hub by registering the user exits.

Specifying the Primary Key Columns for Hard Delete Detection

When you use a function to create a primary key from multiple source columns, the hard delete detection process identifies the source columns by looking at the inputs to the function. The hard delete detection process runs on all the source columns that are inputs to the function.

Note: If all the inputs to the function contribute to the primary key, you can skip this procedure.

If the inputs to the function include source columns that are not used in the primary key, create a list of the column names that contribute to the primary key. Add the list of columns to the `C_REPOS_EXT_HARD_DEL_DETECT` repository table. The hard delete detection process runs on the columns in the list.

1. Identify the columns in the landing table that the MDM Hub uses to create the primary key. Create a comma-separated list of the column names.

2. In an SQL tool, open the C_REPOS_EXT_HARD_DEL_DETECT repository table that you created when you configured hard delete detection.
3. If you did not add the HDD_LANDING_PKEY_COLUMNS column when you created the repository table, add the column. Set the column type to VARCHAR, and enter a length that can contain the comma-separated list of column names.
4. Add the comma-separated list of column names to the column.

When the stage job runs, a validation process verifies the list of column names. The validation process ignores spaces, removes duplicate column names, and verifies that the column names match the column names in the landing table.

Direct Delete

The MDM Hub detects hard-deleted records based on a single source system. The MDM Hub flags a base object record as inactive if the source system record is deleted or if one of its cross-reference records is deleted.

Perform a direct delete when the source data comes from a single source system. You can also perform a direct delete when one source system is responsible for deciding the delete status of a record. The MDM Hub decides the delete status based on the most recent status from that source system. The Post Landing user exit can run direct delete.

Hard Delete Detection Configuration for Direct Delete Flagging

You need to configure the hard delete detection table to set up direct delete flagging. When you perform direct hard delete detection, you need the delete flag value, the optional end date of the record, and some base object properties.

You can set the following metadata for direct delete flagging:

Delete flags

Use the following delete flags when you want to detect hard deletes using direct delete:

- DELETE_FLAG_VAL_ACTIVE = A
- DELETE_FLAG_VAL_INACTIVE = I
- HDD_ENABLED = 1
- HDD_TYPE = DIRECT

End date

If you set HAS_END_DATE = 1, specify the value for end date, END_DATE_VAL. The end date of a record can be a date from the past or in the future. The format of the end date is MM/DD/YYYY.

Base object columns

Specify the columns of the base object, including the delete flag column name and the end date column name.

Setting Up Hard Delete Detection for Direct Delete with End Date

After you complete the general configuration to detect hard deletes, set up the procedure for direct delete with an end date for a staging table.

1. Configure the MDM Hub to detect hard deletes.
2. Identify the stage jobs in which the MDM Hub must detect hard deletes.
3. Insert the staging table details and other required metadata into the hard delete detection table.

Direct Delete with End Date Code Example

Your organization needs to detect hard deletes for a staging table called C_CUSTOMER_STG. They want to perform a direct delete of the hard-deleted records in the base object and assign an end date to the deleted records.

You need to insert the metadata that you need to detect hard deletes into the hard delete detection table for the C_CUSTOMER_STG staging table. When you detect hard deletes, the MDM Hub must provide a delete flag in addition to an end date for the deleted record. The name of the delete flag column in the example is DEL_CODE and the name of the end date column in the staging table in the example is END_DATE.

The following sample code inserts metadata into the hard delete detection table to perform a direct delete with an end date for the deleted records:

On Oracle.

```
INSERT into c_repos_ext_hard_del_detect
    (rowid_table, hdd_enabled, hdd_type, delete_flag_column_name,
    delete_flag_val_active, delete_flag_val_inactive, has_end_date, end_date_column_name,
    end_date_val)
SELECT rowid_table,
    1 AS hdd_enabled,
    DIRECT AS hdd_type,
    'DEL_CODE' AS delete_flag_column_name,
    'A' AS delete_flag_val_active,
    'I' AS delete_flag_val_inactive,
    1 AS has_end_date,
    'END_DATE' AS end_date_column_name,
    'SYSDATE' AS end_date_val
FROM c_repos_table
WHERE table_name = 'C_CUSTOMER_STG';
```

On Microsoft SQL Server.

```
INSERT INTO [dbo].[C_REPOS_EXT_HARD_DEL_DETECT]
    ([ROWID_TABLE], [HDD_ENABLED], [HDD_TYPE], [DELETE_FLAG_COLUMN_NAME],
    [DELETE_FLAG_VAL_ACTIVE], [DELETE_FLAG_VAL_INACTIVE], [HAS_END_DATE],
    [END_DATE_COLUMN_NAME], [END_DATE_VAL],)
SELECT [ROWID_TABLE]
    ,1
    ,'DIRECT'
    ,'DEL_CODE'
    ,'A'
    ,'I'
    ,1
    ,'END_DATE'
    ,GETDATE()
FROM [dbo].[C_REPOS_TABLE]
WHERE table_name = 'C_CUSTOMER_STG';
GO
```


Example of Hard Delete Detection for Direct Delete

Your organization has a single source system that supplies data to a base object record. You need to detect hard deletes in the source system to flag the base object record as deleted. The source system deletes a record, reactivates the record, and then deletes the reactivated record. The MDM Hub accordingly flags the source record as deleted and end dated.

Records Before Hard Delete Detection

Before the MDM Hub detects hard deletes in the source system, the delete flag for the base object record and the cross-reference record is active.

The cross-reference record before hard delete is detected in the source.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Robert	James	Jones	A	null

The base object record before hard delete is detected in the source.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	A	null

Records When Hard Delete Is Detected In the Source

When there is a hard delete in a source system, the MDM Hub performs the following tasks:

1. Detects records that are deleted by comparing the landing table to the previous landing table.
2. Reinserts the column values for the deleted records into the landing table along with a delete flag and the end date.
3. Updates the cross-reference record as inactive with an end date.

The end date contains the system date or the date that you provide when you determine the end date value.

The cross-reference record when hard delete is detected in the source.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Robert	James	Jones	I	04/05/13

The base object record when hard delete is detected in the source.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	I	04/05/13

Records After the Source Record Is Reactivated

If the record is reactivated in the source system, the MDM Hub flags the base object record and the cross-reference record as active. The end date is null when the source record is active.

The cross-reference record after the hard deleted record in the source system is reactivated.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Robert	James	Jones	A	null

The base object record after the hard deleted record in the source system is reactivated.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	A	null

Records When Hard Delete Is Detected Again In the Source

When there is a hard delete in a source system, the MDM Hub performs the following tasks:

1. Detects records that are deleted by comparing the landing table to the previous landing table.
2. Reinserts the column values for the deleted records into the landing table along with a delete flag and the end date.
3. Updates the cross-reference record as inactive with an end date.

The end date contains the system date or the date that you provide when you determine the end date value.

The cross-reference record when hard delete is detected in the source.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Robert	James	Jones	I	04/05/13

The base object record when hard delete is detected in the source.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	I	04/05/13

Consensus Delete

The MDM Hub detects hard-deleted records based on the delete status of a record in all the contributing source systems. You use consensus delete in conjunction with direct delete.

The MDM Hub flags a base object record as inactive if the record is deleted from all the contributing source systems. All the cross-reference records that are associated with the base object record must be in the partially inactive state.

Perform a consensus delete when the source data comes from multiple source systems. The MDM Hub flags a record for delete, when there is a consensus among all the source systems to delete a record. The MDM Hub flags records from source systems as partially inactive. The MDM Hub then uses trust and validation rules to determine another source system record to overwrite the base object record. If delete dates are present, the MDM Hub uses delete dates to determine another source system record to overwrite the base object record.

When you need consensus on the delete state of records from all source systems, implement the Post Landing user exit to call direct delete. Also, the Post Staging user exit must call the consensus delete logic for a consensus delete of base object records.

Hard Delete Detection Configuration for Consensus Delete Flagging

You need to configure the hard delete detection table to set up consensus delete flagging. When you perform a consensus hard delete detection, you need the delete flag value, the end date of the record, and some base object properties.

You can set the following metadata for consensus delete flagging:

Delete flags

Use the following delete flags when you want to perform a consensus delete:

- DELETE_FLAG_VAL_ACTIVE = A
- DELETE_FLAG_VAL_INACTIVE = I
- DELETE_FLAG_VAL_PARTIAL = P
- HDD_ENABLED = 1
- HDD_TYPE = CONSENSUS

End date

If you set HAS_END_DATE = 1, specify the value for end date, END_DATE_VAL. The end date of a record can be a date from the past or in the future. The format is MM/DD/YYYY.

Base object columns

Specify the columns of the base object, including the delete flag column name and the end date column name.

Setting Up Hard Delete Detection for Consensus Delete with End Date

After you complete the general configuration to detect hard deletes, set up the procedure for consensus delete with an end date for a staging table.

1. Configure the MDM Hub to detect hard deletes.
2. Identify the stage jobs in which the MDM Hub must detect hard deletes.
3. Insert the staging table details and other required metadata into the hard delete detection table.
4. In the **Schema** tool, enable the **Trust** and **Validate** checkboxes for the column name that contains the value of the delete flag for any hard deleted records.
5. If you use an end date, enable the **Trust** and **Validate** check boxes for the end date column name.
6. In the **Systems and Trust** tool, specify trust settings for the delete flag column name, and the end date column name for each source system.

The value for **Decay** must be the same for all source systems of the base object.

7. In the **Schema** tool, set up validation rule for the delete flag column name for each base object that you configured for Hard Delete Detection. You must set the partial delete flag to a lower trust value than the active, or fully deleted flag.
 - a. Select **Validation Rules Setup** for the base object.

The **Validation Rules** page appears.
 - b. Click the **Add validation rule** button.

The **Add Validation Rule** dialog box appears.
 - c. Enter a rule name and select the **Custom** rule type from the list.

Note: Do not insert a comma into the rule name. This can disrupt the order of the validation rules.
 - d. Select a downgrade percentage and specify a rule in the following format in the **Rule SQL** section:

```
WHERE S.<Delete flag column name> LIKE '<Value of delete flag column for partial delete>'
```

You must not enable **Reserve Minimum Trust** for the column.

Consensus Delete with End Date Code Example

Your organization needs to detect hard deletes for a staging table called C_CUSTOMER_STG. They want to perform a consensus delete of the hard-deleted records in the base object and assign an end date to the deleted records.

You need to insert the metadata that you need to detect hard deletes into the hard delete detection table for the C_CUSTOMER_STG staging table. When you detect hard deletes, the MDM Hub must provide a delete flag in addition to an end date for the deleted record. Also, the process must verify that all the associated cross-reference records are flagged as deleted before the MDM Hub flags the base object records as deleted. The name of the delete flag column is DEL_CODE and the name of the end date column in the staging table is END_DATE.

The following sample code inserts metadata into the hard delete detection table for a consensus delete with an end date for the deleted records:

On Oracle.

```
INSERT into c_repos_ext_hard_del_detect
    (rowid_table, hdd_enabled, hdd_type, delete_flag_column_name,
    delete_flag_val_active, delete_flag_val_inactive, delete_flag_val_partial, has_end_date,
    end_date_column_name, end_date_val)
SELECT rowid_table,
    1 AS hdd_enabled,
    'CONSENSUS' AS hdd_type,
    'DEL_CODE' AS delete_flag_column_name,
    'A' AS delete_flag_val_active,
    'I' AS delete_flag_val_inactive,
    'P' AS delete_flag_val_partial,
    1 AS has_end_date,
    'END_DATE' AS end_date_column_name,
    'SYSDATE' AS end_date_val
FROM c_repos_table
WHERE table_name = 'C_CUSTOMER_STG';
```

On Microsoft SQL Server.

```
INSERT INTO [dbo].[C_REPOS_EXT_HARD_DEL_DETECT]
    ([ROWID_TABLE], [HDD_ENABLED], [HDD_TYPE], [DELETE_FLAG_COLUMN_NAME],
    [DELETE_FLAG_VAL_ACTIVE], [DELETE_FLAG_VAL_INACTIVE], [DELETE_FLAG_VAL_PARTIAL],
    [HAS_END_DATE], [END_DATE_COLUMN_NAME], [END_DATE_VAL],)
SELECT [ROWID_TABLE]
    ,1
    , 'CONSENSUS'
    , 'DEL_CODE'
    , 'A'
    , 'I'
    , 'P'
    , 1
    , 'END_DATE'
    , 'SYSDATE'
FROM [dbo].[C_REPOS_TABLE]
WHERE table_name = 'C_CUSTOMER_STG';
GO
```

Example of Hard Delete Detection for Consensus Delete

Your organization has multiple source systems that provide data to a base object record. The records in each source system are deleted one after the other, and then the record from source system 1 is reactivated and is hard deleted again. The MDM Hub needs to detect hard deletes in the source systems. If records are hard-deleted in all the source systems, the base object record is flagged as deleted.

The MDM Hub will update the column values of the base object record based on the trust downgrades for the deleted source records. The records in all the source systems must be hard-deleted and the MDM Hub must flag all the associated cross-reference records as partially inactive or inactive.

Records Before Hard Delete Detection

Before detecting hard deletes in the source systems, the delete flag for the base object record and the cross-reference records is active. The MDM Hub merges 3 source records into one base object record. The first name comes from source system 2, the middle name comes from source system 3, and the last name from source system 1.

The cross-reference record before hard delete is detected in the source.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	J	Jones	A	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	A	null

The base object record before hard delete is detected in the source.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	A	null

Records When Hard Delete Is Detected In Source 3

When there is a hard delete in a source system 3, the MDM Hub performs the following tasks:

1. Detects deleted source records from the landing table.
2. Reinserts the column values for the deleted records into the landing table along with an inactive delete flag and the end date.
3. Verifies if all the cross-reference records are inactive or partially inactive and updates the associated cross-reference records as partially inactive.
4. Retains the middle name provided by the record from source system 3 in the base object until a source record with a higher trust score provides an update for that column.

The cross-reference records when hard delete is detected in the source system 3.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	J	Jones	A	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	P	null

The base object record when hard delete is detected in source system 3.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	James	Jones	A	null

Records When Hard Delete Is Detected In Source 1

When there is a hard delete in a source system 1, the MDM Hub performs the following tasks:

1. Detects deleted source records from the landing table.
2. Reinserts the column values for the deleted records into the landing table along with a delete flag and the end date.
3. Verifies if all the cross-reference records are inactive or partially inactive and updates the associated cross-reference records as partially inactive.
4. BVT is calculated and the base object reflects the value from source 1 for the middle name, which is John, as it has a higher trust score even though the record is partially deleted.

The cross-reference records when hard delete is detected in the source system 1.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	John	Jones	P	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	P	null

The base object record when hard delete is detected in source system 1.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	John	Jones	A	null

Records When Hard Delete Is Detected In Source 2

When there is a hard delete in a source system 2, the MDM Hub performs the following tasks:

1. Detects deleted source records from the landing table.
2. Reinserts the deleted records into the landing table along with a delete flag and the end date.
3. Verifies if all the cross-reference records are inactive or partially inactive and updates the associated cross-reference records as inactive.
At this point, no other source systems can provide an active status.
4. If no other source systems can provide an active record, the base object record reflects an inactive state with an end date.

The end date contains the system date or the date that you provide when you determine the end date value.

The cross-reference records when hard delete is detected in the source system 2.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	John	Jones	P	null
2	163	Robert	J	Jones	I	04/05/13
3	163	Bob	James	Jones	P	null

The base object record when hard delete is detected in source system 2.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	John	Jones	I	04/05/13

Records After the Source 1 Record Is Reactivated

If a record is reactivated in a source system, the MDM Hub flags the base object record and the cross-reference record as active. The end date is null when the source record is active.

The source record in source system 1 is reactivated, and the MDM Hub flags the associated cross-reference record as active. The end date is null when the record is active.

The cross-reference record after the hard deleted record in the source system 1 is reactivated.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	John	Jones	A	null
2	163	Robert	J	Jones	I	04/05/13
3	163	Bob	James	Jones	P	null

The base object record after the hard deleted record in the source system 1 is reactivated.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	John	Jones	A	null

Records When Hard Delete Is Detected Again In Source 1

When there is a hard delete in a source system 1, the MDM Hub performs the following tasks:

1. Detects deleted source records from the landing table.
2. Reinserts the column values for the deleted records into the landing table along with a delete flag and the end date.
3. Verifies if all the cross-reference records are inactive or partially inactive and updates the associated cross-reference records as inactive.
At this point, no other source systems can provide an active status.
4. The base object record reflects an inactive state with an end date.

The end date contains the system date or the date that you provide when you determine the end date value.

The cross-reference record when hard delete is detected again in source system 1.

Source	Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
1	25	Bob	John	Jones	I	04/05/13
2	163	Robert	J	Jones	I	04/05/13
3	163	Bob	James	Jones	P	null

The base object record when hard delete is detected again in source system 1.

Customer ID	First Name	Middle Name	Last Name	Del_Code	End Date
25	Robert	John	Jones	I	04/05/13

Using Hard Delete Detection within User Exits

You can implement the Post Landing and Post Stage user exits to detect hard deleted records in source systems. To perform a direct hard delete detection you need the Post Landing user exit. To perform a consensus hard delete detection you need both Post Landing and Post Stage user exits.

1. Create an instance of the `HardDeleteDetection` class to use within the implementation of the Post Landing user exit.

The `HardDeleteDetection` class is available in `mdm-ue.jar`, which is in the following directory:

On Windows. <infamdm installation directory>\hub\server\lib

On UNIX. <infamdm installation directory>/hub/server/lib

2. Add the following lines to the Java code for the Post Landing or Post Stage user exits to detect hard deleted records:

- For Post Landing user exit.

```
public void processUserExit(UserExitContext userExitContext, String
    stagingTableName, String landingTableName, String previousLandingTableName)
    throws Exception
{
    HardDeleteDetection hdd = new
    HardDeleteDetection(userExitContext.getBatchJobRowid(), stagingTableName);
    hdd.startHardDeleteDetection(userExitContext.getDBConnection());
}
```

- For Post Stage user exit.

```
public void processUserExit(UserExitContext userExitContext, String
    stagingTableName, String landingTableName, String previousLandingTableName)
```

```

                                throws Exception
        {
            ConsensusFlagUpdate consensusProcess = new
            ConsensusFlagUpdate(userExitContext.getBatchJobRowid(), stagingTableName);
            consensusProcess.startConsensusFlagUpdate(userExitContext.getDBConnection());
        }

```

3. Package the user exits JAR and upload it to the MDM Hub.
4. Run the stage job.

The MDM Hub supplies input parameter values to detect hard deletes when the stage job calls the user exit.

RELATED TOPICS:

- [“Implementing the User Exit JAR File” on page 578](#)
- [“Uploading User Exits to the MDM Hub” on page 578](#)

CHAPTER 19

Data Cleansing Configuration

This chapter includes the following topics:

- [Data Cleansing Configuration Overview, 333](#)
- [Set Up Data Cleansing in the MDM Hub, 333](#)
- [Configure Process Servers for Data Cleansing, 333](#)
- [Configure Cleanse Functions, 338](#)
- [Testing Functions, 347](#)
- [Using Conditions in Cleanse Functions, 347](#)
- [Configuring Cleanse Lists, 348](#)

Data Cleansing Configuration Overview

You can configure data cleansing in the MDM Hub to cleanse and standardize data during the stage process. When you run matching on cleansed data, the MDM Hub generates a greater number of reliable matches.

Before you perform data cleansing in the MDM Hub, install and configure the cleanse engines. Also, you need to configure a Process Server for the MDM Hub implementation. The Process Server is a servlet that cleanses data and processes batch jobs. You can configure cleanse functions that standardize or verify data values in a record.

Set Up Data Cleansing in the MDM Hub

To set up data cleansing in the MDM Hub, perform the following tasks:

- Configure Process Servers for data cleansing.
- Configure cleanse functions.
- Configure cleanse lists.

Configure Process Servers for Data Cleansing

You can configure a Process Server for the MDM Hub implementation.

The Process Server is a servlet that cleanses data and processes batch jobs. You deploy the Process Server in an application server environment.

The Process Server is multithreaded so that each instance can process multiple requests concurrently.

You can run multiple Process Servers for each Operational Reference Store in the MDM Hub. The cleanse process is generally CPU-bound. Use this scalable architecture to scale the MDM Hub implementation as the volume of data increases. Deploy Process Servers on multiple hosts, to distribute the processing load across multiple CPUs and run cleanse operations in parallel. Additionally, some external adapters are inherently single-threaded, so the MDM Hub architecture can simulate multithreaded operations by running one processing thread for each application server instance.

Note: You can install multiple Process Server instances on different application server instances that might be on the same host or different hosts, and register them in the MDM Hub. You must not register a single installation of a Process Server as multiple Process Server instances with different ports in the MDM Hub.

Modes of Cleanse Operations

Cleanse operations can be classified according to the following modes:

- Online and Batch
- Online Only
- Batch Only

The Mode can be used to specify which classes of operations a particular Process Server will run. If you deploy two Process Servers, you could make one batch-only and the other online-only, or you could make them both accept both classes of requests. Unless otherwise specified, a Process Server will default to running both batch and online requests.

Distributed Data Cleansing

You can run multiple Process Servers in parallel to increase the throughput of cleanse operations and fuzzy match processes. When a specified job size is met or exceeded, the MDM Hub distributes the job among the Process Servers.

To set up distributed data cleansing, for each Process Server, set the following properties in the `cmxcleanse.properties` file:

cmx.server.match.distributed_match

Optional. Must be added manually. Specifies whether the Process Server participates in a distributed processing environment for cleanse operations and fuzzy match processes. Default is 1, which is enabled. To disable distributed processing, set to 0.

For more information about configuring multiple Process Servers, see the *Multidomain MDM Installation Guide*.

cmx.server.cleansesize.min_size_for_distribution

Optional. Must be added manually. Specifies the size at which a job can be distributed among the Process Servers. Default is 1000.

In the Hub Console, configure the Process Servers properties as usual.

Cleanse Requests

All requests for cleansing are issued by batch APIs.

The batch APIs package a cleanse request as an XML payload and send it to a Process Server. When the Process Server receives a request, it parses the XML and invokes the appropriate code:

Mode Type	Description
Online operations	The result is packaged as an XML response and sent back through an HTTP POST connection.
Batch jobs	<p>The Process Server pulls the data to be processed into a flat file, processes it, and then uses a bulk loader to write the data back.</p> <ul style="list-style-type: none">- The bulk loader for Oracle is the SQL*Loader utility.- The bulk loader for IBM DB2 is the IBM DB2 Load utility.- The bulk loader for Microsoft SQL Server uses the Java database connectivity driver.

The Process Server is multi-threaded so that each instance can process multiple requests concurrently. The default timeout for batch requests to a Process Server is one year, and the default timeout for online requests is one minute.

When running a stage or match job, if more than one Process Server is registered, and if the total number of records to be staged or matched is more than 500, and then the job will get distributed in parallel among the available Process Servers.

Starting the Process Server Tool

To view the Process Server information such as name, port, server type, and whether the Process Server is online or off-line, start the Process Server tool.

To start the Process Server tool, in the Hub Console, expand the Utilities workbench, and then click **Process Server**. The Process Server tool displays a list of configured Process Servers.

Process Server Properties

Identifies the server and port for the selected Process Server. The other properties control the behavior of the Process Server, such as which types of processes the server handles and the number of threads to use.

You can enable or disable the following types of processes on the Process Server:

- Cleanse operations
- Fuzzy match and query search processes
- Load and merge batch processes
- Elasticsearch processes

Best practice: Keep it simple. Enable all the operations and processes on all the Process Servers. If performance becomes an issue, you can decide how to distribute the workload among the servers. For example, if you have a heavy cleanse workload, you might configure one Process Server for online cleanse operations and a different one for batch cleanse operations. If batch jobs for cleanse and match run at the same time, consider running the jobs on different Process Servers.

If you change the properties, restart the Process Server. The exception is thread counts, which do not require a restart. The following table describes the properties that you can specify:

Property	Description
Server	IP address or the fully qualified host name of the application server on which you deployed this Process Server. Note: Do not use <code>localhost</code> as the host name.
Port	HTTP or HTTPS port of the application server on which you deployed this Process Server.
Cleanse Operations	Indicates whether this Process Server handles cleanse operations. Default is true. Use the Cleanse Mode option to specify whether this server handles batch jobs, real-time cleanse requests, or both.
Batch Threads for Cleanse and Fuzzy Match	Number of threads to use for cleanse, tokenize, and match batch jobs. Default is 1. Best practices: <ul style="list-style-type: none"> - When you are using fuzzy match or address verification, enable one thread per CPU. For example, on a quad-core machine, set this value to 4. - When you are not using fuzzy match or address verification, and most of the data cleansing involves strings, enable four threads per CPU. For example, on a quad-core machine, set this value to 16. - When you are running the Process Server on a separate machine from the Operational Reference Store, add one additional thread to accommodate latency that might occur with a remote database. - If you run a memory-intensive process, restrict the total memory allocated to all these threads in the JVM to 1 GB.
Cleanse Mode	Specifies the types of cleanse operations that this Process Server handles. <ul style="list-style-type: none"> - Batch Only. Handles only cleanse requests that come from batch jobs. The cleanse functions are called by the mappings in the stage process. - Online Only. Handles only real-time cleanse requests. Requests come from cleanse functions that are called implicitly by the CleansePut SIF API or that are configured explicitly in the IDD Subject Area Cleanse Function. - Both. Handles both batch and online cleanse requests.
Fuzzy Match and Query Search Processing	Indicates whether this Process Server handles fuzzy match. Default is true. Use the Match Mode option to specify whether this server handles batch jobs, real-time match requests, or both.
Match Mode	Specifies the types of fuzzy match processing that this Process Server handles. <ul style="list-style-type: none"> - Batch Only. Participates in matching only when the request comes from batch jobs. - Online Only. Participates in matching only for real-time requests. Requests for real-time matching come from searchMatch SIF API calls and IDD extended search. - Both. Participates in matching for batch and online requests.
Offline	Status of this Process Server. Selecting or clearing the check box does not change the state of the Process Server.
Load and Merge Batch Processing	Indicates whether this Process Server handles data load batch jobs and merge batch jobs. Default is false.
Threads for Load and Merge	Maximum number of threads to use for data load batch jobs and automerge batch jobs. Best practice: Enable four threads per CPU on the machine. For example, on a quad-core machine, set a value of 16. Default is 20.

Property	Description
CPU Rating	Specifies the relative CPU performance of the machines in the cleanse server pool. Best practice: If the performance of the machines that run the Process Servers are similar, keep the default value of 1. If this Process Server machine has double the performance of the other machines, set this value to 2.
Elasticsearch Processing	Indicates whether this Process Server handles the Initially Index Smart Search Data batch job. This batch job creates indexes for all the values of the searchable fields in a business entity.
Secured Connection (HTTPS)	Indicates whether this Process Server uses the HTTPS protocol. If selected, ensure that the Port option is set to an HTTPS port number.

Adding a Process Server

You can use the Process Server tool to add a Process Server. You can configure multiple Process Servers for each Operational Reference Store.

1. Start the Process Server tool.
2. Acquire a write lock.
3. In the right pane of the Process Server tool, click the **Add Process Server** button to add a Process Server.
The **Add/Edit Process Server** dialog box appears.
4. Set the properties for the Process Server.
5. Click **OK**.
6. Click the **Save**.

Enabling Secured Communications for Process Servers

Each Process Server requires a signed certificate. Use the Hub Console to enable the HTTPS protocol and specify a secure port for each Process Server.

1. Create signed certificates for the Process Servers in the certificate store.
2. Ensure that the application server can access the certificate store.
3. Log in to the Hub Console.
4. Select an Operational Reference Store database.
5. Acquire a write lock.
6. In the **Utilities** workbench, select **Process Server**.
7. Select a Process Server and click the **Edit Process Server** icon.
The Add/Edit Process Server dialog box opens.
8. Verify that the **Port** is a secure port.
9. Select the **Secured Connection (HTTPS)** check box.
10. Click **OK**.
11. Verify other Process Servers that appear in the list.

Editing Process Server Properties

You can edit the properties of the Process Server.

1. Start the Process Server tool.
2. Acquire a write lock.
3. Select the Process Server that you want to edit.
4. Click the **Edit Process Server** button.

The **Add/Edit Process Server** dialog box for the selected Process Server appears.

5. Change the properties you want for the Process Server.
6. Click **OK**.
7. Click **Save**.

Deleting a Process Server

You can use the Process Server tool to delete a Process Server.

1. Start the Process Server tool.
2. Acquire a write lock.
3. Select the Process Server that you want to delete.
4. Click the **Delete Process Server** button.

The Process Server tool prompts you to confirm deletion.

5. Click **OK**.

Testing the Process Server Configuration

Whenever you add or change your Process Server information, it is recommended that you check the configuration to ensure that the connection works properly.

1. Start the Process Server tool.
2. Select the Process Server that you want to test.
3. Click the **Test Process Server** button to test the configuration.

If the test succeeds, the Process Server tool displays a window showing the connection information and a success message.

If there is a problem, Informatica MDM Hub displays a window with information about the connection problem.

4. Click **OK**.

Configure Cleanse Functions

In the MDM Hub, you can build and run cleanse functions that cleanse data.

A *cleanse function* is a function that the MDM Hub applies to a data value in a record to standardize or verify it. For example, if the data has a column for salutation, you could use a cleanse function to standardize all

instances of “Doctor” to “Dr.” You can apply cleanse functions successively, or simply assign the output value to a column in the staging table.

Types of Cleanse Functions

The MDM Hub can have one of the following types of cleanse functions:

- Cleanse function defined by the MDM Hub
- Cleanse function defined by a cleanse engine
- Custom cleanse function that you define

The pre-defined functions provide access to specialized cleansing functionality, such as name and address standardization, address decomposition, gender determination, and so on. See the console for more information on the Cleanse Function tool.

Libraries

Functions are organized into *libraries*—Java libraries and user libraries, which are folders used to organize the functions that you can use in the Cleanse Functions tool in the Model workbench.

Cleanse Functions are Secure Resources

You can configure cleanse functions as secure resources and make them SECURE or PRIVATE.

Available Functions Subject to Cleanse Engine

The functions you see in the Hub Console depend on the cleanse engine that you use. The MDM Hub shows the cleanse functions that the cleanse engine makes available. Regardless of which cleanse engine you use, the overall process of data cleansing in the MDM Hub is the same.

Starting the Cleanse Functions Tool

The Cleanse Functions tool provides the interface for defining how you cleanse your data.

To start the Cleanse Functions tool:

- In the Hub Console, expand the Model workbench and then click **Cleanse Functions**.

The Hub Console displays the Cleanse Functions tool.

The Cleanse Functions tool is divided into the following panes:

Pane	Description
Navigation pane	Shows the cleanse functions in a tree view. Clicking on any node in the tree shows you the appropriate properties page in the right-hand pane.
Properties pane	Shows the properties for the selected function. For any of the custom cleanse functions, you can edit properties in the right-hand pane.

The functions you see in the left pane depend on the cleanse engine you are using. Your functions may differ from the ones shown in the previous figure.

Cleanse Function Types

Cleanse functions are grouped in the tree according to their type.

Cleanse function types are high-level categories that are used to group similar cleanse functions for easier management and access.

Cleanse Function Properties

If you expand the list of cleanse function types in the navigation pane, you can select a cleanse function to display its particular properties.

In addition to specific cleanse functions, the Misc Functions include Read Database and Reject functions that provide efficiencies in data management.

Field	Description
Read Database	Allows a map to look up records directly from a database table. Note: This function is designed to be used when there are many references to the same limited number of data items.
Reject	Allows the creator of a map to identify incorrect data and reject the record, noting the reason.

Overview of Configuring Cleanse Functions

To define cleanse functions, you complete the following tasks:

Note: Multidomain MDM 10.5 includes an upgrade of Apache Axis2 to version 1.8.0. Update the custom code of cleanse functions so that all Apache Axis2 libraries point to version 1.8.0. Refresh existing SOAP cleanse functions by completing steps 1 to 3 below.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Click **Refresh** to refresh your cleanse library.
4. Create your own cleanse library, which is simply a folder where you keep your custom cleanse functions.
5. Define regular expression functions in the new library, if applicable.
6. Define graph functions in the new library, if applicable.
7. Add cleanse functions to your graph function.
8. Test your functions.

Configuring User Cleanse Libraries

You can add a user cleanse library or a Java cleanse library when you want to create a customized cleanse function from existing internal or external cleanse functions. Configure a user cleanse library to add graph functions, regular expression functions, and cleanse lists.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Click **Refresh** to refresh the cleanse library.
4. In the Cleanse Functions navigator, right-click **Cleanse Functions**, and then choose **Add User Library**. The **Add User Library** dialog box appears.
5. Specify the following properties:

Name

Unique, descriptive name for the library.

Description

Optional description of the library.

6. Click **OK**.

The library that you add appears in the Cleanse Functions navigator.

Configuring Java Cleanse Libraries

You can add a user cleanse library or a Java cleanse library when you want to create a customized cleanse function from existing internal or external cleanse functions. Configure a Java cleanse library to add custom cleanse function that you generate outside the MDM Hub.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Click **Refresh** to refresh your cleanse library.
4. In the Cleanse Functions navigator, right-click **Cleanse Functions**, and then choose **Add Java Library**.
The **Add Java Library** dialog box appears.
5. Click the **Browse** button to navigate to the location of the Java library JAR file.
6. Specify the following properties:

Name

Unique, descriptive name for the library.

Description

Optional description of the library.

7. If applicable, configure parameters for the library.
 - a. Click the **Parameters** button.
The **Parameters** dialog box appears.
 - b. Add as many parameters as needed for the library.
 - To add a parameter, click the **Add Parameter** button.
The **Add Value** dialog box appears.
Type a name and value, and then click **OK**.
 - To import parameters, click the **Import Parameters** button.
The **Open** dialog box appears.
Select the properties file that contains the parameters you want.

You can add as many parameters as needed for this library.

- To add a parameter, click the **Add Parameter** button. The Cleanse Functions tool displays the Add Value dialog.
Type a name and value, and then click **OK**.
- To import parameters, click the **Import Parameters** button. The Cleanse Functions tool displays the Open dialog, prompting you to select a properties file containing the parameter(s) you want.

The name, value pairs that are imported from the file are available to the user-defined Java function at run time as elements of its Java properties. You can provide custom values such as user ID or target URL in a generic function.

8. Click **OK**.

The library that you add appears in the Cleanse Functions navigator.

Adding Regular Expression Functions

You can add regular expression functions for cleanse operations. Regular expressions are computational expressions that you can use to match and manipulate text data according to common syntactic conventions and symbolic patterns. For more information about syntax and patterns for regular expressions, see the Javadoc for `java.util.regex.Pattern`.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Right-click a user cleanse library name, and then click **Add Regular Expression Function**.

The **Add Regular Expression** dialog box appears.

4. Specify the following properties:

Field	Description
Name	Unique, descriptive name for the regular expression function.
Description	Optional description of the regular expression function.

5. Click **OK**.
The regular expression function that you add appears under the user library that you selected in the Cleanse Functions navigator.
6. Click the **Details** tab of the regular expression function that you added.
7. To enter an input or output expression, click **Edit**, and then click **Accept Edit** to apply the change.
8. Click **Save**.

Configure Graph Functions

A graph function is a cleanse function that you can visualize and configure graphically by using the Cleanse Functions tool in the Hub Console. You can add any pre-defined functions to a graph function.

Graph functions have one or more inputs and outputs. For each graph function, configure all the required inputs and outputs.

Perform the following tasks to configure a graph function:

1. Add graph functions.
2. Add functions to a graph function.

Input and Output Properties

Graph functions have one or more input and output.

Configure the following properties for the input and output of all graph functions:

Name

Unique, descriptive name for the input or output.

Description

Optional description of the input or output.

Data Type

Data type of the input or output parameter.

Use one of the following values:

- Boolean. Accepts boolean values.
- Date. Accepts date values.
- Float. Accepts float values.
- Integer. Accepts integer values.
- String. Accepts any data.

Adding Graph Functions

You can add graph functions.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Right-click a **User Library** name and select **Add Graph Function**.
The Cleanse Functions tool displays the Add Graph Function dialog box.
4. Specify the following properties:

Field	Description
Name	Unique, descriptive name for the graph function.
Description	Optional description of the graph function.

5. Click **OK**.
The Cleanse Functions tool displays the graph function. The graph function is empty. To configure the graph function, see [“Adding Functions to a Graph Function” on page 343](#).

Adding Functions to a Graph Function

You can add as many functions as you want to a graph function.

You can also reuse graph functions. Define the graph functions and then use the graph functions like any other function in the cleanse libraries. For example, you can add a graph function inside another graph function.

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Click the graph function, and then click the **Details** tab.
The graphical representation of the function is called the workspace. You might need to resize the window to see both the input and output in the workspace.
By default, graph functions have one input and one output that are of type String. The gray circles represent the inputs and outputs. Your function might require more inputs or outputs and different data types.
4. Right-click the workspace and select **Add Function**.
The Cleanse Functions tool displays the **Choose Function to Add** dialog box.

5. Expand the folder containing the function you want to add, select the function, and then click **OK**.

Note: The available functions depend on your cleanse engine configuration.

The Cleanse Functions tool displays the function in the workspace. To move a function, drag it to the new location.

6. Right-click the function and select **Expanded Mode**.

The expanded mode shows the labels for all the inputs and outputs. The color of the circle indicates the data type of the input or output. The data types must match.

7. Mouse-over the input connector, which is the little circle on the right side of the input box. The circle turns red when ready for use.

8. Click the node and draw a line to one of the function input nodes.

9. Draw a line from one of the function output nodes to the output box node.

10. Click **Save**.

To learn about testing your the graph function, see ["Testing Functions" on page 347](#).

Function Modes







Function modes determine how the function is displayed on the workspace. Each function has the following modes, which are accessible by right-clicking the function:

Option	Description
Compact	Displays the function as a small box, with just the function name.
Standard	Displays the function as a larger box, with the name and the nodes for the input and output, but the nodes are not labeled. This is the default mode.
Expanded	Displays the function as a large box, with the name, the input and output nodes, and the names of those nodes.
Logging Enabled	<p>Used for debugging. Choosing this option generates a log file for this function when you run a Stage job. The log file records the input and output for every time the function is called during the stage job. There is a new log file created for each stage job.</p> <p>The log file is named <code><jobID><graph function name>.log</code> and is stored in:</p> <pre>\<infamdm_install_dir>\hub\cleanse\tmp\<Operational Reference Store></pre> <p>Note: Do not use this option in production, as it will consume disk space and require performance overhead associated with the disk I/O. To disable this logging, right-click on the function and uncheck Enable Logging.</p>
Delete Object	Deletes the function from the graph function.

You can cycle through the display modes (compact, standard, and expanded) by double-clicking on the function.

Workspace Buttons

The toolbar on the right side of the workspace provides the following buttons.

Button	Description
	Save changes.
	Edit the function inputs.
	Edit the function outputs.
	Add a function.
	Add a constant.
	Add a conditional execution component.
	Edit the selected component.
	Delete the selected component.
	Expand the graph. This makes more room for the workspace on the screen by hiding the left pane.

Using Constants

Constants are useful in cases where you know that you have standardized input.

For example, if you have a data set that you know consists entirely of doctors, then you can use a constant to put Dr. in the title. When you use constants in your graph function, they are differentiated visually from other functions by their grey background color.

Configuring Inputs

To add more inputs:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Select the cleanse function that you want to configure.
4. Click the **Details** tab.
5. Right-click on the input and choose **Edit inputs**.

The Cleanse Functions tool displays the Inputs dialog.

Note: Once you create an input, you cannot later edit the input to change its type. If you must change the type of an input, create a new one of the correct type and delete the old one.

6. Click the **Add** button to add another input.

The Cleanse Functions tool displays the Add Parameter dialog.

7. Specify the following properties:

Field	Description
Name	Unique, descriptive name for this parameter.
Data Type	Data type of this parameter.
Description	Optional description of this parameter.

8. Click **OK**.

Add as many inputs as you need for your functions.

Configuring Outputs

To add more outputs:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Select the cleanse function that you want to configure.
4. Click the **Details** tab.
5. Right-click on the output and choose **Edit outputs**.

The Cleanse Functions tool displays the Outputs dialog.

Note: Once you create an output, you cannot later edit the output to change its type. If you must change the type of an output, create a new one of the correct type and delete the old one.

6. Click the **Add** button to add another output.

The Cleanse Functions tool displays the Add Parameter dialog.

Field	Description
Name	Unique, descriptive name for this parameter.
Data Type	Data type of this parameter.
Description	Optional description of this parameter.

7. Click **OK**.

Add as many outputs as you need for your functions.

Testing Functions

Once you have added and configured a graph or regular expression function, it is recommended that you test it to make sure it is behaving as expected.

This test process mimics a single record coming into the function.

To test a function:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Select the cleanse function that you want to test.
4. Click the **Test** tab.

The Cleanse Functions tool displays the test screen.

5. For each input, specify the value that you want to test by clicking the cell in the Value column and typing a value that complies with the data type of the input.
 - For Boolean inputs, the Cleanse Functions tool displays a true/false drop-down list.
 - For Calendar inputs, the Cleanse Functions tool displays a Calendar button that you can click to select a date from the Date dialog.
6. Click **Test**.

If the test completed successfully, the output is displayed in the output section.

Using Conditions in Cleanse Functions

This section describes how to add conditions to graph functions.

About Conditional Execution Components

Conditional execution components are similar to the construct of a case (or switch) statement in a programming language.

The cleanse function evaluates the condition and, based on this evaluation, applies the appropriate graph function associated with the case that matches the condition. If no case matches the condition, then the default case is used—the case flagged with an asterisk (*).

When to Use Conditional Execution Components

Conditional execution components are useful when, for example, you have segmented data.

Suppose a table has several distinct groups of data (such as customers and prospects). You could create a column that indicated the group of which the record is a member. Each group is called a segment. In this example, customers might have C in this column, while prospects would have P. You could use a conditional execution component to cleanse the data differently for each segment. If the conditional value does not meet any of the conditions you specify, then the default case will be executed.

Adding Conditional Execution Components

To add a conditional execution component:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Select the cleanse function that you want to configure.
4. Right-click on the workspace and choose **Add Condition**.

The Cleanse Functions tool displays the Edit Condition dialog.

5. Click the **Add** button to add a value.

The Cleanse Functions tool displays the Add Value dialog.

6. Enter a value for the condition. Using the customer and prospect example, you would enter C or P. Click **OK**.

The Cleanse Functions tool displays the new condition in the list of conditions on the left, as well as in the input box.

Add as many conditions as you require. You do need to specify a default condition—the default case is automatically created when you create a new conditional execution component. However, you can specify the default case with the asterisk (*). The default case will be executed for all cases that are not covered by the cases you specify.

7. Add as many functions as you require to process all of the conditions.
8. For each condition—including the default condition—draw a link between the input node to the input of the function. In addition, draw links between the outputs of the functions and the output of your cleanse function.

Note: You can specify nested processing logic in graph functions. For example, you can nest conditional components within other conditional components (such as nested case statements). In fact, you can define an entire complex process containing many conditional tests, each one of which contains any level of complexity as well.

Configuring Cleanse Lists

This section describes how to configure cleanse lists in your Informatica MDM Hub implementation.

About Cleanse Lists

A cleanse list is a logical grouping of string functions that are executed at run time in a predefined order.

Use cleanse lists to standardize known string values and to remove extraneous characters (such as punctuation) from input strings.

Adding Cleanse Lists

To add a new cleanse list:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Click **Refresh** to refresh your cleanse library. Used with external cleanse engines.

Important: You must choose **Refresh** after acquiring a write lock and before processing any records. Otherwise, your external cleanse engine will throw an error.

4. Right-click your cleanse library in the list under Cleanse Functions and choose **Add Cleanse List**. The Cleanse Functions tool displays the Add Cleanse List dialog.
5. Specify the following properties:

Field	Description
Name	Unique, descriptive name for this cleanse list.
Description	Optional description of this cleanse list.

6. Click **OK**.

The Cleanse Functions tool displays the details pane for the new (empty) cleanse list on the right side of the screen.

Cleanse List Properties

This section describes the input and output properties of cleanse lists.

Input Properties

The following table describes input properties for cleanse lists.

Property	Description
Input string	String value from the source system. Used as the target of the search.
searchType	<p>Specifies the type of match (comparing cleanse list items with the input string) to be executed against the input string. One of the following values:</p> <p>ENTIRE</p> <p>Compares cleanse list items with the entire string. A match succeeds only when entire input string is the same as a cleanse list item. Default setting if this parameter is not specified.</p> <p>WORD</p> <p>Compares the cleanse list items with each word substring in the input string. A match succeeds only if a cleanse list item is a substring flanked by the following word boundaries in the input string: beginning of string, end of string, or a space character.</p> <p>ANYWHERE</p> <p>Compares the cleanse list items with any part of the input string. A match succeeds if a cleanse list item is a substring of the input string, regardless of where the substring appears in the input string.</p> <p>Note: String comparisons are case sensitive.</p>

Property	Description
replaceAllOccurrences	<p>Specifies the degree to which matched substrings in the input string are replaced with the matching cleanse list item. One of the following values.</p> <p>TRUE</p> <p>Replaces all occurrences of the matching substring in the input string with the matching cleanse list item.</p> <p>FALSE</p> <p>Replaces only the first occurrence of the matching substring in the input string with the matching cleanse list item. Default setting if replaceAllOccurrences is not specified.</p> <p>Note: If the Strip parameter is TRUE, then occurrences of the matching substring are removed rather than replaced.</p>
stopOnHit	<p>Specifies whether to continue processing the rest of the cleanse list after one matching item has been found in the input string. One of the following values.</p> <p>TRUE</p> <p>Stops processing the cleanse list as soon as the first cleanse list item is found in the input string (as long as the searchType condition is met). Default setting if stopOnHit is not specified.</p> <p>FALSE</p> <p>Continues to search the input string for the rest of the items in the cleanse list (in order to find any further matching substrings).</p>
Strip	<p>Specifies whether the matched text in the input string will be stripped from—or replaced in—the input string. One of the following values.</p> <p>TRUE</p> <p>Removes (rather than replaces) the matched text in the input string.</p> <p>FALSE</p> <p>Replaces the match text in the input string. Default setting if Strip is not specified.</p> <p>Note: The replaceAllOccurrences parameter determines whether replacement or removal affects all matches in the input string or just the first match.</p>
defaultValue	<p>Value to use for the output if none of the cleanse list items was found in the input string. If this property is not specified and no match was found, then the original input string is used as the output.</p>

Output Properties

The following table describes output properties for cleanse lists.

Property	Description
output	Output value of the cleanse list function.
matched	Last matched value of the cleanse list.
matchFlag	Indicates whether a match was found in the list (true) or not (false).

Editing Cleanse List Properties

New cleanse lists are empty lists. You need to edit the cleanse list to add match and output strings.

To edit your cleanse list to add match and output strings:

1. Start the Cleanse Functions tool.
2. Acquire a write lock.
3. Select the cleanse list that you want to configure.

The Cleanse Functions tool displays information about the cleanse list in the right pane.
4. Change the display name and description in the right pane, if you want, by clicking the **Edit** button next to a value that you want to change.
5. Click the **Details** tab.

The Cleanse Functions tool displays the details for the cleanse list.
6. Click the **Add** button in the right hand pane.

The Cleanse Functions tool displays the Output String dialog.
7. Specify a search string, an output string, a match type, and click **OK**.

The search string is the input that you want to cleanse, resulting in the output string.

Important: Informatica MDM Hub will search through the strings in the order in which they are entered. The order in which you specify the items can therefore affect the results obtained. To learn more about the types of matches available, see [“Types of String Matches” on page 351](#).

Note: As soon as you add strings to a cleanse list, the cleanse list is saved.

The strings that you specified are shown in the Cleanse List Details section.
8. You can add and remove strings. You can also move string forward or backward in the cleanse list, which affects their order in run-time execution sequence and, therefore, the results obtained.
9. You can also specify the “Default value” for every input string that does not match any of the search strings.

If you do not specify a default value, every input string that does not match a search string is passed to the output string with no changes.





Types of String Matches

For the output string, you can specify one of the following match types:

Match Type	Description
Exact Match	Text string (for example, “IBM”). Note that string matches are not case sensitive. For example, the string test will also match TEST or Test.
Regular Expression	<p>Pattern using the Java syntax for regular expressions (for example, “I.M.*” would match “IBM”, “IB Corp” and “IXM Inc.”). To parse a name field that consists of first, middle, and last names, you could use the regular expression (\S+\$), which gives you the last name no matter what name you give it.</p> <p>The regular expression that is typed in as a parameter is used against the string and the matched output is sent to the outlet. You can also specify the group number to match an inner group of the regular expression. Refer to the Javadoc for java.util.regex.Pattern for the documentation on the regular expression construction and how groups work.</p>
SQL Match	Pattern using the SQL syntax for the LIKE operator in SQL (for example, “I_M%” would match “IBM”, “IBM Corp” and “IXM Inc.”). If you use metacharacters such as the pipeline symbol (), the metacharacter must be delimited by an escape sequence such as backslash (\).


Importing Match Strings

To import match strings (such as a file or a database table):

1. Click the  button in the right hand pane.
The Import Match Strings wizard opens.
2. Specify the connection properties for the source of the data and click **Next**.
The Cleanse Functions tool displays a list of tables available for import.
3. Select the table you want to import and click **Next**.
The Cleanse Functions tool displays a list of columns available for import.
4. Click the columns you want to import and click **Next**.
The Cleanse Functions tool displays a list of match strings available for import.
You can import the records of the sample data either as phrases (one entry for each record) or as words (one entry for each word in each record). Choose whether to import the match strings as words or phrases and then click **Finish**.
The Cleanse List Details box is now populated with data from the specified source.
Note: The imported match strings are not part of the match list. To add them to the match list, you need to move them to the Search Strings on the right hand side.
 - To add match strings to the match list with the match string value in both the Search String and Output String, select the strings in the Match Strings list, and click the  button.
 - If you add match strings to the match list with an Output String value that you want to define, click the record you added and specify a new Search and Output String.
 - To add all Match Strings to the match list, click the  button.
 - To clear all Match Strings from the match list, click the  button.
 - Repeat these steps until you have constructed a complete match list.

Importing Match Output Strings

To import match output strings, such as a file or a database table:

1. Click the  button in the right hand pane.
The Import Match Output Strings wizard opens.
2. Specify the connection properties for the source of the data.
3. Click **Next**.
The Cleanse Functions tool displays a list of tables available for import.
4. Select the table that you want to import.
5. Click **Next**.
The Cleanse Functions tool displays a list of columns available for import.
6. Select the columns that you want to import.
7. Click **Next**.
The Cleanse Functions tool displays a list of match strings available for import.

8. Click **Finish**.

The Cleanse List Details box is now populated with data from the specified source.

CHAPTER 20

Configuring the Load Process

This chapter includes the following topics:

- [Overview, 354](#)
- [Before You Begin, 354](#)
- [Configuration Tasks for Loading Data, 355](#)
- [Configuring Staging Tables, 355](#)
- [Configuring Initial Data Load, 362](#)
- [Configuring Trust for Source Systems, 362](#)
- [Configuring Validation Rules, 368](#)

Overview

This chapter explains how to configure the load process in your Informatica MDM Hub implementation.

Before You Begin

Before you begin to configure the load process, you must have completed the following tasks:

- Installed Informatica MDM Hub and created the Hub Store
- Built the schema
- Defined source systems
- Created landing tables
- Created staging tables
- Learned about the load process

Configuration Tasks for Loading Data

To set up the process of loading data in your Informatica MDM Hub implementation, you must complete the following tasks in the Hub Console:

- Configure staging tables.
- Configure initial data load.
- Configure trust for source systems.
- Configure validation rules.

For additional configuration settings that can affect the load process, see:

- [“Loading by Row ID” on page 308](#)
- [“Distinct Systems” on page 484](#)
- [“Generating Match Tokens \(Optional\)” on page 271](#)
- [“Load Process” on page 265](#)

Configuring Staging Tables

MDM Hub uses staging tables for temporary, intermediate storage in the flow of data from landing tables into base objects.

Staging tables contain data from one source system for one base object table in the Hub Store. The batch stage job populates the staging tables from the landing tables. The batch load job then populates the base object from the staging table.

The structure of a staging table is based on the structure of the target object that will contain the consolidated data. You use the Schema Manager in the Model workbench to configure staging tables. You must have at least one source system defined before you can define a staging table.

Staging Table Columns

Staging tables contain system columns and user-defined columns.

System Staging Table Columns

The Schema Manager creates and maintains the system staging table columns.

The following table describes the system columns of staging tables:

Physical Name	MDM Hub Data type (Size)	Description
PKEY_SRC_OBJECT	VARCHAR (255)	Primary key from the source system. The PKEY_SRC_OBJECT value must be unique. If the source record does not have a single unique column, concatenate the values from multiple columns to uniquely identify the record.
ROWID_OBJECT	CHAR (14)	Primary key from MDM Hub. MDM Hub assigns a unique ROWID_OBJECT value during the stage process.

Physical Name	MDM Hub Data type (Size)	Description
DELETED_IND	INT	Reserved for future use.
DELETED_DATE	TIMESTAMP	Reserved for future use.
DELETED_BY	VARCHAR (50)	Reserved for future use.
LAST_UPDATE_DATE	TIMESTAMP	Date when the source system last updated the record. For base objects, LAST_UPDATE_DATE populates LAST_UPDATE_DATE and SRC_LUD in the cross-reference table, and based on trust settings might also populate LAST_UPDATE_DATE in the base object table.
UPDATED_BY	VARCHAR (50)	User or process responsible for the most recent update.
CREATE_DATE	TIMESTAMP	Date on which the record was created.
PERIOD_START_DATE	TIMESTAMP	Start date of an effective period of a record. PERIOD_START_DATE value is required for timeline-enabled base objects.
PERIOD_END_DATE	TIMESTAMP	End date of an effective period of a record. PERIOD_END_DATE value is required for timeline-enabled base objects.
CREATOR	VARCHAR (50)	User or process responsible for creating the record.
SRC_ROWID	VARCHAR (30)	Database internal Row ID column that traces records back to the landing table from the staging table.
HUB_STATE_IND	INT	For state-enabled base objects. Integer value indicating the state of this record. The following values are valid: <ul style="list-style-type: none"> - 0=Pending - 1=Active - -1=Deleted The default value is 1.
VERSION_SEQ	INT	For timeline-enabled base objects, value shows the order that records with the same primary source key were loaded into the stage table. For base objects that are not timeline-enabled, this value must be 1. The VERSION_SEQ column is mapped to from a user-defined landing table column. The landing table column contains the version sequence for records with the same primary key but different period start and period end dates.

User-defined Staging Table Columns

To avoid decreased performance caused by unnecessary trust calculations, ensure that the columns you add to staging tables come from specific source systems.

When you add columns from source systems, you can restrict the trust columns to columns that have data from multiple staging tables. When staging tables have columns from specific source systems, you do not treat every column as if it comes from every source system. You do not need to add trust for each column. Also, you do not need validation rules to downgrade the trust on null values for all of the sources that do not provide values for the columns.

Preserve Source System Keys

During stage jobs, the MDM Hub can generate keys or use keys from the primary key column of a source system. The MDM Hub can preserve the source system keys for one source system for the initial data load.

You can specify whether the MDM Hub must use key values from the source system or use the key values that the MDM Hub generates. Enable the **Preserving Source System Keys** option to use key values from the source system. Disable to use the key values that the MDM Hub generates. Default is disabled.

You can enable the option to preserve source system keys before you run the stage job. If you enable the option, the MDM Hub does not generate an internal key but instead uses the primary keys from the source system during the stage job. When you run the load job, the MDM Hub takes the value from the PKEY_SOURCE_OBJECT column of the staging table and inserts it into the ROWID_OBJECT column in the target base object.

Note: You cannot change the setting to preserve source system keys after the base object is loaded.

Specify the Highest Reserved Key

The highest reserved key is the highest source system key. To ensure that the keys that the MDM Hub generates does not conflict with the source system keys, reserve the highest source system key. When you preserve the source system keys, you can specify the highest source system key to use for the records.

To insert a gap between the source keys and the keys that the MDM Hub generates, specify the number by which the key value must increase after the first load.

Set the highest reserved key value to the upper boundary of the source system keys. To allow a margin, set the number slightly higher by adding a buffer to the expected range of source system keys.

The load operation can add a record to the base object that does not contain the source system key. If the base object record does not contain the source system key, the MDM Hub assigns a key that is higher than the highest reserved key value.

If you specify the highest reserved key, the MDM Hub processes the ROWID_OBJECT value for the records that it loads in to the base object in the following way:

1. During the initial data load, the MDM Hub takes the value in the PKEY_SOURCE_OBJECT column of the staging table. The MDM Hub inserts the value into the ROWID_OBJECT column of the base object instead of generating internal keys.
2. After the initial data load, the MDM Hub resets the starting position of the key to the value of the highest reserved key that is incremented by 1.
3. The MDM Hub uses the highest reserved key value for subsequent loads from the staging table. The MDM Hub uses the key that it generates for loads from other staging tables. The key that the MDM Hub generates is in continuation with the highest reserved key value.

Note: You can preserve the source system keys for one staging table associated with a base object even if the keys are from the same source system. Set the reserved key range at the initial load.

Highest Reserved Key Example

You preserve the source system keys for a staging table and set the highest source system key to use for the records to 100. You load more records from the staging table for which you preserved the source system

keys. You then load records from a staging table other than the one for which you preserved the source system keys.

You can observe the changes to the key values in the ROWID_OBJECT column during each of the following load operations:

1. Initial data load with primary source system keys 20, 21, and 22 from the staging table for which you preserve the source system keys.
The MDM Hub loads three records to the associated base object with the ROWID_OBJECT values 20, 21, and 22.
2. After the initial data load, load five records with primary source system keys from the staging table for which you preserve the source system keys. The primary source system keys for the records are 23, 24, and 25, AB, and YZ.
The MDM Hub loads five records to the associated base object with the ROWID_OBJECT values 23, 24, 25, 101 and 102.
3. Load three records with any primary key from a staging table for which you do not preserve the source system keys.
The MDM Hub loads three records to the associated base object with the ROWID_OBJECT values from 103 to 105.

Enable Cell Update

To increase performance and to ensure that the MDM Hub updates cells with changed values, enable the option to update cells. By default, the load process replaces the cell value in the target base object for each inbound record that has a higher trust level. The load process replaces cell values even if the value replaced is identical.

Even if the value does not change, the MDM Hub updates the last update date for the cell to the date associated with the incoming record, and assigns the same trust level to the cell as the new value. To change the behavior enable cell update when you configure a staging table. If you enable cell update, during load jobs, the MDM Hub compares the cell value with the current contents of the cross-reference table before it updates the target record in the base object. If the cross-reference record for the system has an identical value in the cell, the MDM Hub does not update the cell in the Hub Store.

Enable cell update to increase performance during load jobs if the MDM Hub does not require updates to the last update date and trust value in the target base object record.

Properties for Columns in Staging Tables

The properties for columns in the staging table provide information about foreign key lookups. The properties also allow you to configure the batch load and Put API behavior when the staging table columns contain null values.

Note: In the MDM Hub, an empty string is the equivalent of a null value regardless of the database type that contributes the empty string.

Staging table columns have the following properties:

Column

Name of the column as defined in the associated base object.

Lookup System

Name of the lookup system if the lookup table is a cross-reference table.

Lookup Table

For foreign key columns in the staging table, the name of the table containing the lookup column.

Lookup Column

For foreign key columns in the staging table, the name of the lookup column in the lookup table.

Allow Null Foreign Key

When enabled, a load batch job or Put API can load data when the lookup column contains a null value. Do not enable **Allow Null Foreign Keys** if the foreign key relationship is required.

When disabled, a load batch job or Put API cannot load data when the lookup column contains a null value. The Hub Console rejects the record and does not load the record.

Allow Null Update

Controls what happens when a source contributes a null value for the column, while other sources have non-null values for the same column. All processes that perform a Best Version of the Truth (BVT) calculation use this property: load, load update, put, cleanse put, merge, unmerge, recalculate BVT, and revalidate.

- True. When enabled, if the null value is the most trustworthy value for this column, a process can write the null value to the base object record.
- False. Default. When disabled, a process cannot write a null value to the base object record if another source contributes a non-null value for the column.

When a process runs, for each source that contributes a null value, the process checks the staging table for the source. If the setting of the **Allow Null Update** property is false on a column, the process downgrades the trust on that column to less than zero. The process then calculates the BVT using the adjusted trust scores. This method ensures that the process selects the most trustworthy non-null value to write to the base object record.

In the following special cases, a process ignores the **Allow Null Update** property in the staging tables and uses the **Apply Null Values** property for the column in the base object table instead:

- The process finds multiple staging tables associated with a source, where the staging tables have a mixture of settings for the **Allow Null Update** property for the column.
- The process finds a staging table for the source, but the column is not configured in the staging table. For example, a service call always updates a column value, so the column is not configured in a staging table.
- The process cannot find a staging table for a source. For example, there is no value in the STG_ROWID_TABLE column in the cross-reference record and alternative methods for determining the staging table are not definitive.

A process ignores both the **Allow Null Update** property and the **Apply Null Values** property in the following scenarios:

- When all sources contribute values that are not null, the value from the most trusted source survives.
- When all sources contribute null values, the null value survives.
- When a base object has a single source system, the value—null or not null—from that source is written to the base object.

Allow Null Update Example

You have a customer base object with three contributing sources. The load update process loads data from Source A where the middle name has been deleted, that is, the value is null. Source B and Source C have middle names for the customer.

The following table shows the three sources, the settings for the Middle Name column in the staging tables, the trust adjustment, and the result of the BVT calculation:

Source	Staging Table Middle Name Trust	Staging Table Middle Name Allow Null Update	XREF Record Middle Name Value	Trust After Adjustment	Base Object Record BVT Value
Source A	90	false	null	< 0	-
Source B	60	false	Edward	60	-
Source C	80	true	Edwin	80	Edwin

The load update process starts the BVT calculation for the Middle Name column. Initially, Source A has the highest trust at 90, but the value is null. The process finds the staging table for Source A and checks the **Allow Null Update** property on the Middle Name column. The property is false. The process downgrades the trust on the Middle Name column in Source A to less than zero. After the trust adjustment, Source C has the highest trust at 80. The process selects the middle name from Source C and writes Edwin to the base object record.

Changing Properties in Staging Tables

You can change the staging table properties when you need.

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Base Objects** node, and then expand the node for the base object associated with this staging table.
If the staging table is associated with the base object, then expand the **Staging Tables** node to display it.
4. Select the staging table that you want to configure.
The Schema Manager displays the properties for the selected table.
5. Specify the staging table properties.
For each property that you want to edit, click the **Edit** button next to it, and specify the new value.
Note: You can change the source system if the staging table and its related support tables such as the raw and primary landing tables are empty.
Do not change the source system if the staging table or its related tables contain data.
6. From the list of the columns in the base object, change the columns that the source system will provide.
 - Click the **Select All** button to select all of the columns without needing to click each column individually.
 - Click the **Clear All** button to uncheck all selected columns.**Note:** The Rowid Object and the Last Update Date are automatically selected. You cannot uncheck these columns or change their properties.
7. If you want, change column properties.
8. If you want, change lookups for foreign key columns. Select the column and click the **Edit** button to configure the lookup column.
9. If you want to change cell updating, click the **Cell update** check box.

10. Change the column configuration for your staging table, if you want.
11. If you want, configure an Audit Trail and Delta Detection for this staging table.
12. Click the **Save** button to save your changes.

Lookups for Foreign Key Columns

You can use lookups to retrieve data from a parent table during load jobs. If a foreign key column in the staging table is related to the primary key in a parent table, configure a lookup to retrieve data from the parent table.

The target column in the lookup table must be a unique column such as the primary key.

After you define a lookup, if a load job is run on the base object, the MDM Hub looks up the Consumer code value of the source system in the primary key from the source system column of the Consumer code cross-reference table. After the lookup, the MDM Hub returns the customer type ROWID_OBJECT value that corresponds to the source consumer type.

Example

The MDM Hub implementation in your organization has two base objects, a Consumer parent base object and an Address child base object. The base objects have the following relationships:

```
Consumer.Rowid_object = Address.Consumer_Fkey
```

In this case, the Consumer_Fkey is in the Address Staging table and it will look up data on some column.

Note: The Address.Consumer_Fkey must be the same as Consumer.Rowid_object.

In this example, you can configure the following types of lookups:

- Lookup to the ROWID_OBJECT, which is the primary key of the Consumer base object lookup table.
- Lookup to the PKEY_SRC_OBJECT column, which is the primary key of the cross-reference table for the Consumer base object.
In this case, you must also define the lookup system. If you configure a lookup to the PKEY_SRC_OBJECT column of a cross-reference table, point to parent tables associated with a source system that differ from the source system associated with this staging table.
- Lookup to any other unique column, if available, in the base object or its cross-reference table.

Configuring Lookups

You can configure a lookup through foreign key relationships.

1. Start the Schema Manager.
2. Acquire a write lock.
3. In the schema tree, expand the **Base Objects** node, and then expand the node for the base object associated with this staging table.
4. Select the staging table that you want to configure.
5. Select the row of the foreign key column that you want to configure.
The **Edit Lookup** button is enabled for foreign key columns.
6. Click the **Edit Lookup** button.

The Schema Manager displays the Define Lookup dialog box.

The Define Lookup dialog box contains the parent base object and its cross-reference table, along with any unique columns.

7. Select the target column for the lookup.
 - To define the lookup to a base object, expand the base object and select Rowid_Object, which is the primary key for this base object.
 - To define the lookup to a cross-reference table, select PKey Src Object (the primary key for the source system in this cross-reference table).
 - To define the lookup to any other unique column, select the column.

Note: When you delete a relationship, it clears the lookup.
8. If the lookup column is PKey Src Object in the relationship table, select the lookup system from the Lookup System list.
9. Click **OK**.
10. If you want, configure the **Allow Null Update** check box to specify what happens if a load job specifies a null value for a cell that already contains a non-null value.
11. For each column, configure the **Allow Null Foreign Key** option to specify what happens if the foreign key column contains a null value (no lookup value is available).
12. Click the **Save** button to save your changes.

Configuring Initial Data Load

During initial data load into a base object, you can parallelize the batch job to improve performance.

Configure the following parameter in C_REPOS_TABLE to improve the performance of initial data load:

PARALLEL_BATCH_JOB_THRESHOLD

Parallelizes the batch job. Set the value to a number that is less than the number of available CPU cores that the MDM Hub can use. For example, if a machine has 4 CPUs, but only 2 are available for the MDM Hub to use, and each CPU has 8 processor cores, then the maximum value that you can set is 15. Default is 1,000.

Configuring Trust for Source Systems

This section describes how to configure trust in your Informatica MDM Hub implementation.

About Trust

Several source systems may contain attributes that correspond to the same column in a base object table.

For example, several systems may store a customer's address. However, one system might be a more reliable source for that data than others. If these systems disagree, then Informatica MDM Hub must decide which value is the best one to use.

To help with comparing the relative reliability of column data from different source systems, Informatica MDM Hub allows you to configure trust for a column. Trust is a designation the confidence in the relative accuracy of a particular piece of data. For each column from each source, you can define a trust level represented by a number between 0 and 100, with zero being the least trustworthy and 100 being the most

trustworthy. By itself, this number has no meaning. It becomes meaningful only when compared with another trust number to determine which is higher.

Trust takes into account the age of data, how much its reliability has decayed over time, and the validity of the data. Trust is used to determine survivorship (when two records are consolidated), and whether updates from a source system are sufficiently reliable to update the master record.

Trust Levels

A trust level is a number between 0 and 100. By itself, this number has no meaning. It has meaning only when compared with another trust number.

Data Reliability Decays Over Time

The reliability of data from a given source system can decay (diminish) over time. In order to reflect this fact in trust calculations, Informatica MDM Hub allows you to configure decay characteristics for trust-enabled columns. The *decay period* is the amount of time that it takes for the trust level to decay from the maximum trust level to the minimum trust level.

Trust Calculations

The load process calculates trust for trust-enabled columns in the base object. For records with trust-enabled columns, the load process assigns a trust score to cell data. This trust score is initially based on the configured trust settings for that column. The trust score may be subsequently downgraded when the load process applies validation rules—if configured for a trust-enabled column—after the trust calculations.

All trust calculations are based on the system date with one exception; trust calculations for historic queries on timeline-enabled base objects are based on the historical date. Trust calculations for trust-enabled base objects are not based on effective dates.

Trust Calculations for Load Update Operations

During the load process, if a record in the staging table will be used for a load update operation, and if that record contains a changed cell value in a trust-enabled column, the load process calculates trust scores for:

- the cell data in the source record in the staging table (which contains the updated information)
- the cell data in the target record in the base object (which contains the existing information)

If the cell data in the source record has a higher trust score than the cell data in the target record, then Informatica MDM Hub updates the cell in the base object record with the cell data in the staging table record.

Trust Calculations When Consolidating Two Base Object Records

When two records in a base object are consolidated, Informatica MDM Hub calculates the trust score for each trusted column in the two records being merged. Cells with the highest trust scores survive in the final consolidated record. If the trust scores are the same, then Informatica MDM Hub compares records.

Control Tables for Trust-Enabled Columns

For each trust-enabled column in a base object record, Informatica MDM Hub maintains a record in a corresponding control table that contains the last update date and an identifier of the source system. Based on these settings, Informatica MDM Hub can always calculate the current trust for the column value.

If history is enabled for a base object, Informatica MDM Hub also maintains a separate history table for the control table, in addition to history tables for the base object and its cross-reference table.

Cell Values in Base Object Records and Cross-Reference Records

The cross-reference table for a base object contains the most recent value from each source system. By default (without trust settings), the base object contains the most recent value no matter which source system it comes from.

For trust-enabled columns, the cell value in a base object record might not have the same value as its corresponding record in the cross-reference table. Validation rules, which are run during the load process after trust calculations, can downgrade trust for a cell so that a source that had previously provided the cell value might not update the cell.

Overriding Trust Scores

Data stewards can manually override a calculated trust setting if they have direct knowledge that a particular value is correct. Data stewards can also enter a value directly into a record in a base object. For more information, see the *Multidomain MDM Data Steward Guide*.

Trust for State-Enabled Base Objects

For state-enabled base objects, trust is calculated for records with the ACTIVE, PENDING, and DELETE states. In case of records with the DELETE state, the trust is downgraded further, to ensure that records with the DELETE state do not win over records with ACTIVE and PENDING states.

Batch Job Constraints on Number of Trust-Enabled Columns

Synchronize batch jobs can fail for base objects with a large number of trust-enabled columns.

Similarly, Automerge jobs can fail if there is a large number of trust-enabled or validation-enabled columns. The exact number of columns that cause the job to fail is variable and is based on the length of the column names and the number of trust-enabled columns (or, for Automerge jobs, validation-enabled columns as well). Long column names are at—or close to—the maximum allowable length of 26 characters. To avoid this problem, keep the number of trust-enabled columns below 100 and/or the length of the column names short. A work around is to enable all trust/validation columns before saving the base object to avoid running the synchronization job.

Trust Properties

This section describes the trust properties that you can configure for trust-enabled columns.

Trust properties are configured separately for each source system that could provide records for trust-enabled columns in a base object.

Maximum Trust

The maximum trust (starting trust) is the trust level that a data value will have if it has just been changed. For example, if source system X changes a phone number field from 555-1234 to 555-4321, the new value will be given system X's maximum trust level for the phone number field. By setting the maximum trust level relatively high, you can ensure that changes in the source systems will usually be applied to the base object.

Minimum Trust

The minimum trust is the trust level that a data value will have when it is old (after the decay period has elapsed). This value must be less than or equal to the maximum trust.

Note: If the maximum and minimum trust are equal, then the decay curve is a flat line and the decay period and decay type have no effect.

Units

Specifies the units used in calculating the decay period—day, week, month, quarter, or year.




Decay

Specifies the number (of days, weeks, months, quarters, or years) used in calculating the decay period.

Note: For the best graph view, limit the decay period you specify to between 1 and 100.

Graph Type

Decay follows a pattern in which the trust level decreases during the decay period. The graph types show these decay patterns have any of the following settings.

Icon	Graph Type	Description
	Linear	Simplest decay. Decay follows a straight line from the maximum trust to the minimum trust.
	Rapid Initial Slow Later (RISL)	Most of the decrease occurs toward the beginning of the decay period. Decay follows a concave curve. If a source system has this graph type, then a new value from the system will probably be trusted, but this value will soon become much more likely to be overridden.
	Slow Initial Rapid Later (SIRL)	Most of the decrease occurs toward the end of the decay period. Decay follows a convex curve. If a source system has this graph type, it will be relatively unlikely for any other system to override the value that it sets until the value is near the end of its decay period.

Test Offset Date

By default, the start date for trust decay shown in the Trust Decay Graph is the current system date. To see the impact of trust decay based on a different start date for a given source system, specify a different test offset date.

Considerations for Setting Trust Values

Choosing the correct trust values can be a complex process.

It is not enough to consider one system in isolation. You must ensure that the combinations of trust settings for all of the source systems that contribute to a particular column produce the behavior that you want. Trust levels for a source system are not absolute—they are meaningful only in relation to the trust levels of other source systems that contribute data for the trust-enabled column.

When determining trust, consider the following questions.

- Does the source system validate this data value? How reliably does it do this?
- How important is this data value to the users of the source system, as compared with other data values? Users are likely to put the most effort into validating the data that is central to their work.
- How frequently is the source system updated?
- How frequently is a particular attribute likely to be updated?

Column Trust

You enable and configure trust on base object columns. You cannot enable trust for any other tables in an Operational Reference Store.

Informatica MDM Hub disables column trust by default. When trust is disabled, Informatica MDM Hub uses the data from the most recently executed load process regardless of which source system the data comes from. If base object column data for a comes from one system, keep trust disabled for that column.

Trust should be enabled for columns in which data can come from multiple source systems. If you enable trust for a column, you then assign trust levels to specify the relative reliability of the particular source systems that could provide data for the column.

As you add trust columns and validation rules, batch load and batch merge performance decreases. Also, as you add trust columns and validation rules, the length of control table update statements increases. If you enable more than 40 trust columns, Informatica MDM Hub updates the control table in chunks of up to 40 columns at a time. Do not enable trust for a column when you consider the most recently loaded data for the column to be the best version of the truth.

Ensure you do not enable trust and validation on an excessive number of columns. On Microsoft SQL Server, the page size limit is 8 KB. To support Unicode characters, Informatica MDM Hub uses the NCHAR and NVARCHAR data types. Due to double-byte support, the maximum record size is 4000 characters. Batch processes can fail unexpectedly if record size exceeds 4000 characters.

Enabling Trust for a Column

To enable trust for a base object column, edit the base object column properties.

1. In the Hub Console under the **Model** workbench, select **Schema**.
2. Acquire a write lock.
3. In the left pane of the Schema Manager, expand the base object with the column to which you want to apply trust. Select **Columns**.
4. Enable the **Trust** checkbox for the column. Click the **Save** button.

Before You Configure Trust for Trust-Enabled Columns

Before you configure trust for trust-enabled columns, you must have:

- enabled trust for base object columns
- configured staging tables in the Schema Manager, including associated source systems and staging table columns that correspond to base object columns

Specifying Trust for the Administration Source System

At a minimum, you must specify trust settings for trust-enabled columns in the administration source system (called *Admin* by default). This source system represents manual updates that you make within Informatica MDM Hub. This source system can contribute data to any trust-enabled column. Set the trust settings for this source system to high values (relative to other source systems) to ensure that manual updates override any existing values from other source systems.

Assigning Trust Levels to Trust-Enabled Columns in a Base Object

To assign trust levels to trust-enabled columns in a base object:

1. Start the Systems and Trust tool.

2. Acquire a write lock.
3. In the navigation pane, expand the Trust node.
The Systems and Trust tool displays all base objects with trust-enabled columns.
4. Select a base object.
The Systems and Trust tool displays a read-only view of the trust-enabled columns in the selected base object, indicating with a check mark whether a given source system supplies data for that column.
Note: The association between trust-enabled columns and source systems is specified in the staging tables for the base object.
5. Expand a base object to see its trust-enabled columns.
6. Select the trust-enabled column that you want to configure.
For the selected trust-enabled column, the Systems and Trust tool displays the list of source systems associated with the column, along with editable trust settings to be configured per source system, and a trust decay graph.
7. Specify the trust properties for each column.
8. Optionally, you can change the offset date.
9. Click the **Save** button to save your changes.
The Systems and Trust tool refreshes the Trust Decay Graph based on the trust settings you specified for each source system for this trust-enabled column.
The X-axis displays the trust score and the Y-axis displays the time.

Changing the Offset Date for a Trust-Enabled Column

By default, the Trust Decay Graph shows the trust decay across all source systems from the current system date. You can specify a different date (such as a future date) to test your current trust settings and see how trust would decay from that date. Note that offset dates are not saved.

To change the offset date for a trust-enabled column:

1. In the Systems and Trust tool, select a trust-enabled column.
2. Click the **Calendar** button next to the source system for which you want to specify a different offset date.
The Systems and Trust tool prompts you to specify a date.
3. Select a different date.
4. Choose **OK**.
The Systems and Trust tool updates the Trust Decay Graph based on your current trust settings and the Offset Date you specified.

To remove the Offset Date:

- Click the **Delete** button next to the source system for which you want to remove the Offset Date.
The Systems and Trust tool updates the Trust Decay Graph based on your current trust settings and the current system date.

Running Synchronize Batch Jobs After Changes to Trust Settings

After records have been loaded into a base object, if you enable trust for any column, or if you change trust settings for any trust-enabled column(s) in that base object, then you must run the Synchronize batch job before running the consolidation process. If this batch job is not run, then errors will occur during the consolidation process.

Configuring Validation Rules

This section describes how to configure validation rules in your Informatica MDM Hub implementation.

About Validation Rules

A validation rule downgrades trust for a cell value when the cell value matches a given condition.

Note: The MDM Hub does not support custom validation rules for IBM DB2 or Microsoft SQL Server.

Each validation rule specifies:

- a condition that determines whether the cell value is valid
- an action to take if the condition is met (downgrade trust by a certain percentage)

For example, the following validation rule:

```
Downgrade trust on First_Name by 50% if Length < 3'
```

consists of:

Condition

Length < 3

Action

Downgrade trust on First_Name by 50%

If the Reserve Minimum Trust flag is set for the column, then the trust cannot be downgraded below the column's minimum trust. You use the Schema Manager to configure validation rules for a base object.

Validation rules are executed during the load process, after trust has been calculated for trust-enabled columns in the base object. If validation rules have been defined, then the load process applies them to determine the final trust scores, and then uses the final trust values to determine whether to update records in the base object with cell data from the updated records.

Validation Checks

A validation check can be done on any column in a base object. The downgrade resulting from the validation check can be applied to the same column, as well as to any other columns that can be validated. Invalid data in one column can therefore result in trust downgrades on many columns.

For example, supposed you used an address verification flag in which the flag is OK if the address is complete and BAD if the address is not complete. You could configure a validation rule that downgrades the trust on all address fields if the verification flag is not OK. Note that, in this case, the verification flag should also be downgraded.

Required Columns

Validation rules are applied regardless of the source of the incoming data. However, validation rules are applied only if the staging table or if the input—a Services Integration Framework (SIF) request—contains all of the required columns. If any required columns are missing, validation rules are not applied.

Recalculating Trust Scores After Changing Validation Rules

If a base object contains existing data and you change validation rules, you must run the Revalidate job to recalculate trust scores for new and existing data.

Validation Rules and State-Enabled Base Objects

For state-enabled base objects, validation rules are applied to records with the ACTIVE, PENDING, and DELETED states. While calculating BVT, the trust is downgraded further in case of records with the DELETED state, to ensure that records with the DELETE state do not win over records with ACTIVE and PENDING states.

Automerger Job Constraints on Number of Validation Columns

Automerger jobs can fail if there is a large number of validation-enabled columns.

The exact number of columns that cause the job to fail is variable and is based on the length of the column names and the number of validation-enabled columns. Long column names are at—or close to—the maximum allowable length of 26 characters. To avoid this problem, keep the number of validation-enabled columns below 60 and/or the length of the column names short. A work around is to enable all trust/validation columns before saving the base object to avoid running the synchronization job.

Enabling Validation Rules for a Column

A validation rule is enabled and configured on a per-column basis for base objects in the Schema Manager.

Validation rules do not apply to columns in any other tables in an Operational Reference Store.

Validation rules are disabled by default. Validation rules should be enabled, however, for any trust-enabled columns that will use validation rules for trust downgrades.

When you update a base object, and if you do not provide values for trust-enabled columns, the validation rules that depend on the trust-enabled columns are not applied to the associated cross-reference records.

How the Downgrade Percentage is Applied

Validation rules downgrade trust scores according to the following algorithm:

$$\text{Final trust} = \text{Trust} - (\text{Trust} * \text{Validation_Downgrade} / 100)$$

For example, with a validation downgrade percentage of 50%, and a trust level calculated at 60:

$$\text{Final Trust Score} = 60 - (60 * 50 / 100)$$

The final trust score is:

$$\text{Final Trust Score} = 60 - 30 = 30$$

Sequence of Validation Rules

The MDM Hub runs the validation rules in the sequence in which you specify them.

If you configure multiple validation rules for a column, the MDM Hub runs the validation rules in the sequence that you configure. Also, the MDM Hub considers the downgrade percentage of each validation rule.

Downgrade percentages are not cumulative. The validation rule with the highest downgrade percentage overwrites other changes.

If multiple validation rules have the same highest downgrade percentage, the MDM Hub checks the reserve minimum trust setting. If the reserve minimum trust setting is enabled for all the validation rules, the MDM Hub applies the first validation rule with the highest downgrade percentage.

Navigating to the Validation Rules Node

To configure validation rules, you navigate to the Validation Rules node for a base object in the Schema Manager:

1. Start the Schema Manager.
2. Acquire a write lock.
3. Expand the tree for the base object that you want to configure, and then click its **Validation Rules Setup** node.

The Schema Manager displays the Validation Rules editor.

The Validation Rules editor is divided into the following sections.

Pane	Description
Number of Rules	Number of configured validation rules for the selected base object.
Validation Rules	List of configured validation rules for the selected base object.
Properties Pane	Properties for the selected validation rule.

Validation Rule Properties

Validation rules have the following properties.

Rule Name

A unique, descriptive name for this validation rule.

Note: Do not insert a comma into the rule name. This can disrupt the order of the validation rules.

Rule Type

The type of validation rule. The rule type can have one of the following values:

Rule Type	Description
Existence Check	Trust is downgraded if the cell has a null value, that is, the cell value does not exist.
Domain Check	Trust is downgraded if the cell value does not fall within a list or range of allowed values.
Referential Integrity	Trust is downgraded if the value in a cell does not exist in the set of values in a column on a different table. This rule is for use in cases where an explicit foreign key has not been defined, and an incorrect cell value can be allowed if there is no correct cell value that has higher trust.
Pattern Validation	Trust is downgraded if the value in a cell conforms (LIKE) or does not conform (NOT LIKE) to the specified pattern.
Custom	Use custom rules to enter complex validation rules. Use custom rules only when you require SQL functions such as LENGTH and ABS, or if you require a complex join to a static table. Custom SQL code must conform with the SQL syntax for the database platform. The SQL that you enter is not validated at design time. SQL syntax errors that are not valid cause problems when the load process executes.

Rule Columns

For each column, you specify the downgrade percentage and whether to reserve minimum trust.

Downgrade Percentage

Percentage by which the trust level of the specified column will be decreased if this validation rule condition is met. The larger the percentage, the greater the downgrade. For example, 0% has no effect on the trust, while 100% downgrades the trust completely (unless the reserve minimum trust is specified, in which case 100% downgrades the trust so that it equals minimum trust).

If trust is downgraded by 100% and you have not enabled minimum reserve trust for the column, then the value of that column will not be populated into the base object.

Reserve Minimum Trust

Specifies what will happen if the downgrade causes the trust level to fall below the column's minimum trust level. You can retain the minimum trust (so that the trust level will be reduced to the minimum trust but no lower). If this box is cleared (unchecked), then the trust level will be reduced by the specified percentage even if this means going below the minimum trust.

Rule SQL

Specifies the condition for the validation rule as an SQL WHERE clause. The load process executes the validation rule. If a record meets the condition specified in the Rule SQL field, the trust value is downgraded by the downgrade percentage that is configured for the validation rule.

The Validation Rules editor prompts you to configure the SQL WHERE clause based on the selected rule type for the validation rule. During the load process, this query is used to check the validity of the data in the staging table.

The following table lists the rule types and contains examples of SQL WHERE clauses for each rule type:

Rule Type	WHERE clause	Examples	Result
Existence Check	WHERE S.ColumnName IS NULL	WHERE S.MIDDLE_NAME IS NULL	Affected columns will be downgraded for records with middle names that are null. Records that do not meet the condition are not affected.
Domain Check	WHERE S.ColumnName IN ('?', '?', '?')	WHERE S.Gender NOT IN ('M', 'F', 'U')	Affected columns will be downgraded if the Gender is any value other than M, F, or U.
Referential Integrity	WHERE NOT EXISTS (SELECT <blank>'a' FROM ? WHERE ?? = S.<Column_Name> WHERE NOT EXISTS (SELECT <blank>'a' FROM <Ref_Table> WHERE <Ref_Table>.<Ref_Column> = S.<Column_Name>	WHERE NOT EXISTS (SELECT DISTINCT 'a' FROM ACCOUNT_TYPE WHERE ACCOUNT_TYPE.Account_Type = S.Account_Type	Affected columns will be downgraded for records with Account Type values that are not on the Account Type table.

Rule Type	WHERE clause	Examples	Result
Pattern Validation	WHERE S.ColumnName LIKE 'Pattern'	WHERE S.eMail_Address NOT LIKE '%@%'	Downgrade will be applied if the e-mail address does not contain an @ character.
Custom	WHERE	WHERE LENGTH(S.ZIP_CODE) > 4	Downgrade will be applied if the length of the zip code column is less than 4.

Table Aliases and Wildcards

You can use the wildcard character (*) to reference tables using an alias.

- s.* aliases the staging table
- l.* aliases a temporary table and provides ROWID_OBJECT, PKEY_SRC_OBJECT, and ROWID_SYSTEM information for the records being updated. The l alias can only be used for ROWID_OBJECT, PKEY_SRC_OBJECT, and ROWID_SYSTEM columns in a validation rule.

Custom Rule Types and SQL WHERE Syntax

For custom validation rules, write SQL statements that are well formed and well tuned. The SQL statement is executed as part of the WHERE condition for incoming data.

Note: Custom validation rules decrease the performance of the load process. Use sparingly.

Use the SQL syntax that is required by your database platform. If you need more information about SQL WHERE clause syntax and wild card patterns, refer to the product documentation for the database platform that is used in your Informatica MDM Hub implementation.

Be sure to use parentheses to specify precedence. Incorrect or omitted parentheses can have unexpected results and long-running queries. For example, the following statement is ambiguous and leaves it up to the database server to determine precedence:

```
WHERE conditionA AND conditionB OR conditionC
```

The following statements use parentheses to explicitly specify precedence:

```
WHERE (conditionA AND conditionB) OR conditionC
WHERE conditionA AND (conditionB OR conditionC)
```

These two statements yield very different results when evaluating records.

Example of a Custom Validation Rule with a Join Between Tables

You can create a condition for a custom validation rule that joins data from two tables. One table contains the parent record in the incoming data. The other table is a lookup table with a static list of values.

Note: Use values in the parent record for conditions. If you create a condition that requires a value from a child record, the rule does not return results. You cannot join to child tables from the parent table in a custom validation rule.

You reference the incoming data as a table with the alias S. You reference the joined table with the alias l.

The following code shows a validation rule SQL statement for the join expressed by Put (BO - C_AAA_BO):

```
WHERE NOT EXISTS (SELECT 1 FROM C_B_LU_ADDR_TP T WHERE T.ADDR_TP = S.COL1)
```


The following code shows the generated SQL statement:

```
SELECT S.PKEY_SRC_OBJECT ,
      (SELECT 'a' FROM dual WHERE NOT EXISTS (SELECT 1 FROM C_B_LU_ADDR_TP T WHERE
T.ADDR_TP = S.COL1 )
      AND ROWNUM <= 1 ) RULE1
FROM
      (SELECT NULL AS ROWID_OBJECT ,
        'SVR1.3GDX' AS PKEY_SRC_OBJECT ,
        'SYS0' AS ROWID_SYSTEM
      FROM dual) I
CROSS JOIN
      (SELECT '1' AS COL2 ,
        '1' AS COL1 ,
        'SVR1.3GDX' AS PKEY_SRC_OBJECT ,
        'SYS0' AS ROWID_SYSTEM
      FROM dual) S
```

Adding Validation Rules

To add a validation rule:

1. Navigate to the Validation Rules editor.
2. Click the **Add Validation Rule** button.

The Schema Manager displays the Add Validation Rule dialog.

3. Specify the properties for the validation rule.
4. If you want, select the rule column(s) for the validation rule by clicking the **Edit** button.

The Validation Rules editor displays the Select Rule Columns dialog.

The available columns are those that have the Validate flag enabled.

Select the column(s) for which the trust level will be downgraded if the condition specified in the WHERE clause for this validation rule is met, and then click **OK**.

Note: If you must use date in a validation rule, then use the `to_date` function and specify the actual format of the date or ensure that the date is specified in the format expected by the database.

5. Click **OK**.

The Schema Manager adds the new rule to the list of validation rules.

Note: If a base object contains existing data and you change validation rules, you must run the Revalidate job to recalculate trust scores for new and existing data.

Editing Validation Rule Properties

To edit a validation rule:

1. Navigate to the Validation Rules editor in the Schema Manager.
2. In the Navigation Rules list, select the navigation rule that you want to configure.

The Validation Rules editor displays the properties for the selected validation rule.

3. Specify the editable properties for this validation rule. You cannot change the rule type.
4. If you want, select the rule column(s) for this validation rule by clicking the **Edit** button.

The Validation Rules editor displays the Select Rule Columns dialog.

The available columns are those that have the Validate flag enabled.

Select the column(s) for which the trust level will be downgraded if the condition specified in the WHERE clause for this validation rule is met, and then click **OK**.



5. Click the **Save** button to save changes.

Note: If a base object contains existing data and you change validation rules, you must run the Revalidate job to recalculate trust scores for new and existing data.

Changing the Sequence of Validation Rules

The execution order for validation rules is extremely important.

Use the following buttons to change the sequence of validation rules in the list.

Click	To....
	Move the selected validation rule higher in the sequence.
	Move the selected validation rule further down in the sequence.

Removing Validation Rules

To remove a validation rule:

1. Navigate to the Validation Rules editor in the Schema Manager.
2. In the Validation Rules list, select the validation rule that you want to remove.
3. Click the **Delete** button.

The Schema Manager prompts you to confirm deletion.

4. Click **Yes**.

Note: If a base object contains existing data and you change validation rules, you must run the Revalidate job to recalculate trust scores for new and existing data.

CHAPTER 21

Configuring the Match Process

This chapter includes the following topics:

- [Before You Begin, 375](#)
- [Configuration Tasks for the Match Process, 375](#)
- [Navigating to the Match/Merge Setup Details Dialog, 377](#)
- [Configuring Match Properties for a Base Object, 378](#)
- [Configuring Match Paths for Related Records, 382](#)
- [Configuring Path Components, 389](#)
- [Configuring Match Columns, 392](#)
- [Configuring Match Column Rules for Match Rule Sets, 405](#)
- [Configuring Primary Key Match Rules, 426](#)
- [Investigating the Distribution of Match Keys, 428](#)
- [Excluding Records from the Match Process, 431](#)
- [Proximity Search, 431](#)
- [Lightweight Matching, 433](#)

Before You Begin

Before you begin, you must install Informatica MDM Hub, create the Hub Store by using the instructions in *Multidomain MDM Installation Guide*, and build the schema.

Configuration Tasks for the Match Process

This section provides an overview of the configuration tasks associated with the match process.

Understanding Your Data

Before you define match rules, you must be very familiar with your data and understand:

- The distribution of the values in the columns you intend to use to determine duplicate records
- The general proportion of the total number of records that are duplicates.

Base Object Properties Associated with the Match Process

The following base object properties affect the behavior of the match process.

Property	Description
Duplicate Match Threshold	Used only with the Match for Duplicate Data job for initial data loads.
Max Elapsed Match Minutes	Timeout (in minutes) when executing a match rule. If exceeded, the match process exits.
Match Flag audit table	If enabled, then an audit table (<i>BusinessObjectName_FMHA</i>) is created and populated with the userID of the user who, in Merge Manager, queued a manual match record for automerging.

Configuration Steps for Defining Match Rules

To define match rules:

1. Configure the match properties for the base object.
2. Define your match columns.
3. Define a match rule set for your match rules.
4. Define your match rules for the rule set.
5. Repeat steps 3 and 4 until you are finished creating match rules.
6. Based on your knowledge of your data, determine whether you require matching based on primary keys.
7. If your data is appropriate for primary key matching, create your primary key match rules.
8. Tune your rules. This is an iterative process by which you apply your match rules to a representative data set, analyze the results, and adjust your settings to optimize the match performance.

Configuring Base Objects with International Data

Informatica MDM Hub supports matching for base objects that contain data from non-United States populations, as well as base objects that contain data from different populations such as the United States and China.

Distributed Match Configuration

You can configure distributed match if you configure multiple Process Servers for the Operational Reference Store. You can run multiple Process Servers in parallel to increase the throughput of the match process.

You need to configure distributed match in the `cmxcleanse.properties` file. You must set the `cmx.server.match.distributed_match` property to 1 to enable it. Default is disabled.

Configuring Data Load

You can configure the data load process to use an intermediate file or to directly load data into the database for the tokenization and match process. You can configure properties in the `cmxcleanse.properties` file to specify the method of data load and the batch size. The default is direct load.

To change the default behavior, add data load properties to the `cmxcleanse.properties` files. The `cmxcleanse.properties` file is in the following directory:

On Windows. <MDM Hub installation directory>\hub\cleanse\resources

On UNIX. <MDM Hub installation directory>/hub/cleanse/resources

The following table describes the data load properties for tokenization and matching:

Properties	Description
<code>cmx.server.tokenize.file_load</code>	Specifies whether to use an intermediate file to load data into the database for tokenization. Set to <code>true</code> to use an intermediate file to load data. Set to <code>false</code> for direct data load. Default is <code>true</code> for Oracle and IBM DB2 environments. Default is <code>false</code> for Microsoft SQL Server environments. Note: Use of intermediate files to load data is not applicable to Microsoft SQL Server.
<code>cmx.server.tokenize.loader_batch_size</code>	Maximum number of insert statements to send to the database during direct load. Default is 1000.
<code>cmx.server.match.file_load</code>	Specifies whether to use an intermediate file to load data into the database for matching. Set to <code>true</code> to use an intermediate file to load data. Set to <code>false</code> for direct data load. Default is <code>true</code> for Oracle and IBM DB2 environments. Default is <code>false</code> for Microsoft SQL Server environments and IBM DB2 environments configured for external match. Note: Use of intermediate files to load data is not applicable to Microsoft SQL Server.
<code>cmx.server.match.loader_batch_size</code>	Maximum number of insert statements to send to the database during direct load. Default is 1000.

Navigating to the Match/Merge Setup Details Dialog

To set up the match and merge process for a base object, begin by completing the following steps:

1. Start the Schema Manager.
2. In the schema navigation tree, expand the base object for which you want to define match properties.
3. In the schema navigation tree, select **Match/Merge Setup**.

The Schema Manager displays the Match/Merge Setup Details dialog box.

If you want to change settings, you need to Acquire a write lock.

The Match/Merge Setup Details dialog contains the following tabs:

Tab Name	Description
Properties	Summarizes the match/merge setup and provides various configurable match/merge settings.
Paths	Allows you to configure the match path for parent/child relationships for records in different base objects or in the same base object.
Match Columns	Allows you to configure match columns for match column rules.
Match Rule Sets	Allows you to define a search strategy and rules using match rule sets.
Primary Key Match Rules	Allows you to define primary key match rules.
Match Key Distribution	Shows the distribution of match keys.
Merge Settings	Allows you to merge and link settings.

Configuring Match Properties for a Base Object

You must set the match properties for a base object before you can configure other match features, such as match columns and match rules.

These match properties apply to all rules for the base object.

Setting Match Properties

You configure match properties for each base object. These settings apply to all of its match rules and rule sets.

To configure match properties for a base object:

1. In the Schema Manager, display the Match/Merge Setup Details pane for the base object that you want to configure.
2. In the Match/Merge Setup Details pane, click the **Properties** tab.
The Schema Manager displays the Properties tab.
3. Acquire a write lock.
4. Edit the property settings that you want to change, clicking the **Edit** button next to the field if applicable.
5. Click the **Save** button to save your changes.

Match Properties

This section describes the configuration settings on the Properties tab.

Calculated, Read-Only Fields

The Properties tab displays the following read-only fields.

Table 1. Read-Only Match Properties

Property	Description
Match Columns	Number of match columns configured for this base object. Read-only.
Match Rule Sets	Number of match rule sets configured for this base object. Read-only.
Match Rules in Active Set	Number of match rules configured for this base object in the rule set currently selected as active. Read-only.
Primary key match rules	Number of primary key match rules configured for this base object. Read-only.

Maximum Matches for Manual Consolidation

This setting helps prevent data stewards from being overwhelmed with thousands of matches for manual consolidation.

This sets the limit on the list of possible matches that must be decided upon by a data steward (default is 1000). Once this limit is reached, Informatica MDM Hub stops the match process until the number of records for manual consolidation has been reduced.

This value is calculated by checking the count of records with a consolidation_ind=2. At the end of each automatch and merge cycle, this count is checked and, if the count exceeds the maximum number of matches for manual consolidation, then the automatch-and-merge process will exit.

Number of Rows per Match Job Batch Cycle

This setting specifies an upper limit on the number of records that Informatica MDM Hub will process for matching during match process execution (Match or Auto Match and Merge jobs). When the match process starts executing, it begins by flagging records to be included in the *match job batch*. From the pool of new/unconsolidated records that are ready for match (CONSOLIDATION_IND=4), the match process changes CONSOLIDATION_IND to 3. The number of records flagged is determined by the Number of Rows per Match Job Batch Cycle. The match process then matches those records in the match job batch against all of the records in the base object.

The number of records in the match job batch affects how long the match process takes to execute. The value to specify depends on the size of your data set, the complexity of your match rules, and the length of the time window you have available to run the match process. The default match batch size is low (10). You increase this based on the number of records in the base object, as well as the number of matches generated for those records based on its match rules.

- The lower your match batch size, the more times you will need to run the match and consolidation processes.
- The higher your match batch size, the more work each match and consolidation process does.

For each base object, there is a medium ground where you reach the optimal match batch size. You need to identify this optimal batch size as part of performance tuning in your environment. Start with a match batch size of 10% of the volume of records to be matched and merged, run the match job only, see how many matches are generated by your match rules, and then adjust upwards or downwards accordingly.

Accept All Unmatched Rows as Unique

Enable (set to **Yes**) this feature to have Informatica MDM Hub mark as unique (CONSOLIDATION_IND=1) any records that have been through the match process, but for which no matches were identified.

If enabled, for such records, Informatica MDM Hub automatically changes their state to *consolidated* (changes the consolidation indicator from 2 to 1). Consolidated records are removed from the data steward's queue via the Automerge batch job.

By default, this option is disabled. In a development environment, you might want this option disabled, for example, while iteratively testing and tuning match rules to determine which records are found to be unique for a given set of match rules.

This option should always be enabled in a production environment. Otherwise, you can end up with a large number of records with a consolidation indicator of 2. If this backlog of records exceeds the Maximum Matches for Manual Consolidation setting, then you will need to process these records first before you can continue matching and consolidating other records.

Match/Search Strategy

Select the match/search strategy to specify the reliability of the match versus the performance you require.

Select one of the following options:

Strategy Option	Description
Fuzzy	Probabilistic match that takes into account spelling variations, possible misspellings, and other differences that can make matching records non-identical. This is the primary means of matching data in a base object. Referred to in this document as <i>fuzzy-match base objects</i> . Note: If you specify a Fuzzy match/search strategy, you must specify a fuzzy match key.
Exact	Matches only records with identical values in the match columns. If you specify an exact match, you can define only exact-match columns for this base object. Exact-match base objects cannot have fuzzy-match columns. Referred to in this document as <i>exact-match base objects</i> .

An exact match strategy is faster, but an exact match will miss some matches if the data is imperfect. The best option to choose depends on the characteristics of the data, your knowledge of the data, and your particular match and consolidation requirements.

Fuzzy Population

If the match/search strategy is Fuzzy, then you must select a *population*, which defines certain characteristics about the records that you are matching.

Data characteristics can vary from country to country. By default, Informatica MDM Hub comes with a Demo population, but Informatica provides standard populations for each country. If you require another population, contact Informatica support. If you chose an exact match/search strategy, then this value is ignored.

Populations perform the following functions for matching:

- accounts for the inevitable variations and errors that are likely to exist in name, address, and other identification data
For example, the population for the US has some intelligence about the typical identification numbers used in US data, such as the social security number. Populations also have some intelligence about the distribution of common names. For example, the US population has a relatively high percentage of the surname Smith. But a population for a non-English-speaking country would not have Smith among the common names.
- specifies how Informatica MDM Hub builds match tokens
- specifies how search strategies and match purposes operate on the population of data to be matched

Match Only Previous Rowid Objects

Enable this property to match the current records against records with lower ROWID_OBJECT values.

For example, if the current record has a ROWID_OBJECT value of 100, then the record is matched only against other records in the base object with a ROWID_OBJECT value that is less than 100. Records that have a ROWID_OBJECT value that is higher than 100 are ignored during the match process.

Use the Match Only Previous Rowid Objects property to reduce the number of matches required and speed performance. However, if you run PUT API calls, or if records are inserted out of rowid order, then records might not be fully matched. You must assess the trade-off between performance and match quantity based on the characteristics of the data and your particular match requirements.

You can enable the Match Only Previous Rowid Objects property to improve the performance of initial data loads. You might lose potential matches if you enable this property for incremental data loads. By default, this option is disabled.

Match Only Once

Available only for fuzzy key matching and only if [“Match Only Previous Rowid Objects” on page 381](#) is checked (selected).

If Match Only Once is enabled (checked), then once a record has found a match, Informatica MDM Hub will not match it any further within this search range (the set of similar match key values). Using this feature can reduce duplicates and increase performance. Instead of finding every match for a record in a search range, Informatica MDM Hub can find a single match for each. In subsequent match cycles, the merge process will put these into large groups of XREF records associated with the base object.

By default, this option is unchecked (disabled). If this feature is enabled, however, you can miss matches. For example, suppose record A matches record B, and record A matches record C, but record B and C do not match. You must assess the trade-off between performance and match quantity based on the characteristics of your data and your particular match requirements.

Dynamic Match Analysis Threshold

During the match process, dynamic match analysis determines whether the match process will take an unacceptably long period of time.

This threshold value specifies the maximum acceptable number of comparisons.

To enable the dynamic match threshold, specify a non-zero value. Enable this feature if you have data that is very similar (with high concentrations of matches) to reduce the amount of work expended for a hot spot in your data. A hotspot is a group of records representing overmatched data—a large intersection of matches. If Dynamic Match Analysis Threshold is enabled, then records that produce more than the specified number of

potential match candidates will be skipped during the match process. By default, this option is zero (disabled).

Before conducting a match on a given search range, Informatica MDM Hub calculates the number of search records (records being searched for matches), and multiplies it by the number of file records (the number of records returned from the match key table that need to be compared). If the result is greater than the specified Dynamic Match Analysis Threshold, then no comparisons are performed on that range of data, and the range is noted in the application server log for further investigation.

Enable Match on Pending Records

By default, the match process includes only ACTIVE records and ignores PENDING records. For state management-enabled objects, select this check box to include PENDING records in the match process. Note that, regardless of this setting, DELETED records are ignored by the match process.

Supporting Long ROWID_OBJECT Values

If a base object has such a large number of records that the ROWID_OBJECT values might exceed 12 digits or more, you need to explicitly enable support for longer values in the Process Server.

To enable the Process Server to use long Rowid Object values, edit the `cmxcleanse.properties` file and configure the `cmx.server.bmg.use_longs` setting:

```
cmx.server.bmg.use_longs=1
```

By default, this option is disabled.

Configuring Match Paths for Related Records

This section describes how to configure match paths for related records, which are used for matching in your Informatica MDM Hub implementation.

Match Paths

A *match path* allows you to traverse the hierarchy between records—whether that hierarchy exists between base objects (*inter-table paths*) or within a single base object (*intra-table paths*). Match paths are used for configuring match column rules involving related records in either separate tables or in the same table.

Foreign Key Relationships and Filters

Configuring match paths that point to other records involves two main components:

Component	Description
foreign key relationships	Used to traverse the relationships to other records. Allows you to specify parent-to-child and child-to-parent relationships.
filters (optional)	Allow you to selectively include or exclude records based on values in a given column, such as ADDRESS_TYPE or PARTY_TYPE.

Relationship Base Objects

In order to configure match rules for these kinds of relationships, particularly many-to-many relationships, you need create a separate base object that serves as a *relationship base object* to describe to Informatica MDM Hub the relationships between records. You populate this relationship base object with information about the relationships using a data management tool (outside of Informatica MDM Hub) rather than using the Informatica MDM Hub processes (land, stage, and load).

You configure a separate relationship base object for each type of relationship. You can include additional attributes of the relationship type, such as start date, end date, and other relationship details. The relationship base object defines a match path that enables you to configure match column rules.

Important: Do not run the match and consolidation processes on a base object that is used to define relationships between records in inter-table or intra-table match paths. Doing so will change the relationship data, resulting in the loss of the associations between records.

Inter-Table Paths

An inter-table path defines the relationship between records in two different base objects. In many cases, this relationship can be defined simply by configuring a foreign key relationship: a key column in the child base object points to the primary key of the parent base object.

In some cases, however, the relationship between records can be more complex, requiring an intermediary base object that defines the relationship between records in the two tables.

Example Base Objects for Inter-Table Paths

Consider the following example in which a Informatica MDM Hub implementation has two base objects:

Base Object	Description
Person	Contains any type of person, such as employees for your organization, employees for some other organizations (prospects, customers, vendors, or partners), contractors, and so on.
Address	Contains any type of address—mailing, shipping, home, work, and so on.

In this example, there is the potential for many-to-many relationships:

- A person could have multiple addresses, such as a home and work address.
- A single address could have multiple persons, such as a workplace or home.

In order to configure match rules for this kind of relationship between records in different base objects, you would create a separate base object (such as PersAddrRel) that describes to Informatica MDM Hub the relationships between records in the two base objects.

Columns in the Example Base Objects

Suppose the Person base object had the following columns:

Column	Type	Description
ROWID_OBJECT	CHAR(14)	Primary key. Uniquely identifies this person in the base object.
TYPE	CHAR(14)	Type of person, such as an employee or customer contact.
NAME	VARCHAR(50)	Person's name (simplified for this example).
EMPLOYER	VARCHAR(50)	Person's employer.
...

Suppose the Address base object had the following columns:

Column	Type	Description
ROWID_OBJECT	CHAR(14)	Primary key. Uniquely identifies this employee.
TYPE	CHAR(14)	Type of address, such as their home, work, mailing, or shipping address.
NAME	VARCHAR(50)	Name of the individual or organization residing at this address.
ADDRESS_1	VARCHAR(50)	First address line.
ADDRESS_2	VARCHAR(50)	Second address line.
CITY	VARCHAR(50)	City
STATE_PROV	VARCHAR(50)	State or province
POSTAL_CODE	VARCHAR(50)	Postal code
...

To define the relationship between records in the two base objects, the PersonAddrRel base object could have the following columns:

Column	Type	Description
ROWID_OBJECT	CHAR(14)	Primary key. Uniquely identifies this person in the base object.
PERS_FK	CHAR(14)	Foreign key to the ROWID_OBJECT column in the Person base object.
ADDR_FK	CHAR(14)	Foreign key to the ROWID_OBJECT column in the Address base object.

Note that the column type of the foreign key columns—CHAR(14)—matches the primary key to which they point.

Example Configuration Steps

After you have configured the relationship base object (PersonAddrRel), you would complete the following tasks:

1. Configure foreign keys from this base object to the ROWID_OBJECT of the Person and Address base objects.
2. Load the PersAddrRel base object with data that describes the relationships between records.

ROWID_OBJECT	PERS_FKEY	ADDR_FKEY
1	380	132
2	480	920
3	786	432
4	786	980
5	12	1028
6	922	1028
7	1302	110
...

In this example, note that Person #786 has two addresses, and that Address #1028 has two persons.

3. Use the PersonAddrRel base object when configuring match column rules for the related records.

Intra-Table Paths

Within a base object, parent/child relationships can exist between individual records. Informatica MDM Hub allows you to clarify relationships between records in the same base object, and then use those relationships when configuring column match rules.

Example Base Object for Intra-Table Paths

Consider the following figure shows an Employee base object in which reporting relationships exist between employees:

Employee Name	Title
Maria Alvarez	CEO
Todd Garvey	President
Lisa Cho	Vice President
Anil Shahani	Director
Jane Rickets	Manager
Mark Sullivan	Analyst
Rajan Vir	Analyst
Roberta Zimmer	Analyst
...	...



The relationships among employees is hierarchical. The CEO is at the top of the hierarchy, representing what is called the *global ultimate parent* record.

Columns in the Example Base Object

Suppose the Employee base object had the following columns:

Column	Type	Description
ROWID_OBJECT	CHAR(14)	Primary key. Uniquely identifies this employee in the base object.
NAME	VARCHAR(50)	Employee name.
TITLE	VARCHAR(50)	Employee's job title.
...

Create a Relationship Base Object

To configure match rules for this kind of object, you would create a separate base object to describe to Informatica MDM Hub the relationships between records.

For example, you could create and configure a EmplRepRel base object with the following columns:

Column	Type	Description
ROWID_OBJECT	CHAR(14)	Primary key. Uniquely identifies this relationship record.
EMPLOYEE_FK	CHAR(14)	Foreign key to the ROWID_OBJECT of the employee record.
REPORTS_TO_FK	CHAR(14)	Foreign key to the ROWID_OBJECT of a manager record.

Note: The column type of the foreign key columns—CHAR(14)—matches the primary key to which they point.

Example Configuration Steps

After you have configured this base object, you must complete the following tasks:

1. Configure foreign keys from this base object to the ROWID_OBJECT of the Employee base object.
2. Load this base object with data that describes the relationships between records.

ROWID_OBJECT	EMPLOYEE	REPORTS_TO
1	7	93
2	19	71
3	24	82
4	29	82

ROWID_OBJECT	EMPLOYEE	REPORTS_TO
5	31	82
6	31	71
7	48	16
8	53	12

Note that you can define many-to-many relationships between records. For example, the employee whose ROWID_OBJECT is 31 reports to two different managers (ROWID_OBJECT=82 and ROWID_OBJECT=71), while this manager (ROWID_OBJECT=82) has three reports (ROWID_OBJECT=24, 29, and 31).

3. Use the EmplRepRel base object when configuring match column rules for the related records. For example, you could create a match rule that takes into account the employee's manager to produce more accurate matches.

Note: This example used a REPORTS_TO field to define the relationship, but you could use piece of information to associate the records—even something more generic and flexible like RELATIONSHIP_TYPE.

Navigating to the Paths Tab

To navigate to the Paths tab for a base object:

1. In the Schema Manager, navigate to the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Click the **Paths** tab.

The Schema Manager displays the Path Components window.

Sections of the Paths Tab

The Paths tab has two sections:

Section	Description
Path Components	Configure the foreign keys used to traverse the relationships.
Filters	Configure filters used to include or exclude records for matching.

Root Base Object

The root base object is displayed automatically in the Path Components section of the screen and is always available.

The root base object represents an entity without child or parent relationships. If you want to configure match rules that involve parent or child records, you need to explicitly add path components to the root base object, and these relationships must have been configured beforehand.

Configuring Filters for Match Paths

The filter of a match path includes or excludes records for matching based on the values of the column that you specify. To define a filter for a column, specify the filter condition with one or more values that determine the records that qualify for match processing. For example, for an Address base object that contains shipping and billing addresses, you can configure a filter that includes billing addresses and excludes shipping addresses. When the match process runs, the MDM Hub matches records in the match batch with the billing address records.

Note: If you specify a filter condition on a date column, use the correct string representation of a date value that is compatible with the database locale.

In Informatica MDM Hub, filters have the following properties:

Setting	Description
Column	Column to configure in the currently-selected base object.
Operator	Operator to use for this filter. One of the following values: <ul style="list-style-type: none">- IN—Include columns that contain the specified values.- NOT IN—Exclude columns that contain the specified values.
Values	One or more values to use for this filter.

For example, if you wanted to match only on mailing addresses in an Address base object, you could specify:

Setting	Example Value
Column	ADDR_TYPE
Operator	IN
Values	MAILING

In this example, only mailing addresses would qualify for matching—records in which the COLUMN field contains “MAILING”. All other records would be ignored.

Adding Filters

If you add multiple filters, Informatica MDM Hub evaluates the entire expression using the logical AND operator. For example:

```
xExpr AND yExpr AND zExpr
```

To add a filter:

1. In the Schema Manager, navigate to the **Paths** tab.
2. Acquire a write lock.
3. In the Filters section, click the **Add** button.
The Schema Manager displays the Add Filter dialog.
4. Specify the properties for this path component.
5. Specify the value(s) for this filter.
6. Click the **Save** button to save your changes.

Editing Values for a Filter

To edit values for a filter:

1. Do one of the following:
 - Add a filter.
 - Edit filter properties.
2. In either the Add Filter or Edit Filter dialog, click the **Edit** button next to the Values field.
The Schema Manager displays the Edit Values dialog.
3. Configure the values for this filter.
 - To add a value, click the **Add** button. When prompted, specify a value and then click **OK**.
 - To delete a value, select it in the Edit Values dialog, click the **Delete** button, and then click **Yes** when prompted to delete the value.
4. Click **OK**.
5. Click the **Save** button to save your changes.

Editing Filter Properties

To edit filter properties:

1. In the Schema Manager, navigate to the **Paths** tab.
2. Acquire a write lock.
3. In the Filters section, click the **Edit** button.
The Schema Manager displays the Add Filter dialog.
4. Specify the properties for this path component.
5. Specify the value(s) for this filter.
6. Click the **Save** button to save your changes.

Deleting Filters

To delete a filter:

1. In the Schema Manager, navigate to the **Paths** tab.
2. Acquire a write lock.
3. In the Filters section, select the filter that you want to delete, and then click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
4. Click **Yes**.

Configuring Path Components

This section describes how to configure path components in the Schema Manager. Path components provide a way to define the connection between parent and child tables using foreign keys for the purpose of using columns from that table in a match column.

Display Name

The name of this path component as it will be displayed in the Hub Console.

Physical Name

Actual name of the path component in the database. Informatica MDM Hub will suggest a physical name for the path component based on the display name that you enter.

Allow Missing Child Records

The **Allow missing child records** option indicates whether parent records must be considered for matching based on a check for the existence of records in child base objects in the match path.

You can enable the option to allow missing child records at the match path component level. The match path component level might consist of many child base object levels in the match path of the main parent base object.

When you enable the option to allow missing child records on a match path component, matching occurs between the parent base object records and their associated child base object records. Matching occurs even if the parent base object records do not have child records in the child base object for which the option is enabled. By default, the option to allow missing child records is enabled for all child base objects in the match path of the parent base object. When the option is enabled, it is inclusive and therefore implements a database outer join between a parent and a child base table that has the option enabled. The Informatica Data Director basic search returns results for records that do not have a child record.

When you disable the option to allow missing child records for all match path components, parent base object records that have records in all child base objects undergo matching. If a parent record does not have a child record in a child base object for which the option is disabled, the parent record does not undergo matching. When the option is disabled, it is exclusive and therefore similar to a regular database equi-join where, if the parent record does not have child records, no parent or child records are returned. When you disable the option, you avoid the performance impact associated with an outer join. The Informatica Data Director basic search does not return results for records that do not have a child record.

If you need to perform a fuzzy match on a base object, tokenization of a parent base object record must occur. Tokenization of a parent base object record occurs if all child base objects that have the option to allow missing child records disabled have a related child base object record. If a parent base object record has a child base object, where the option to allow missing child records is disabled yet contains no record, the parent record is not tokenized.

If the data is complete, matching between parent records occurs as expected and the MDM Hub includes all parent records for matching according to the match criteria. Your data is complete if parent records have child records in each child base object in the match path of the parent, and you include the child column in the match rule on the parent. However, if your data is incomplete, enable the check for missing children option on the path component for child base objects that do not have records for the parent records to be eligible for matching. The data is incomplete if your data contains parent records with records missing from one child base object and other parent records with records missing from a different child base object. Also, if you do not include the child column in the parent match rule, the data is incomplete. This ensures that parent records and their associated child base object records on whose columns the match path columns are based, are not excluded from a match when the parent has no record in that child base object.

Note: The Informatica MDM Hub performs an outer join between the parent and child tables when the option to allow missing child records is enabled. This impacts performance on each match path component on which the option is enabled. Therefore, when not needed, it is more efficient to disable this option.

Constraints

Property	Description
Table	List of tables in the schema.
Direction	Direction of the foreign key: Parent-to-Child Child-to-Parent N/A
Foreign Key On	Column to which the foreign key points. This column can be either in a different base object or the same base object.

Adding Path Components

To add a path component:

1. In the Schema Manager, navigate to the Paths tab.
2. Acquire a write lock.
3. In the Path Components section, click the **Add** button.
The Schema Manager displays the Add Path Component dialog.
4. Specify the properties for this path component.
5. Click **OK**.
6. Click the **Save** button to save your changes.

Editing Path Components

To edit a path component:

1. In the Schema Manager, navigate to the **Paths** tab.
2. Acquire a write lock.
3. In the Path Components tree, select the path component that you want to delete.
4. In the Path Components section, click the **Edit** button.
The Schema Manager displays the Edit Path Component dialog.
5. Specify the properties for this path component. You can change the following values:
 - Display Name
 - Allow missing child records
6. Click **OK**.
7. Click the **Save** button to save your changes.

Deleting Path Components

You can delete path components but *not* the root base object. To delete a path component:

1. In the Schema Manager, navigate to the **Paths** tab.
2. Acquire a write lock.

3. In the Path Components tree, select the path component that you want to delete.
4. In the Path Components section, click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
5. Click **Yes**.
6. Click the **Save** button to save your changes.

Configuring Match Columns

This section describes how to configure match columns so that you can use them in match column rules. If you want to configure primary key match rules instead, see the instructions in [“Configuring Primary Key Match Rules” on page 426](#).

About Match Columns

A *match column* is a column that you want to use in a match rule, such as name or address columns. Before you can use a column in rule definitions, you must first designate it as a column that can be used in match rules, and provide information about the data it contains.

Note: When you select columns to define a match rule, columns with a numeric or decimal data type are not available as match columns.

Match Column Types

You can configure a fuzzy or exact column type for match rules.

The following table describes the column types for match rules:

Column Type	Description
Fuzzy	<i>Probabilistic match.</i> Use for columns that contain data that vary in spelling, abbreviations, word sequence, completeness, reliability, and other inconsistencies. For example, fuzzy columns include street addresses, geographic coordinates such as latitude, longitude, and elevation, and names of people or organizations.
Exact	<i>Deterministic match.</i> Use for columns that contain consistent and predictable patterns. Exact match columns match only on identical data. For example, IDs, postal codes, industry codes, or any other well-defined piece of information.

Match Columns Depend on the Search Strategy

The types of match columns that you can configure depend on the type of the base object that you are configuring.

The type of base object is defined by the selected match / search strategy.

Match Strategy	Description
Fuzzy-match base objects	Allows you to configure fuzzy-match columns as well as exact-match columns.
Exact-match base objects	Allows you to configure exact-match columns but not fuzzy-match columns.

Path Component

The path component is either the source table to use for a match column definition, or the match path used to navigate a hierarchy of records. Match paths are used for configuring match column rules involving related records in either separate tables or in the same table. Before you can specify a path component, the match path must be configured.

To specify a path component for a match column:

1. Click the **Edit** button next to the Path Component field.
The Schema Manager displays the Select Match Path Component dialog.
2. Select the match path component.
3. Click **OK**.

Field Names

When you add a fuzzy match column to a match rule, you can select a field name from a list.

The following table describes the field names that you can select when you add a fuzzy match column for a match rule:

Field Name	Description
Address_Part1	<p>Includes the part of address up to, but not including, the locality last line. The position of the address components must be in the normal word order, as in the data population. Pass this data in one field. Based on your base object, you might concatenate these attributes into one field before matching. For example, in the US, an Address_Part1 string includes the following fields: Care-of + Building Name + Street Number + Street Name + Street Type + Apartment Details. Address_Part1 uses methods and options that are designed specifically for addresses.</p> <p>To prevent potential overmatching, the match process only considers transposition errors when the field contains 10 or more characters. For example, 'PO Box 38' does not match 'PO Box 83', but '13 Capital Street' matches '31 Capital Street'.</p>
Address_Part2	<p>Locality line in an address. For example, in the US, a typical Address_Part2 includes: City + State + Zip (+ Country). Matching on Address_Part2 uses methods and options designed specifically for addresses.</p>
Attribute1, Attribute2	<p>Two general purpose fields. The MDM Hub matches the attribute fields using a general purpose, string matching algorithm that compensates for transpositions and missing characters or digits.</p>

Field Name	Description
Date	Matches any type of date, such as date of birth, expiry date, date of contract, date of change, and creation date. Pass the date in the Day+Month+Year format. The SSA_Date field name supports the use or absence of delimiters between the date components. Matching on dates uses methods and options designed specifically for dates. Matching on dates overcomes the typical error and variation found in this data type.
Geocode	Matches the geographic coordinates, latitude, longitude, and elevation. Specify the geographic coordinates in the following order: 1. Latitude 2. Longitude 3. Elevation You can pass this data in one field or multiple fields. If you concatenate the geocode data in one field, separate the values by a comma or a space. Geocode uses a string matching algorithm that compensates for transpositions and missing characters or digits.
ID	Matches any type of ID, such as account number, customer number, credit card number, drivers license number, passport number, policy number, SSN, or other identity code and VIN. The ID field uses a string matching algorithm that compensates for transpositions and missing characters or digits.
Organization_Name	Matches the names of organizations, such as organization names, business names, institution names, department names, agency names, and trading names. This field supports matching on a single name or on a compound name such as a legal name and its trading style. You might also use multiple names such as a legal name and a trading style in a single Organization_Name column for the match.
Person_Name	Matches the names of people. Use the full person name. The position of the first name, middle names, and family names, must be the normal word order used in your population. For example, in English-speaking countries, the normal order is: First Name + Middle Names + Family Names. Based on your base object design, you can concatenate these fields into one field before matching. This field supports matching on a single name, or an account name such as JOHN & MARY SMITH. You might also use multiple names, such as a married name and a former name.
Postal_Area	Use to place more emphasis on the postal code by not using the Address_Part2 field. Use for all types of postal codes, including ZIP codes. The Postal_Area field name uses a string matching algorithm that compensates for transpositions and missing characters or digits.
Telephone_Number	Use to match telephone numbers. The Telephone_Number field name uses a string matching algorithm that compensates for transpositions and missing digits or area codes.

Selecting Multiple Columns for Matching

If you specify more than one column for matching:

- Values are concatenated into the field used by the match purpose, with a space inserted between each value. For example, you can select first, middle, last, and suffix columns in your base object. The concatenated fields will look like this (a space follows the last word in the string):

```
first middle last suffix
```

For example:

```
Anna Maria Gonzales MD
```

- For data containing spaces or null data:
 - If there are spaces in the data, then the spaces remain and the field is not NULL.
 - If all the fields are null, then the combined value is null.
 - If any component on the combined field is null, then no extra space will be added to replace the null.

Note: Concatenating columns is not recommended for exact match columns.

Configuring Match Columns for Fuzzy-match Base Objects

Fuzzy-match base objects can have both fuzzy and exact-match columns. For exact-match base objects instead, see [“Configuring Match Columns for Exact-match Base Objects” on page 398](#).

Navigating to the Match Columns Tab for a Fuzzy-match Base Object

To define match columns for a fuzzy-match base object:

1. In the Schema Manager, select the fuzzy-match base object that you want to configure.
2. Click the **Match/Merge Setup** node.
3. Click the **Match Columns** tab.

The Schema Manager displays the Match Columns tab for the fuzzy-match base object.

The Match Columns tab for a fuzzy-match base object has the following sections.

Property	Description
Fuzzy Match Key	Properties for the fuzzy match key.
Match Columns	Match columns and their properties: <ul style="list-style-type: none"> - Field Name - Column Type - Path Component - Source Table—table referenced in the path component, or the base object (if the path component is root)
Match Column Contents	List of available columns in the base object, as well as columns that have been selected for match.

Configuring Fuzzy Match Key Properties

This section describes how to configure the match column properties for fuzzy-match base objects. The *Fuzzy Match Key* is a special column in the base object that the Schema Manager adds if a match column uses the fuzzy match / search strategy. This column is the primary field used during searching and matching to generate match candidates for this base object. All fuzzy base objects have one and only one Fuzzy Match Key.

Key Types

The *match key type* describes important characteristics about a column to Informatica MDM Hub. Informatica MDM Hub has some intelligence about names and addresses, so this information helps Informatica MDM Hub generate keys correctly and conduct better searches. This is the main criterion for the search that builds the initial list of potential match candidates. This key type should be based on the main type of data that is in physical column(s) that make up the fuzzy match key.

For a fuzzy-match base object, you can select one of the following key types:

Key Type	Description
Person_Name	Used if your fuzzy match key contains data for individuals only.
Organization_Name	Used if your fuzzy match key contains data for organizations only, or if it contains data for both organizations and individuals.
Address_Part1	Used if your fuzzy match key contains address data to be consolidated.

Note: Key types are based on the population you select. The above list of key types applies to the default population (US). Other populations might have different key types. If you require another population, contact Informatica support.

Key Widths

The *match key width* determines the thoroughness of the analysis of the fuzzy match key, the number of possible match candidates returned, and how much disk space the keys consume. Key widths apply to fuzzy-match objects only.

Key Width	Description
Standard	Appropriate for most fuzzy match keys, balancing reliability and space usage.
Extended	Might result in more match candidates, but at the cost of longer processing time to generate keys. This option provides some additional matching capability due to the concatenation of columns. This key width works best when: <ul style="list-style-type: none">- your data set is not extremely large- your data set is not complete- you have sufficient resources to handle the processing time and disk space requirements
Limited	Trades some match reliability for disk space savings. This option might result in fewer match candidates, but searches can be faster. This option works well if you are willing to undermatch for faster searches that use less disk space for the keys. Limited keys match fewer records with word-order variations than standard keys. This choice provides a subset of the Standard key set, but might be the best option if disk space is restricted or the data volume is extremely large.
Preferred	Generates a single key per base object record. This option trades some match reliability for performance (reduces the number of matches that need to be performed) and disk space savings (reduces the size of the match key table). Depending on characteristics of the data, a preferred key width might result in fewer match candidates.

Steps to Configure Fuzzy Match Key Properties

To configure fuzzy match key properties for a fuzzy-match base object:

1. In the Schema Manager, navigate to the Match Columns tab.
2. Acquire a write lock.

3. Configure the following settings for this fuzzy-match base object.

Property	Description
Key Type	Type of field primarily used in the match. This is the main criterion for the search that builds the initial list of potential match candidates. This key type should be based on the main type of data stored in the base object.
Key Width	Size of the search range for which keys are generated.
Path Component	Path component for this fuzzy match key. This is a table containing the column(s) to designate as the key type: Base Object, Child Base Object table, or Cross-reference table.

4. Click the **Save** button to save your changes.

Adding a Fuzzy-match Column for Fuzzy-match Base Objects

To define a fuzzy-match column for a fuzzy-match base object:

1. In the Schema Manager, navigate to the Match Columns tab.
2. Acquire a write lock.
3. To add a fuzzy-match column, click the **Add** button.
The Schema Manager displays the Add Fuzzy-match Column dialog.
4. Specify the following settings.

Property	Description
Match Path Component	Match path component for this fuzzy-match column. For a fuzzy-match column, the source table can be the parent table, a parent cross-reference table, or any child base object table.
Field Name	Name of the field. Select the type of data for the match column.

5. Specify the base object columns for the fuzzy match.
To add a column to the **Selected Columns** list, select a column name and click the right-arrow button.
Note: If you add multiple columns, the values are concatenated, with a separator space between the values.
6. Click **OK**.
The Schema Manager adds the match column to the Match Columns list.
7. Click the **Save** button to save your changes.

Adding Exact-match Columns for Fuzzy-match Base Objects

To define an exact-match column for a fuzzy-match base object:

1. In the Schema Manager, navigate to the **Match Columns** tab.
2. Acquire a write lock.
3. To add an exact-match column, click the **Add** button.
The Schema Manager displays the Add Exact-match Column dialog.

- Specify the following settings.

Property	Description
Match Path Component	Match path component for this exact-match column. For an exact-match column, the source table can be the parent table and / or child physical columns.
Field Name	Name for the field that you want the Hub Console to display.

- Specify the base object columns for the exact match.
- To add a column to the **Selected Columns** list, select a column name and click the right-arrow button.
Note: If you add multiple columns, the values are concatenated, with a separator space between values. Do not concatenate columns for exact-match columns.
- Click **OK**.
The Schema Manager adds the match column to the Match Columns list.
- Click the **Save** button to save your changes.

Editing Match Column Properties for Fuzzy-match Base Objects

Instead of editing match column properties, you must:

- delete the match column
- add a new match column

Deleting Match Columns for Fuzzy-match Base Objects

To delete a match column for a fuzzy-match base object:

- In the Schema Manager, navigate to the **Match Columns** tab.
- Acquire a write lock.
- In the Match Columns list, select the match column that you want to delete.
- Click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
- Click **Yes**.
- Click the **Save** button to save your changes.

Configuring Match Columns for Exact-match Base Objects

Before you define match column rules, you must define the match columns on which they will be based. Exact-match base objects can have only exact-match columns. For more information about configuring match columns for fuzzy-match base objects instead, see [“Configuring Match Columns for Fuzzy-match Base Objects” on page 395](#).

Navigating to the Match Columns Tab for an Exact-match Base Object

To define match columns for an exact-match base object:

- In the Schema Manager, display the Match/Merge Setup Details dialog for the exact-match base object that you want to configure.

2. Click the **Match Columns** tab.

The Schema Manager displays the Match Columns tab for the exact-match base object.

The Match Columns tab for an exact-match base object has the following sections.

Property	Description
Match Columns	Match columns and their properties: <ul style="list-style-type: none">- Field Name- Column Type- Path Component- Source Table—table referenced in the path component, or the base object (if the path component is root)
Match Column Contents	List of available columns and columns selected for matching.

Adding Match Columns for Exact-match Base Objects

You can add only exact-match columns for exact-match base objects. Fuzzy-match columns are not allowed.

To add an exact-match column for an exact-match base object:

1. In the Schema Manager, navigate to the **Match Columns** tab.
2. Acquire a write lock.
3. To add an exact-match column, click the **Add** button.

The Schema Manager displays the Add Exact-match Column dialog.

4. Specify the following settings.

Property	Description
Match Path Component	Match path component for this exact-match column. For an exact-match column, the source table can be the parent table and / or child physical columns.
Field Name	Name of this field as it will be displayed in the Hub Console.

5. Specify the base object column(s) for the exact match.
6. To add a column to the Selected Columns list, select a column name and then click the right arrow.

Note:

- If you add multiple columns, the values are concatenated, with a separator space between values.
- Concatenating columns is not recommended for exact match columns.

7. Click **OK**.

The Schema Manager adds the selected match column(s) to the Match Columns list.

8. Click the **Save** button to save your changes.

Editing Match Column Properties for Exact-match Base Objects

If an exact match column has a path that is not a root path, you can set the match column as a **Default Match Subtype**. In a Match Rule Set, when you add a fuzzy match rule with this exact match column, the **Match Subtype** option is selected.

Note: If you want to edit any other properties, delete the match column, save, and then add a match column with the updated properties.

1. In the **Match Columns** tab, select a match column with the following attributes:
 - **Column Type** = Exact
 - **Path Component** = *<any path except Root>*
2. Click **Edit**.
3. Select **Default Match Subtype**, and click **OK**.
4. Click **Save**.

RELATED TOPICS:

- [“Match Subtype” on page 414](#)

Deleting Match Columns for Exact-match Base Objects

To delete a match column for an exact-match base object:

1. In the Schema Manager, navigate to the **Match Columns** tab.
2. Acquire a write lock.
3. In the Match Columns list, select the match column that you want to delete.
4. Click the **Delete** button.

The Schema Manager prompts you to confirm deletion.
5. Click **Yes**.
6. Click the **Save** button to save your changes.

Match Rule Sets

A *match rule set* is a logical collection of match column rules that have some properties in common.

Match rule sets are associated with match column rules only—not primary key match rules.

Match rule sets allow you to run different sets of match column rules at different times. Each time the match process is run, only one match rule set is used. To match using a different match rule set, select the match rule set and run the match process again.

The match rule sets that you configure are used in the Data Director to create extended queries. When you modify or delete a match rule set, the queries that use the match rule set are deleted.

Note: Only one match column rule in the match rule set needs to succeed in order to declare a match between records.

What Match Rule Sets Specify

Match rule sets include:

- a search level that dictates the search strategy
- any number of automatic and manual match column rules

- optionally, a filter that allows you to selectively include or exclude records from the match batch during the match process

Multiple Match Rule Sets and the Specified Default

You can configure any number of rule sets.

When users want to run the Match batch job, they select one rule set from the list of rule sets that have been defined for the base object.

In the Schema Manager, you designate one match rule set as the default.

When to Use Match Rule Sets

Match rule sets allow you to accommodate different match column rule requirements at different times.

For example, you might use one match rule set for an initial data load and a different match rule set for subsequent incremental loads. Similarly, you might use one match rule set to process all records, and another match rule set with a filter to process just a subset of records.

Rule Set Evaluation

Before saving any changes to a match rule set (including any changes to match rules in the match rule set), the Schema Manager analyzes the match rule set and prompts you with a warning message if the match rule set has any issues.

Note: This is only a warning message. You can choose to ignore the message and save changes anyway.

Example issues include a match rule set that:

- is identical to an already existing match rule set
- is empty—no match column rules have been added
- contains no fuzzy-match column rules for a fuzzy-match base object
- contains one or more fuzzy-match columns but no exact-match column (can impact match performance)
- contains fuzzy and exact-match columns with the same source columns

Match Rule Set Properties

This section describes the properties for match rule sets.

Name

The name of the rule set. Specify a unique, descriptive name.

Search Levels

Used with fuzzy-match base objects only. When you configure a match rule set, you define a *search level* that instructs Informatica MDM Hub on how stringently and thoroughly to search for candidate matches.

The goal of the match process is to find the optimal number of matches for your data:

- not too few (called *undermatching*), which misses relevant matches, or
- not too many (called *overmatching*), which generates too many matches, including matches that are not relevant

For any name or address in a fuzzy match key, Informatica MDM Hub uses the defined search level to generate different key ranges for the purpose of determining which records are possible match candidates—and to which records the match column rules will be applied.

You can choose one of the following search levels:

Search Level	Description
Narrow	Most stringent level in searching for possible match candidates. This search level is fast, but it can result in fewer matches than other search levels might generate and possibly result in undermatching. Narrow can be appropriate if your data set is relatively correct and complete, or for very large data sets with highly matchy data.
Typical	Appropriate for most rule sets.
Exhaustive	Generates a larger set of possible match candidates than the Typical level. This can result in more matches than other search levels might generate, possibly result in overmatching, and take more time. This level might be appropriate for smaller data sets that are less complete.
Extreme	Generates a still larger set of possible match candidates, which can result in overmatching and take more much more time. This level might be appropriate for smaller data sets that are less complete, or to identify the highest possible number of matching records.

The search level you choose should be determined by the size of your data set, your time constraints, and how critical the matches are. Depending on your circumstances and requirements, it is sometimes more appropriate to undermatch, while at other times, it is more appropriate to overmatch. Implementations dealing with relatively reliable and complete data can use the Narrow level, while implementations dealing with less reliable data or with more critical problems should use Exhaustive or Extreme.

The search level might also differ depending on the phase of a project. It might be necessary to have a looser level (exhaustive or extreme) for initial matching, and tighten as the data is deduplicated.

Enable Search by Rules

When you enable a match rule set for Enable Search by Rules, you reserve that particular match rule set for use with the **SearchMatch** API. To use an Enable Search by Rules match rule set, the **SearchMatch** MatchType must be **NONE** and you must specify the match rule set in the **SearchMatch** request. If you use Enable Search by Rules then the MDM Hub only uses fuzzy matches.

You might want to perform a search with some empty fields. When SearchMatch performs a search using an Enable Search by Rules match rule set, it ignores empty fields provided by the search request. This prevents the exclusion of relevant records from the search results.

For more information about **SearchMatch**, see the *Multidomain MDM Services Integration Framework Guide*.

Note: If you enable the optional Hub Server property to perform exact matches on fuzzy base objects, it affects match rules. The MDM Hub does not return matches if you use the same source columns for both exact and fuzzy matching. To avoid this problem, you must disable the `cmx.server.match.exact_match_fuzzy_bo_api` property in the `cmxserver.properties` file. For more information about the Hub Server properties, see [“Hub Server Properties” on page 610](#).

Enable Filtering

Specifies whether filtering is enabled for this match rule set.

- If checked (selected), allows you to define a filter for this match rule set. When running a Match job, users can select the match rule set with a filter defined so that the Match job processes only the subset of records that meet the filter criteria.
- If unchecked (not selected), then all records will be processed by the match rule set when the Match batch job runs.

For example, if you had an Organization base object that contained multiple types of organizations (customers, vendors, prospects, partners, and so on), you could define different match rule sets that selectively processed only the type of records you want to match: MatchAll (no filter), MatchCustomersOnly, MatchVendorsOnly, and so on.

Note: You can enable filters on a Match rule set, but filters on child base objects and group functions is not allowed.

Filtering SQL

By default, when the Match batch job is run, the match rule set processes all records.

If the Enable Filtering check box is selected (checked), you can specify a filter condition to restrict processing to only those rules that meet the filter condition. A *filter* is analogous to a WHERE clause in a SQL statement. The filter expression can be any expression that is valid for the WHERE clause syntax used in your database platform.

Note: The match rule set filter is applied to the base object records that are selected for the *match batch* only (the records to match from)—not the records in the match pool (the records to match to).

For example, suppose your implementation had an Organization base object that contained multiple types of organizations (customers, vendors, prospects, partners, and so on). Using filters, you could define a match rule set (MatchCustomersOnly) that processed customer data only.

```
org_type='C'
```

All other, non-customer records would be ignored and not processed by the Match job.

Note: It is the administrator's responsibility to specify an appropriate SQL expression that correctly filters records during the Match job. The Schema Manager validates the SQL syntax according to your database platform, but it does not check the logic or suitability of your filter condition.

Match Rules

This area of the window displays a list of match column rules that have been configured for the selected match rule set.

Navigating to the Match Rule Set Tab

To navigate to the Match Rule Set tab:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Click the **Match Rule Sets** tab.

The Schema Manager displays the Match Rule Sets tab for the selected base object.

3. The Match Rule Sets tab consists of the following sections:

Section	Description
Match Rule Sets	List of configured match rule sets.
Properties	Properties for the selected match rule set.

Adding Match Rule Sets

To add a new match rule set:

1. In the Schema Manager, display the **Match Rule Sets** tab in the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Add** button.
The Schema Manager displays the Add Match Rule Set dialog.
4. Enter a unique, descriptive name for this new match rule set.
5. Click **OK**.
The Schema Manager adds the new match rule set to the list.
6. Configure the match rule set.

Editing Match Rule Set Properties

To edit the properties of a match rule set:

1. In the Schema Manager, display the **Match Rule Sets** tab in the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Select the match rule set that you want to configure.
The Schema Manager displays its properties in the properties panel.
4. Configure properties for this match rule set.
5. Configure match columns for this match rule set.
6. Click the **Save** button to save your changes.
Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruences.
7. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Renaming Match Rule Sets

To rename a match rule set:

1. In the Schema Manager, display the **Match Rule Sets** tab in the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Select the match rule set that you want to rename.
4. Click the **Edit** button.

The Schema Manager displays the Edit Rule Set Name dialog.

5. Specify a unique, descriptive name for this match rule set.
6. Click **OK**.

The Schema Manager updates the name of the match rule set in the list.

Deleting Match Rule Sets

To delete a match rule set:

1. In the Schema Manager, display the **Match Rule Sets** tab in the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Select the name of the match rule set that you want to delete.
4. Click the **Delete** button.

The Schema Manager prompts you to confirm deletion.

5. Click **Yes**.

The Schema Manager removes the deleted match rule set, along with all of the match column rules it contains, from the list.

Configuring Match Column Rules for Match Rule Sets

A *match column rule* determines what constitutes a match during the match process.

Match column rules determine whether two records are similar enough to consolidate. Each match rule is defined as a set of one or more match columns that it needs to examine for points of similarity. Match rules are configured by setting the conditions for identifying matching records within and across source systems.

Prerequisites for Configuring Match Column Rules

You can configure match column rules only after you have performed the following prerequisites:

- configured the columns that you intend to use in your match rules
- created at least one match rule set

Match Column Rules Differ Between Exact-Match and Fuzzy-Match Base Objects

The properties for match column rules differ between exact match and fuzzy-match base objects.

- For exact-match base objects, you can configure only exact column types.
- For fuzzy-match base objects, you can configure fuzzy or exact column types.

For each match column rule, decide whether matched records should be automatically or manually consolidated.

Specifying Consolidation Options for Matched Records

For each match column rule, decide whether matched records should be automatically or manually consolidated.

Match Rule Properties for Fuzzy-match Base Objects Only

This section describes match rule properties for fuzzy-match base objects. These properties do not apply to exact-match base objects.

Match/Search Strategy

For fuzzy-match base objects, the match/search strategy defines the strategy that the match process uses for searching and matching records. The match/search strategy determines how to match candidate A with candidate B by using fuzzy or exact options.

Note: For more information about fuzzy-match base objects, see [“Match/Search Strategy” on page 380](#).

The match/search strategy can affect the quantity and quality of the match candidates. An exact strategy requires clean and complete data; if the data is not cleansed or if the data is incomplete, the match process might miss some matches. A fuzzy strategy finds many more matches, but many might not be duplicates. When defining match rule properties, you must find the optimal balance between finding all possible candidates and avoiding irrelevant candidates.

Select one of the following strategy options:

Match/Search Strategy Option	Description
Fuzzy strategy	A probabilistic match that takes into account spelling variations, possible misspellings, and other differences.
Exact strategy	An exact match that matches records that are identical.

All fuzzy-match base objects have a fuzzy match key, which is generated based on the columns that you specify in the match path component. When you use the fuzzy strategy, the match process searches the match key index to identify match candidates. To ensure that the match process can identify candidates, you must specify at least one of the columns from the fuzzy match key in the match rule. For example, if the match key is generated based on the Name column, then the match rule must include the Name column.

The following table lists the types of match rules, the match/search strategies, and a description for each combination:

Match Rule Type	Match/Search Strategy Used	Description
Fuzzy-match rule	Fuzzy strategy	Contains fuzzy-match columns and might contain exact-match columns. Fuzzy-match rules are recommended for columns that contain data variations, such as misspelled names.
Exact-match rule	Exact strategy	Contains only exact-match columns. The match process identifies matches by running SQL statements on the rows in the database. Exact-match rules are best for columns, such as ID and date of birth, where you do not need fuzzy matching capabilities.
Filtered-match rule	Fuzzy strategy	Contains only exact-match columns. The rule runs on match candidates that are filtered based on the associated match key. The match process identifies match candidates by searching the match key index. It is recommended to use a filtered-match rule when one of the exact-match columns has the same source columns as the fuzzy match key. Filtered-match rules improve performance by running the exact-match rules in the Process Server instead of the database.

Tip: If you are constrained by performance issues related to the database server, consider using filtered-match rules instead of exact-match rules. Filtered-match rules let you run batches larger than what you can run on the exact-match rules. Also, for a large increment in batch size on filtered matching, the duration of the match job comparatively increases by a small margin.

Match Purpose

For fuzzy-match base objects, the *match purpose* defines the primary goal behind a match rule. For example, if you're trying to identify matches for people where address is an important part of determining whether two records are for the same person, then you would choose the Match Purpose called Resident.

For every match rule you define, you must choose the purpose of the rule from a list of predefined match purposes provided by Informatica. Each match purpose contains knowledge about how best to compare two records to achieve the purpose of the match. Informatica MDM Hub uses the selected match purpose as a basis for applying the match rules to determine matched records. The behavior of the rules is dependent on the selected purpose. The list of available match purposes depends on the population used.

What the Match Purpose Determines

The match purpose determines:

- how your match rules behave
- which columns are required
- how much attention Informatica MDM Hub pays to each of the columns used in the match process

Two rules with all attributes identical (except for the purpose) will return different sets of matches because of the different purpose.

Mandatory and Optional Fields

Each match purpose supports a combination of mandatory and optional fields. Each field is weighted according to its influence in the match decision. Some fields in some purposes may be grouped. There are two types of groupings:

- Required—requires at least one of the field members to be non-null
- Best of—contributes only the best score from the fields in the group to the overall match score

For example, in the Individual match purpose:

- Person_Name is a mandatory field
- One of either ID Number or Date of Birth is required
- Other attributes are optional

The overall score returned by each purpose is calculated by adding the participating field scores multiplied by their respective weight and divided by the total of all field weights. If a field is optional and is not provided, it is not included in the weight calculation.

Name Formats

Informatica MDM Hub match has a default name format which tells it where to expect the last name. The options are:

- Left—last name is at the start of the full name, for example Smith Jim
- Right—last name is at the end of the full name, for example, Jim Smith

The name format that the Informatica MDM Hub uses depends on the purpose that you use. If you use Organization, the default is Last name, First name, Middle name. If you use Person or Resident, the default is First Middle Last.

Note: When you format data for matching, remember that the name format that the Informatica MDM Hub uses depends on the purpose that you use. In special cases, particularly for names that are not in the selected population, the name format and the match purpose that you select makes a difference.

List of Match Purposes

The following table describes match purposes that Informatica provides:

Match Purpose	Description
Person_Name	<p>Identifies a person by name. Use the purpose for online searches when a name-only lookup is required and a human is available to make the choice. Matching requires other attributes in addition to name to make match decisions. When you use this purpose, the rule must not contain address fields. This purpose matches people with an address and those without an address. If the rules contain address fields, use the Resident purpose instead.</p> <p>This purpose uses the following fields:</p> <ul style="list-style-type: none">- Person_Name (Required)- Address_Part1- Address_Part2- Postal_Area- Telephone_Number- ID- Date- Attribute1- Attribute2- Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, use Postal_Area as a repeat value in the Address_Part2 field.</p>
Individual	<p>Identifies an individual by the name, ID number, and date of birth attributes.</p> <p>Use the Individual match purpose after a search by Person_Name as this match purpose requires additional information that the Person_Name match purpose provides.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none">- Person_Name (Required)- ID (Required)- Date (Required)- Attribute1- Attribute2
Resident	<p>Identifies a person at an address. Use this match purpose after a search by either Person_Name or Address_Part1. Optional input fields help qualify or rank a match if more information is available.</p> <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none">- Person_Name (Required)- Address_Part1 (Required)- Address_Part2- Postal_Area- Telephone_Number- ID- Date- Attribute1- Attribute2- Geocode

Match Purpose	Description
Household	<p>Identifies matches where individuals with the same or similar family names share the same address.</p> <p>Use this match purpose after a search by Address_Part1.</p> <p>Note: A search by Person_Name is not practical because ultimately one word from the Person_Name must match, and a one-word search is not effective in most situations.</p> <p>Emphasis is on the Last Name, the major word of the Person_Name field, so this is one of the few cases where word order is important in the way the records are presented for matching.</p> <p>However, a reasonable score is generated, provided that a match occurs between the major word in one name and any other word in the other name.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Person_Name (Required) - Address_Part1 (Required) - Address_Part2 - Postal_Area - Telephone_Number - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>
Family	<p>Identifies matches where individuals with the same or similar family names share the same address or the same telephone number.</p> <p>Use this match purpose after a tiered search (multisearch) by Address_Part1 and Telephone_Number.</p> <p>Note: A search by Person_Name is not practical because ultimately one word from the Person_Name must match, and a one-word search is not effective in most situations.</p> <p>Emphasis is on the Last Name, the major word of the Person_Name field, so this is one of the few cases where word order is important in the way the records are presented for matching.</p> <p>However, a reasonable score is generated provided that a match occurs between the major word in one name and any other word in the other name.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Person_Name (Required) - Address_Part1 (Required) - Telephone_Number (Required) (Score is based on best of Address_Part_1 and Telephone_Number) - Address_Part2 - Postal_Area - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>

Match Purpose	Description
Wide_Household	<p>Identifies matches where the same address is shared by individuals with the same family name or with the same telephone number.</p> <p>Use this match purpose after a search by Address_Part1.</p> <p>Note: A search by Person_Name is not practical because ultimately one word from the Person_Name must match, and a one-word search is not effective in most situations.</p> <p>Emphasis is on the last name, the major word of the Person_Name field, so this is one of the few cases where word order is important in the way the records are presented for matching.</p> <p>However, a reasonable score is generated provided that a match occurs between the major word in one name and any other word in the other name.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Address_Part1 (Required) - Person_Name (Required) - Telephone_Number (Required) (Score is based on best of Person_Name and Telephone_Number) - Address_Part2 - Postal_Area - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>
Address	<p>Identifies an address match. The address might be postal, residential, delivery, descriptive, formal, or informal.</p> <p>The required field is Address_Part1. The fields Address_Part2, Postal_Area, Telephone_Number, ID, Date, Attribute1 and Attribute2 are optional input fields to further differentiate an address. For example, if the name of a City and/or State is provided as Address_Part2, it will help differentiate between a common street address [100 Main Street] in different locations.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Address_Part1 (Required) - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2. In that case, the Address_Part2 score used is the higher of the two scored fields.</p>

Match Purpose	Description
Organization	<p>Matches organizations primarily by name. It is targeted at online searches when a name-only lookup is required and a human is available to make the choice. Matching in batch requires other attributes in addition to name to make match decisions. Use this match purpose when the rule does not contain address fields. This match purpose allows matches between organizations with an address and those without an address. If the rules contain address fields, use the Division match purpose.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Organization_Name (Required) - Address_Part1 - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>
Division	<p>Identifies an Organization at an Address. Use this match purpose after a search by Organization_Name or by Address_Part1, or both.</p> <p>This match purpose is in essence the same match purpose as Organization, except that Address_Part1 is a required field. For example, this match purpose is designed to match company X at an address of Y or company Z at address W if multiple addresses are supplied.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Organization_Name (Required) - Address_Part1 (Required) - Address_Part2 - Postal_Area - Telephone_Number - ID - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>
Contact	<p>Identifies a contact within an organization at a specific location.</p> <p>Use the match purpose after a search by Person_Name. However, either Organization_Name or Address_Part1 can be used as the search criteria.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Person_Name (Required) - Organization_Name (Required) - Address_Part1 (Required) - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>

Match Purpose	Description
Corporate_Entity	<p>Identifies an Organization by its legal corporate name, including the legal endings such as INC and LTD. This match purpose is designed for applications that need to honor the differences between such names as ABC TRADING INC and ABC TRADING LTD.</p> <p>Use this match purpose after a search by Organization_Name. This match purpose is in essence the same match purpose as Organization, except that tighter matching is performed and legal endings are not treated as noise.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Organization_Name (Required) - Address_Part1 - Address_Part2 - Postal_Area - Telephone_Number - ID - Attribute1 - Attribute2 - Geocode <p>To achieve a best of score between Address_Part2 and Postal_Area, pass Postal_Area as a repeat value in the Address_Part2 field.</p>
Wide_Contact	<p>Loosely Identifies a contact within an organization without regard to location.</p> <p>Use this match purpose after a search by Person_Name.</p> <p>In addition to the required fields, ID, Attribute1 and Attribute2 can be provided optionally for matching to further qualify a contact.</p> <p>This match purpose uses the following fields:</p> <ul style="list-style-type: none"> - Person_Name (Required) - Organization_name (Required) - ID - Attribute1 - Attribute2
Fields	<p>Provided for general, nonspecific use. This match purpose has no required fields. All field types are available as optional input fields.</p>

Match Levels

For fuzzy-match base objects, the *match level* determines how precise the match is. You can specify one of the following match levels for a fuzzy-match base object:

Table 2. Match Levels

Level	Description
Typical	Appropriate for most matches.
Conservative	Produces fewer matches than the Typical level. Some data that actually matches may pass through the match process without being flagged as a match. This situation is called <i>undermatching</i> .
Loose	Produces more matches than the Typical level. Loose matching may produce a significant number of match candidates that are not really matches. This situation is called <i>overmatching</i> . You might choose to use this in a match rule for manual merges, to make sure that other, tighter match rules have not missed any potential matches.

Select the level based on your knowledge of the data to be matched: Typical, Conservative (fewer matches), or Looser (more matches). When in doubt, use Typical.

Geocode Radius

Use geocode radius as a match rule property for a fuzzy base object when you want to identify records that lie within the specified radius. The Geocode Radius field is enabled if you select the Geocode field name for the match column.

Specify the geocode radius in meters. The geocode radius is stored in the GEOCODE_RADIUS column in the C_REPOS_MATCH_RULE table. When records are close to the geocode radius that you specify, the match scores are high. Match scores decrease when records are located away from the geocode radius that you specify.

Accept Limit Adjustment

For fuzzy-match base objects, the *accept limit* is a number that determines the acceptability of a match. This setting does the exact same thing as the match level, but to a more granular degree. The accept limit is defined by Informatica within a population in accordance with its match purpose. The Accept Limit Adjustment allows a coarse adjustment to what is considered to be a match for this match rule.

- A positive adjustment results in more conservative matching.
- A negative adjustment results in looser matching.

For example, suppose that, for a given field and a given population, the accept limit for a typical match level is 80, for a loose match level is 70, and for a conservative match level is 90. If you specify a positive number (such as 3) for the adjustment, then the accept level becomes slightly more conservative. If you specify a negative number (such as -2), then the accept level becomes looser.

Configuring this setting provides a optional refinement to your match settings that might be helpful in certain circumstances. Adjusting the accept limit even a few points can have a dramatic effect on your matches, resulting in overmatching or undermatching. Therefore, it is recommended that you test different settings iteratively, with small increments, to determine the best setting for your data.

Match Column Properties for Match Rules

This section describes the match column properties that you can configure for match rules.

Match Subtype

For base objects containing different types of data, the *match subtype* option allows you to apply match rules to specific types of data within the same base object. You have the option to enable or disable match subtyping for *exact-match columns* that have parent/child path components. Match subtype is available only for:

- exact-match column types that are based on a non-root Path Component, and
- match rules that have a fuzzy match / search strategy

To use match subtyping, for each match rule, specify one or more *exact-match column(s)* that will serve as the “subtyping” column(s) to use. The subtype indicator can be set for any of the exact-match columns regardless of whether they are used for segment match or not. During the match process, evaluation of the subtype column precedes evaluation of the other match columns. Use match subtyping judiciously, because it can have a performance impact on the match process.

Match Subtype behaves just like a standard parent/child matching scenario with the additional requirement that the match column marked as Match Subtype must be the same across all records being matched. In the

following example, the Match Subtype column is Address Type and the match rule consists of Address Line1, City, and State.

Parent ID	Address Line 1	City	State	Address Type
3	123 Main	NYC	ON	Billing
3	50 John St	Toronto	NY	Shipping
5	123 Main	Toronto	BC	Billing
5	20 Adelaide St	Markham	AB	Shipping
5	50 John St	Ottawa	ON	Billing
7	50 John St	Barrie	BC	Billing
7	20 Adelaide St	Toronto	NB	Shipping
7	90 Yonge St	Toronto	ON	Billing

Without Match Subtype, Parent ID 3 would match with 5 and 7. With Match Subtype, however, Parent ID 3 will not match with 5 nor 7 because the matching rows are distributed between different Address Types. Parent ID 5 and 7 will match with each other, however, because the matching rows all fall within the 'Billing' Address Type.

RELATED TOPICS:

- [“Editing Match Column Properties for Exact-match Base Objects ” on page 400](#)

NULL Matching

Use NULL matching to specify how the match process handles null values that match other null values.

Note: Null matching and segment matching are mutually exclusive. NULL matching applies to exact-match columns.

By default, null matching is disabled and the MDM Hub treats nulls as unequal values when it searches for matches. When null matching is disabled, a null value does not match any value.

To enable null matching, you can select one of the following null matching options for the match columns:

- NULL Matches NULL
- NULL Matches Non-NULL

NULL Matches NULL

You can enable the **NULL Matches NULL** option to specify that the match process considers two null values as matches.

Based on the null matching scenario, one of the following value pairs is considered a match:

- Both cell values are null.
- Both cell values are identical.

NULL Matches Non-NULL

You can enable the **NULL Matches Non-NULL** option to specify that the match process considers null values as matches of any value that is not null.

Based on the null matching scenario, any of the following value pairs is considered a match:

- One cell value is null and the other cell value is not null.
- Both cell values are identical.

Important: When the NULL Matches Non-NULL option is enabled, Build Match Groups allows only a single null to non-null match into any group, which reduces the possibility of unwanted transitive matches. To ensure identical values are not dropped, create two exact match rules that are identical. In one exact match rule, enable the NULL Matches Non-NULL option. In the other exact match rule, disable the NULL Matches Non-NULL option. In the match rule execution sequence, ensure that the exact match rule with the NULL Matches Non-NULL option disabled precedes the exact match rule with the NULL Matches Non-NULL option enabled.

Null Matches Null When One Cell Value is Non-Null Example

Your organization has the CUSTOMER base object for which you perform a merge job. You enable the option to match null values with other null values for the exact match rule. The CUSTOMER base object contains three records for John Smith, two of which have null values for the phone extension.

The base object records for John Smith show that the phone extension for two of the three records is null.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	-
2	John Smith	6053128215	-
3	John Smith	6053128215	236

After a match and merge job, the base object shows that the records with the null values merge.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	-
3	John Smith	6053128215	236

After a match and merge job, the cross-reference table shows that the Rowid_Object of one of the records with the null value and the record with the non-null value survives.

Rowid_XREF	Rowid_Object	Person Name	Phone	Extension
1	1	John Smith	6053128215	-
2	1	John Smith	6053128215	-
3	3	John Smith	6053128215	236

Note: In the scenario, because a null value matches another null value, you might be unsure if the two records are a match. To ensure that the records with the null values are a match, you can configure the MDM Hub to flag them for a manual merge.

Null Matches Non-Null When One Cell Value is Null Example

Your organization has the CUSTOMER base object for which you perform a merge job. You enable the option to match null values with non-null values for the exact match rule. The CUSTOMER base object contains records for John Smith and one of the records has a null value for the phone extension.

The base object records for John Smith show that the phone extension is either 236 or null.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236
2	John Smith	6053128215	-

After a match and merge job, the base object shows that the record with the null value merges into the record with the non-null value.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236

After a match and merge job, the cross-reference table shows that the Rowid_Object of the record into which the record with the null value merges survives.

Rowid_XREF	Rowid_Object	Person Name	Phone	Extension
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	-

Note: In the scenario, because a null value matches another null value, you might be unsure if the two records are a match. To ensure that the records with null and non-null values are a match, you can configure the MDM Hub to flag them for a manual merge.

Null Matches Non-Null When Both Cell Values are Identical Example

Your organization has the CUSTOMER base object for which you perform the match and merge job. You enable the option to match null values with non-null values for the exact match rule. The CUSTOMER base object contains records for John Smith. Both the records have identical values for the phone extension.

The base object records for John Smith show that the phone extension is 236.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236
2	John Smith	6053128215	236

After a match and merge job, the base object shows one record.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236

After a match and merge job, the record with Rowid_Object 2 that contains a value identical to the record with Rowid_Object 1 merges into the record with Rowid_Object 1.

After a match and merge job, the cross-reference table shows that the Rowid_Object of the record into which the record with the identical value merges survives.

Rowid_XREF	Rowid_Object	Person Name	Phone	Extension
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	236

The records with the identical value that merges into the another record with an identical value has the Rowid_Object of the record into which it merges.

Null Matches Non-Null When One Cell Value is Null and Another is Identical Example

Your organization has the CUSTOMER base object for which you perform the match and merge job. You enable the option to match null values with non-null values for the exact match rule. The CUSTOMER base object contains records for John Smith. One of the records has a null value for the phone extension while the other record has an identical value.

The base object records for John Smith show that the phone extension is either 236 or null.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236
2	John Smith	6053128215	-
3	John Smith	6053128215	236

After a match and merge job, the base object shows two records.

Rowid_Object	Person Name	Phone	Extension
1	John Smith	6053128215	236
3	John Smith	6053128215	236

After a match and merge job, the record with Rowid_Object 2 merges into the record with Rowid_Object 1 because Build Match Groups allows only a single null to non-null match into any group. The record with Rowid_Object 3 that contains a value identical to the value in the record with Rowid_Object 1 does not merge.

After a match and merge job, the cross-reference table shows that the Rowid_Object of the record into which the record with the null value merges survives.

Rowid_XREF	Rowid_Object	Person Name	Phone	Extension
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	-
3	3	John Smith	6053128215	236

The record with the null value that merges into a record with a non-null value has the Rowid_Object of the record it merges into. The Rowid_Object value of the record that does not merge remains the same.

Note: In the scenario, because a null value matches another non-null value, you might be unsure if the two records are a match. To ensure that the records with null and non-null values are a match, you can configure the MDM Hub to flag them for a manual merge. Also, to ensure that the identical records with Rowid_Object 1 and Rowid_Object 3 are matches, create two exact match rules that are identical. In one exact match rule, enable the NULL Matches Non-NULL option. In the other exact match rule, disable the NULL Matches Non-NULL option. In the match rule execution sequence, ensure that the exact match rule with the NULL Matches Non-NULL option disabled precedes the exact match rule with the NULL Matches Non-NULL option enabled.

Non-equal Matching

Use the Non-equal Matching option in match rules to prevent equal values in a column from matching each other. Non-equal Matching applies only to exact-match columns.

Note: Non-equal Matching and Segment Matching are mutually exclusive. If one is selected, then the other cannot be selected.

You can think about the Non-equal Matching option as an anti-match option. The match result when the option is enabled is the opposite of the match result when the option is disabled.

You can use Non-equal Matching to match a record with a fuzzy organization name column and an exact, non-equal organization type column. Even though organization names are identical, the records will match only if the organization types are not identical.

Non-equal Matching without NULL Matching

First consider the effect of the Non-equal Matching option when NULL matching is disabled. As you can see from the following table, the NULL values never match, and the equal values match. With the Non-equal Matching option is enabled, the opposite is true: all the NULL values match and equal values do not match.

The following table shows simple match results before and after Non-equal Matching is enabled:

Values	Non-equal Matching=False	Non-equal Matching=True
- Record 1 = NULL - Record 2 = "Fred"	No match	Match
- Record 1 = NULL - Record 2 = NULL	No match	Match
- Record 1 = "Bill" - Record 2 = "Fred"	No match	Match
- Record 1 = "Fred" - Record 2 = "Fred"	Match	No match

Non-equal Matching with NULL Matches NULL

Similarly, when you enable the Non-equal Matching option with the NULL Matches NULL option, the match results switch to the opposite results.

The following table shows the match results before and after Non-equal Matching is enabled with NULL Matches NULL:

Values	Non-equal Matching=False NULL Matches NULL=True	Non-equal Matching=True NULL Matches NULL=True
- Record 1 = NULL - Record 2 = "Fred"	No match	Match
- Record 1 = NULL - Record 2 = NULL	Match	No match
- Record 1 = "Bill" - Record 2 = "Fred"	No match	Match
- Record 1 = "Fred" - Record 2 = "Fred"	Match	No match

Non-equal Matching with NULL Matches Non-NULL

When you enable the Non-equal Matching option with the NULL Matches Non-NULL option, the match result switches to the opposite result. In one circumstance, the records with NULL values do not match. See the explanation below the table.

The following table shows the match results before and after Non-equal Matching is enabled with NULL Matches Non-NULL:

Values	Non-equal Matching=False NULL Matches Non-NULL=True	Non-equal Matching=True NULL Matches Non-NULL=True
- Record 1 = NULL - Record 2 = "Fred"	Match	No match
- Record 1 = NULL - Record 2 = NULL	No match	Match*
- Record 1 = "Bill" - Record 2 = "Fred"	No match	Match
- Record 1 = "Fred" - Record 2 = "Fred"	Match	No match

* When a match rule uses Filtered Exact, the result is a match. However, when a match rule uses Exact, the result is no match.

Segment Matching

Note: Segment Matching and Non-equal Matching are mutually exclusive. If one is selected, then the other cannot be selected. Segment Matching and NULL Matching are also mutually exclusive. If one is selected, then the other cannot be selected.

For exact-match columns only, you can use *segment matching* to limit match rules to specific subsets of data. For example, you could define different match rules for customers in different countries by using segment matching to limit certain rules to specific country codes. Segment matching applies to both exact-match and fuzzy-match base objects.

If the Segment Matching check box is checked (selected), you can configure two other options: Segment Matches All Data and Segment Match Values.

Segment Matches All Data

When unchecked (the default), Informatica MDM Hub will only match records within the set of values defined in Segment Match Values.

For example, suppose a base object contained Leads, Partners, Customers, and Suppliers. If Segment Match Values contained the values Leads and Partners, and Segment Matches All Data were unchecked, then Informatica MDM Hub would only match within records that contain Leads or Partners. All Customers and Suppliers records will be ignored.

With Segment Matches All Data checked (selected), then Leads and Partners would match with Customers and Suppliers, but Customers and Suppliers would not match with each other.

Concatenation of Values in Multiple Columns

For exact matches with segment matching enabled on concatenated columns, a space character must be added to each piece of data present in the concatenated fields.

Note: Concatenating columns is not recommended for exact match columns.

Segment Match Values

For segment matching, specifies the list of segment values to use for segment matching.

You must specify one or more values (for a match column) that defines the segment matching. For example, for a given match rule, suppose you wanted to define segment matching by Gender. If you specified a segment match value of M (for male), then, for that match rule, Informatica MDM Hub searches for matches (based on the other match columns) only on male records—and can only match to other male records, unless you also enabled Segment Matches All Data.

Note: Segment match values are case-sensitive. When using segment matching on fuzzy and exact base objects, the values that you set are case-sensitive when executing the Match batch job.






Requirements for Exact-match Columns in Match Column Rules



Exact-match columns are subject to the following rules:

- The names of exact-match columns cannot be longer than 26 characters.
- Exact-match columns must be of type VARCHAR or CHAR.
- Match columns can be used to match on any text column or combination of text columns from a base object.
- If you want to use numerics or dates, you must convert them to VARCHAR using cleanse functions before they are loaded into your base object.
- Match columns can also be used to match on a column from a child base object, which in turn can be based on any text column or combination of text columns in the child base object. Matching on the match columns of a child base object is called intertable matching.
- When using intertable match and creating match rules for the child table (via a foreign key), you must include the foreign key from the parent table in each match rule on the child. If you do not, when the child is merged, the parent records would lose the child records that had previously belonged to them.

Command Buttons for Configuring Column Match Rules

In the Match Rule Sets tab, if you select a match rule set in the list, the Schema Manager displays the following command buttons.

Button	Description
	Adds a match rule.
	Edits properties for the selected a match rule.
	Deletes the selected match rule.
	Moves the selected match rule up in the sequence.
	Moves the selected match rule down in the sequence.

Button	Description
	Changes a manual consolidation rule to an automatic consolidation rule. Select a manual consolidation record and then click the button.
	Changes an automatic consolidation rule to a manual consolidation rule. Select an automatic consolidation record and then click the button.

Note: If you change your match rules after matching, you are prompted to reset your matches. When you reset your matches, it deletes everything in the match table and, in records where the consolidation indicator is 2, resets the consolidation indicator to 4.

Adding Match Column Rules

You can add fuzzy, exact, or filtered match rules.

To add a new exact or fuzzy match rule using match columns:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Select a match rule set in the list.

The Schema Manager displays the properties for the selected match rule set.

5. In the Match Rules section of the screen, click the **Add** button.

The Schema Manager displays the Edit Match Rule dialog. This dialog differs slightly between exact match and fuzzy-match base objects.

6. For fuzzy-match base objects, configure the match rule properties at the top of the dialog box.
7. Configure the match columns for the match rule.

- a. Click the **Edit** button next to the Match Columns list.

The Add/Remove Match Columns dialog box appears and displays only the match columns that you defined. For exact-match base objects or match rules with an exact match/search strategy, only exact column types are available. For fuzzy-match base objects, you can select fuzzy or exact column types.

- b. Select the fields that you want to add to the match column rule.
- c. Click **OK**.

The Schema Manager displays the selected fields in the Match Columns list.

8. Configure the match properties for each match column in the Match Columns list.
9. Click **OK**.
10. If this is an exact match, specify the match properties for this match rule. Click **OK**.
11. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains any incongruousness.

12. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Configuring Filtered Match Rules

Filtered match rules can be configured for fuzzy-match base objects that include exact match columns.

You must have already added exact match rules for the fuzzy-match base object for which you must configure a filtered match rule.

To add a filtered match rule using exact match columns:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the fuzzy-match base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Choose a match rule of an Exact Match Rule Type and click the **Edit** button.

The Edit Match Rule dialog appears.

Note: A fuzzy match rule cannot be changed to a filtered match rule directly.

5. From the Match/Search Strategy drop-down list, choose **Fuzzy** and click **OK**.

The Match Rule type changes to Filtered.

6. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains any incongruousness.

7. If you are prompted to confirm saving changes, click the **OK** button to save the changes.

Editing Match Column Rules

To edit the properties for an existing match rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the exact-match base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Select a match rule set in the list.

The Schema Manager displays the properties for the selected match rule set.

5. In the Match Rules section of the screen, click the **Edit** button.

The Schema Manager displays the Edit Match Rule dialog. This dialog differs slightly between exact match and fuzzy-match base objects.

6. For fuzzy-match base objects, change the match rule properties at the top of the dialog box, if you want.
7. Configure the match column(s) for this match rule, if you want.

Only columns you have previously defined as match columns are shown.

- For exact-match base objects or match rules with an exact match / search strategy, only exact column types are available.
- For fuzzy-match base objects, you can choose fuzzy or exact columns types.

- a. Click the **Edit** button next to the Match Columns list.

The Schema Manager displays the Add/Remove Match Columns dialog.

- b. Check (select) the check box next to any column that you want to include
- c. Uncheck (clear) the check box next to any column that you want to omit.

- d. Click **OK**.

The Schema Manager displays the selected columns in the Match Columns list.

8. Change the match properties for any match column that you want to edit.
9. Click **OK**.
10. If this is an exact match, specify the match properties for this match rule. Click **OK**.
11. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruities.

12. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Deleting Match Column Rules

To delete a match column rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the exact-match base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Select a match rule set in the list.
5. In the Match Rules section, select the match rule that you want to delete.
6. Click the **Delete** button.

The Schema Manager prompts you to confirm deletion.

7. Click **Yes**.

Changing the Execution Sequence of Match Column Rules

To change the execution sequence of match column rules:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the exact-match base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Select a match rule set in the list.
5. In the Match Rules section, select the match rule that you want to move up or down.
6. Do one of the following:
 - Click the **Up** button to move the selected match rule up in the execution sequence.
 - Click the **Down** button to move the selected match rule down in the execution sequence.
7. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruities.

8. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Specifying Consolidation Options for Match Column Rules

During the match process, a match column rule must determine whether matched records should be queued for manual or automatic consolidation.

Note: A base object cannot have more than 200 user-defined columns if it will have match rules that are configured for automatic consolidation.

To toggle between manual and automatic consolidation for a match rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the exact-match base object that you want to configure.
2. Acquire a write lock.
3. Click the **Match Rule Sets** tab.
4. Select a match rule set in the list.
5. In the Match Rules section, select the match rule that you want to configure.
6. Do one of the following:
 - Click the **Up** button to change a manual consolidation rule to an automatic consolidation rule.
 - Click the **Down** button to change an automatic consolidation rule to a manual consolidation rule.
7. Click the **Save** button to save your changes.


Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruences.
8. If you are prompted to confirm saving changes, click the **OK** button to save your changes.

Configuring the Match Weight of a Column

For a fuzzy-match column, you can change its match weight in the Edit Match Rule dialog box. For each column, Informatica MDM Hub assigns an internal *match weight*, which is a number that indicates the importance of this column (relative to other columns in the table) for matching. The match weight varies according to the selected match purpose and population. For example, if the match purpose is Person_Name, then Informatica MDM Hub, when evaluating matches, views a data match in the name column with greater importance than a data match in a different column (such as the address).

By adjusting the match weight of a column, you give added weight to, and elevate the significance of, that column (relative to other columns) when Informatica MDM Hub analyzes values for matches.

To configure the match weight of a column:

1. In the Edit Match Rule dialog box, select a column in the list.
2. Click the  **Match Weight Adjustment** button.

If adjusted, the name of the selected column shows in a bold font.
3. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruences.
4. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Configuring Segment Matching for a Column

Segment matching is used with exact-match columns to limit match rules to specific subsets of data.

To configure segment matching for an exact-match column:

1. In the Edit Match Rule dialog box, select an exact-match column in the Match Columns list.
2. Check (select) the **Segment Matching** check box to enable this feature.
3. Check (select) the **Segment Matches All Data** check box, if you want.
4. Specify the segment match values for segment matching.
 - a. Click the **Edit** button.

The Schema Manager displays the Edit Values dialog.
 - b. Do one of the following:
 - To add a value, click the **Add** button, type the value you want to add, and click **OK**.
 - To delete a value, select it in the list, click the **Delete** button, and choose **Yes** when prompted to confirm deletion.
5. Click **OK**.
6. Click the **Save** button to save your changes.

Before saving changes, the Schema Manager analyzes the match rule set and prompts you with a message if the match rule set contains certain incongruences.
7. If you are prompted to confirm saving changes, click **OK** button to save your changes.

Configuring Primary Key Match Rules

This section describes how to configure primary key match rules for your Informatica MDM Hub implementation.

If you want to configure match column match rules instead, see the instructions in [“Configuring Match Columns” on page 392](#).

About Primary Key Match Rules

Matching on primary keys can be used when two or more different source systems for a base object have identical primary key values.

This situation occurs infrequently in source systems, but when it does occur, you can make use of the primary key matching option in Informatica MDM Hub to rapidly match and automatically consolidate records from the source systems that have the matching primary keys.

For example, two systems might use the same set of customer IDs. If both systems provide information about customer XYZ123 using identical primary key values, the two systems are certainly referring to the same customer and the records should be automatically consolidated.

When you specify a primary key match, you simply specify which source systems have the same primary key values. You also check the **Auto-merge matching records** check box to have Informatica MDM Hub automatically consolidate matching records when a Merge or Link batch job is run.

Note: After you specify the primary key match rules, you must run the Key Match batch job.

Adding Primary Key Match Rules

To add a new primary key match rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Primary Key Match Rules** tab.

The Schema Manager displays the Primary Key Match Rules tab.

The Primary Key Match Rules tab has the following columns.

Column	Description
Key Combination	Two source systems for which this primary match key rule will be used for matching. These source systems must already be defined in Informatica MDM Hub, and staging tables for this base object must be associated with these source systems.
Auto-Merge	Specifies whether this primary key match rule results in automatic or manual consolidation.

4. Click the **Add** button to add a primary match key rule.
The Add Primary Key Match Rule dialog is displayed.
5. Check (select) the check box next to two source systems for which you want to match records based on the primary key.
6. Check (select) the **Auto-merge matching records** check box if you are certain that records with identical primary keys are matches.
You can change your choice for **Auto-merge matching records** later, if you want.
7. Click **OK**.
The Schema Manager displays the new rule in the **Primary Key Rule** tab.
8. Click the **Save** button to save your changes.
The Schema Manager asks you whether you want to reset existing matches.
9. Choose **Yes** to delete all matches currently stored in the match table, if you want.

Editing Primary Key Match Rules

Once you have defined a primary key match rule, you can change the value of the **Auto-merge matching records** check box.

To edit an existing primary key match rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Primary Key Match Rules** tab.
The Schema Manager displays the **Primary Key Match Rules** tab.
4. Scroll to the primary key match rule that you want to edit.
5. Check or uncheck the **Auto-merge matching records** check box to enable or disable auto-merging, respectively.
6. Click the **Save** button to save your changes.

The Schema Manager asks you whether you want to reset existing matches.

7. Choose **Yes** to delete all matches currently stored in the match table, if you want.

Deleting Primary Key Match Rules

To delete an existing primary key match rule:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Primary Key Match Rules** tab.

The Schema Manager displays the Primary Key Match Rules tab.

4. Select the primary key match rule that you want to delete.
5. Click the **Delete** button.

The Schema Manager prompts you to confirm deletion.

6. Choose **Yes**.

The Schema Manager removes the deleted rule from the Primary Key Match Rules tab.

7. Click the **Save** button to save your changes.

The Schema Manager asks you whether you want to reset existing matches.

8. Choose **Yes** to delete all matches currently stored in your Match table, if you want.

Investigating the Distribution of Match Keys

This section describes how to investigate the distribution of match keys in the match key table.

About Match Keys Distribution

Match keys are strings that encode data in the fuzzy match key column used to identify candidates for matching.

The tokenize process generates match keys for all the records in a base object and stores them in its match key table. Depending on the nature of the data in the base object record, the tokenize process generates at least one match key—and possibly multiple match keys—for each base object record. Match keys are used subsequently in the match process to help determine possible matches between base object records.

In the Match / Merge Setup Details pane of the Schema Manager, the Match Keys Distribution tab allows you to investigate the distribution of match keys in the match key table. This tool can assist you with identifying potential *hot spots* in your data—high concentrations of match keys that could result in *overmatching*—where the match process generates too many matches, including matches that are not relevant. By knowing where hot spots occur in your data, you can refine data cleansing and match rules to reduce hot spots and generate an optimal distribution of match keys for use in the match process. Ideally, you want to have a relatively even distribution across all keys.

Navigating to the Match Keys Distribution Tab

To navigate to the Match Keys Distribution tab:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Click the **Match Keys Distribution** tab.

The Schema Manager displays the Match Keys Distribution tab.

Components of the Match Keys Distribution Tab

The Match Key Distribution tab displays a histogram, match keys, and match columns.

Histogram

The histogram displays the statistical distribution of match keys in the match key table.

Axis	Description
Key (X-axis)	Starting character(s) of the match key. If no filter is applied (the default), this is the starting character of the match key. If a filter is applied, this is the starting sequence of characters in the match key, beginning with the left-most character.
Count (Y-axis)	Number of match keys in the match key table that begins with the starting character(s). Hotspots in the match key table show up as disproportionately tall spikes (high number of match keys), relative to other characters in the histogram.

Match Keys List

The Match Keys List on the Match Keys Distribution tab displays records in the match key table.




For each record, it displays cell data for the following columns:

Column Name	Description
ROWID	ROWID_OBJECT that uniquely identifies the record in the base object that is associated with this match key.
KEY	Generated match key. SSA_KEY column in the match key table.

Depending on the configured match rules and the nature of the data in a record, a single record in the base object table can have multiple generated match keys.

Paging Through Records in the Match Key Table

Use the following command buttons to navigate the records in the match key table.

Button	Description
	Displays the first page of records in the match key table.
	Displays the previous page of records in the match key table.
	Displays the next page of records in the match key table.
<input type="text" value="1"/>	Jumps to the page number you enter.

Match Columns

The Match Columns area on the Match Keys Distribution tab displays match column data for the selected record in the match keys list.

This is the SSA_DATA column in the match key table. For each match column that is configured for this base object, it displays the column name and cell data.

Filtering Match Keys

You can use a match key filter to focus your investigation on hotspots or other match key distribution patterns.

A match key filter restricts the data in the Histogram and the Match Keys List to the subset of match keys that meets the filter condition. By default, no filter is defined—all records in the match key table are displayed.

The filter condition specifies the beginning string sequence for qualified match keys, evaluated from left to right. For example, to view only match keys beginning with the letter M, you would select M for the filter. To further restrict match keys and view data for only the match keys that start with the letters MD you would add the letter D to the filter. The longer the filter expression, the more restrictive the display.

Setting a Filter



To set a filter:

1. Click the vertical bar in the Histogram associated with the character you want to add to the filter.
2. Click the vertical bar above a character along the X-axis, the Histogram refreshes and displays the distribution for all match keys beginning with that character.

The Match Keys list displays only those match keys that match the filter condition.

Navigating Filters

Use the following command buttons to navigate filters.

Button	Description
	Clears the filter. Displays the default view (no filter).
	Displays the previously-selected filter (removes the right-most character from the filter).

Excluding Records from the Match Process

Informatica MDM Hub provides a mechanism for selectively excluding records from the match process. You might want to do this if, for example, your data contained records that you wanted the match process to ignore.

To configure this feature, in the Schema Manager, you add a column named `EXCLUDE_FROM_MATCH` to a base object. This column must be an integer type with a default value of zero (0).

Once the table is populated and before running the Match job, to exclude a record from matching, change its value in the `EXCLUDE_FROM_MATCH` column to a one (1) in the Data Manager. When the Match job runs, only those records with an `EXCLUDE_FROM_MATCH` value of zero (0) will be tokenized and processed—all other records will be ignored.

Excluding records from the match process is available for:

- Both exact-match base objects and fuzzy-match base objects
- Match column rules only (not primary key match rules) that do not match for duplicates

Proximity Search

You can search for records that are in the proximity of the geocode radius that you specify. You can perform a proximity search on base objects for which columns are populated with latitude, longitude, and optionally elevation.

Ensure that the base object you select has a single column or multiple columns for geographic coordinates such as latitude, longitude, and elevation. The geographic coordinates must be in the signed degrees format and can be in different columns or in a single column separated by a comma or a space. For example, you can have a single column called Geocode with all the geographic coordinates `-23.399437, -52.090904, 203`. Alternatively, you can have 3 columns, latitude with the value `-23.399437`, longitude with the value `-52.090904`, and elevation with the value `203`.

When you specify a geocode radius such as 1000 meters, you can search for records that are within 1000 meters. To use proximity search you need another fuzzy match column such as `Person_Name` that the MDM Hub uses to generate match keys. The MDM Hub groups records based on these match keys. The MDM Hub then uses the match rule that includes the geocode radius to match records that are within 1000 meters of each other in each of the matched groups.

Configuring Proximity Search

To configure proximity search, you must configure the Geocode column and one or more other fuzzy match columns. The MDM Hub generates the match keys for proximity search based on the fuzzy match column other than the Geocode column that you configure.

1. Select the Schema tool and acquire a write lock.
The Schema Navigator appears.
2. Expand the base object for which you want to configure proximity search, and select Match/Merge Setup.
The **Match/Merge Setup Details** pane appears.
3. Click the **Properties** tab and configure the match properties for the base object.
Ensure that you set the value of the **Match/Search Strategy** property to Fuzzy.
4. Click the **Paths** tab and configure match paths if related records exist.
5. Click the **Match Columns** tab and configure match columns for match rules.
 - a. In the Fuzzy Match Key section, configure the fuzzy match key properties.
Ensure that you select a key type from the **Key Type** list, based on which you want match keys generated.
A fuzzy match column is added to the Match Columns section.
 - b. Select and move source columns for the fuzzy match column to the Selected Columns list.
 - c. Click **Add Fuzzy Match Column**.
The **Add Fuzzy Match Column** dialog box appears.
 - d. From the **Field Name** list, select Geocode.
 - e. Move the base object columns that contain geographic coordinate data to the Selected Columns list to create a match column for proximity search.
If all the geographic coordinate data is in a single column, move that column to the Selected Columns list.
 - f. Click **OK**.
6. Click the **Match Rule Sets** tab and configure a match rule set.
 - a. Click **Add** in the Match Rule Set section.
The **Add Match Rule Set** dialog box appears.
 - b. Enter a name for the match rule set, and click **OK**.
The match rule set appears in the Match Rule Set section.
 - c. Click **Add** in the Match Rules section.
The **Edit Match Rule** dialog box appears.
 - d. Click **Edit**.
The **Add/Remove Match Columns** dialog box appears.
 - e. Select **Geocode** and the fuzzy match column based on which you want match keys generated.
 - f. Click **OK**.
The **Edit Match Rule** dialog box appears.
 - g. Set match rule properties including **Geocode Radius**, and click **OK**.
The match rule appears in the Match rules section.

You can now use a batch job or a Services Integration Framework API to perform a proximity search.

Lightweight Matching

If you want to improve match performance, configure lightweight matching. Lightweight matching generates extremely fast score estimates. The algorithm rejects candidates that contain obvious mismatches instead of passing them to full scoring. On a typical data set, the algorithm rejects greater than 99% of the candidates, which results in improved performance.

Tip: If you already use tight exact-match rules on columns that contain values like identification numbers or date of birth, lightweight matching might not provide significant performance improvement.

SSA-NAME3 rejects the candidates based on the lightweight matching score. It is possible for SSA-NAME3 to reject the candidates that might have matched high with the SSA-NAME3 scoring. You can mitigate this risk by carefully selecting the fields to which you apply lightweight matching and by using threshold tuning.

Use the LWM_FIELDS control to apply lightweight matching to the fields. Use the LWM_LIMIT control to set the reject and accept limits for the lightweight matching score.

Configuring Lightweight Matching

Configure lightweight matching by setting properties in the `cmxcleanse.properties` file:

For example, the following example enables lightweight matching on the specified fields.

```
cmx.server.match.lwm=true
cmx.server.match.lwm_param=LWM_FIELDS=Organization_Name,50,Address_Part1,50
LWM_LIMIT=75,85
cmx.server.match.stats=false
```

The following property definitions describe how to use the lightweight matching properties together:

cmx.server.match.lwm

Optional. Must be added manually. Controls the lightweight matching feature. To enable the lightweight matching feature with full scoring on the matching records, set to `Y`. To enable the lightweight matching feature without full scoring on the matching records, set to `ONLY`. Default is `N`.

Use this property with the `cmx.server.match.lwm_param` and `cmx.server.match.stats` properties.

cmx.server.match.lwm_param

Optional. Must be added manually. Requires that the `cmx.server.match.lwm` property is set to `Y` or `ONLY`. Set the property value to the SSA-NAME3 controls in the following format:

```
LWM=Y LWM_FIELDS=<field1>,<weight1>[,...,<fieldn>,<weightn>]
LWM_LIMIT=<Reject>[,<Accept>]
```

cmx.server.match.stats

Optional. Must be added manually. Requires that the `cmx.server.match.lwm` property is set to `Y` or `ONLY`.

SSA-NAME3 Controls

You can add the following SSA-NAME3 controls within the `cmx.server.match.lwm_param` property. Separate the controls with a space.

LWM=Y/N/ONLY

Enables or disables lightweight matching. Use the value `Y` to enable lightweight matching. Lightweight matching uses a fast score estimate to reject the obvious mismatches. The records that lightweight matching passes go to the full scoring for robust scoring and ranking. SSA-NAME3 returns the full score and the decision to the caller.

Note: If you create system definition files by using the SDF Wizard, the lightweight matching is enabled by default.

Use the value **N** to disable lightweight matching. SSA-NAME3 matching performs full scoring on all the matching records.

Use the value **ONLY** to enable lightweight matching and disable full scoring. Lightweight matching returns the estimate as the final score to the caller.

LWM_FIELDS

Specifies the fields to which you want to apply lightweight matching and their weights. These values override the values that you have defined in the match purpose during the run time. Based on the lightweight matching scores, SSA-NAME3 rejects the obvious mismatches. If you do not set any value, SSA-NAME3 retrieves the fields from the match purpose and assigns equal weight to them.

The syntax of the LWM_FIELDS control is as follows:

```
LWM_FIELDS=<field1>,<weight1>[,...,<fieldn>,<weightn>]
```

where **field** is a valid field name that you have defined in the Purpose control, and **weight** is the relative significance of the specified field (0-100) when compared to the other fields.

For example, `LWM_FIELDS=Person_Name,5,Address_Part1,1`

Lightweight matching is useful when you apply it to the fields that have low variations such as addresses. Lightweight matching is not efficient for the fields with high variations, where SSA-NAME3 handles the variations through Edit-list, and lightweight matching might incorrectly reject the records.

LWM_LIMIT

Specifies the accept and reject limits for the lightweight matching score. Based on the limits, SSA-NAME3 accepts or rejects the search results.

The syntax of the LWM_LIMIT control is as follows:

```
LWM_LIMIT=<Reject>[,<Accept>]
```

where **Reject** and **Accept** are the integer values ranging from 0 through 100.

For example, `LWM_LIMIT=50,90`

If **LWM=N**, the **LWM_LIMIT** control has no effect.

If **LWM=Y**, SSA-NAME3 rejects the lightweight matching scores that are less than the reject limit. The accept limit has no effect, and you can omit it.

If **LWM=ONLY**, SSA-NAME3 rejects the lightweight matching scores that are less than the reject limit. It accepts the scores that are greater than the accept limit. It marks the scores of the records that are greater than or equal to the reject limit and less than the accept limit as undecided.

The default reject limit is 65, and the default accept limit is 90. If you have not set the accept limit and the reject limit is greater than 90, the accept limit is equal to the reject limit.

CHAPTER 22

Match Rule Configuration Example

This chapter includes the following topics:

- [Match Rule Configuration Example Overview, 435](#)
- [Match Rule Configuration Scenario, 436](#)
- [Configuring Match Rules, 437](#)
- [Step 1. Review the Data, 437](#)
- [Step 2. Identify the Base Objects for the Match Job, 438](#)
- [Step 3. Configure Match Properties, 438](#)
- [Step 4. Define the Match Path, 439](#)
- [Step 5. Define Match Columns, 444](#)
- [Step 6. Define a Match Rule Set, 449](#)
- [Step 7. Add Match Rules, 451](#)
- [Step 8. Set Merge Options for Match Rules, 454](#)
- [Step 9. Review the Match Properties, 455](#)
- [Step 10. Test the Match Rules, 457](#)

Match Rule Configuration Example Overview

The match rules configuration example shows you how to configure match rules to match and merge duplicate records in the MDM Hub. The match rules configuration example is based on the data available in the MDM Hub sample Operational Reference Store (ORS) that is included in the MDM Hub Resource Kit.

To determine records that are duplicates, that is, matches of each other, the MDM Hub uses match rules. You configure match rules for records in base objects. You can define a match rule for exact or fuzzy matching. An exact match rule matches records that have identical values in match columns. A fuzzy match rule matches similar records based on probabilistic match determinations that consider likely variations in data patterns, such as misspellings, transpositions, omissions, and phonetic variations. The match rules that you configure depend on the characteristics of the data, and the particular match and merge requirements.

The match rules configuration example walks you through the configuration process in a step-by-step manner. The example is based on the Party base object data that is available in the MDM Hub sample ORS. The example shows you how to configure match rules for data of individuals in the Party base object to match and merge data based on names and addresses. To view the preconfigured match rules that are

similar to the match rules described in the example, set up the MDM Hub sample ORS. For more information about the sample Operational Reference Store, see the *Multidomain MDM Sample ORS Guide*.

Match Rule Configuration Scenario

Your organization has customer information stored in the Party base object and other related base objects. The Party base object contains duplicate records for many customers. You want to configure match rules that can identify and queue up duplicate records for merge.

The Party base object contains duplicate customer records that might have the first name and the last name missing, or with incorrect or inconsistent entries. The Display Name column has values for each record. The records are categorized as a Person or an Organization party type. The addresses of the customers are stored in the related Address base object. The Party Address Rel relationship base object defines the relationship between the records in the Party and the Address base objects.

You want to create match rules to match duplicate customer records that have the Person party type in the Party base object. Records with the Person party type belong to individuals. You can perform the match based on names and addresses stored in the Party and Address base objects.

To match duplicate records, you need to create an automerger match rule for records that are sufficiently similar and can be queued for automerger. Also, you need to create a manual merge match rule for records that are possible duplicates, but need to be reviewed by a data steward before the merge process.

The Party base object shows important columns and sample records.

Rowid	Object	First Name	Last Name	Organization Name	Display Name	Party Type
1019		NULL	NULL	NULL	WILL R DE HAAN	Person
1072		NULL	NULL	NULL	AHMED RAUF	Person
1106		RACHEL	ARSEN	NULL	RACHEL ARSEN	Person
1154		RACHEL	ARSEN	NULL	RACHEL ARSEN	Person
1191		WILLIAM	DE HAAN	NULL	WILLIAM DE HAAN	Person
1419		BILL	DE HAAN	NULL	BILL ROGER DE HAAN	Person
1475		NULL	NULL	RYERSON AREA	RYERSON AREA MED CTR	Organization
1642		AHMED	RAUF	NULL	AHMED RAUF	Person
1800		NULL	NULL	NULL	RYERSON AREA MED CTR	Organization

The Address base object shows important columns and sample records.

Rowid	Object	Address Line1	Address Line2	City Name	State Cd	Postal Cd
920		69 BUTLER ST	NULL	ATLANTA	NULL	30303
991		7610 ROSENWALD LN	NULL	NOKESVILLE	NULL	20181
1221		RR 1 BOX 4	NULL	SUGAR RUN	NULL	18846-9701
1279		RR 1 BOX 3	NULL	SUGAR RUN	NULL	18846-9701
1711		69 JESSE HILL JR DR SE	NULL	ATLANTA	NULL	30303-3033
1860		5493 S QUEENS RD	NULL	ROCHELLE	NULL	61068
1909		5193 S QUEENS RD	NULL	ROCHELLE	NULL	61068
1960		669 BUTLER ST SE	NULL	ATLANTA	NULL	30303-3033
2005		7601 ROSENWALD LN	NULL	NOKESVILLE	NULL	20181

The LU Address Type lookup base object shows important columns and sample address type lookups.

Rowid	Object	Address Type	Address Type Disp	Address Type Desc
1		BILL	BILLING	BILLING
4		LGL	LEGAL	LEGAL
10		RSID	RESIDENCE	RESIDENCE
11		SHIP	SHIPPING	SHIPPING

The Party Address Rel relationship base object shows important columns and sample records.

Rowid	Object	Party ID	Address ID	Address Type	Status CD
976		1019	920	BILL	NULL
1051		1072	991	BILL	NULL
1080		1191	1960	LGL	NULL
1100		1419	1711	BILL	NULL
2001		1154	1909	LGL	NULL
2002		1106	1860	LGL	NULL
2004		1642	2005	LGL	NULL
2049		1475	1279	LGL	NULL
2050		1800	1221	LGL	NULL

Configuring Match Rules

To configure match rules, perform the following tasks:

1. Review the data.
2. Identify the base objects for the match job.
3. Configure match properties.
4. Define the match path.
5. Define match columns.
6. Define a match rule set.
7. Add match rules.
8. Set merge options for match rules.
9. Review the match properties.
10. Test the match rules.

Step 1. Review the Data

Before you create the match rules, review and understand the data. The base objects might have incorrect, inconsistent, and missing values.

Review the customer data for quality of the data values, consistency, uniqueness, and logic. To create match rules to match individuals, you must understand the attributes related to individuals.

The sample data set includes the Party and Address base objects. The Party base object has the First Name and the Last Name columns with missing values. It is not ideal to base the match rules on the First Name and the Last Name columns, because of missing values. The Display Name column has values for all records and is good to use as a match column in match rules.

The Party Type column in the Party base object identifies a customer record as a person or an organization. You can use the Party Type column to filter out customer records that belong to organizations because you do not want to find matches for customer records that belong to organizations. You can improve match performance if you filter data that is not relevant for the match process.

The Address base object has columns such as Address Line1, City Name, and Postal Cd that you can use as match columns in match rules. The columns in the Address base object would help identify duplicate records

in the Party base objects. You can create match rules to match individuals based on the name and address attributes.

Step 2. Identify the Base Objects for the Match Job

You want to match the data of individuals based on the name and address attributes. Before you create match rules for the data of individuals, you must identify the base objects that contain specific data on which you want to base the match rules.

In the example, the data of individuals is stored in the Party and Address base objects. The names of individuals are in the Party base object and the addresses of the individuals are in the Address base object. You can determine matches and eliminate duplicate records of individuals based on names and addresses in the Party and Address base objects. To find duplicates, you can run the match job on the Party base object that contains the names of individuals.

Step 3. Configure Match Properties

Before you configure the match columns for the match rules, configure the match properties for the Party base object, such as the match and search strategy. You want to configure the match properties for the Party base object because it has duplicate records for individuals.

You must decide on the match properties based on the characteristics of the data, your knowledge of the data, and the match and merge requirements. The data in the Party base object has imperfections such as spelling variations and transpositions. The fuzzy match and search strategy, which is based on probabilistic matching, is suitable for the Party base object data. When you set the fuzzy match and search strategy, you need to set a population for the match rule. The population defines the characteristics of the data that you want to use for matching. The data that you chose for matching is US-centric, and therefore you need to set the `us` population as a match property.

Configuring Match Properties

To configure the match properties, use the Schema tool in the Hub Console.

1. In the Hub Console, start the Schema tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Party base object for which you want to configure match properties.
4. Click **Match/Merge Setup**.
The **Match/Merge Setup Details** page appears.
5. Click the **Properties** tab.
The **Party Match/Merge Setup Details** section appears.

6. To configure match rules, you must set the following properties:

Property	Value and Description
Match/Merge Strategy	Fuzzy. Probabilistic match strategy that takes into account spelling variations, possible misspellings, and transpositions that are present in the data of individuals.
Fuzzy Population	US. A population that defines certain characteristics about the records that you want to match.

The following image shows the **Properties** tab of the **Match/Merge Setup Details** page:

The screenshot shows the 'Match/Merge Setup Details' page with the 'Properties' tab selected. The page has a header bar with four tabs: 'Match Rule Sets', 'Primary key match rules', 'Match Key Distribution', and 'Merge Settings'. Below the header, there are three sub-tabs: 'Properties', 'Paths', and 'Match Columns'. The 'Properties' sub-tab is active, showing a table with the following rows:

Party Match/Merge Setup Details	
Match Columns	0
Match Rule Sets	0
Match Rules in Active Set	0
Primary key match rules	0
Maximum matches for manual consolidation	1000
Number of rows per match job batch cycle	10
Accept All Unmatched Rows as Unique	No
Match/Search Strategy	Fuzzy
Fuzzy Population	US
Match Only Previous Rowid Objects	<input type="checkbox"/>
Match Only Once	<input type="checkbox"/>
Dynamic Match Analysis Threshold (0=disa...	0

Step 4. Define the Match Path

To configure match rules for related records, configure match path components. Related records might reside in one base object or in multiple base objects.

To configure match rules that involve child records, you need to add path components to the root base object. When you add components to the match path, you define the connection between related parent and child base objects.

Add the following components to the match path:

- Party. Base object that contains data of individuals and organizations.
- Address. Base object that contains addresses of customers that are either individuals, or organizations.
- LU Address Type. Look up base object that contains address type lookups.

- Party Address Rel. Relationship base object that defines the relationships between the records in the Party and Address base objects.

Adding Match Path Components

Match paths define the connection between parent and child base objects. To configure match rules that include parent and child records, add match path components to the root base object. Also, add filters to include or exclude data during the match process.

1. In the **Match/Merge Setup Details** page for the Party base object, click the **Paths** tab.
2. Add the Party Address Rel relationship base object as a match path component.
 - a. From the Path Components section, select the **Root for C_Party** match path component.

The following image shows the **Paths** tab of the **Match/Merge Setup Details** page:

The screenshot shows the 'Match/Merge Setup Details' window with the 'Paths' tab selected. The 'Path Components' section contains a table with the following data:

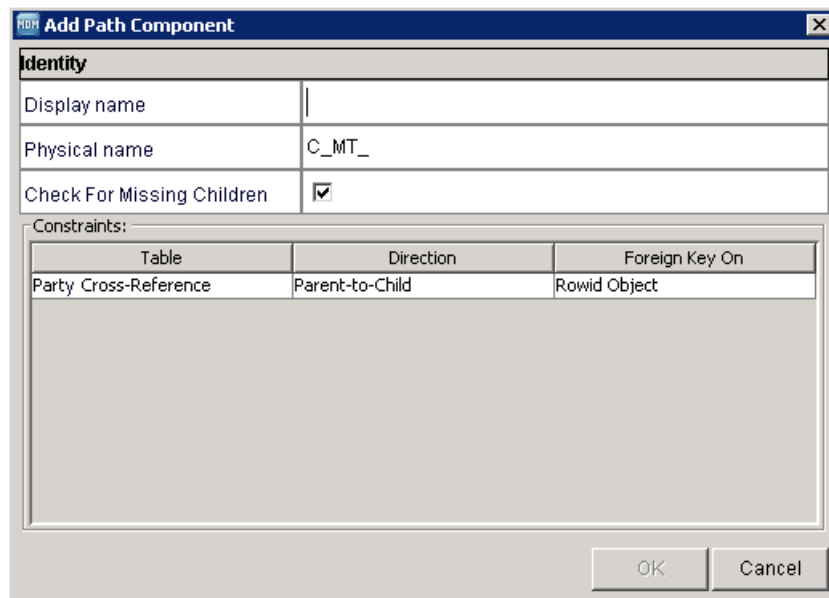
Display name	Component Name	Table Name	Direction	Check Mis...
Root for C_PARTY	N/A	Party	N/A	N/A

Below the table is a 'Filters' section with columns for 'Column', 'Operator', and 'Values'.

- b. In the **Path Components** section, click **Add**.

The following image shows the **Add Path Component** dialog box:

The **Add Path Component** dialog box appears.



The **Add Path Component** dialog box is shown. It has a title bar with a close button. The **Identity** section contains three fields: **Display name** (empty), **Physical name** (C_MT_), and **Check For Missing Children** (checked). Below this is a **Constraints** section with a table. The table has three columns: **Table**, **Direction**, and **Foreign Key On**. The first row contains the values **Party Cross-Reference**, **Parent-to-Child**, and **Rowid Object**. At the bottom are **OK** and **Cancel** buttons.

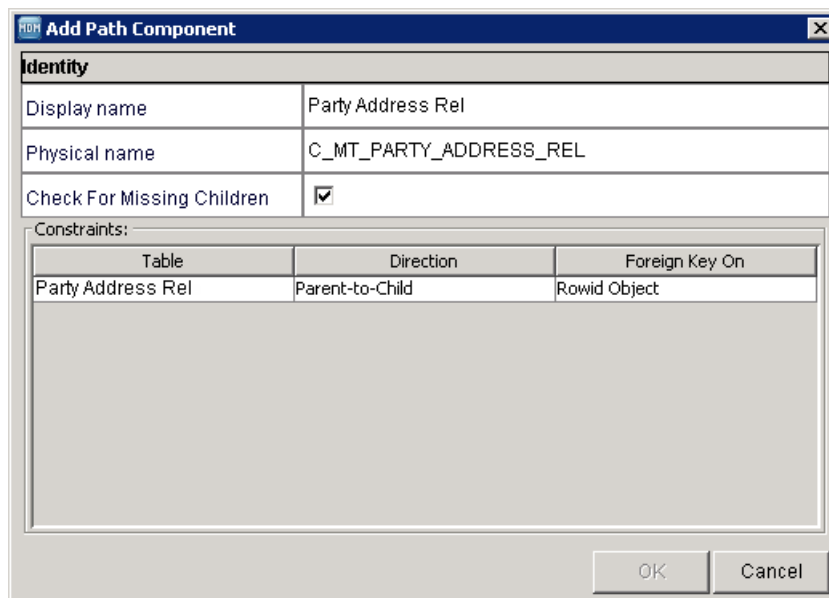
Table	Direction	Foreign Key On
Party Cross-Reference	Parent-to-Child	Rowid Object

- c. In the **Display Name** field, enter **Party Address Rel** as the name for the match path component that you want to display in the Hub Console.

The **Physical Name** field is populated based on the display name for the match path component that you enter. Physical name is the name of the path component in the database.

The following image shows the **Add Path Component** dialog box:

The **Add Path Component** dialog box appears with the name fields populated.



The **Add Path Component** dialog box is shown with the **Display name** field populated with **Party Address Rel**. The **Physical name** field is now **C_MT_PARTY_ADDRESS_REL**. The **Check For Missing Children** checkbox remains checked. The **Constraints** table is identical to the previous image. The **OK** and **Cancel** buttons are at the bottom.

Table	Direction	Foreign Key On
Party Address Rel	Parent-to-Child	Rowid Object

- d. Ensure that the **Check for missing children** option is enabled.

The option is enabled by default. When enabled, matching occurs between the parent base object records and their associated child base object records. Matching occurs even if the parent base object records do not have child records in the child base object for which the option is enabled.

Note: Match key generation depends on the configuration of the **Check for missing children** option. If the **Check for missing children** option is disabled, and if the parent record has no child records, match keys are not generated for the parent record.

- e. From the Constraints section, select a constraint, and click **OK**.

The Party Address Rel path component appears in the Path Components section.

The following image shows the **Paths** tab of the **Match/Merge Setup Details** page:

Match/Merge Setup Details

Primary key match rules | Match Key Distribution | Merge Settings

Properties | **Paths** | Match Columns | Match Rule Sets

Path Components

Display name	Component Name	Table Name	Direction	Check Mis...	
[-] Root for C_PARTY	N/A	Party	N/A	N/A	+
[-] Party Address Rel	C_MT_PARTY_ADDRESS_REL	Party Address Rel	Parent-to-Child	Yes	-

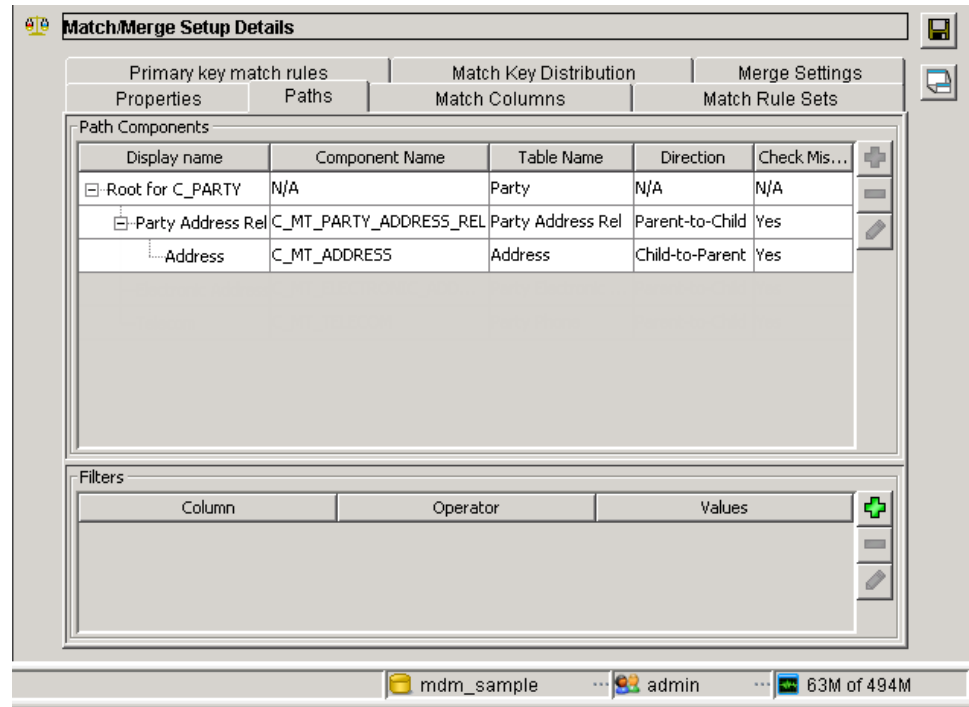
Filters

Column	Operator	Values	
			+

3. Add the Address base object to the match path as a child of the Party Address Rel match path component.
 - a. From the Path Components section on the **Paths** tab, select the **Party Address Rel** match path component.
 - b. Click **Add**.
The **Add Path Component** dialog box appears.
 - c. In the **Display Name** field, enter the name that you want to display in the Hub Console for the Address base object.

- d. From the Constraints section, select a constraint, and click **OK**.

The Address path component appears in the Path Components section as a child of the Party Address Rel match path component. The direction of the relationship appears as Child-to-Parent. The following image shows the **Paths** tab of the **Match/Merge Setup Details** page with path components:



4. To include individuals and exclude organizations for matching, configure a filter for the Party base object that has the Party Type column.
 - a. From the Path Components section, select the **Root for C_Party** match path component.
 - b. In the Filters section, click **Add**.
The **Add Filter** dialog box appears.
 - c. From the **Column** list, select **Party Type**.
 - d. From the **Operator** list, select **IN**.
 - e. Beside the **Values** field, click **Edit**.
The **Edit Values** dialog box appears.
 - f. Click **Add**.
The **Add Value** dialog box appears.
 - g. In the Value field, enter **Individual**, and click **OK**.
 - h. Click **OK**.

The filter for the Party Type column appears in the Filters section. The filter for the Party Type column ensures that records such as Ryerson Area Med Ctr that belong to the party type Organization are excluded for match key generation. Only records such as Rachel Arsen and Ahmed Rauf that belong to the Person party type are included for match key generation.

Step 5. Define Match Columns

After you configure the match path components, define the match columns that you want to use in the match rules. You can configure both fuzzy and exact match columns because you chose a fuzzy match and search strategy for the Party base object.

You want to create match rules to match duplicate records of individuals. The match rules must include columns with names of individuals and address data to match duplicate records.

To match duplicate records of individuals, define the following columns as match columns:

- Display Name
- Address Line 1
- Address Line 2
- City Name
- State Cd
- Postal Cd

Defining Match Columns

Define match columns for the Party base object.

1. In the **Match/Merge Setup Details** page for the Party base object, click the **Match Columns** tab.
2. Configure the fuzzy match key settings for the Party base object.
 - a. From the **Key Type** list, select **Person Name**.

The **Path Component** is set to **Root (Party)**, which is the path component for the **Person_Name** fuzzy match key that is added to the Match Columns section.

- b. From the **Key Width** list, select **Standard**.

The search range for which the MDM Hub generates match keys is set to the standard size. The following image shows the **Match Columns** tab with the Person Name fuzzy match key configured:

The screenshot shows the 'Match/Merge Setup Details' window with the 'Match Columns' tab selected. The 'Fuzzy Match Key' section is configured with 'Person Name' as the Key Type, 'Standard' as the Key Width, and 'Root (Customer)' as the Path Component. The 'Match Columns' table lists 'Person_Name' as a 'Fuzzy Match Key' column with a 'Root' path component and 'Party' as the source table. The 'Match Column Contents - Source Table: Customer' section shows 'Display Name' selected in the 'Available columns' list, which is being moved to the 'Selected columns' list.

Field Name	Column Type	Path Component	Source Table
Person_Name	Fuzzy Match Key	Root	Party

Match Column Contents - Source Table: Customer

Available columns:

- Display Name
- First Name
- Last Name
- Middle Name
- Organization Name
- Party Type

Selected columns:

The MDM Hub will generate match keys based on the Person Name fuzzy match key that you configured.

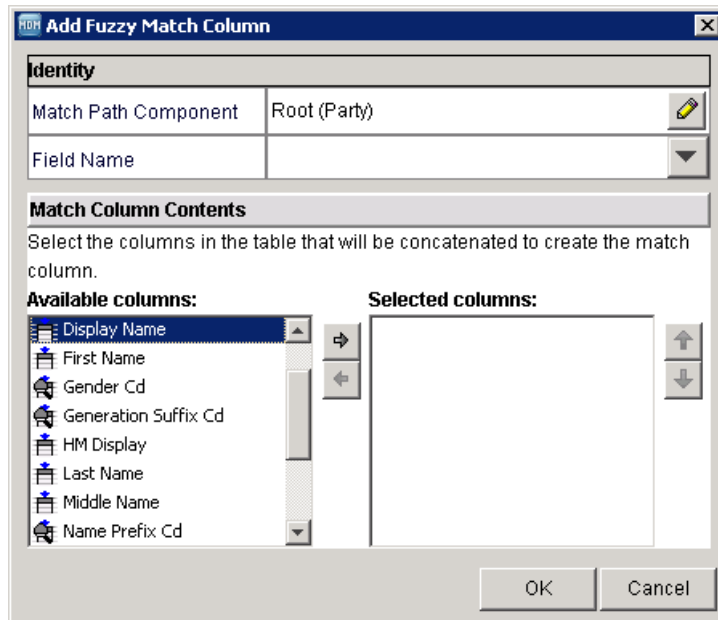
3. Define the Display Name column as the source column for the Person_Name fuzzy match key column.
- a. In the Match Column Contents section, select the **Display Name** column from the **Available columns** list.
- b. Click the right arrow.

The Display Name column moves to the **Selected columns** list, and the source column for the Person_Name match column is defined.

4. Add the fuzzy match column, Address_Part1, to contain street addresses.
 - a. Click **Add Fuzzy Match Column**.

The **Add Fuzzy Match Column** dialog box appears.

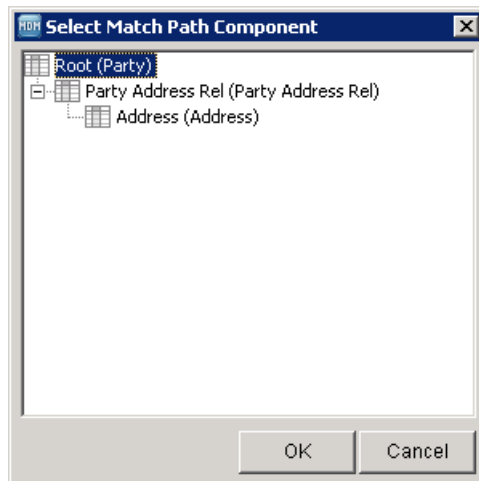
The following image shows the **Add Fuzzy Match Column** dialog box:



- b. Beside the **Match Path Component** field, click **Edit**.

The **Match Path Component** dialog box appears.

The following image shows the **Select Match Path Component** dialog box:



- c. Select the Address base object that contains the customer address data, and click **OK**.
 - d. From the **Field Name** list, select **Address_Part1**.
 - e. From the **Available columns** list, select the **Address Line 1** and **Address Line 2** columns that you want to concatenate to create the Address_Part1 match column.
 - f. To move the columns to the **Selected columns** list, click the right arrow.

The Address Line 1 and Address Line 2 columns move to the **Selected columns** list. The columns are concatenated to create the Address_Part1 match column. The **Add Fuzzy Match Column** dialog box appears.

The following image shows the **Add Fuzzy Match Column** dialog box:

Add Fuzzy Match Column

Identity

Match Path Component	Address (Address)
Field Name	Address_Part1

Match Column Contents

Select the columns in the table that will be concatenated to create the match column.

Available columns:

- Address Line3
- Bldg Name
- Care Of Address
- City Name
- Country Code
- County Name
- Dept Name
- Floor Number

Selected columns:

- Address Line1
- Address Line2

OK Cancel

- g. Click **OK**.

The Address_Part1 match column is added to the Match Columns section.

The following image shows the **Match Columns** tab with the match columns configured for person name and address data:

The screenshot shows the 'Match/Merge Setup Details' dialog box with the 'Match Columns' tab selected. The 'Fuzzy Match Key' section is at the top, followed by the 'Match Columns' table. Below that is the 'Match Column Contents - Source Table: Address' section, which includes 'Available columns' and 'Selected columns' lists.

Match/Merge Setup Details				
Primary key match rules		Match Key Distribution	Merge Settings	
Properties	Paths	Match Columns	Match Rule Sets	
Fuzzy Match Key				
Key Type	Person Name			
Key Width	Standard			
Path Component	Root (Party)			
Match Columns				
Field Name	Column Type	Path Component	Source Table	
Address_Part1	Fuzzy	Address	Address	
Person_Name	Fuzzy Match Key	Root	Party	
Match Column Contents - Source Table: Address				
Available columns:				
City Name				
Country Code				
County Name				
Dept Name				
Floor Number				
Is Valid Ind				
Latitude				
Selected columns:				
			Address Line1	
			Address Line2	

5. Add the fuzzy match column, Address_Part2, for customer address data to contain city names and the state codes.

- a. Click **Add Fuzzy Match Column**.

The **Add Fuzzy Match Column** dialog box appears.

- b. Beside the **Match Path Component** field, click **Edit**.

The **Match Path Component** dialog box appears.

- c. Select the Address base object that contains the customer address data, and click **OK**.

- d. From the **Field Name** list, select **Address_Part2**.

- e. From the **Available columns** list, select the **City Name** and **State Cd** columns that you want to concatenate to create the Address_Part2 match column.

- f. To move the columns to the **Selected columns** list, click the right arrow.

The City Name and State Cd columns move to the **Selected columns** list. The columns are concatenated to create the Address_Part2 match column.

- g. Click **OK**.

The Address_Part2 match column is added to the Match Columns section.

6. Add the exact match column, Postal_Area, to contain postal codes for customer addresses.
 - a. Click **Add Exact Match Column**.
The **Add Exact Match Column** dialog box appears.
 - b. Beside the **Match Path Component** field, click **Edit**.
The **Select Match Path Component** dialog box appears.
 - c. Select the Address base object that contains the customer address data, and click **OK**.
 - d. From the **Field Name** list, select **Postal_Area**.
 - e. In the **Available columns** field name, select the **Postal Cd** column that you want to use to create the Postal_Area match column.
 - f. To move the column to the **Selected columns** list, click the right arrow.
The Postal Cd column moves to the **Selected columns** list.
 - g. Click **OK**.
The Postal_Area match column is added to the Match Columns section.

The following image shows the **Match Columns** tab with the match columns configured for person name and address data:

Match/Merge Setup Details

Primary key match rules | Match Key Distribution | Merge Settings
 Properties | Paths | **Match Columns** | Match Rule Sets

Fuzzy Match Key

Key Type	Person Name
Key Width	Standard
Path Component	Root (Customer)

Match Columns

	Field Name	Column Type	Path Compon...	Source Table
	Address_Part1	Fuzzy	Address	Address
	Address_Part2	Fuzzy	Address	Address
	Person_Name	Fuzzy Match Key	Root	Party
	Postal_Area	Exact	Address	Address

Match Column Contents - Source Table: Address

Available columns:

- Address Line3
- Bldg Name
- Care Of Address
- City Name
- Country Code

Selected columns:

- Address Line1
- Address Line2

Step 6. Define a Match Rule Set

You want to define a match rule set that has a search level set to Typical. You can create match rules in the match rule set that you define. You need to create at least one match rule set. Match rule sets can contain a logical set of match rules that have some common properties.

Defining a Match Rule Set

Define a rule set to which you can add match rules.

1. In the **Match/Merge Setup Details** page for the Party base object, click the **Match Rule Sets** tab.
2. To add a match rule set, click **Add**.

The **Add Match Rule Set** dialog box appears.

3. Enter a unique name, such as WS, for the rule set, and click **OK**.
4. From the **Search Level** list, select **Typical**.

The search level to search for match candidates is set to Typical.

5. To ensure that the match rule set is not reserved for exclusive use with the SearchMatch API, disable the **Enable Search by Rules** option.
6. Ensure that the **Enable Filtering** option is disabled.

Note: If the **Enable Filtering** option is disabled, all records are processed by the match rule set when a match batch job is run. If the **Enable Filtering** option is enabled, you can define a filter for this match rule set, and only the filtered records are processed by the match rule set.

The following image shows a Match/Merge Setup Details page with the WS match rule set configured:

The screenshot shows the 'Match/Merge Setup Details' window. The 'Match Rule Sets' tab is selected. On the left, a list shows 'WS (*)' with a green plus icon. The main area displays the configuration for 'WS':

Match Rule Set	
Name	WS
Search Level	Typical
Enable Search by Rules	<input type="checkbox"/>
Enable Filtering	<input type="checkbox"/>
Filtering SQL	

Below this, the 'Match Rules' section is visible, showing a table with columns: Rule #, Auto, Type, Accept Limit. The table is currently empty, and a green plus icon is visible on the right side of the table header.

Step 7. Add Match Rules

Before you add match rules, you must decide whether you want to set the match rules for automerge or manual merge. To match individuals at an address, you need a match purpose that has the name and address components.

If you set the match rules for automerge, the MDM Hub merges the records that are matched without the intervention of a data steward. If you configure the match rules for manual merge, the MDM Hub does not automatically merge the records that are matched. A data steward can review the matched records and start the merge job.

Also, you must decide whether you want to set up fuzzy, exact, or filtered match rules. A fuzzy match rule is required for data that might have inconsistencies. If the quality of the data is good, you can configure exact match rules. If you want to run a large batch job and at the same time ensure optimal performance, you might want to configure filtered match rules. Filtered match rules are exact match rules that use the fuzzy match key in addition to exact match columns.

To match individuals at an address, use the Resident match purpose that has the name and address components.

Adding Match Rules

To match duplicate individuals based on name and address, add fuzzy match rules with the Resident match purpose to the WS match rule set. You created the WS match rule set in the preceding step.

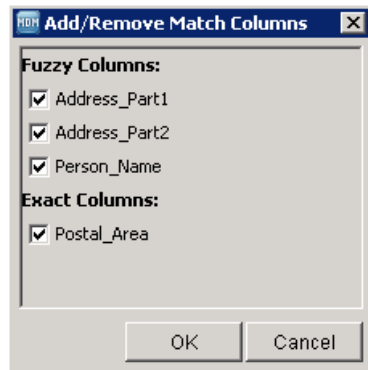
1. In the **Match/Merge Setup Details** page for the Party base object, click the **Match Rule Sets** tab.
2. Add an manual merge match rule with the Typical match level.
 - a. In the Match Rules section, click **Add**.

The **Edit Match Rule** dialog box appears.
 - b. From the **Match/Search Strategy** list, select **Fuzzy**.

A fuzzy match/search strategy is a probabilistic match strategy that includes records with spelling variations, misspellings, and transpositions.
 - c. Beside the **Match Columns** field, click **Edit**.

The **Add/Remove Match Columns** dialog box appears.
 - d. Select the following columns:
 - Address_Part1
 - Address_Part2
 - Person_Name
 - Postal_Area

The following image shows the **Add/Remove Match Columns** dialog box with the match columns selected:



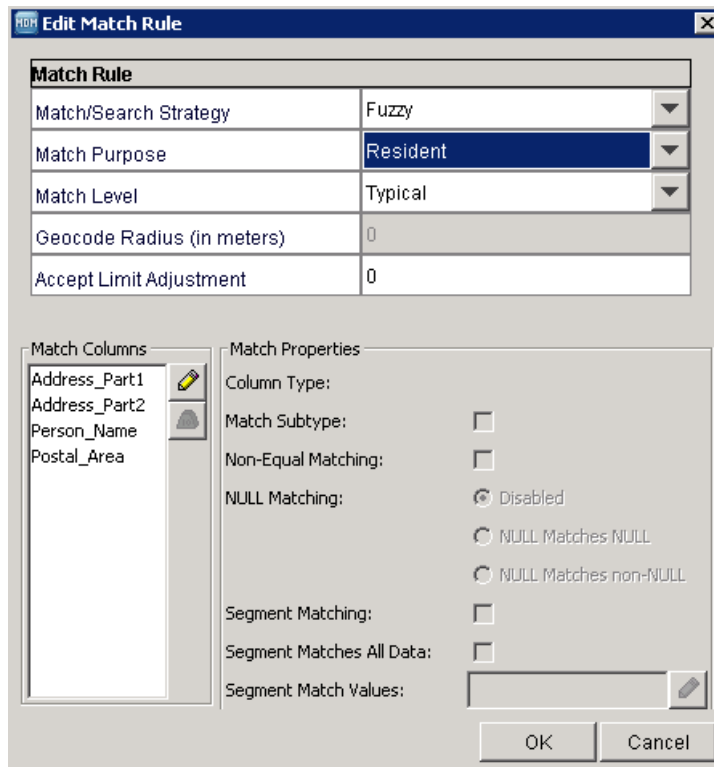
- e. Click **OK**.

The match column names appear in the **Match Columns** field.

- f. To identify matches for person names based on addresses, from the **Match Purpose** list, select **Resident**.
- g. To define the precision of the match between records, from the **Match Level** field, select **Typical**.

The Typical match level is suitable for most match jobs that match individuals. The Conservative match level results in fewer matches than the Typical match level and some potential matches might pass through the match process without being flagged as a match. The Loose match level might result in a significantly higher number of matches than the Typical match level, and might include matches that are not really matches.

The following image shows the **Edit Match Rule** dialog box with the fuzzy match rule settings:



- h. Click **OK**.

A manual merge match rule with the Resident match purpose appears in the Match Rules section.

3. Add a manual merge match rule with the Loose match level.

- a. In the Match Rules section, click **Add**.

The **Edit Match Rule** dialog box appears.

- b. From the **Match/Search Strategy** list, select **Fuzzy**.

A fuzzy match/search strategy is a probabilistic match that includes records with spelling variations, misspellings, and transpositions.

- c. Beside the **Match Columns** field, click **Edit**.

The **Add/Remove Match Columns** dialog box appears.

- d. Select the following fuzzy columns from the **Fuzzy Columns** options:

- Address_Part1
- Person_Name

- e. Click **OK**.

The fuzzy match columns appear in the **Match Columns** field.

- f. To identify matches for person names based on address, from the **Match Purpose** list, select **Resident**.

- g. To define the precision of the match between records, from the **Match Level** field, select **Loose**.

- h. Click **OK**.

A manual merge match rule with the Resident match purpose appears in the Match Rules section.

The following image shows the **Match Rule Sets** tab with two fuzzy, manual merge match rules configured for the WS match rule set:

Match/Merge Setup Details

Properties Paths Match Columns Match Rule Sets Primary key match rules Match Key Distribution Merge Settings

Match Rule Set

WS (*)

Match Rule Set

Name WS

Search Level Typical

Enable Search by Rules ☐

Enable Filtering ☐

Filtering SQL

Match Rules

Rule #	Auto	Type	Accept Limit	Purpose(Level)	Columns
1	No	Fuzzy	0	Resident(Typical)	Address_Part1 (Fuzzy) Address_Part2 (Fuzzy) Person_Name (Fuzzy) Postal_Area (Fuzzy)
2	No	Fuzzy	0	Resident(Loose)	Address_Part1 (Fuzzy) Person_Name (Fuzzy)

Step 8. Set Merge Options for Match Rules

During the match process, the match rule must determine whether to queue matched records for automerge or manual merge. You can toggle between manual merge and automerge for a match rule. To automerge records after the match process, you can set the typical match level for the match rule.

The precision of the match rule with the Resident match purpose and the typical match level lends itself for automerge.

Setting a Match Rule as an Automerge Match Rule

To queue matched records for automerge during the match process, set the match rule as an automerge match rule.

1. Select the manual merge match rule with the typical match level that you want to change to an automerge match rule.
2. Click **Move Up**.

The manual merge match rule changes to an automerge match rule.

The following image shows the **Match Rule Sets** tab with an automerge match rule and a manual merge match rule that is configured for the WS match rule set:

Match/Merge Setup Details

Properties Paths Match Columns Match Rule Sets Primary key match rules Match Key Distribution Merge Settings

Match Rule Set

WS (*)

Match Rule Set

Name WS

Search Level Typical

Enable Search by Rules ☐

Enable Filtering ☐

Filtering SQL

Match Rules

Rule #	Auto	Type	Accept Limit	Purpose(Level)	Columns
1	Yes	Fuzzy	0	Resident(Typical)	Address_Part1 (Fuzzy) Address_Part2 (Fuzzy) Person_Name (Fuzzy) Postal_Area (Fuzzy)
2	No	Fuzzy	0	Resident(Loose)	Address_Part1 (Fuzzy) Person_Name (Fuzzy)

3. Click **Save**.
The **Rule Set Evaluation** dialog box appears.
4. Click **OK**.
The match rules that you created are saved.

Step 9. Review the Match Properties

After you save the match rules configuration, review the match properties. The match properties summarize the match and merge setup.

The match and merge properties must show the following details:

- Number of match columns configured
- Number of match rule sets configured
- Number of match rules in the active match rule set
- Maximum matches for manual consolidation
- Number of rows for each match batch job cycle
- Type of match and search strategy

- Name of the fuzzy population

Reviewing the Match Properties

For a summary of the match rules setup, review the match properties.

- In the **Match/Merge Setup Details** page for the Party base object, click the **Properties** tab.

The **Party Match/Merge Setup Details** view appears.

The following table summarizes the match and merge setup:

Properties	Value
Match Columns	4
Match Rule Sets	1
Match Rules in Active Set	2
Primary key match rules	0
Maximum matches for manual consolidation	1000
Number of rows per match job batch cycle	10
Accept All Unmatched Rows as Unique	No
Match/Search Strategy	Fuzzy
Fuzzy Population	US
Match Only Previous Rowid Objects	Disabled
Match Only Once	Disabled
Dynamic Match Analysis Threshold (0=disabled)	0

The following image shows the **Properties** tab of the **Match/Merge Setup Details** page:

Match/Merge Setup Details	
Match Rule Sets	Primary key match rules
Properties	Paths
Match Columns	
Match Columns	4
Match Rule Sets	1
Match Rules in Active Set	2
Primary key match rules	0
Maximum matches for manual consolidation	1000
Number of rows per match job batch cycle	10
Accept All Unmatched Rows as Unique	No
Match/Search Strategy	Fuzzy
Fuzzy Population	US
Match Only Previous Rowid Objects	<input type="checkbox"/>
Match Only Once	<input type="checkbox"/>
Dynamic Match Analysis Threshold (0=disa...	0

Step 10. Test the Match Rules

To test the match rules, run the match job and review the results.

You can test the match rules on a sample data set that is representative of the larger data set. To test the match rules, run the match job on the sample, representative data set. After the match job completes, review the match results.

You can view the match results in the C_PARTY_MTCH match table that is associated with the Party base object. Also, you can view the match results through Merge Manager in the Hub Console, and through the Potential Matches option in Informatica Data Director. Review the matched records to verify their accuracy.

The C_PARTY_MTCH match table shows important columns that contain the sample match result.

ROWID_OBJECT	ROWID_OBJECT_MATCHED	ROWID_MATCH_RULE	AUTOMERGE_IND
1191	1019	SVR1.JJ4J	0
1191	1419	SVR1.JJ4J	0
1154	1106	SVR1.JJ4E	1
1642	1072	SVR1.JJ4E	1

The ROWID_OBJECT column contains the row ID of the records that are matched to the records with row ID in the ROWID_OBJECT_MATCHED column.

The ROWID_MATCH_RULE column contains the row ID of the match rules that you create. The match rules are in the C_REPOS_MATCH_RULE table in the repository. SVR1.JJ4J is the row ID of the match rule with the Resident match purpose and a loose match level. SVR1.JJ4E is the row ID of the match rule with the Resident match purpose and a typical match level.

The example shows that the record for William De Haan that has row ID 1191 has two matches. The record matches the individual with the row ID 1019, Will R De Haan, and the individual with the row ID 1419, Bill Roger De Haan. The matches are based on the name and address columns and are similar but need to be reviewed by a data steward before the merge. Both the matches of the record for William De Haan are queued for manual merge.

The match table shows that the record for Rachel Arsen that has row ID 1154 matches another record for Rachel Arsen that has row ID 1106. Both the records for Rachel Arsen have addresses in the Address base object that are similar and can be safely queued for automerge.

The following table shows the names of some individuals whose records were matched to other similar records:

Display Name	Display Name Matched
WILLIAM DE HAAN	WILL R DE HAAN
WILLIAM DE HAAN	BILL ROGER DE HAAN
RACHEL ARSEN	RACHEL ARSEN
AHMED RAUF	AHMED RAUF

The match result appears correct in the example. You can use the match rules that you configured for the larger match job.

Note: To view any issues that might occur during the match job, review the `cmxserver.log` file located in the following directory:

On UNIX. `<infamdm_install_directory>/hub/server/logs`

On Windows. `<infamdm_install_directory>\hub\server\logs`

Timeouts can occur because of match properties such as the number of rows for each match job batch cycle. You can change the properties in the **Properties** tab of the **Match/Merge Setup** page.

To check the progress of the match job and to get summary statics, review the Process Server log.

The Process Server log, `cmxserver.log`, is in the following directory:

On UNIX. `<Process Server installation directory>/hub/cleanse/logs`

On Windows. `<Process Server installation directory>\hub\cleanse\logs`

CHAPTER 23

Search with Elasticsearch

This chapter includes the following topics:

- [Search with Elasticsearch Overview, 459](#)
- [Search with Elasticsearch Architecture, 459](#)
- [Installing and Configuring Search, 460](#)
- [Step 1. Install and Set Up Elasticsearch, 461](#)
- [Step 2. Configure the MDM Hub Properties for Search, 466](#)
- [Step 3. Configure Search by Using the Provisioning Tool, 470](#)
- [Step 4. Validate the Operational Reference Store, 480](#)
- [Step 5. Index the Search Data, 481](#)
- [Creating Keystores, Truststore, and Certificates \(Optional\), 481](#)

Search with Elasticsearch Overview

You can use the Data Director application or a custom application to search for data within a specific business entity. MDM uses Elasticsearch, which is an open-source, full-text search engine.

Note: The search with Elasticsearch replaces the search with Solr, which is no longer supported.

You must install and set up Elasticsearch. Set up Elasticsearch as a single node cluster or as a multi-node cluster to provide distributed indexing and search.

Before a data search is performed, business users and data stewards must configure the Hub Server and the Process Server, and then index the data. When you index the data, the MDM Hub processes the records and adds the indexed records to the Elasticsearch server. If the data is large for one node, you can use multiple nodes and split the index into multiple shards.

Search with Elasticsearch Architecture

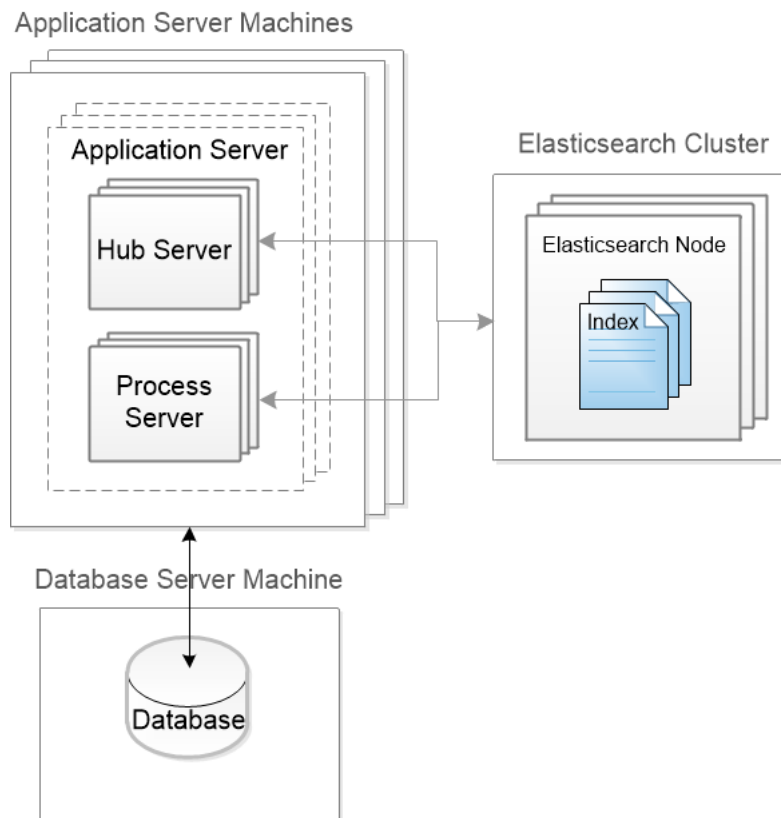
The MDM Hub uses the Hub Server and the Process Server for search. The Elasticsearch client is embedded in the Hub Server to perform real-time indexing. The Process Server performs only the initial indexing after

the initial data load. Whether you install all the MDM Hub components on a single machine or on multiple machines, you must configure all the Hub Server and Process Server instances for search.

You can install Elasticsearch on any machine where the MDM Hub components are installed or on a separate machine. Elasticsearch requires an exclusive supported version of the Java Virtual Machine (JVM), which the MDM Hub components do not share.

To configure the MDM Hub components to search with Elasticsearch, set up the Elasticsearch cluster. A cluster is a collection of one or more nodes. Each node is a single server that stores your data and participates in indexing and search operations. Based on your MDM Hub topology and the amount of data to index, you can configure one or more nodes for the cluster. Each node can in turn have multiple indexes. To prevent data loss and maintain cluster stability, you can configure Zen Discovery for multi-node clusters.

The following image shows a sample installation topology configured for search:



Installing and Configuring Search

To use search, configure the MDM Hub with Elasticsearch.

1. Install and set up Elasticsearch.
2. Configure the MDM Hub properties for search.
3. Configure search by using the Provisioning tool.
4. Validate the Operational Reference Store (ORS).
5. Index the search data.

Step 1. Install and Set Up Elasticsearch

To configure search, you must install and set up Elasticsearch.

To set up Elasticsearch, perform the following tasks:

1. Complete the pre-installation tasks.
2. Install Elasticsearch.
3. Configure the Elasticsearch Java Virtual Machine (JVM).
4. Configure the Elasticsearch properties file.
5. Secure Elasticsearch.
6. Install the analysis plugins.
7. Configure stop words, synonyms, and character mappings
8. Start Elasticsearch.

Complete Pre-Installation Tasks

Before you install and set up Elasticsearch clusters, prepare the environment and determine whether you want to configure high availability.

Tasks for All Environments

Perform the following tasks to prepare the installation environment:

- Ensure that each machine satisfies the hardware requirements for the supported version of Elasticsearch. For information about hardware, see the Elasticsearch documentation.
- Ensure that each machine satisfies the software requirements for the supported version of Elasticsearch, such as supported operating systems and Java version. For information about the software requirements, see the *Elasticsearch Support Matrix*.
- Complete important system configurations, such as swapping, file descriptors, and virtual memory. For information about important system configurations, see the Elasticsearch documentation.

Tasks for UNIX Environments

In a UNIX environment, perform the following tasks:

- To avoid data loss due to insufficient number of file descriptors, set the number of file descriptors to 65536 or higher.
- To prevent memory swapping, configure the system to prevent swapping. You can configure the Java Virtual Machine (JVM) to lock the heap in memory through `mlockall`.

High Availability Requirements

If you have a large amount of data to index and search, the best practice is to implement a highly available Elasticsearch cluster. A highly available cluster has multiple nodes, and the cluster can distribute the workload among the nodes. If one node fails in a production environment, the cluster distributes the workload to the other nodes.

As a pre-installation task, decide if you want to implement a highly available Elasticsearch cluster. If so, configure the Elasticsearch cluster as usual, but ensure that you satisfy the following additional requirements:

- The Elasticsearch cluster has three or more nodes.

Tip: You can set up a small cluster to start and scale it as necessary. Analyze the workload and make sure that you have enough capacity to handle a node failure.

- Each node is configured on a separate, dedicated machine.
- At least three of the nodes are master nodes to ensure stability and performance. Note that Elasticsearch recommends an odd number of master nodes.
 - If the cluster has only three nodes, configure all the nodes as master nodes.
 - If the cluster has more than three nodes, configure three nodes as master nodes and configure the rest of the nodes as data nodes.
- Based on the Elasticsearch cluster size, decide on the number of replicas. When you use the Provisioning tool to configure the Elasticsearch index, you can specify the number of replicas to use.
- For each node, set the following additional properties in the `elasticsearch.yml` configuration file:
 - `discovery.zen.minimum_master_nodes`
 - `discovery.zen.ping.unicast.hosts`

For more information about highly available clusters, including hardware requirements, system configurations, and property values, see the Elasticsearch documentation.

Install Elasticsearch

After you install the Hub Server and the Process Server, to configure search, install and set up Elasticsearch clusters.

Ensure that you use a supported operating system and Java version for your Elasticsearch installation. For more information, see the Elasticsearch Support Matrix.

For more information about how to install Elasticsearch and set up clusters, see the Elasticsearch documentation.

1. From the Elastic website, download the supported version of the Elasticsearch archive file.
For information about the supported versions, see the Product Availability Matrix (PAM). You can access PAMs at <https://network.informatica.com/community/informatica-network/product-availability-matrices>.
2. Extract the Elasticsearch archive file.

Configure the Elasticsearch Java Virtual Machine (JVM)

Configure the Elasticsearch Java Virtual Machine (JVM) to use a heap size based on the amount of RAM available on your machine. To configure the JVM, edit the `jvm.options` file.

1. Find the `jvm.options` file in the following directory:
`<elasticsearch installation directory>/config`

2. Use a text editor to open the file, and edit the following properties:

Property	Description
-Xms	Minimum heap size. Default is 1 GB.
-Xmx	Maximum heap size. Default is 1 GB.
-XX:HeapDumpPath	Heap dump path. Default is <code>/var/lib/elasticsearch</code> . In a multi-cluster environment, you must set this property to an alternative path.

Note: Set the minimum heap size (Xms) and the maximum heap size (Xmx) to the same value. Use the default settings for other properties.

Configure the Elasticsearch Properties File

Informatica provides a sample Elasticsearch properties file. To configure Elasticsearch, edit the properties file.

1. Find the `elasticsearch.yml` file in the following directory:
`<elasticsearch installation directory>/config`
2. Use a text editor to open the file, and edit the following properties:

Property	Description
<code>bootstrap.memory_lock</code>	Sets up memory locking. To prevent any Elasticsearch memory from being swapped out, set to <code>true</code> . Default is <code>true</code> .
<code>cluster.name</code>	Specify a unique name for the Elasticsearch cluster. If you have multiple clusters, ensure that the name of each cluster is unique. If a cluster has multiple nodes, ensure that on each node of the cluster, the same cluster name is specified.
<code>discovery.zen.minimum_master_nodes</code>	Required for a multi-node cluster to prevent data loss and maintain cluster stability. Set to the following value: $(\text{number of master-eligible nodes} / 2) + 1$ For example, if a cluster has three nodes, all of which are master-eligible nodes and can contain data, then set the property to $(3 / 2) + 1$, which is rounded to 2.
<code>discovery.zen.ping.unicast.hosts</code>	Required for a multi-node cluster. This property is used to specify the discovery setting, which is a list of IP addresses and transport ports of the nodes in the cluster. Use the following format to set the property: <code>["host1:port1", "host2:port2", "host3:port3"]</code>
<code>http.port</code>	Port for the HTTP requests. Default is 9200.
<code>network.host</code>	The IP address of the host to use as the bind address.
<code>node.data</code>	Enables a node as a data node that performs data related operations, such as CRUD and search. Default is <code>true</code> .

Property	Description
node.ingest	Enables a node as an ingest node that transforms and enriches the data before indexing. Default is <code>true</code> .
node.master	Enables a node as a master node that controls the cluster. If a cluster has multiple nodes, enable at least one of the nodes as a master node. For high availability, set multiple nodes as master nodes. Default is <code>true</code> .
node.name	Specify a unique name for the node.
path.data	Path to the directory where you want to store the data. You can configure multiple data directories. For more information about configuring multiple data directories, see the Elasticsearch documentation.
path.logs	Path to the log files.
transport.tcp.port	The TCP bind port. Default is <code>9300</code> .

3. Save the properties file with the same name, `elasticsearch.yml`.

Secure Elasticsearch (Optional)

After you install Elasticsearch, secure the communication between the MDM Hub and Elasticsearch. Also, secure the Elasticsearch cluster.

For information about securing Elasticsearch, see the Elasticsearch security documentation.

Install Analysis Plugins

Install the Phonetic and Japanese (kuromoji) analysis plugins, which extend Elasticsearch by adding new analyzers, tokenizers, token filters, and character filters. The Phonetic analysis plugin analyzes and converts tokens into their phonetic equivalent. The Japanese (kuromoji) analysis plugin analyzes Japanese by using the Kuromoji analyzer.

1. Download the Phonetic and Japanese (kuromoji) analysis plugin from the Elastic website.
2. Install the Phonetic analysis plugin on each cluster node by running the following command:

```
sudo bin/elasticsearch-plugin install analysis-phonetic
```
3. Install the Japanese (kuromoji) analysis plugin on each cluster node by running the following command:

```
sudo bin/elasticsearch-plugin install analysis-kuromoji
```
4. Restart each cluster node.

Configure Stop Words, Synonyms, and Character Mappings

When you perform a search, MDM can ignore common words such as "and", "an", and "is". MDM can also search for synonyms of the search string. For example, when you search for "William", the search result can include the synonyms "Will" and "Willy".

To configure common words to ignore or include synonyms in search results, Informatica provides text files that contain stop words and synonyms, or you can configure your own.

To use the default Elasticsearch analyzers for languages such as Chinese, Japanese, and Korean, Informatica provides a mappings file, `mapping-FoldToASCII.txt`. The character filter of these default analyzers uses the mappings file to convert alphabetic, numeric, and symbolic characters that are not in the Basic Latin Unicode block to their ASCII equivalent.

To get the `stopwords.txt`, `synonyms.txt`, `stopwords_ja.txt`, and `mapping-FoldToASCII.txt` files, contact Informatica Global Customer Support.

To configure stop words, synonyms, and character mappings, perform the following steps:

1. Create an analysis directory in the following location:
`<elasticsearch installation directory>/config`
2. Copy the `stopwords.txt` and `synonyms.txt` files to the analysis directory.
3. To configure stop words for languages such as Japanese, create a `lang` directory in the following location:
`<elasticsearch installation directory>/config/analysis`
4. Copy the stop words files for other languages, such as `stopwords_ja.txt`, and the `mapping-FoldToASCII.txt` file to the `lang` directory.

Customize the List of Stop Words for Searches

To customize the list of words to ignore in a search, edit the `stopwords.txt` file.

1. Use a text editor to open the `stopwords.txt` file in the following location:
`<elasticsearch installation directory>/config/analysis`
2. Edit and save the `stopwords.txt` file.
3. If data was indexed before you edited the `stopwords.txt` file, manually delete the indexes, restart Elasticsearch, and then, reindex the data.

For more information about updating the `stopwords.txt` file, see the Elasticsearch documentation.

Customize the List of Synonyms to Include in Searches

To customize the synonyms to use in a search, edit the `synonyms.txt` file.

1. Use a text editor to open the `synonyms.txt` file in the following location:
`<elasticsearch installation directory>/config/analysis`
2. Edit and save the `synonyms.txt` file.
3. If data was indexed before you edited the `synonyms.txt` file, manually delete the indexes, restart Elasticsearch, and then, reindex the data.

For more information about updating the `synonyms.txt` file, see the Elasticsearch documentation.

Start Elasticsearch

After you set up Elasticsearch, start each node of the Elasticsearch cluster for the changes to take effect.

Tip: When you start Elasticsearch, if memory locking issues occur, you might need to set `soft memlock unlimited` and `hard memlock unlimited`.

1. Open a command prompt, and change to the following directory:
`<elasticsearch installation directory>/bin`

2. Run the following command:

On UNIX. `elasticsearch.sh`

On Windows. `elasticsearch.bat`

Upgrade Elasticsearch

After you add an upgraded Elasticsearch node, indicate whether to update the new node with the index details. Then manually run the Initially Index Smart Search Data batch jobs to add indexes to the upgraded node. The existing node continues to manage search requests. When the batch jobs run successfully and the node is updated, the new node can manage search requests.

Note: Applies when you upgrade from Elasticsearch Version 6 to Version 7. Elasticsearch authentication support is valid only for Version 7.x.

For more information about configuring the Elasticsearch node, see [“Configure the Elasticsearch Cluster” on page 470](#).

Step 2. Configure the MDM Hub Properties for Search

To configure the MDM Hub properties, use the Hub Console, the Process Server properties file, and the Hub Server properties file.

1. Configure the Process Server properties.
2. Configure the Hub Server properties.

Configure the Hub Server Properties

You must configure all the Hub Server instances to enable search. Use the Hub Server tool in the Hub Console and the `cmxserver.properties` file to configure the Hub Server properties for search.

1. Use a text editor to open the `cmxserver.properties` file in the following location: <MDM Hub Installation Directory>\hub\server\resources\cmxserver.properties

2. Configure the following properties for search:

cmx.ss.enabled

Indicates whether to enable search. In a new installation, the default is `true`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `false`.

ex.max.conn.per.host

Sets the maximum number of Elasticsearch nodes that you want to connect to the host. Set to the number of Elasticsearch cluster nodes on the host.

ex.max.threads

Sets the maximum number of threads that you want the Apache asynchronous non-blocking receiver to use for each node in the Elasticsearch cluster. Default is 1.

Change the value only when suggested by Informatica Global Customer Support.

es.index.refresh.interval

Sets the interval, in seconds, for Elasticsearch to commit the changes to the data after an Initially Index Smart Search Data batch job is run. The data is available for search after this time interval. Default is 30.

This property impacts the high indexing volume encountered during initial indexing. Change the value only when suggested by Informatica Global Customer Support.

cmx.e360.view.enabled

When MDM administrators implement the Entity 360 framework, IDD users use the **Search** box to find records and a entity tabs to edit and manage records. In a new installation, the default is `true`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `false`.

search.provisioning.numshards

Optional. Number of shards to create on your Elasticsearch environment. The value depends on the maximum number of shards and the total number of nodes. For example, if the maximum number of shards is 1 and the number of nodes is 3, you can create 3 shards. Default is the total number of Hub Servers.

search.provisioning.numreplicas

Optional. Number of copies of the Elasticsearch engine documents that you want to create on different nodes. Use the replication factor to create multiple copies of the documents in the shards of different nodes. You require multiple copies of the documents to achieve high availability if one or more nodes shut down unexpectedly. For example, if the replication factor is 2, you get two copies of the documents in two nodes. For Elasticsearch, the default is 0.

cmx.task.search.records.return

Controls the Elasticsearch pagination when users search for tasks in the Task Manager of Data Director with business entities. Default is 1000.

Change the value only when suggested by Informatica Global Customer Support.

cmx.server.batch.smartsearch.initial.block_size

Maximum number of records that the Initially Index Smart Search Data batch job can process in each block. Default is 250. When you index a large data set, increase the number of records. The recommended maximum value is 1000.

cmx.server.enrichcopager.thread_pool

Manually add the property. Sets the number of threads the EnrichCoPager property uses from the thread pool to perform parallel ReadCO operations. Default is 30. If you set the number of threads to 1, the property is disabled.

cmx.server.enrichcopager.min_rec_for_multithreading

Sets the minimum number of records to return before the EnrichCoPager property uses multithreading. Default is 2.

ssl.keyStore

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Absolute path and file name of the keystore file.

ssl.keyStore.password

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Plain text password for the keystore file.

ssl.trustStore

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Absolute path and file name of the truststore file.

ssl.trustStore.password

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Plain text password for the truststore file.

After you update the Hub Server properties, you must validate the Operational Reference Store (ORS), and restart the Hub Console.

Configure the Process Server Properties

Configure all the Process Server instances to enable search. Use the Process Server tool in the Hub Console and the `cmxcleanse.properties` file to configure the Process Server properties for search.

1. In the Hub Console of a node, start the Process Server tool.
2. Click **Write Lock > Acquire Lock**.
3. In the right pane of the Process Server tool, click the **Add Process Server** button.
The **Add/Edit Process Server** dialog box appears.
4. Set the following properties of a Process Server for search:

Property	Description
Server	IP address or the fully qualified host name of the application server on which you deployed this Process Server. Note: Do not use <code>localhost</code> as the host name.
Port	HTTP or HTTPS port of the application server on which you deployed this Process Server.

Property	Description
Elasticsearch Processing	Indicates whether this Process Server handles the Initially Index Smart Search Data batch job. This batch job creates indexes for all the values of the searchable fields in a business entity.
Secured Connection (HTTPS)	Indicates whether this Process Server uses the HTTPS protocol. If selected, ensure that the Port option is set to an HTTPS port number.

- Click **OK**, and then click **Save**.
- Use a text editor to open the `cmxcleanse.properties` file in the following location:

```
<MDM Hub Installation Directory>\hub\cleanse\resources
```

- Configure the following properties for search:

cmx.ss.enabled

Indicates whether to use the Process Server to perform search. Default is false. Set to true if you want to use the Process Server to perform search.

ex.max.conn.per.host

Sets the maximum number of Elasticsearch nodes that you want to connect to the host. Set to the number of Elasticsearch cluster nodes on the host.

ex.max.threads

Sets the maximum number of threads that you want the Apache asynchronous non-blocking receiver to use for each node in the Elasticsearch cluster. Default is 1.

Change the value only when suggested by Informatica Global Customer Support.

search.provisioning.numreplicas

Optional. Number of copies of the Elasticsearch engine documents that you want to create on different nodes. Use the replication factor to create multiple copies of the documents in the shards of different nodes. You require multiple copies of the documents to achieve high availability if one or more nodes shut down unexpectedly. For example, if the replication factor is 2, you get two copies of the documents in two nodes. For Elasticsearch, the default is 0.

MAX_INITIAL_RESULT_SIZE_TO_CONSIDER

Optional. Manually add the property. Total number of search results to display in the Data Director application. The recommended maximum value is 250. Default is 130. Any value higher than 130 affects the performance of the Data Director application.

ssl.keyStore

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Absolute path and file name of the keystore file.

ssl.keyStore.password

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Plain text password for the keystore file.

ssl.trustStore

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Absolute path and file name of the truststore file.

ssl.trustStore.password

Required if you use the HTTPS port of the application server to configure the Process Server.
Manually add the property. Plain text password for the truststore file.

cmx.websphere.security.ssl.config.url

Required if you use the HTTPS port of the application server to configure the Process Server. For WebSphere only. Manually add the property. Absolute path of the `ssl.client.props` file with the file name.

8. Save the `cmxcleanse.properties` file.
9. Restart the application server.

Step 3. Configure Search by Using the Provisioning Tool

After you set up Elasticsearch and configure the MDM Hub properties, use the Provisioning tool to configure the search environment.

1. Configure the Elasticsearch cluster.
2. Optionally, create custom Elasticsearch index settings.
3. Configure the searchable fields.
4. Configure the search and query results display.
5. Optionally, configure the layout to display similar records.

Configure the Elasticsearch Cluster

Use the Provisioning tool to configure the Elasticsearch cluster for the MDM applications. The search APIs use the configuration. The Data Director application and any custom applications use the search APIs.

Note: When you configure the Elasticsearch cluster, only master nodes of the cluster must be specified.

1. Open a supported browser, and enter the following URL:
`https://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/`
The **Login** page appears.
2. Enter the user name and password, and click **Log In**.
3. From the **Database** list, select the database for which you want to configure the Elasticsearch cluster.
4. Click **Configuration > Infrastructure Settings**.
The **Infrastructure Settings** page appears.
5. Select **Elasticsearch Cluster** from the list, and then click **ESCluster**.
The **ESCluster** appears in the tree view panel.
6. To configure an Elasticsearch cluster node, select **esNode** in the tree view panel, and then click **Create**.

- Specify the following properties of the configured Elasticsearch cluster:

Property	Description
Name	Name of the master node in the Elasticsearch cluster.
URL	URL of the master node in the Elasticsearch cluster. The URL format is <code>https://<host name>:<port></code> .
Update Elasticsearch Node	<p>Indicates whether to update the node with the index details when the existing node continues to manage search requests. After you manually run the Initially Index Smart Search Data batch jobs so that the indexes are added to the new node, you can remove the existing node. You can then clear the check box so that the new node can manage search requests.</p> <p>Note: Applies when you upgrade from Elasticsearch Version 6 to Version 7. Elasticsearch authentication support is valid only for Version 7.x.</p> <p>For more information about installing and configuring Elasticsearch, see the <i>Multidomain MDM Configuration Guide</i>.</p>

- Click **Apply**.
- If you want to create additional master nodes, repeat steps [6](#) through [8](#).
- Publish the changes to the MDM Hub.
 - Click **Publish**.

A confirmation dialog box appears that prompts you to publish or review the changes.
 - Review the changes or publish without a review.
 - To publish without a review, click **Publish**.
 - To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.

Create Custom Elasticsearch Index Settings (Optional)

If the Elasticsearch index settings that Informatica provides do not meet your requirements, you can create custom index settings. The custom index settings must include analyzers, which convert text into tokens or terms that are added to the inverted index for searching.

An analyzer must have only one tokenizer, and can have zero or more character filters and token filters. Tokenizers receive a stream of characters that are converted into tokens. Token filters receive a stream of tokens generated by a tokenizer and might add, remove, or change tokens. Character filters receive a stream of characters and can add, remove, or change characters in the stream.

The tokenizer, token filters, and character filters that you use in a custom analyzer can be the Informatica default, custom, or Elasticsearch built-in components. You cannot edit the default settings. When you configure an analyzer, the Elasticsearch built-in tokenizers and token filters are available for selection.

For more information about Elasticsearch index settings, see the Elasticsearch documentation.

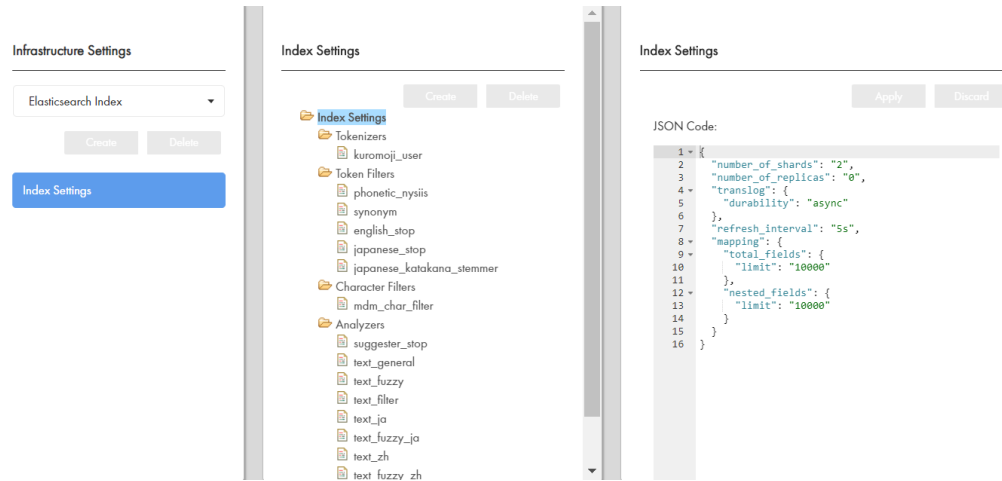
- Log in to the Provisioning tool.
- From the **Database** list, select the database for which you want to configure the Elasticsearch index settings.

3. Click **Configuration > Infrastructure Settings**.

The **Infrastructure Settings** page appears.

4. From the infrastructure settings list, select **Elasticsearch Index**, and click **Index Settings**.

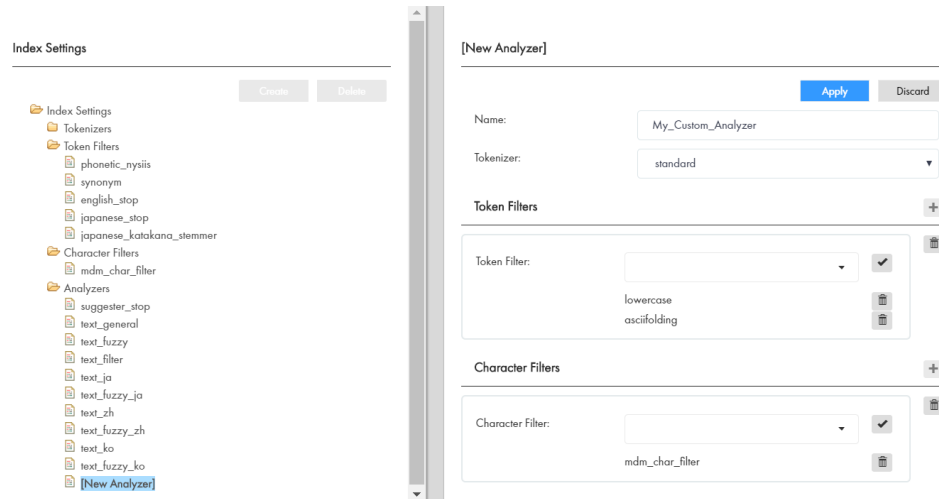
The **Index Settings** appear in the tree view panel, and the **JSON Code** box for the index settings appears in the properties panel. If the index settings were not changed, the page displays the default settings.



5. In the **JSON Code** box, enter index settings for modules other than the **Analysis** module. Also, enter the index settings that are not associated with any specific index module, such as the number of shards, number of replicas, and refresh interval.
6. Configure the analyzer components, such as a tokenizer, token filters, and character filters.
 - a. In the tree view panel, select the component that you want to configure, and click **Create**.
 - b. In the properties panel, enter the name and JSON code for the component.
 - c. Click **Apply**.
7. Configure an analyzer.
 - a. In the tree view panel, select **Analyzers**, and then click **Create**.
 - b. In the properties panel, specify the name, tokenizer, token filters, and character filters for the analyzer.

Ensure that you specify the token filters in the order in which you want the analyzer to use them.

The following image shows an example of a custom analyzer configuration:



- c. Click **Apply**.
8. Publish the changes to the MDM Hub.
 - a. Click **Publish**.
A confirmation dialog box appears that prompts you to publish or review the changes.
 - b. Review the changes or publish without a review.
 - To publish without a review, click **Publish**.
 - To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.
9. Review the application server logs for any validation errors related to the index settings and make changes.

Elasticsearch Built-In Tokenizers and Token Filters

You can select the available Elasticsearch built-in tokenizers and token filters for your custom analyzers.

The following Elasticsearch built-in tokenizers are available for the custom analyzers:

- standard
- letter
- lowercase
- whitespace
- uax_url_email
- classic
- thai
- keyword

The following Elasticsearch built-in token filters are available for the custom analyzers:

- asciifolding
- standard
- lowercase

- uppercase
- porter_stem
- trim
- cjk_width
- cjk_bigram
- classic
- apostrophe
- kuromoji_baseform

For more information about the custom and built-in Elasticsearch analyzer components, see the Elasticsearch documentation.

Configure the Searchable Fields

You can use the Provisioning tool to configure a field as a searchable field and set the field properties. A search request searches only the fields that you configure as searchable fields.

Multiple searchable fields might affect the performance of the search requests, so configure only the significant fields as searchable fields. For example, configure fields that contain full names, organization names, or email addresses as searchable fields rather than fields that contain country codes, gender codes, or address types.

1. Log in to the Provisioning tool.
2. From the **Database** list, select the database for which you want to configure the fields.
3. Click **Business Entity > Modeling**.
The **Modeling** page appears.
4. Select **Business Entities** from the list, and select the business entity for which you want to configure the searchable fields.
5. In the tree view panel, select **Fields**, and click **Create**.
6. Configure the following properties based on your requirement:

Name

Name of the field that that you want to appear in the tree view panel.

Label

Label for the field that you want to appear within views in Data Director.

Read Only

Optional. Indicates whether you can edit the field in the Entity View. To configure the field as a non-editable field, select the property.

Required

Optional. Indicates whether the field is a required field. To configure the field as a required field, select **Required**. By default, the field is not a required field.

URI

Defines the namespace where data types are declared. The data type that you want to configure depends on the URI that you select.

To configure basic data types, such as string, integer, and boolean, select `commonj.sdo`. To configure Informatica data types, such as image URL, hyperlink, and file attachment, select `urn:co-types.informatica.mdm`.

Type

The data type of the field. The data type that you can configure depends on the URI that you select. You can configure the following data types: string, integer, decimal, date, boolean, image URL, hyperlink, and file attachment.

By default, the data type of a business entity field is as close as possible to the data type of the base object column that the field is associated with. For example, a string column in a base object might contain information for an image, so you configure the image URL data type for the business entity field. If you want a field to display web URL, file URI, FTP link, and email address as hyperlinks, configure the hyperlink data type for the field.

Note: You cannot open file URIs with Google Chrome and Mozilla Firefox because of security reasons.

Display Format

The display format for the date fields. Before you specify the display format for the date fields, specify the URI and set the Type property to **Date**.

You can select the **Date** or **Datetime** format for business entity fields that display a date. The display format that you configure is applied to the date field in all the Data Director views.

If a display format is not set for a date field, the date format of the field is the same as the format of the associated base object column.

Filter

Defines a static filter to regulate the data that users can or cannot enter in a field. When you enable the Filter property, specify the following fields:

- **Operator.** Controls whether the specified values are allowed or not allowed in the field.

Option	Description
In	Displays the values to the user. If you add a list of comma-separated values as allowed values, Data Director users can choose a value from a lookup list. If you also configure a field filter on the same field, the displayed values reflect the intersection of the static filter and the field filter.
Not In	Does not display the values to the user.

- **Value.** Add values that match the data type of the field. To add multiple values, enter a comma-separated list of values. To allow users to leave the field blank, add empty quotes ("") as a value.

Note: The Provisioning tool does not validate the values. If you add a value of the wrong data type, the value does not appear in Data Director.

Column

Name of the base object column that you want to associate with the field.

7. Select **Searchable**.

Additional field properties appear.

8. Based on your requirement, select one or more of the following properties:
 - Search Analyzer
 - Suggester
 - Sortable
 - Filterable
 - Facet Range
 - Facet
 - Displayable
9. Optionally, if you select **Facet**, in the **Facet Range** field, specify the range for the numeric or date fields that you configure as facets in the following format:
`<Start Value>,<End Value>,<Frequency>`
For example, 1000,2000,50
Note: The facet ranges are not visible in the Data Director application. When you use the REST web service to perform the search, the response can return the facet ranges.
10. Click **Apply**.
11. Publish the changes to the MDM Hub.
 - a. Click **Publish**.
A confirmation dialog box appears that prompts you to publish or review the changes.
 - b. Review the changes or publish without a review.
 - To publish without a review, click **Publish**.
 - To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.

Searchable Field Properties

To configure the searchable field properties, you can use the Provisioning tool or apply a change list to the repository.

When you enable a searchable reference entity field as a filterable facet, the Data Director record view displays the filter field label in the following format:

`<Business entity lookup field label> - <Reference entity lookup field label>`

If you configure the Filterable property for a reference entity, for filtering to work, ensure that you configure the Filterable property for all the dependent reference entities.

The following table describes the searchable field properties:

Property	Description
Searchable	<p>Indicates whether a search request can search the field for a search string. To include the field in search requests, enable the property. If you do not want to include the field in search requests, disable the property.</p> <p>When you enable the Searchable property, you can configure additional properties for search.</p> <p>Note: System columns cannot be configured as searchable except for the consolidationInd and hubStateInd columns.</p> <p>The following additional properties are available for configuration:</p> <ul style="list-style-type: none"> - Search Analyzer - Suggester - Sortable - Filterable - Facet Range - Facet - Displayable
Search Analyzer	Specifies the custom search analyzer that you want to use for the field. Based on the type of data that the field will contain, decide on a suitable search analyzer.
Suggester	<p>Indicates whether you want to suggest the values of the field as the search string in the Data Director application. To suggest the values of the field as the search string, enable the property. If you do not want to suggest the values of the field as the search string, disable the property.</p> <p>Important: To ensure data security, do not enable the Suggester property for fields that contain sensitive data.</p>
Sortable	Do not use this property.
Filterable	Indicates whether you want to enable filtering on a field. The Data Director application displays the filterable fields as filters in the Search workspace. To configure the field as a filter, enable the property. If you do not want to configure the field as a filter, disable the property.
Facet Range	<p>Indicates the range for the numeric or date fields that you configure as facets. Use the following format to specify the range:</p> <p><Start Value>,<End Value>,<Frequency></p> <p>The start value is inclusive, and the end value is exclusive in the range. For example, if you set the facet range to 1000,2000,500 for an integer field, a search request returns the following ranges:</p> <pre>[1000 to 1500] [1500 to 2000]</pre> <p>The range 1000 to 1500 includes the values from 1000 to 1499, and the range 1500 to 2000 includes the values from 1500 to 1999.</p> <p>Ensure that you set a valid minimum and maximum value for the range and an offset that limits the number of ranges to 10.</p> <p>Facets cannot be configured for negative numbers, but a search request still displays negative values.</p> <p>For a date field, suffix Y M D to the frequency, where Y indicates year, M indicates month, and D indicates day. For example, 2M indicates 2 months.</p>

Property	Description
Facet	<p>Indicates whether you want to set the field as a facet. A facet field groups the search result values and shows the count of each group.</p> <p>The Data Director application displays the facet fields, the field values grouped based on the search results, and the count of each group in the Search workspace.</p> <p>When a child record field is set as a facet field, the Data Director application displays a tooltip for the facet field. The format of the tooltip text is <Child record name>/<Child record field name>.</p> <p>The Facet property functions in conjunction with the Filterable property, so enable the Filterable property if you want to configure the field as a facet. If you do not want to configure the field as a facet, disable the Facet property.</p> <p>Note: When a data security filter is configured for a facet field, faceting is disabled for the field to protect sensitive data.</p> <p>Enable the Facet property, if you want to configure the Facet Label Format menu options, used for the facet values on the search page.</p> <p>The following options are available:</p> <ul style="list-style-type: none"> - Lowercase - Displays the facet values in lowercase. For example, john autumn. - Uppercase - Displays the facet values in uppercase. For example, JOHN AUTUMN. - Title Case - Displays the facet values title case. For example, John Autumn.
Displayable	Do not use this property.

Configure the Search or Query Results Display

You can use the Provisioning tool to configure the business entity views that you want to use for search. A search result includes only the fields that are part of the business entity view that you configure for search results. You can also configure the order in which the search filters appear.

Before you configure the searchable views, create the business entity views that you want to use for the search results.

Note: To display child record fields of a business entity in the search results, use a business entity view that is transformed from a business entity. Ensure that the view includes child record fields at the root record level.

1. Log in to the Provisioning tool.
2. From the **Database** list, select the database with which your application is associated.
3. Click **Configuration > Application Editor**.
The **Applications** page appears.
4. From the **Applications** list, select the application for which you want to configure search.
If you do not have an application, create one before you can configure search.
5. In the tree view panel, select **Search Configuration**, and click **Create**.
6. In the properties panel, select a business entity and the business entity view that you want to use to display the search or query results.
If you do not select a business entity view, the search and query results contain all the business entity fields.

7. Optionally, if you configured search, select the filters and configure the display order of the search filters.
 - a. Click the **Edit** icon next to **Filter Display Order**.

The **Edit Filter Display Order** dialog box appears. The dialog box contains filters, which are fields that are configured as filterable in the business entity model.
 - b. Drag the filters from the **Available Filters** section to the **Selected Filters** section.
 - c. To configure the order, drag and move the filters up or down.
 - d. Click **OK**.
8. Click **Apply**.

The search configuration is saved to the temporary workspace.
9. Publish the changes to the MDM Hub.
 - a. Click **Publish**.

A confirmation dialog box appears that prompts you to publish or review the changes.
 - b. Review the changes or publish without a review.
 - To publish without a review, click **Publish**.
 - To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.

Configure the Layout to Display Similar Records (Optional)

When you enter data in the Data Director application to create a record, you can view similar records that are retrieved based on the data that you enter. To view the similar records, you must configure the layout to define the fields based on which you want to search for the similar records.

1. Log in to the Provisioning tool.
2. From the **Database** list, select the database for which you want to configure the application.
3. Click **Configuration > Component Editor**.

The **Component Editor** appears.
4. From the Component type list, select **Similar Records**, and click **Create**.
5. In the properties panel, enter a name for the Similar Records component.
6. In the **XML** field, enter an XML configuration that includes a list of fields to search for similar records:

The following table describes the XML elements that you can use to configure the Similar Records component:

Element	Description
searchableFields	Specifies one or more fields on which you want to base the search. The <code>searchableFields</code> element is the parent of the <code>field name</code> element.
field name	Specifies the name of a field on which you want to base the search for similar records. The <code>field name</code> element is a child of the <code>searchableFields</code> element. You can configure multiple <code>field name</code> elements.

Element	Description
searchType	Specifies the type of search to perform. The <code>searchType</code> element can contain the following child elements: <ul style="list-style-type: none"> - <code>smartSearch</code> - <code>searchMatch</code>
smartSearch	Specifies that you want to use search to find similar records.
searchMatch	Specifies that you want to use queries to find similar records. The <code>searchMatch</code> element can contain the following child elements: <ul style="list-style-type: none"> - <code>fuzzy</code> - <code>matchRuleSet</code>
fuzzy	Specifies whether you want to perform a fuzzy search. To perform a fuzzy search, set to <code>true</code> . The <code>fuzzy</code> element is a child of the <code>searchMatch</code> element. If you do not add this element, exact search is performed.
matchRuleSet	Specifies the name of the match rule set to use for finding similar records. The <code>matchRuleSet</code> element is a child of the <code>searchMatch</code> element.
label	Specifies the label format for the search field values. To configure the label format, use the <code>existsFormat</code> attribute.
column	Specifies a single column to use in the label format. To configure the column for the label, use the <code>columnUid</code> attribute, which is the unique identifier of the column. The <code>column</code> element is a child of the <code>label</code> element. You can specify more than one column for the label.

For sample configurations, see the *Multidomain MDM Provisioning Tool Guide*.

7. Click **Apply**.

The Similar Records component that you created appears in the **Component Editor** panel and in the tree view panel.

8. Publish the changes to the MDM Hub.

a. Click **Publish**.

A confirmation dialog box appears that prompts you to publish or review the changes.

b. Review the changes or publish without a review.

- To publish without a review, click **Publish**.
- To publish after a review, click **Review Changes** and follow the instructions that appear on the screen.

Step 4. Validate the Operational Reference Store

To validate the metadata of the Operational Reference Store (ORS) that is affected by the Elasticsearch configuration, use the Repository Manager tool in the Hub Console.

1. Start the Hub Console and connect to the MDM Hub Master Database.
2. Expand the **Configuration** workbench, and click **Repository Manager**.

- The Repository Manager appears.
3. Click the **Validate** tab, and select the repository to validate.
 4. Click **Validate**.
The **Select Validation Checks** dialog box appears.
 5. Select the validation checks to perform.
 6. Click **OK**.
The Repository Manager validates the repository and displays any issues in the **Issues Found** pane.
 7. To repair issues, click **Repair**.

Step 5. Index the Search Data

If your environment contains data, manually run the Initially Index Smart Search Data batch job to index the data. If your environment does not contain any data, you do not need to run the Initially Index Smart Search Data job. When you run the Load batch job to load data, the Load batch job automatically runs the Initially Index Smart Search Data batch job and indexes the data. A search request uses the indexes to search for records.

Run the Initially Index Smart Search Data batch job on all the base objects that contribute to the business entities. When you run the Initially Index Smart Search Data batch job on a base object, the Elasticsearch server indexes the data in the searchable fields. The job then adds the indexed data to all the collections that represent the business entities to which the searchable fields belong. If a collection is too large, you can split the collection into one or more shards. Shards are the logical pieces of a collection split over multiple nodes. When you perform a search, the Elasticsearch server reads the collections and returns the matching fields.

The Initially Index Smart Search Data batch job indexes the records asynchronously and reports successful completion after the job queues the indexing request for all the records. A search request can show the indexed records only after the successful completion of the index request, which might take a few minutes.

Important: If you update the searchable properties of a field after you index your data, the indexes are deleted. You must run the Initially Index Smart Search Data batch job to index the data. In addition, the indexing process is a resource-intensive process, so do not run multiple Initially Index Smart Search Data batch jobs in parallel.

Creating Keystores, Truststore, and Certificates (Optional)

After you install Elasticsearch, you can create keystores, truststore, and security certificates that are required to secure the communication between the MDM Hub and Elasticsearch. To create keystores, truststore, and certificates, run the `sip_ant` script on only one of the machines that has the Hub Server installed. Then, copy the keystores, truststore, and certificates to all the other machines on which the Hub Server is installed.

Note: You can create keystores, truststore, and certificates without using the `sip_ant` script.

The following table describes the keystores and truststore that are required:

Keystore/Truststore Name	Description
MDM_ESCLIENT_FILE_JKS.keystore	Elasticsearch keystore that contains the client certificate and its key.
MDM_ESKEYSTORE_FILE_JKS.keystore	Elasticsearch keystore that contains the client and node certificates. If the Elasticsearch cluster has multiple nodes, all the nodes use the certificates.
MDM ESTRUSTSTORE_FILE_JKS.keystore	Elasticsearch truststore that contains the signed certificate for the client and Elasticsearch nodes.

1. Open a command prompt, and navigate to the following directory on one of the machines that has the Hub Server installed:
`<MDM Hub installation directory>/hub/server/bin`
2. To create the keystores, truststore, and certificates, run the following command:
On UNIX. `sip_ant.sh generate_mdm_es_store`
On Windows. `sip_ant.bat generate_mdm_es_store`
3. When prompted for a password for the keystores and truststore, specify a password.
The keystores, truststore, and certificates are created in the following directory:
`<MDM Hub installation directory>/hub/server/resources/certificates`
4. Copy the following keystores and truststore to the `<Elasticsearch installation directory>/config` directory of each Elasticsearch installation:
 - MDM_ESCLIENT_FILE_JKS.keystore
 - MDM_ESKEYSTORE_FILE_JKS.keystore
 - MDM ESTRUSTSTORE_FILE_JKS.keystore
5. Copy the following keystore and truststore to the `<MDM Hub installation directory>/hub/server/resources/certificates` directory of each Hub Server node that is part of the Elasticsearch cluster:
 - MDM_ESCLIENT_FILE_JKS.keystore
 - MDM ESTRUSTSTORE_FILE_JKS.keystore

CHAPTER 24

Configuring the Consolidate Process

This chapter includes the following topics:

- [Configuring the Consolidate Process Overview, 483](#)
- [Consolidation Settings, 483](#)
- [Changing Consolidation Settings, 487](#)

Configuring the Consolidate Process Overview

The consolidate process merges the match pairs into a single master record. After you configure the match process, configure the consolidate process for the MDM Hub implementation.

To configure the consolidate process, use the Merge Settings tab in the Match/Merge Setup Details page. You can specify the source system characteristics and specify how to unmerge records.

When Business Entity Services or Data Director is used for merging records, users can override values in the records to be merged. If a user overrides values in the records to be merged and a workflow is configured, the details of the pending records that are awaiting a task action are stored in a pending control table.

Consolidation Settings

Consolidation settings affect the behavior of the consolidate process in Informatica MDM Hub. This section describes the settings that you can configure on the Merge Settings tab in the Match/Merge Setup Details page.

Immutable Rowid Object

For a given base object, you can designate a source system as an immutable source, which means that records from that source system are accepted as unique (`CONSOLIDATION_IND = 1`), even in the event of a merge. Once a record from that source is fully consolidated, it is not changed subsequently, nor is it matched to any other record (although other records can be matched to it). Only one source system can be configured as an immutable source.

Note: If the Requeue on Parent Merge setting for a child base object is set to UNCONSOLIDATED ONLY, in the event of a merging parent, the consolidation indicator is set to 4 for the child record, except for records with consolidation indicator set to 1. To requeue child records with consolidation indicator set to 1, the Requeue on Parent Merge setting must be manually set to 2.

Immutable sources are also distinct systems. All records are stored in the Informatica MDM Hub as master records. For all source records from an immutable source system, the consolidation indicator for Load and PUT is always 1 (consolidated record).

To specify an immutable source for a base object, click the drop-down list next to Immutable Rowid Object and select a source system.

This list displays the source system(s) associated with this base object. Only one source system can be designated an immutable source system.

Immutable source systems are applicable when, for example, Informatica MDM Hub is the only persistent store for the source data. Designating an immutable source system streamlines the load, match, and merge processes by preventing intra-source matches and automatically accepting records from immutable sources as unique. If two immutable records must be merged, then a data steward needs to perform a manual verification in order to allow that change. At that point, Informatica MDM Hub allows the data steward to choose the key that remains.

Distinct Systems

A *distinct system* provides data that gets inserted into the base object without being consolidated. Records from a distinct system will never match with other records from the *same* system, but they can be matched to and from other records in other systems (their CONSOLIDATION_IND is set to 4 on load). You can specify distinct source systems and configure whether, for each source system, records are consolidated automatically or manually.

Distinct Source Systems

You can designate a source system as a *distinct source* (also known as a *golden source*), which means that records from that source will not be merged. For example, if the ABC source has been designated as a distinct source, then the match rules will never match (or merge) two records that come from the same source. Records from a distinct source will not match through a transient match in an Auto Match and Merge process. Such records can be merged only manually by flagging them as matches.

To designate a distinct source system:

1. From the list of source systems on the Merge Settings tab, select (check) any source system that should not allow intra-system merges to prevent records from merging.
2. For each distinct source system, designate whether you want it to use Auto Rules only.

Auto Rules Only

For distinct systems only, you can enable this option to allow you to configure what types of rules are executed for the associated distinct source system. Check (select) this check box if you want Informatica MDM Hub to apply only the automatic consolidation rules (not the manual consolidation rules) for this distinct system. By default, this option is disabled (unchecked).

Unmerge Child When Parent Unmerges (Cascade Unmerge)

Important: This feature applies only to child base objects with configured match rules and foreign keys.

For child base objects, Informatica MDM Hub provides a *cascade unmerge* feature that allows you to specify what happens if records in the parent base object are unmerged. By default, this feature is disabled, so that unmerging parent records does not unmerge associated child records. In the Unmerge Child When Parent Unmerges portion near the bottom of the Merge Settings tab, if you check (select) the Cascade Unmerge check box for a child base object, when records in the parent object are unmerged, Informatica MDM Hub also unmerges affected records in the child base object.

Prerequisites for Cascade Unmerge

To enable cascade unmerge:

- the parent-child relationship must already be configured in the child base object
- the foreign key column in the child base object must be a match-enabled column

In the Unmerge Child When Parent Unmerges portion near the bottom of the Merge Settings tab, the Schema Manager displays only those match-enabled columns in the child base object that are configured with a foreign key.

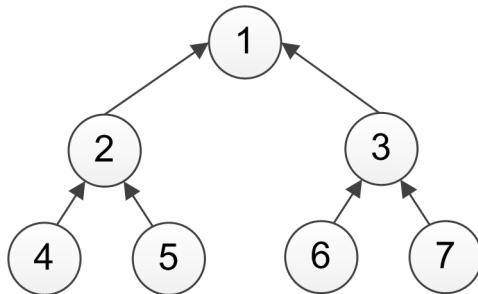
Child Base Object Cascade Unmerge Behavior

The affect that the cascade unmerge process has on a child base object depends on whether the parent underwent the tree unmerge process or the linear unmerge process.

The following list shows the the configuration of the parent base object record and the child base object record before and after the cascade unmerge processes:

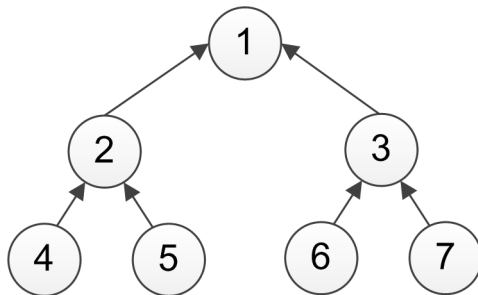
Merged parent base object record

The following graphic shows a parent base object record '1', which is the best version of the truth that consists of data from records '2' through '7':



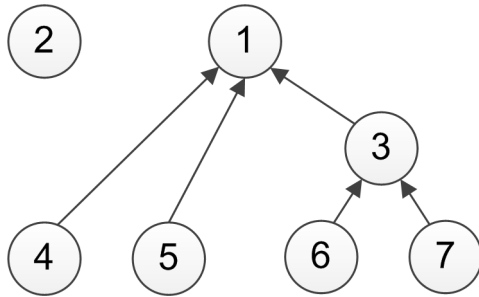
Merged child base object record

The following graphic shows the child base object record, '1', that has the same data structure as the parent base object record. The child base object record generally has a different structure from the parent base object record.



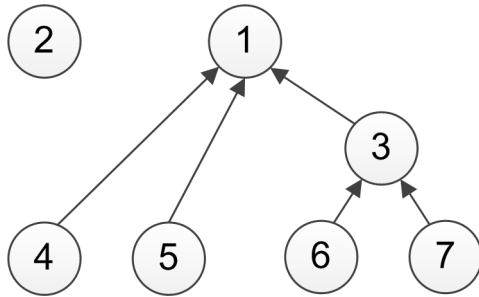
Parent base object after a linear unmerge

The following graphic shows record '2' as a separate base object after it undergoes the linear unmerge process:



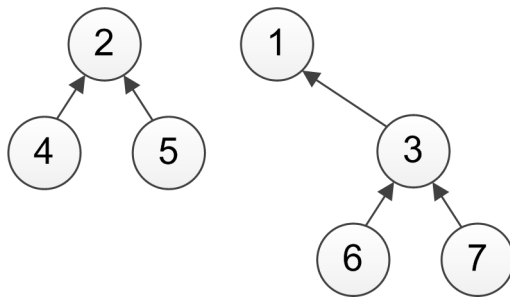
Child base object after a linear unmerge

When the parent base object record undergoes the linear unmerge process, the child base object record undergoes the same process. The following graphic shows that the affect of the parent base object record linear unmerge process has on the child base object record:



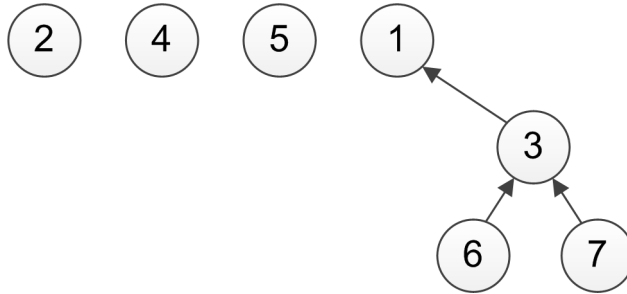
Parent base object after a tree unmerge

The following graphic shows that record '2' is a separate base object that maintains its tree structure after it undergoes the tree unmerge process.



Child base object after a tree unmerge

When the parent base object record undergoes the tree unmerge process, the unmerged tree in the child base object undergoes the linear unmerge process. The following graphic shows the affect that the parent base object record tree unmerge process has on the child base object record:



Parents with Multiple Children

In situations where a parent base object has multiple child base objects, you can explicitly enable cascade unmerge for each child base object. Once configured, when the parent base object is unmerged, then all affected records in all associated child base objects are unmerged as well.

Considerations for Using Cascade Unmerge

A full unmerge of affected records is not required in all implementations, and it can have a performance overhead on the unmerge because many child records can be affected. In addition, it does not always make sense to enable this property. One example is when Customer is a child of Customer Type. In this situation, you might not want to unmerge Customers if Customer Type is unmerged. However, in most cases, it is a good idea to unmerge addresses linked to customers if Customer unmerges.

Note: When cascade unmerge is enabled, the child record may not be unmerged if a previous manual unmerge was done on the child base object.

When you enable the unmerge feature, it applies to the child table and the child cross-reference table. Once enabled, if you then unmerge the parent cross-reference, the original child cross-reference should be unmerged as well. This feature has no impact on the parent—the feature operates on the child tables to provide additional flexibility.

Changing Consolidation Settings

To change consolidation settings on the Merge Settings tab:

1. In the Schema Manager, display the Match/Merge Setup Details dialog for the base object that you want to configure.
2. Acquire a write lock.
3. Click the **Merge Settings** tab.
The Schema Manager displays the Merge Settings tab for the selected base object.
4. Change any of the following settings:
 - Immutable Rowid Object

- Distinct Systems
 - Unmerge Child When Parent Unmerges (Cascade Unmerge)
5. Click the **Save** button to save your changes.

CHAPTER 25

Pending Control Table

The pending control table is a system table that the MDM Hub automatically creates for a base object. The table tracks information about records with overrides to trusted values that are waiting for a merge task action. The format of the table name is C_<base object name>_PCTL. The pending control table contains all the columns of a control table and an additional column to store the interaction ID.

The pending control table is used when a task workflow is configured for merge operations. The table is used by Business Entity Services APIs only and not by Services Integration Framework (SIF) APIs and batch jobs.

When users manually override trusted values in a record, the merge process inserts the records into the pending control table. The records that are pending a merge action are protected by an interaction ID. Each trust-enabled column of the records have four columns in the pending control table. When users manually enter correct values, a cross-reference record is created in the pending state. When the merge task is approved, the pending cross-reference records and the pending overrides are promoted and the records are merged.

The following table describes the columns in the pending control table:

Column Name	Description
INTERACTION_ID	An identifier that groups the related trust overrides that are part of a merge task. Required for a merge task action to complete.
<trust-enabled column name>_LRS	Last row ID of the source system that provides the most recent update to the base object record.
<trust-enabled column name>_LUD	Last update date of the cross-reference record that provided the final value for the record.
<trust-enabled column name>_SRX	Identifier of the cross-reference record that provided the final value for the record.
<trust-enabled column name>_OTS	Encoded trust settings to facilitate trust overrides.

If history is enabled for a base object, the MDM Hub maintains a separate history table for the pending control table. The name format of the history table associated with the pending control table is C_<base object name>_HPCT.

CHAPTER 26

Configuring the Publish Process

This chapter includes the following topics:

- [Publish Process Overview, 490](#)
- [Configuration Steps for the Publish Process, 491](#)
- [Starting the Message Queues Tool, 491](#)
- [Configuring Global Message Queue Settings, 491](#)
- [Configuring Message Queue Servers, 492](#)
- [Configuring Outbound Message Queues, 494](#)
- [Configuring Parallel Processing of JMS Messages, 496](#)
- [Configuring JMS Security, 496](#)
- [Message Trigger Configuration, 496](#)
- [Disabling Message Queue Polling, 501](#)
- [JMS Message XML Reference, 501](#)
- [Legacy JMS Message XML Reference, 514](#)

Publish Process Overview

You can configure the MDM Hub publish process to generate XML messages about data changes in the Hub Store and publish the messages to an outbound Java Messaging System (JMS) message queue. External applications can retrieve the XML messages that the MDM Hub publishes to the JMS message queue.

Before you configure the publish process, set up the JMS message queue and the connection factory for the application servers on which the Hub Server and the Process Server are deployed. If you want to enhance the performance of the publish process, configure parallel processing for the JMS messages.

Important: Services Integration framework (SIF) uses a message-driven bean on the JMS message queue, `siperian.sif.jms.queue`, to process incoming asynchronous SIF requests. The `siperian.sif.jms.queue` JMS message queue is set up during the MDM Hub installation process and is an absolute requirement for all MDM Hub installations. The JMS message queues that you configure for the MDM Hub publish process are required if you use external applications to retrieve JMS messages from the JMS message queue.

Configuration Steps for the Publish Process

After installing Informatica MDM Hub, you use the Message Queues tool in the Hub Console to configure message queues for your Informatica MDM Hub implementation. The following tasks are mandatory if you want to publish events in the outbound message queue:

1. Configure the message queues on your application server.
The Informatica MDM Hub installer automatically sets up message queues and the connection factory configuration. For more information, see the *Multidomain MDM Installation Guide* for your platform.
2. Configure global message queue settings.
3. Add at least one message queue server.
4. Add at least one message queue to the message queue server.
5. Generate the JMS event message schema for each ORS that has data that you want to publish.
6. Configure message triggers for your message queues.

After you have configured message queues, you can review run-time activities using the Audit Manager.

Starting the Message Queues Tool

1. In the Hub Console, connect to the Master Database.
Message queues are defined in the Master Database.
2. In the Hub Console, expand the Configuration workbench, and then click **Message Queues**.
The Hub Console displays the Message Queues tool. The Message Queues tool is divided into two panes.

Pane	Description
Navigation pane	Shows (in a tree view) the message queues that are defined for this Informatica MDM Hub implementation.
Properties pane	Shows the properties for the selected message queue.

Configuring Global Message Queue Settings

To configure the global message queue settings for your Informatica MDM Hub implementation:

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. Specify settings for Data Changes Monitoring, which monitors the queue for outgoing messages.
4. To enable or disable Data Changes Monitoring, click the **Toggle Data Changes Monitoring Status** button.

Specify the following monitoring settings:

Monitoring Setting	Description
Receive Timeout(milliseconds)	Default is 0. Amount of time allowed to receive the messages from the queue.
Receive Batch Size	Default is 100. Maximum number of events processed and placed in the message queue in a single pass.
Message Check Interval (milliseconds)	Default is 300000. Amount of time to pause before polling for inbound messages or processing outbound messages. The same value applies to both inbound and outbound message queues.
Out of sync check interval (milliseconds)	<p>If configured, periodically polls for ORS metadata and regenerates the XML message schema if subsequent changes have been made to design objects in the ORS.</p> <p>By default, this feature is disabled—set to zero (0)—and is available only if: Data Changes Monitoring is enabled.</p> <p>ORS-specific XML message schema has been generated using the JMS Event Schema Manager.</p> <p>Note: Make sure that this value is greater than or equal to the Message Check Interval.</p>

5. Click the **Edit** button next to any property that you want to change.
6. Click the **Save** button to save your changes.

Configuring Message Queue Servers

Configure message queue servers for the MDM Hub implementation.

Before you define message queues in the MDM Hub, configure the message queue servers that the MDM Hub needs to use for handling message queues. Before you add message queue servers to the MDM Hub, configure message queue servers on the application server instances on which the Hub Server and the Process Servers are deployed. To define the message queue servers in the MDM Hub, you need the connection factory name of the message queue servers for JBoss and WebLogic and the server name and the port number of the message queue servers for WebSphere.

Message Queue Server Properties

This section describes the settings that you can configure for message queue servers.

WebLogic and JBoss Properties

You can configure the following message queue server properties.

Property	Description
Connection Factory Name	Name of the connection factory for this message queue server.
Display Name	Name of this message queue server as it will be displayed in the Hub Console.
Description	Descriptive information for this message queue server.

WebSphere Properties

IBM WebSphere implementations have the following properties.

Property	Description
Server Name	Name of the server where the message queue is defined.
Channel	Channel of the server where the message queue is defined.
Port	Port on the server where the message queue is defined.

Adding Message Queue Servers

Add a message queue server to connect to a message queue.

To add a message queue server:

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. Right-click anywhere in the navigation pane and choose **Add Message Queue Server**.
The Add Message Queue Server dialog box appears.
4. Specify the message queue server properties.

Editing Message Queue Server Properties

To edit the properties of an existing message queue server:

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. In the navigation pane, select the name of the message queue server that you want to configure.
4. Change the editable properties for this message queue server.
Click the **Edit** button next to any property that you want to change.
5. Click the **Save** button to save your changes.

Deleting Message Queue Servers

To delete an existing message queue server:

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. In the navigation pane, right-click the name of the message queue server that you want to delete, and then choose **Delete** from the pop-up menu.

The Message Queues tool prompts you to confirm deletion.

4. Click **Yes**.

Configuring Outbound Message Queues

This section describes how to configure outbound JMS message queues for your Informatica MDM Hub implementation.

About Message Queues

Before you can define outbound JMS message queues in Informatica MDM Hub, you must define the message queue server(s) that will service the message queue. In JMS, a message queue is a staging area for XML messages. Informatica MDM Hub publishes XML messages to the message queue. External applications retrieve these published XML messages from the message queue.

Message Queue Properties

You can configure the following message queue properties.

Property	Description
Queue Name	Name of this message queue. This must match the JNDI queue name as configured on your application server.
Display Name	Name of this message queue as it will be displayed in the Hub Console.
Description	Descriptive information for this message queue.

Adding Message Queues to a Message Queue Server

Add a message queue to which the message queue server must connect to.

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. In the navigation pane, right-click the name of the message queue server to which you want to add a message queue, and choose **Add Message Queue**.

The Add Message Queue dialog box appears.

4. Specify the message queue properties.

5. Click **OK**.

The Message Queues tool prompts you to choose the queue assignment.

6. Select one of the following queue assignment options:

Option	Description
Leave Unassigned	Queue is currently unassigned and not in use. Select the option to use the queue as the outbound queue for Informatica MDM Hub API responses, or to indicate that the queue is currently unassigned and is not in use.
Use with Message Queue Triggers	Queue is currently assigned and is available for use by message triggers that are defined in the Schema Manager.
Use Legacy XML	Informatica MDM Hub implementation requires you to use the legacy XML message format instead of the current version of the XML message format.

7. Click **Save**.

Editing Message Queue Properties

To edit the properties of an existing message queue:

1. In the Hub Console, start the Message Queues tool.
2. Acquire a write lock.
3. In the navigation pane, select the name of the message queue that you want to configure.
4. Change the editable properties for this message queue.
5. Click the **Edit** button next to any property that you want to change.
6. Change the queue assignment, if you want.
7. Click the **Save** button to save your changes.

Deleting Message Queues

Use the Message Queues tool to delete message queues.

Ensure that message triggers are not associated with the message queue that you want to delete. If a message trigger is associated with the message queue, delete the message trigger before you delete the message queue.

1. Start the **Message Queues** tool under the Configuration workbench.
2. Acquire a write lock.
3. In the navigation pane, right-click the name of the message queue that you want to delete, and then click **Delete** from the pop-up menu.

The Message Queues tool prompts you to confirm deletion.

4. Click **Yes**.

Configuring Parallel Processing of JMS Messages

If you configure parallel processing of JMS messages, the Hub Server distributes the message processing load in batches to multiple Process Servers. You must configure the batch size for parallel processing of JMS messages. Configure parallel processing of JMS messages in the Hub Server properties file.

1. Open the the `cmxserver.properties` file in the following directory:

`<MDM Hub installation directory>/hub/server/resources`

2. Add the properties required for parallel processing of JMS messages and save the file.

The following table describes the properties for parallel processing of JMS messages:

Property	Description
<code>mq.data.change.threads</code>	Number of threads to use to process JMS messages during the publish process. Default is 1.
<code>mq.data.change.batch.size</code>	Number of JMS messages to process in each batch for the publish process. Default is 500.
<code>mq.data.change.timeout</code>	Amount of time in seconds that is allowed to process the JMS messages. Default is 120.

Configuring JMS Security

You must configure JMS security to secure message queues.

You must first configure JMS security in the application server. Configure the MDM Hub to use the user credentials that you set in the application server by setting the following properties in the `cmxserver.properties` file:

```
<connection factory name>.qcf.username=<user name>
<connection factory name>.qcf.password=<password>
```

Message Trigger Configuration

Use the Schema Manager tool to configure message triggers for your MDM Hub implementation.

Message triggers identify the actions that the MDM Hub communicates to external applications. When an action occurs for which a rule is defined, the MDM Hub puts an XML message in a JMS message queue. A message trigger specifies the JMS message queue in which messages are placed.

For an example of how message triggers work, consider the following scenario:

1. You insert a record in a base object.
2. The insert action initiates a message trigger.
3. The MDM Hub evaluates the message trigger and sends a message to the appropriate message queue.
4. An outside application polls the message queue, picks up the message, and processes it.

You can use the same message queue for all triggers, or you can use a different message queue for each trigger. For an action to trigger a message trigger, configure the message queues and define a message trigger for that base object and action.

Types of Events for Message Triggers

The following types of events can cause a message trigger to be fired and a message placed in the queue.

The following table describes events for which you can define message queue rules:

Event	Description	Message Type Code
Add new data	Used to add data by using the following approaches: <ul style="list-style-type: none"> - Through the load process - By using the Data Manager - Through the API verb using PUT or CLEANSE_PUT through protocols such as HTTP, SOAP, and EJB. 	1
Add new pending data	A record is created with the PENDING state. Applies to state-enabled base objects.	10
Update existing data	Used to update data by using the following approaches: <ul style="list-style-type: none"> - Through the load process - By using the Data Manager - Through the API verb by using PUT or CLEANSE_PUT through protocols such as HTTP, SOAP, and EJB. <p>If trust rules prevent the base object columns to be updated, no message is generated.</p> <p>If one or more of the specified columns are updated, a single message is generated. The message includes data from all of the cross-reference records in all output systems.</p>	2
Update existing pending data	An existing record with a PENDING state is updated. Applies to state-enabled base objects.	11
Update, only XREF changed	Used to update data in the following scenarios: <ul style="list-style-type: none"> - When only the cross-reference has changed through the load process - When only the cross-reference has changed through the API using PUT or CLEANSE_PUT through protocols such as HTTP, SOAP, and EJB. 	6
Pending update, only XREF changed	A cross-reference record with a PENDING state is updated. Also, includes the promotion of a record. Applies to state-enabled base objects.	12
Merging data	Used to merge data by using the following approaches: <ul style="list-style-type: none"> - Through Merge Manager - Through the API Verb through protocols such as HTTP, SOAP, and EJB. - Through the Automatch and Merge process 	4

Event	Description	Message Type Code
Merging data, base object updated	<p>Used to merge data in the following scenarios:</p> <ul style="list-style-type: none"> - When the base object is updated - When records load by row ID to insert new cross-reference and the base object is updated <p>If trust rules prevent the base object columns to be merged, the MDM Hub does not generate any messages.</p> <p>If one or more of the specified columns are updated, a single message is generated. The message includes data from all of the cross-reference records in all output systems.</p>	7
Unmerging data	<p>Used to unmerge data by using the following approaches:</p> <ul style="list-style-type: none"> - Through the Data Manager - Through the API verb by using UNMERGE through protocols such as HTTP, SOAP, and EJB. 	8
Accepting data as unique	<p>Used to accept records as unique by using the following approaches:</p> <ul style="list-style-type: none"> - Through the Merge Manager - Through the Schema tool by enabling the option when you configure match rules <p>When a record is accepted as unique, either automatically through a match rule or manually by a data steward, the MDM Hub generates a message with the record information, including the cross-reference information for all output systems. The message is placed in the queue.</p>	3
Delete base object data	A base object record is soft deleted and the state is changed to DELETED. Applies to state-enabled base objects.	14
Delete XREF data	A cross-reference record is soft deleted and the state is changed to DELETED. Applies to state-enabled base objects.	15
Delete pending BO data	A base object record with a PENDING state is hard deleted. Applies to state-enabled base objects.	16
Delete pending XREF data	A cross-reference record with a PENDING state is hard deleted. Applies to state-enabled base objects.	17
Restore	A deleted cross-reference record is restored and the state is changed to ACTIVE. Applies to state-enabled base objects.	19

Best Practices for Message Triggers

Create message triggers for your implementation effectively with the help of suggested best practices.

- If a base object message trigger definition uses a message queue, the MDM Hub displays the following message: `The message queue is currently in use by message triggers.` In this case, you cannot edit the properties of the message queue. Instead, you must create another message queue to make the required changes.
- The MDM Hub installer creates a default JMS queue called `siperian.sif.jms.queue`. If you use this queue when you configure a message trigger, the MDM Hub generates an error.

- Message triggers apply to just one base object, and they run only when a specific action occurs directly on that base object. If you have two tables that are in a parent-child relationship, you must explicitly define message queues for each table separately. The MDM Hub detects specific changes made to a base object, such as load INSERT or PUT. Changes to a record of the parent table can cause a message trigger to run for the parent record only. If changes in the parent record affect associated child records, you must explicitly configure a separate message trigger for the child table.

Message Trigger System Properties

When you configure a message trigger, you must select at least one Triggering system and one In Message system.

A system for a message trigger has the following properties.

Triggering

Select the system or systems that triggers the action. You must select at least one.

In Message

Select the system or systems that displays the message. You must select at least one.

Each message in a message queue includes the `pkey_src_object` value for each cross-reference that it has in one of the 'In Message' systems.

For example, your implementation has three source systems: A, B, and C. You select system A as the Triggering system. A base object record has cross-reference records for A and B. You want to update the cross-reference in system A for this base object record. The following table shows possible message trigger configurations and the resulting message:

In Message Systems	Resulting Message
A	Message with cross-reference for system A.
B	Message with cross-reference for system B.
C	No message. No cross-references from In Message.
A and B	Message with cross-reference for systems A and B.
A and C	Message with cross-reference for system A.
B and C	Message with cross-reference for system B.
A, B, and C	Message with cross-reference for systems A and B.

Adding Message Triggers

1. Configure the message queue to be usable with message triggers.
2. Start the Schema Manager.
3. Acquire a write lock.
4. Expand the base object that you want to monitor, and select the **Message Trigger Setup** node.
If no message triggers have been created, the Schema Tool displays an empty screen.

5. Do one of the following:
 - If no message triggers have been defined, click **Add Message Trigger**.
 - If message triggers have been defined, click **Add**.The Schema Manager displays the Add Message Trigger wizard.
6. Specify a name and description for the new message trigger.
7. Click **Next**.

The Add Message Trigger wizard prompts you to specify the messaging package.
8. Select the package that will be used to build the message.
9. Click **Next**.

The Add Message Trigger wizard prompts you to specify the target message queue.
10. Select the message queue that carries the message.
11. Click **Next**.

The Add Message Trigger wizard prompts you to specify the rules for this message trigger.
12. Select the event types for this message trigger.
13. Configure the system properties for this message trigger. You must select at least one Triggering system and one In Message system.
14. Click **Next** if you have selected an Update option. Otherwise click **Finish**.

If you have clicked the Update action, the Schema Manager prompts you to select the columns to monitor for update actions.
15. Do one of the following:
 - Select the columns to monitor for the events associated with this message trigger, or
 - Select the **Trigger message if change on any column** check box to monitor all columns for updates.
16. Click **Finish**.

Editing Message Triggers

1. Start the Schema Manager.
2. Acquire a write lock.
3. Expand the base object that will be monitored, and select the **Message Trigger Setup** node.
4. In the Message Triggers list, click the message trigger that you want to configure.

The Schema Manager displays the settings for the selected message trigger.
5. Change the settings you want.
6. Click the **Edit** button next to editable property that you want to change.
7. Click the **Save** button to save your changes.

Deleting Message Triggers

1. Start the Schema Manager.
2. Acquire a write lock.
3. Expand the base object that will be monitored, and select the **Message Trigger Setup** node.
4. In the Message Triggers list, click the message trigger that you want to delete.

5. Click the **Delete** button.
The Schema Manager prompts you to confirm deletion.
6. Click **Yes**.

Disabling Message Queue Polling

In a multinode environment, you can disable message queue polling for individual nodes. To disable message queue polling, set the `mq.data.change.monitor.thread.start` property to `false` in the `cmxserver.properties` and `cmxcleanse.properties` files.

You can confirm the operation of message queue polling while in DEBUG mode.

- On nodes where the value of `mq.data.change.monitor.thread.start` is `false`, the message **Monitoring is disabled** appears in the logs.
- On nodes where the value of `mq.data.change.monitor.thread.start` is `true`, the message **Data changes monitoring started** appears in the logs.

By default, message queue polling is enabled on all Java virtual machines where an MDM Hub EAR file is deployed. When you use multiple poller threads for data change monitoring and for publishing messages, the messages might not publish in the correct sequence. To control the sequence in which the messages are published, use a single poller thread.

JMS Message XML Reference

This section describes the structure of Informatica MDM Hub XML messages and provides example messages.

Note: If your Informatica MDM Hub implementation requires that you use the legacy XML message format (Informatica MDM Hub XU version) instead of the current version of the XML message format (described in this section), see [“Legacy JMS Message XML Reference” on page 514](#) instead.

Generating ORS-specific XML Message Schemas

To create XML messages, the publish process relies on an ORS-specific schema file (`<ors-name>-siperian-mrm-event.xsd`) that you generate using the JMS Event Schema Manager tool in the Hub Console.

Elements in an XML Message

The following table describes the elements in an XML message.

Field	Description
Root Node	
<code><siperianEvent></code>	Root node in the XML message.

Field	Description
Event Metadata	
<eventMetadata>	Root node for event metadata.
<messageId>	Unique ID for siperianEvent messages.
<eventType>	Type of event with one of the following values: <ul style="list-style-type: none"> - Insert - Update - Update XREF - Accept as Unique - Merge - Unmerge - Merge Update
<baseObjectUid>	UID of the base object affected by this action.
<packageUid>	UID of the package associated with this action.
<messageDate>	Date/time when this message was generated.
<orsId>	ID of the Operational Reference Store (ORS) associated with this event.
<triggerUid>	UID of the rule that triggered the event that generated this message.
Event Details	
<eventTypeEvent>	Root node for event details.
<sourceSystemName>	Name of the source system associated with this event.
<sourceKey>	Value of the PKEY_SRC_OBJECT associated with this event.
<eventDate>	Date/time when the event was generated.
<rowid>	RowID of the base object record that was affected by the event.
<xrefKey>	Root node of a cross-reference record affected by this event.
<systemName>	System name of the cross-reference record affected by this event.
<sourceKey>	PKEY_SRC_OBJECT of the cross-reference record affected by this event.
<packageName>	Name of the secure package associated with this event.
<columnName>	Each column in the package is represented by an element in the XML file. Examples: rowidObject and consolidationInd. Defined in the ORS-specific XSD that is generated using the JMS Event Schema Manager tool.
<mergedRowid>	List of ROWID_OBJECT values for the losing records in the merge. This field is included in messages for Merge events only.

Filtering Messages

You can use the custom JMS header named `MessageType` to filter incoming messages based on the message type. The following message types are indicated in the message header.

Message Type	Description
siperianEvent	Event notification message.
<serviceNameReturn>	For Services Integration Framework (SIF) responses, the response begins with the name of the SIF request, as in the following fragment of a response to a get request: <getReturn> <message>The GET was executed successfully - retrieved 1 records</message> <recordKey> <ROWID>2</ROWID> </recordKey> ...

Example XML Messages

This section provides listings of example XML messages.

Accept As Unique Message

The following is an example of an Accept As Unique message:

```
<?xml version="1.0" encoding="UTF-8"?>
  <siperianEvent>
    <eventMetadata>
      <eventType>Accept as Unique</eventType>
      <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
      <packageUid>PACKAGE.CONTACT_PKG</packageUid>
      <orsId>localhost-mrm-CMX_ORIS</orsId>
      <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
      <messageId>192</messageId>
      <messageDate>2008-09-10T16:33:14.000-07:00</messageDate>
    </eventMetadata>
    <acceptAsUniqueEvent>
      <sourceSystemName>Admin</sourceSystemName>
      <sourceKey>SVR1.1T1</sourceKey>
      <eventDate>2008-09-10T16:33:14.000-07:00</eventDate>
      <rowid>2</rowid>
      <xrefKey>
        <systemName>Admin</systemName>
        <sourceKey>SVR1.1T1</sourceKey>
      </xrefKey>
      <contactPkg>
        <rowidObject>2</rowidObject>
        <creator>admin</creator>
        <createDate>2008-08-13T20:28:02.000-07:00</createDate>
        <updatedBy>admin</updatedBy>
        <lastUpdateDate>2008-09-10T16:33:14.000-07:00</lastUpdateDate>
        <consolidationInd>1</consolidationInd>
        <lastRowidSystem>SYS0</lastRowidSystem>
        <dirtyInd>0</dirtyInd>
        <firstName>Joey</firstName>
        <lastName>Brown</lastName>
      </contactPkg>
    </acceptAsUniqueEvent>
  </siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

AMRule Message

The following is an example of an AMRule message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>AM Rule Event</eventType>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
    <interactionId>12</interactionId>
    <activityName>Changed Contact and Address </activityName>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdateLegacy</triggerUid>
    <messageId>291</messageId>
    <messageDate>2008-09-19T11:43:42.979-07:00</messageDate>
  </eventMetadata>
  <amRuleEvent>
    <eventDate>2008-09-19T11:43:42.979-07:00</eventDate>
    <contactPkgAmEvent>
      <amRuleUid>AM_RULE.RuleSet1|Rule1</amRuleUid>
      <contactPkg>
        <rowidObject>64 </rowidObject>
        <creator>admin</creator>
        <createDate>2008-09-08T16:24:35.000-07:00</createDate>
        <updatedBy>admin</updatedBy>
        <lastUpdateDate>2008-09-18T16:26:45.000-07:00</lastUpdateDate>
        <consolidationInd>2</consolidationInd>
        <lastRowidSystem>SYS0 </lastRowidSystem>
        <dirtyInd>1</dirtyInd>
        <firstName>Johnny</firstName>
        <lastName>Brown</lastName>
        <hubStateInd>1</hubStateInd>
      </contactPkg>
      <cContact>
        <event>
          <eventType>Update</eventType>
          <system>Admin</system>
        </event>
        <event>
          <eventType>Update XREF</eventType>
          <system>Admin</system>
        </event>
        <xrefKey>
          <systemName>CRM</systemName>
          <sourceKey>PK1265</sourceKey>
        </xrefKey>
        <xrefKey>
          <systemName>Admin</systemName>
          <sourceKey>64</sourceKey>
        </xrefKey>
      </cContact>
    </contactPkgAmEvent>
  </amRuleEvent>
</siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

BoDelete Message

The following is an example of a BoDelete message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>BO Delete</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
```

```

    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>328</messageId>
    <messageDate>2008-09-19T14:35:53.000-07:00</messageDate>
  </eventMetadata>
  <boDeleteEvent>
    <sourceSystemName>Admin</sourceSystemName>
    <eventDate>2008-09-19T14:35:53.000-07:00</eventDate>
    <rowid>107          </rowid>
    <xrefKey>
      <systemName>CRM</systemName>
    </xrefKey>
    <xrefKey>
      <systemName>Admin</systemName>
    </xrefKey>
    <xrefKey>
      <systemName>WEB</systemName>
    </xrefKey>
    <contactPkg>
      <rowidObject>107          </rowidObject>
      <creator>sifuser</creator>
      <createDate>2008-09-19T14:35:28.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-19T14:35:53.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM          </lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <hubStateInd>-1</hubStateInd>
    </contactPkg>
  </boDeleteEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

BoSetToDelete Message

The following is an example of a BoSetToDelete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>BO set to Delete</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>319</messageId>
    <messageDate>2008-09-19T14:21:03.000-07:00</messageDate>
  </eventMetadata>
  <boSetToDeleteEvent>
    <sourceSystemName>Admin</sourceSystemName>
    <eventDate>2008-09-19T14:21:03.000-07:00</eventDate>
    <rowid>102          </rowid>
    <xrefKey>
      <systemName>CRM</systemName>
    </xrefKey>
    <xrefKey>
      <systemName>Admin</systemName>
    </xrefKey>
    <xrefKey>
      <systemName>WEB</systemName>
    </xrefKey>
    <contactPkg>
      <rowidObject>102          </rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-19T13:57:09.000-07:00</createDate>

```

```

        <updatedBy>admin</updatedBy>
        <lastUpdateDate>2008-09-19T14:21:03.000-07:00</lastUpdateDate>
        <consolidationInd>4</consolidationInd>
        <lastRowidSystem>SYS0                </lastRowidSystem>
        <dirtyInd>1</dirtyInd>
        <hubStateInd>-1</hubStateInd>
    </contactPkg>
</boSetToDeleteEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Delete Message

The following code is an example of a delete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Delete</ACTION>
    <MESSAGE_ID>11010297</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:35:53.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>107</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_ORS</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>WEB</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>107</ROWID_OBJECT>
      <CREATOR>sifuser</CREATOR>
      <CREATE_DATE>19 Sep 2008 14:35:28</CREATE_DATE>
      <UPDATED_BY>admin</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 14:35:53</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <DELETED_IND />
      <DELETED_BY />
      <DELETED_DATE />
      <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
      <INTERACTION_ID />
      <FIRST_NAME>John</FIRST_NAME>
      <LAST_NAME>Smith</LAST_NAME>
      <HUB_STATE_IND>-1</HUB_STATE_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Insert Message

The following is an example of an Insert message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Insert</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdateLegacy</triggerUid>
    <messageId>114</messageId>
    <messageDate>2008-09-08T16:02:11.000-07:00</messageDate>
  </eventMetadata>
  <insertEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>PK12658</sourceKey>
    <eventDate>2008-09-08T16:02:11.000-07:00</eventDate>
    <rowid>66</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>PK12658</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>66</rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-08T16:02:11.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-08T16:02:11.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM</lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>Joe</firstName>
      <lastName>Brown</lastName>
    </contactPkg>
  </insertEvent>
</siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Merge Message

The following is an example of a Merge message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Merge</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdateLegacy</triggerUid>
    <messageId>130</messageId>
    <messageDate>2008-09-08T16:13:28.000-07:00</messageDate>
  </eventMetadata>
  <mergeEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>PK126566</sourceKey>
    <eventDate>2008-09-08T16:13:28.000-07:00</eventDate>
    <rowid>65</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>PK126566</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>Admin</systemName>
      <sourceKey>SVR1.28E</sourceKey>
    </xrefKey>
  </mergeEvent>
</siperianEvent>
```

```

</xrefKey>
<mergedRowid>62</mergedRowid>
<contactPkg>
  <rowidObject>65</rowidObject>
  <creator>admin</creator>
  <createDate>2008-09-08T15:49:17.000-07:00</createDate>
  <updatedBy>admin</updatedBy>
  <lastUpdateDate>2008-09-08T16:13:28.000-07:00</lastUpdateDate>
  <consolidationInd>4</consolidationInd>
  <lastRowidSystem>SYS0</lastRowidSystem>
  <dirtyInd>1</dirtyInd>
  <firstName>Joe</firstName>
  <lastName>Brown</lastName>
</contactPkg>
</mergeEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Merge Update Message

The following is an example of a Merge Update message:

```

<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Merge Update</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>269</messageId>
    <messageDate>2008-09-10T17:25:42.000-07:00</messageDate>
  </eventMetadata>
  <mergeUpdateEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>P45678</sourceKey>
    <eventDate>2008-09-10T17:25:42.000-07:00</eventDate>
    <rowid>83</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>P45678</sourceKey>
    </xrefKey>
    <mergedRowid>58</mergedRowid>
    <contactPkg>
      <rowidObject>83</rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-10T16:44:56.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-10T17:25:42.000-07:00</lastUpdateDate>
      <consolidationInd>1</consolidationInd>
      <lastRowidSystem>CRM</lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>Thomas</firstName>
      <lastName>Jones</lastName>
    </contactPkg>
  </mergeUpdateEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

No Action Message

The following is an example of a No Action message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>No Action</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>267</messageId>
    <messageDate>2008-09-10T17:25:42.000-07:00</messageDate>
  </eventMetadata>
  <noActionEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>P45678</sourceKey>
    <eventDate>2008-09-10T17:25:42.000-07:00</eventDate>
    <rowid>83</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>P45678</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>P45678</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>P45678</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>83</rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-10T16:44:56.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-10T17:25:42.000-07:00</lastUpdateDate>
      <consolidationInd>1</consolidationInd>
      <lastRowidSystem>CRM</lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>Thomas</firstName>
      <lastName>Jones</lastName>
    </contactPkg>
  </noActionEvent>
</siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

PendingInsert Message

The following is an example of a PendingInsert message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Pending Insert</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>302</messageId>
    <messageDate>2008-09-19T13:57:10.000-07:00</messageDate>
  </eventMetadata>
  <pendingInsertEvent>
    <sourceSystemName>Admin</sourceSystemName>
    <sourceKey>SVR1.2V3</sourceKey>
    <eventDate>2008-09-19T13:57:10.000-07:00</eventDate>
  </pendingInsertEvent>
</siperianEvent>
```

```

<rowid>102          </rowid>
<xrefKey>
  <systemName>Admin</systemName>
  <sourceKey>SVR1.2V3</sourceKey>
</xrefKey>
<contactPkg>
  <rowidObject>102          </rowidObject>
  <creator>admin</creator>
  <createDate>2008-09-19T13:57:09.000-07:00</createDate>
  <updatedBy>admin</updatedBy>
  <lastUpdateDate>2008-09-19T13:57:09.000-07:00</lastUpdateDate>
  <consolidationInd>4</consolidationInd>
  <lastRowidSystem>SYS0          </lastRowidSystem>
  <dirtyInd>1</dirtyInd>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <hubStateInd>0</hubStateInd>
</contactPkg>
</pendingInsertEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

PendingUpdate Message

The following is an example of a PendingUpdate message:

```

<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Pending Update</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>306</messageId>
    <messageDate>2008-09-19T14:01:36.000-07:00</messageDate>
  </eventMetadata>
  <pendingUpdateEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>CPK125</sourceKey>
    <eventDate>2008-09-19T14:01:36.000-07:00</eventDate>
    <rowid>102          </rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>CPK125</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>Admin</systemName>
      <sourceKey>SVR1.2V3</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>102          </rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-19T13:57:09.000-07:00</createDate>
      <updatedBy>sifuser</updatedBy>
      <lastUpdateDate>2008-09-19T14:01:36.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM          </lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <hubStateInd>1</hubStateInd>
    </contactPkg>
  </pendingUpdateEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

PendingUpdateXref Message

The following is an example of a PendingUpdateXref message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Pending Update XREF</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORIS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>306</messageId>
    <messageDate>2008-09-19T14:01:36.000-07:00</messageDate>
  </eventMetadata>
  <pendingUpdateXrefEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>CPK125</sourceKey>
    <eventDate>2008-09-19T14:01:36.000-07:00</eventDate>
    <rowid>102          </rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>CPK125</sourceKey>
    </xrefKey>
    <xrefKey>
      <systemName>Admin</systemName>
      <sourceKey>SVR1.2V3</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>102          </rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-19T13:57:09.000-07:00</createDate>
      <updatedBy>sifuser</updatedBy>
      <lastUpdateDate>2008-09-19T14:01:36.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM          </lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <hubStateInd>1</hubStateInd>
    </contactPkg>
  </pendingUpdateXrefEvent>
</siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Unmerge Message

The following is an example of an unmerge message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>UnMerge</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORIS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>145</messageId>
    <messageDate>2008-09-08T16:24:36.000-07:00</messageDate>
  </eventMetadata>
  <unmergeEvent>
    <sourceSystemName>CRM</sourceSystemName>
```

```

<sourceKey>PK1265</sourceKey>
<eventDate>2008-09-08T16:24:36.000-07:00</eventDate>
<rowid>65</rowid>
<xrefKey>
  <systemName>CRM</systemName>
  <sourceKey>PK1265</sourceKey>
</xrefKey>
<mergedRowid>64</mergedRowid>
<contactPkg>
  <rowidObject>65</rowidObject>
  <creator>admin</creator>
  <createDate>2008-09-08T15:49:17.000-07:00</createDate>
  <updatedBy>admin</updatedBy>
  <lastUpdateDate>2008-09-08T16:24:35.000-07:00</lastUpdateDate>
  <consolidationInd>4</consolidationInd>
  <lastRowidSystem>SYS0</lastRowidSystem>
  <dirtyInd>1</dirtyInd>
  <firstName>Joe</firstName>
  <lastName>Brown</lastName>
</contactPkg>
</unmergeEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Update Message

The following is an example of an update message:

```

<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Update</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR5</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>120</messageId>
    <messageDate>2008-09-08T16:05:13.000-07:00</messageDate>
  </eventMetadata>
  <updateEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>PK12658</sourceKey>
    <eventDate>2008-09-08T16:05:13.000-07:00</eventDate>
    <rowid>66</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>PK12658</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>66</rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-08T16:02:11.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-08T16:05:13.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM</lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>Joe</firstName>
      <lastName>Black</lastName>
    </contactPkg>
  </updateEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Update XREF Message

The following is an example of an Update XREF message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Update XREF</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>121</messageId>
    <messageDate>2008-09-08T16:05:13.000-07:00</messageDate>
  </eventMetadata>
  <updateXrefEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>PK12658</sourceKey>
    <eventDate>2008-09-08T16:05:13.000-07:00</eventDate>
    <rowid>66</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>PK12658</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>66</rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-08T16:02:11.000-07:00</createDate>
      <updatedBy>admin</updatedBy>
      <lastUpdateDate>2008-09-08T16:05:13.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM</lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <firstName>Joe</firstName>
      <lastName>Black</lastName>
    </contactPkg>
  </updateXrefEvent>
</siperianEvent>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

XRefDelete Message

The following is an example of an XRefDelete message:

```
<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>XREF Delete</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>314</messageId>
    <messageDate>2008-09-19T14:14:51.000-07:00</messageDate>
  </eventMetadata>
  <XrefDeleteEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>CPK1256</sourceKey>
    <eventDate>2008-09-19T14:14:51.000-07:00</eventDate>
    <rowid>102</rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>CPK1256</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>102</rowidObject>
      <creator>admin</creator>
    </contactPkg>
  </XrefDeleteEvent>
</siperianEvent>
```

```

        <createDate>2008-09-19T13:57:09.000-07:00</createDate>
        <updatedBy>sifuser</updatedBy>
        <lastUpdateDate>2008-09-19T14:14:54.000-07:00</lastUpdateDate>
        <consolidationInd>4</consolidationInd>
        <lastRowidSystem>CRM                </lastRowidSystem>
        <dirtyInd>1</dirtyInd>
        <hubStateInd>1</hubStateInd>
    </contactPkg>
</XrefDeleteEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

XRefSetToDelete Message

The following is an example of an XRefSetToDelete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<siperianEvent>
  <eventMetadata>
    <eventType>XREF set to Delete</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORs</orsId>
    <triggerUid>MESSAGE_QUEUE_RULE.ContactUpdate</triggerUid>
    <messageId>314</messageId>
    <messageDate>2008-09-19T14:14:51.000-07:00</messageDate>
  </eventMetadata>
  <XrefSetToDeleteEvent>
    <sourceSystemName>CRM</sourceSystemName>
    <sourceKey>CPK1256</sourceKey>
    <eventDate>2008-09-19T14:14:51.000-07:00</eventDate>
    <rowid>102                </rowid>
    <xrefKey>
      <systemName>CRM</systemName>
      <sourceKey>CPK1256</sourceKey>
    </xrefKey>
    <contactPkg>
      <rowidObject>102                </rowidObject>
      <creator>admin</creator>
      <createDate>2008-09-19T13:57:09.000-07:00</createDate>
      <updatedBy>sifuser</updatedBy>
      <lastUpdateDate>2008-09-19T14:14:54.000-07:00</lastUpdateDate>
      <consolidationInd>4</consolidationInd>
      <lastRowidSystem>CRM                </lastRowidSystem>
      <dirtyInd>1</dirtyInd>
      <hubStateInd>1</hubStateInd>
    </contactPkg>
  </XrefSetToDeleteEvent>
</siperianEvent>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Legacy JMS Message XML Reference

If the MDM Hub implementation requires legacy JMS messages, use the legacy XML message format instead of the current version of the XML message format. To use the legacy XML messages, select the **Use Legacy XML** check box in the Message Queues tool.

Message Fields for Legacy XML

The contents of the data area of the message are determined by the package specified in the trigger.

The data area can contain the following message fields:

Field	Description
ACTION	Action type: Insert, Update, Update XREF, Accept as Unique, Merge, Unmerge, or Merge Update.
MESSAGE_ID	ID of the message. The value of the ID corresponds to the value from the C_REPOS_MQ_DATA_CHANGE table.
MESSAGE_DATE	Time when the event was generated.
TABLE_NAME	Name of the base object table or cross-reference object table affected by this action.
RULE_NAME	Name of the rule that triggered the event that generated this message.
RULE_ID	ID of the rule that triggered the event that generated this message.
ROWID_OBJECT	Unique key for the base object affected by this action.
MERGED_OBJECTS	List of ROWID_OBJECT values for the losing records in the merge. This field is included in messages for MERGE events only.
SOURCE_XREF	The SYSTEM and PKEY_SRC_OBJECT values for the cross-reference that triggered the UPDATE event. This field is included in messages for UPDATE events only.
XREFS	List of SYSTEM and PKEY_SRC_OBJECT values for all of the cross-references in the output systems for this base object.

Filtering Messages for Legacy XML

You can use the custom JMS header named MessageType to filter incoming messages based on the message type. The following message types are indicated in the message header.

Message Type	Description
SIP_EVENT	Event notification message.
<serviceNameReturn>	For Services Integration Framework (SIF) responses, the response begins with the name of the SIF request, as in the following fragment of a response to a get request: <pre><getReturn> <message>The GET was executed successfully - retrieved 1 records</message> <recordKey> <ROWID>2</ROWID> </recordKey> ...</pre>

Example Messages for Legacy XML

The following example messages can be used as a reference.

Accept as Unique Message

The following code is an example of an accept as unique message:

```
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Accept as Unique</ACTION>
    <MESSAGE_ID>11010294</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:37:00.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>74      </ROWID_OBJECT>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>196      </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>SFA</SYSTEM>
        <PKEY_SRC_OBJECT>49      </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>74      </ROWID_OBJECT>
      <CONSOLIDATION_IND>1</CONSOLIDATION_IND>
      <FIRST_NAME>Jimmy</FIRST_NAME>
      <MIDDLE_NAME>Neville</MIDDLE_NAME>
      <LAST_NAME>Darwent</LAST_NAME>
      <SUFFIX>Jr</SUFFIX>
      <GENDER>M      </GENDER>
      <BIRTH_DATE>1938-06-22</BIRTH_DATE>
      <SALUTATION>Mr</SALUTATION>
      <SSN_TAX_NUMBER>659483774</SSN_TAX_NUMBER>
      <FULL_NAME>Jimmy Darwent, Stony Brook Ny</FULL_NAME>
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

BO Delete Message

The following code is an example of a BO delete message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>BO Delete</ACTION>
    <MESSAGE_ID>11010295</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:35:53.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>107</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_ORS</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
    </XREFS>
  </CONTROLAREA>
</SIP_EVENT>
```



```

        <XREF>
          <SYSTEM>WEB</SYSTEM>
          <PKEY_SRC_OBJECT />
        </XREF>
      </XREFS>
    </CONTROLAREA>
    <DATAAREA>
      <DATA>
        <ROWID_OBJECT>107</ROWID_OBJECT>
        <CREATOR>sifuser</CREATOR>
        <CREATE_DATE>19 Sep 2008 14:35:28</CREATE_DATE>
        <UPDATED_BY>admin</UPDATED_BY>
        <LAST_UPDATE_DATE>19 Sep 2008 14:35:53</LAST_UPDATE_DATE>
        <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
        <DELETED_IND />
        <DELETED_BY />
        <DELETED_DATE />
        <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
        <INTERACTION_ID />
        <FIRST_NAME>John</FIRST_NAME>
        <LAST_NAME>Smith</LAST_NAME>
        <HUB_STATE_IND>-1</HUB_STATE_IND>
      </DATA>
    </DATAAREA>
  </SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

BO set to Delete

The following code is an example of a BO set to delete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>BO set to Delete</ACTION>
    <MESSAGE_ID>11010296</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:21:03.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_OR</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>WEB</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>102</ROWID_OBJECT>
      <CREATOR>admin</CREATOR>
      <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
      <UPDATED_BY>admin</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 14:21:03</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

```

        <DELETED_IND />
        <DELETED_BY />
        <DELETED_DATE />
        <LAST_ROWID_SYSTEM>SYS0</LAST_ROWID_SYSTEM>
        <INTERACTION_ID />
        <FIRST_NAME />
        <LAST_NAME />
        <HUB_STATE_IND>-1</HUB_STATE_IND>
    </DATA>
</DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Delete Message

The following code is an example of a delete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Delete</ACTION>
    <MESSAGE_ID>11010297</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:35:53.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>107</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_ORS</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
      <XREF>
        <SYSTEM>WEB</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>107</ROWID_OBJECT>
      <CREATOR>sifuser</CREATOR>
      <CREATE_DATE>19 Sep 2008 14:35:28</CREATE_DATE>
      <UPDATED_BY>admin</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 14:35:53</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <DELETED_IND />
      <DELETED_BY />
      <DELETED_DATE />
      <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
      <INTERACTION_ID />
      <FIRST_NAME>John</FIRST_NAME>
      <LAST_NAME>Smith</LAST_NAME>
      <HUB_STATE_IND>-1</HUB_STATE_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Insert Message

The following code is an example of an insert message:

```
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Insert</ACTION>
    <MESSAGE_ID>11010298</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:07:26.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>33          </ROWID_OBJECT>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>49 </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>33          </ROWID_OBJECT>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <FIRST_NAME>James</FIRST_NAME>
      <MIDDLE_NAME>Neville</MIDDLE_NAME>
      <LAST_NAME>Darwent</LAST_NAME>
      <SUFFIX>Unknown</SUFFIX>
      <GENDER>M                </GENDER>
      <BIRTH_DATE>1938-06-22</BIRTH_DATE>
      <SALUTATION>Mr</SALUTATION>
      <SSN_TAX_NUMBER>216275400</SSN_TAX_NUMBER>
      <FULL_NAME>James Darwent,Stony Brook Ny</FULL_NAME>
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Merge Message

The following code is an example of a merge message:

```
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Merge</ACTION>
    <MESSAGE_ID>11010299</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:34:28.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>74          </ROWID_OBJECT>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>196          </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>SFA</SYSTEM>
        <PKEY_SRC_OBJECT>49          </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
    <MERGED_OBJECTS>
      <ROWID_OBJECT>7          </ROWID_OBJECT>
    </MERGED_OBJECTS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
```

```

        <ROWID_OBJECT>74                </ROWID_OBJECT>
        <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
        <FIRST_NAME>Jimmy</FIRST_NAME>
        <MIDDLE_NAME>Neville</MIDDLE_NAME>
        <LAST_NAME>Darwent</LAST_NAME>
        <SUFFIX>Jr</SUFFIX>
        <GENDER>M                        </GENDER>
        <BIRTH_DATE>1938-06-22</BIRTH_DATE>
        <SALUTATION>Mr</SALUTATION>
        <SSN_TAX_NUMBER>659483774</SSN_TAX_NUMBER>
        <FULL_NAME>Jimmy Darwent, Stony Brook Ny</FULL_NAME>
    </DATA>
</DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Merge Update Message

The following code is an example of a merge update message:

```

<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Merge Update</ACTION>
    <MESSAGE_ID>11010310</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:34:28.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>74                </ROWID_OBJECT>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>196          </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>SFA</SYSTEM>
        <PKEY_SRC_OBJECT>49           </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
    <MERGED_OBJECTS>
      <ROWID_OBJECT>7                </ROWID_OBJECT>
    </MERGED_OBJECTS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>74                </ROWID_OBJECT>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <FIRST_NAME>Jimmy</FIRST_NAME>
      <MIDDLE_NAME>Neville</MIDDLE_NAME>
      <LAST_NAME>Darwent</LAST_NAME>
      <SUFFIX>Jr</SUFFIX>
      <GENDER>M                        </GENDER>
      <BIRTH_DATE>1938-06-22</BIRTH_DATE>
      <SALUTATION>Mr</SALUTATION>
      <SSN_TAX_NUMBER>659483774</SSN_TAX_NUMBER>
      <FULL_NAME>Jimmy Darwent, Stony Brook Ny</FULL_NAME>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Pending Insert Message

The following code is an example of a pending insert message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Pending Insert</ACTION>
    <MESSAGE_ID>11010309</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 13:57:10.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_OR</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT>SVR1.2V3</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>102</ROWID_OBJECT>
      <CREATOR>admin</CREATOR>
      <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
      <UPDATED_BY>admin</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 13:57:09</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <DELETED_IND />
      <DELETED_BY />
      <DELETED_DATE />
      <LAST_ROWID_SYSTEM>SYS0</LAST_ROWID_SYSTEM>
      <INTERACTION_ID />
      <FIRST_NAME>John</FIRST_NAME>
      <LAST_NAME>Smith</LAST_NAME>
      <HUB_STATE_IND>0</HUB_STATE_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Pending Update Message

The following code is an example of a pending update message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Pending Update</ACTION>
    <MESSAGE_ID>11010308</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:01:36.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_OR</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>CPK125</PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>102</ROWID_OBJECT>
      <CREATOR>admin</CREATOR>
      <CREATE_DATE>19 Sep 2008 14:01:36</CREATE_DATE>
      <UPDATED_BY>admin</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 14:01:36</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <DELETED_IND />
      <DELETED_BY />
      <DELETED_DATE />
      <LAST_ROWID_SYSTEM>SYS0</LAST_ROWID_SYSTEM>
      <INTERACTION_ID />
      <FIRST_NAME>John</FIRST_NAME>
      <LAST_NAME>Smith</LAST_NAME>
      <HUB_STATE_IND>0</HUB_STATE_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

```

        <PKEY_SRC_OBJECT>SVR1.2V3</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
<DATAAREA>
  <DATA>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <CREATOR>admin</CREATOR>
    <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
    <UPDATED_BY>sifuser</UPDATED_BY>
    <LAST_UPDATE_DATE>19 Sep 2008 14:01:36</LAST_UPDATE_DATE>
    <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
    <DELETED_IND />
    <DELETED_BY />
    <DELETED_DATE />
    <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
    <INTERACTION_ID />
    <FIRST_NAME>John</FIRST_NAME>
    <LAST_NAME>Smith</LAST_NAME>
    <HUB_STATE_IND>1</HUB_STATE_IND>
  </DATA>
</DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Pending Update XREF Message

The following code is an example of a pending update XREF message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Pending Update XREF</ACTION>
    <MESSAGE_ID>11010307</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:01:36.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_ADDRESS_PKG</PACKAGE>
    <RULE_NAME>ContactAM</RULE_NAME>
    <RULE_ID>SVR1.1VU</RULE_ID>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_ORS</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>CPK125</PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT>SVR1.2V3</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
<DATAAREA>
  <DATA>
    <ROWID_CONTACT>102</ROWID_CONTACT>
    <CREATOR>admin</CREATOR>
    <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
    <UPDATED_BY>sifuser</UPDATED_BY>
    <LAST_UPDATE_DATE>19 Sep 2008 14:01:36</LAST_UPDATE_DATE>
    <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
    <DELETED_IND />
    <DELETED_BY />
    <DELETED_DATE />
    <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
    <INTERACTION_ID />
    <FIRST_NAME>John</FIRST_NAME>
  </DATA>
</DATAAREA>
</SIP_EVENT>

```

```

        <LAST_NAME>Smith</LAST_NAME>
        <HUB_STATE_IND>1</HUB_STATE_IND>
        <CITY />
        <STATE />
    </DATA>
</DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Update Message

The following code is an example of an update message:

```

<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Update</ACTION>
    <MESSAGE_ID>11010305</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:44:53.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>74          </ROWID_OBJECT>
    <SOURCE_XREF>
      <SYSTEM>Admin</SYSTEM>
      <PKEY_SRC_OBJECT>196          </PKEY_SRC_OBJECT>
    </SOURCE_XREF>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>196          </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>SFA</SYSTEM>
        <PKEY_SRC_OBJECT>49          </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT>74          </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>74          </ROWID_OBJECT>
      <CONSOLIDATION_IND>1</CONSOLIDATION_IND>
      <FIRST_NAME>Jimmy</FIRST_NAME>
      <MIDDLE_NAME>Neville</MIDDLE_NAME>
      <LAST_NAME>Darwent</LAST_NAME>
      <SUFFIX>Jr</SUFFIX>
      <GENDER>M          </GENDER>
      <BIRTH_DATE>1938-06-22</BIRTH_DATE>
      <SALUTATION>Mr</SALUTATION>
      <SSN_TAX_NUMBER>659483773</SSN_TAX_NUMBER>
      <FULL_NAME>Jimmy Darwent, Stony Brook Ny</FULL_NAME>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Update XREF Message

The following code is an example of an update XREF message:

```
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>Update XREF</ACTION>
    <MESSAGE_ID>11010303</MESSAGE_ID>
    <MESSAGE_DATE>2005-07-21 16:44:53.0</MESSAGE_DATE>
    <TABLE_NAME>C_CUSTOMER</TABLE_NAME>
    <RULE_NAME>CustomerRule1</RULE_NAME>
    <RULE_ID>SVR1.8EO</RULE_ID>
    <ROWID_OBJECT>74          </ROWID_OBJECT>
    <SOURCE_XREF>
      <SYSTEM>Admin</SYSTEM>
      <PKEY_SRC_OBJECT>196      </PKEY_SRC_OBJECT>
    </SOURCE_XREF>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>196      </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>SFA</SYSTEM>
        <PKEY_SRC_OBJECT>49      </PKEY_SRC_OBJECT>
      </XREF>
      <XREF>
        <SYSTEM>Admin</SYSTEM>
        <PKEY_SRC_OBJECT>74      </PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>74          </ROWID_OBJECT>
      <CONSOLIDATION_IND>1</CONSOLIDATION_IND>
      <FIRST_NAME>Jimmy</FIRST_NAME>
      <MIDDLE_NAME>Neville</MIDDLE_NAME>
      <LAST_NAME>Darwent</LAST_NAME>
      <SUFFIX>Jr</SUFFIX>
      <GENDER>M                </GENDER>
      <BIRTH_DATE>1938-06-22</BIRTH_DATE>
      <SALUTATION>Mr</SALUTATION>
      <SSN_TAX_NUMBER>659483773</SSN_TAX_NUMBER>
      <FULL_NAME>Jimmy Darwent, Stony Brook Ny</FULL_NAME>
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Unmerge Message

The following code is an example of an unmerge message:

```
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>UnMerge</ACTION>
    <MESSAGE_ID>11010302</MESSAGE_ID>
    <MESSAGE_DATE>2006-11-07 21:37:56.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONSUMER</TABLE_NAME>
    <PACKAGE>CONSUMER_PKG</PACKAGE>
    <RULE_NAME>Unmerge</RULE_NAME>
    <RULE_ID>SVR1.97S</RULE_ID>
    <ROWID_OBJECT>10</ROWID_OBJECT>
    <DATABASE>edsel-edsel-sp2-CMX_AT</DATABASE>
    <XREFS>
      <XREF>
    </XREF>
  </CONTROLAREA>
</SIP_EVENT>
```



```

        <SYSTEM>Retail System</SYSTEM>
        <PKEY_SRC_OBJECT>8</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
    <MERGED_OBJECTS>
      <ROWID_OBJECT>0</ROWID_OBJECT>
    </MERGED_OBJECTS>
  </CONTROLAREA>
<DATAAREA>
  <DATA>
    <ROWID_OBJECT>10</ROWID_OBJECT>
    <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
    <LAST_ROWID_SYSTEM>SVR1.7NK</LAST_ROWID_SYSTEM>
    <INTERACTION_ID />
    <CONSUMER_ID>8</CONSUMER_ID>
    <FIRST_NAME>THOMAS</FIRST_NAME>
    <MIDDLE_NAME>L</MIDDLE_NAME>
    <LAST_NAME>KIDD</LAST_NAME>
    <SUFFIX />
    <TELEPHONE>2178952323</TELEPHONE>
    <GENDER>M</GENDER>
    <DOB>1940</DOB>
  </DATA>
</DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

XREF Delete Message

The following code is an example of an XREF delete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>XREF Delete</ACTION>
    <MESSAGE_ID>11010301</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:14:51.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>102 </ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_OR<S>S</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>CPK1256</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
<DATAAREA>
  <DATA>
    <ROWID_OBJECT>102</ROWID_OBJECT>
    <CREATOR>admin</CREATOR>
    <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
    <UPDATED_BY>sifuser</UPDATED_BY>
    <LAST_UPDATE_DATE>19 Sep 2008 14:14:54</LAST_UPDATE_DATE>
    <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
    <DELETED_IND />
    <DELETED_BY />
    <DELETED_DATE />
    <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
    <INTERACTION_ID />
    <FIRST_NAME />
    <LAST_NAME />
    <HUB_STATE_IND>1</HUB_STATE_IND>
  </DATA>
</DATAAREA>
</SIP_EVENT>

```

```

        </DATA>
    </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

XREF set to Delete

The following code is an example of an XREF set to delete message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SIP_EVENT>
  <CONTROLAREA>
    <ACTION>XREF set to Delete</ACTION>
    <MESSAGE_ID>11010300</MESSAGE_ID>
    <MESSAGE_DATE>2008-09-19 14:14:51.0</MESSAGE_DATE>
    <TABLE_NAME>C_CONTACT</TABLE_NAME>
    <PACKAGE>CONTACT_PKG</PACKAGE>
    <RULE_NAME>ContactUpdateLegacy</RULE_NAME>
    <RULE_ID>SVR1.28D</RULE_ID>
    <ROWID_OBJECT>102 </ROWID_OBJECT>
    <DATABASE>localhost-mrm-CMX_ORS</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT>CPK1256</PKEY_SRC_OBJECT>
      </XREF>
    </XREFS>
  </CONTROLAREA>
  <DATAAREA>
    <DATA>
      <ROWID_OBJECT>102</ROWID_OBJECT>
      <CREATOR>admin</CREATOR>
      <CREATE_DATE>19 Sep 2008 13:57:09</CREATE_DATE>
      <UPDATED_BY>sifuser</UPDATED_BY>
      <LAST_UPDATE_DATE>19 Sep 2008 14:14:54</LAST_UPDATE_DATE>
      <CONSOLIDATION_IND>4</CONSOLIDATION_IND>
      <DELETED_IND />
      <DELETED_BY />
      <DELETED_DATE />
      <LAST_ROWID_SYSTEM>CRM</LAST_ROWID_SYSTEM>
      <INTERACTION_ID />
      <FIRST_NAME />
      <LAST_NAME />
      <HUB_STATE_IND>1</HUB_STATE_IND>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

Your messages will not look exactly like this. The data will reflect your data, and the fields will reflect your packages.

Part V: Executing Informatica MDM Hub Processes

This part contains the following chapters:

- [Using Batch Jobs, 528](#)
- [User Exits, 576](#)

CHAPTER 27

Using Batch Jobs

This chapter includes the following topics:

- [Using Batch Jobs Overview, 528](#)
- [Batch Job Thread Configuration, 529](#)
- [Launching Batch Jobs, 530](#)
- [Support Tables Used By Batch Jobs, 531](#)
- [Running Batch Jobs in Sequence, 531](#)
- [Best Practices for Working With Batch Jobs, 532](#)
- [Limiting the Parallel Degree for Gathering Statistics in Oracle environments, 533](#)
- [Batch Job Creation, 533](#)
- [Information-Only Batch Jobs \(Not Run in the Hub Console\), 534](#)
- [Process Server Configuration, 535](#)
- [Running Batch Jobs Using the Batch Viewer Tool, 535](#)
- [Running Batch Jobs Using the Batch Group Tool, 543](#)
- [Batch Jobs Reference , 552](#)

Using Batch Jobs Overview

You can configure and run MDM Hub batch jobs through the Batch Viewer and Batch Group tools in the Hub Console.

In the MDM Hub, a batch job is a program that completes a discrete unit of work when it runs. This discrete unit of work is called a process. For example, the Match job carries out the match process. The MDM Hub searches for match candidates, applies the match rules to the match candidates, generates the matches, and then queues the matches for either automatic or manual consolidation. For merge-style base objects, the Automerger job handles automatic consolidation. The Manual Merge job handles manual consolidation.

All the MDM Hub batch jobs are multi-threaded. Multi-threading allows multiple threads to exist within the context of a single process. Also, the MDM Hub batch jobs can run in parallel on all the child base objects that are in the match path of the parent base object.

Before you use batch jobs, you must have performed the following prerequisites:

- Installed the MDM Hub and created the Hub Store.
- Built the schema.

Batch Job Thread Configuration

Multi-threading is a common programming model that allows multiple threads to exist within the context of a single process. All MDM Hub batch jobs, including load jobs and revalidate jobs, are multi-threaded.

When the MDM Hub runs a batch job, the MDM Hub divides the records queued for batch processing into blocks that the MDM Hub can process in parallel. You can configure the number of threads to use for each of the batch jobs in the `cmxserver.properties` file.

When the MDM Hub runs a load job on a parent base object, the MDM Hub applies a batch lock on the corresponding child base object. A batch lock on a child base object prevents another parent from running a load or merge job in parallel. The MDM Hub applies a batch lock on the child base object if the child has a unique key relationship with the parent base object.

Multi-threaded Batch Job Process

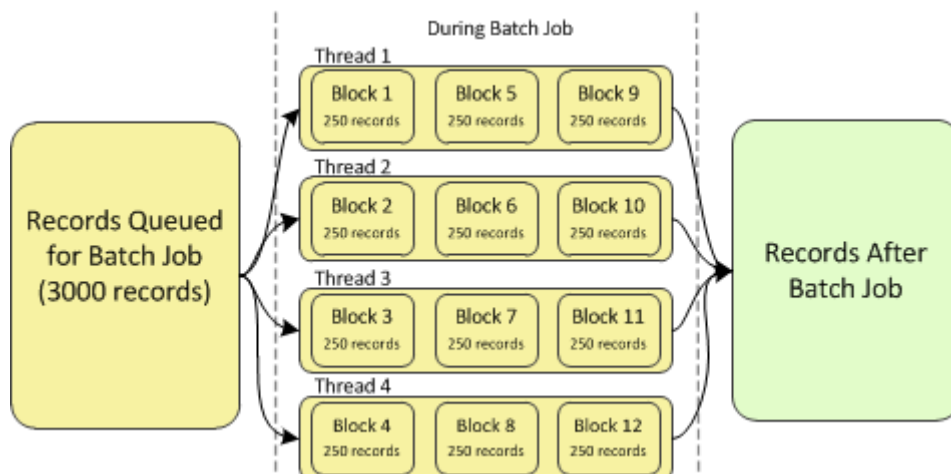
When the MDM Hub processes a batch job, it creates threads and blocks to process the records in parallel. When you configure a batch job, you configure the number of threads and the block size to process the records in a batch job.

1. When you run a batch job, the MDM Hub divides the total number of records by block size that you configured.
2. The MDM Hub creates threads based on the number of threads you configured.
3. The MDM Hub assigns a block to each thread to process.
4. After a thread completes processing a block, the MDM Hub assigns the next unprocessed block in the queue to the thread. This step is repeated until all blocks are processed.

Multi-threaded Batch Job Example

The MDM Hub uses the block size you configure to determine the number of records that it must process in each block. The MDM Hub divides the total number of records to process by the block size.

Consider a scenario where the number of records to be processed in a batch is 3000, the number of threads that you configured is 4, and the block size is 250. The MDM Hub divides the total number of records by the block size, which results in 12 blocks of 250 records each. The MDM Hub then assigns a block to each thread for processing. When a thread completes processing a block, the next block in the queue is assigned to it. The MDM Hub continues assigning blocks to threads that free up, until all blocks are processed.



Multi-threaded Batch Performance

Batch performance can decrease if the thread count is greater than the number of threads your environment can handle efficiently. Also, performance and block size depend on the database environment.

On the MDM Hub, you must configure multi-threaded load or merge jobs on multiple child base objects to run serially.

Multi-threaded Batch Job Properties

You need to configure the number of threads to use and the block size to process for multi-threaded batch jobs.

You configure the number of threads to use for each of the batch jobs in the `cmxserver.properties` file. The following table describes the properties to configure for multi-threaded batch jobs:

Property	Description
<code>cmx.server.automerge.threads_per_job</code>	Number of threads that the MDM Hub uses to process the automerge batch jobs. The default is 1.
<code>cmx.server.automerge.block_size</code>	Number of records to process in each block for the automerge job. The default is 250.
<code>cmx.server.batch.threads_per_job</code>	Number of threads that the MDM Hub uses to process the load, recalculate BVT, and revalidate batch jobs. The default is 10. The value of <code>cmx.server.batch.threads_per_job</code> must be equal to or less than the total number of threads for batch processing available from all Process Servers .
<code>cmx.server.batch.load.block_size</code>	Number of records to process in each block for the load job. The default is 250.
<code>cmx.server.batch.recalculate.block_size</code>	Number of records to process in each block for the recalculate BVT and revalidate jobs. The default is 250.

Launching Batch Jobs

You can launch batch jobs individually or as a group from the MDM Hub console or with Services Integration Framework APIs.

You can use the following tools to launch batch jobs:

The batch viewer tool

Use the batch viewer tool in the MDM Hub console to launch an individual batch job.

The batch group tool

Use the batch group tool in the MDM Hub console to launch batch jobs as a group. You can use the batch group tool to set the launch sequence of the batch jobs or run batch jobs in parallel.

The individual Services Integration Framework APIs

Each batch job available in the MDM Hub console has a corresponding Service Integration Framework API. Use the APIs to launch an individual batch job.

The ExecuteBatchGroup Services Integration Framework API

Use the ExecuteBatchGroup Services Integration Framework API to launch batch groups.

See the *Multidomain MDM Services Integration Framework Guide* for more information about the Services Integration Framework APIs.

Support Tables Used By Batch Jobs

The following table lists the various support tables used by Informatica MDM Hub batch jobs:

- Landing table.
- Staging table.
- Raw table.
- Reject table.
- Match key table.
- Match table.
- System control and history tables.
- XREF table.

Running Batch Jobs in Sequence

Certain batch jobs require that other batch jobs be completed first. For example, the landing tables for a base object must be populated before running any batch jobs. Similarly, before you can run a Match job for a base object, you must run its corresponding Stage and Load jobs. Finally, when a base object has *dependencies* (for example, it is the child of a parent table, or it has foreign key relationships that point to other base objects), batch jobs must be run first for the tables on which the base object depends. You or your organization should consider the best practice of developing an administration or operations plan that specifies which batch processes and dependencies should be completed before running batch jobs.

Populating Landing Tables Before Running Batch Jobs

One of the tasks Informatica MDM Hub batch jobs perform is to move data from landing tables to the appropriate target location in Informatica MDM Hub. Therefore, before you run Informatica MDM Hub batch jobs, you must first have your source systems or an ETL tool write data into the landing tables. The landing tables are Informatica MDM Hub's interface for batch loads. You deliver the data to the landing tables, and Informatica MDM Hub batch procedures manipulate the data and copy it to the appropriate location(s). For more information, see the description of the Informatica MDM Hub data management process in the Informatica MDM Hub Overview.

Match Jobs and Subsequent Consolidation Jobs

Batch jobs need to be executed in a certain sequence. For example, a Match job must be run for a base object before running the consolidation process. For merge-style base objects, you can run the Auto Match and Merge job, which executes the Match job and then Automerge job repeatedly, until either all records in

the base object have been checked for matches, or until the maximum number of records for manual consolidation limit is reached.

Loading Data from Parent Tables First

The general guideline is that all parent tables (tables that other tables reference) must be loaded first.

Loading Data for Objects With Foreign Key Relationships

If two tables have a foreign key relationship between them, you must load the table that is being referenced gets loaded first, and the table doing the referencing gets loaded second. The following foreign key relationships can exist in Informatica MDM Hub: from one base object (child with foreign key) to another base object (parent with primary key).

In most cases, you will schedule these jobs to run on a regular basis.

Best Practices for Working With Batch Jobs

While you design and plan your batch jobs, consider the following issues:

- Define your schema.
The schema is fundamental to all your Informatica MDM Hub tasks. Without a schema, your batch jobs have nothing to do.
- In Oracle environments, limit the parallel degree by which Oracle gathers statistics to avoid using an excessive amount of processes. Excessive processes for gathering statistics could leave insufficient resources for Hub processing. See ["Limiting the Parallel Degree for Gathering Statistics in Oracle environments" on page 533](#).
- Define mappings before executing Stage jobs.
Mappings define the transformations performed in Stage jobs. If you have no mappings defined, then the Stage job will not perform any transformations in the staging process.
- Define match rules before executing Match jobs.
If you have no match rules, then the Match job will produce no matches.
- Before running production jobs:
 - Run tests with small data sets.
 - Run tests of your cleanse engine and other components to determine whether each component is working as expected.
 - After testing each of the components separately, test the integrated system in its entirety to determine whether the overall system is working as expected.

Limiting the Parallel Degree for Gathering Statistics in Oracle environments

The parallel degree for gathering statistics in Oracle environments should be limited to ensure Hub process performance is not negatively impacted.

You must be logged in as a user with database administrator privileges to set the parallel degree.

Perform the following steps to assign an appropriate parallel degree for gathering statistics:

1. Calculate the appropriate parallel degree by using the following formula:

`APPROPRIATE PARALLEL DEGREE = CPU_COUNT * PARALLEL_THREADS_PER_CPU`

`CPU_COUNT` is the number of CPUs available for Oracle to use. The `PARALLEL_THREADS_PER_CPU` is usually 2.

Note: If the server has many CPUs, select a parallel degree value equal to or less than the number of CPU cores.

Note: If other applications are running on the same server as the Hub, decide how many CPU resources can be allocated to the Hub and then use this number when setting the appropriate parallel degree.

2. Check the current parallel degree setting by running the following SQL*Plus command:

`select DBMS_STATS.GET_PREFS('DEGREE') from dual;`

3. If necessary, set the appropriate parallel degree by running one of the following SQL*Plus commands:

- In Oracle 10g: `DBMS_STATS.SET_PARAM ('DEGREE', <appropriate parallel degree value>);`
- In Oracle 11g: `DBMS_STATS.SET_GLOBAL_PREFS ('DEGREE', <appropriate parallel degree value>);`

4. Test performance using the new parallel degree value by running the following SQL command for a large table:

`DBMS_STATS.GATHER_TABLE_STATS`

5. Repeat steps 3 and 4, reducing the parallel degree each time, until wait events are eliminated and performance is acceptable.

Batch Job Creation

Batch jobs are created in either of two ways:

- automatically when you configure Hub Store, or
- when certain changes occur in your Informatica MDM Hub configuration, such as changes to trust settings for a base object

Batch Jobs That Are Created Automatically

When you configure your Hub Store, the MDM Hub creates the following types of batch jobs automatically:

- Auto Match and Merge Jobs
- Autolink Jobs
- Automerge Jobs
- BVT Snapshot Jobs

- External Match Jobs
- Generate Match Tokens Jobs
- Initially Index Smart Search Data Jobs
- Load Jobs
- Manual Link Jobs
- Manual Merge Jobs
- Manual Unlink Jobs
- Manual Unmerge Jobs
- Match Jobs
- Match Analyze Jobs
- Promote Jobs
- Stage Jobs

Batch Jobs That Are Created When Changes Occur

The MDM Hub creates the following batch jobs when you make changes to the match and merge setup, set properties, or enable trust settings after initial loads:

- Accept Non-Matched Records As Unique
- Key Match Jobs
- Reset Match Table Jobs
- Revalidate Jobs (if you enable validation for a column)
- Synchronize Jobs

Information-Only Batch Jobs (Not Run in the Hub Console)

The following batch jobs are for information only and cannot be manually run from the Hub Console.

- Accept Non-Matched Records As Unique
- BVT Snapshot Jobs
- Batch Unmerge Jobs
- Manual Link Jobs
- Manual Merge Jobs
- Manual Unlink Jobs
- Manual Unmerge Jobs
- Migrate Link Style To Merge Style Jobs
- Multi Merge Jobs
- Reset Match Table Jobs

Other Batch Jobs

- Hub Delete Jobs

Process Server Configuration

A Process Server performs batch jobs such as load, recalculate BVT, revalidate, delete, and batch unmerge. The Process Server is deployed in an application server environment. You must configure the Process Servers to perform the batch jobs.

You can configure multiple Process Servers for each Operational Reference Store. You can deploy Process Servers on multiple hosts to distribute the processing load across multiple CPUs and run batch jobs in parallel. Also, the Process Server is multi-threaded so that each instance can process multiple requests concurrently.

RELATED TOPICS:

- [“Process Server Properties” on page 335](#)
- [“Adding a Process Server” on page 337](#)
- [“Editing Process Server Properties” on page 338](#)
- [“Deleting a Process Server” on page 338](#)

Running Batch Jobs Using the Batch Viewer Tool

This section describes how to use the Batch Viewer tool in the Hub Console to run batch jobs individually. To run batch jobs in a group, see [“Running Batch Jobs Using the Batch Group Tool” on page 543](#).

Batch Viewer Tool

The Batch Viewer tool provides a way to execute batch jobs individually and to view the job execution logs. The Batch Viewer is useful for starting the run of a single job, or for running jobs that do not need to run often, such as the Synchronize job that is run after trust settings change. The job execution log shows job completion status with any associated messages, such as success, failure, or warning. The Batch Viewer tool also shows job statistics, if applicable.

Note: The Batch Viewer does not provide automated scheduling.

Starting the Batch Viewer Tool

To start the Batch Viewer tool:

- In the Hub Console, expand the Utilities workbench, and then click **Batch Viewer**.

The Hub Console displays the Batch Viewer tool.

Grouping by Table, Data, or Procedure Type

You can change the top-level view of the navigation tree by right-clicking **Group By** control at the bottom of the tree.

Note: The grayed-out item with the check mark represents the current selection.

Selecting one of the following options:

Group By Option	Description
Table	Displays items in the hierarchy at the following levels: <ul style="list-style-type: none">- top level: tables- second level: procedure type- third level: batch job- fourth level: date / timestamp
Date	Displays items in the hierarchy at the following levels: <ul style="list-style-type: none">- top level: date / timestamp- second level: batch jobs by date/timestamp
Procedure Type	Displays items in the hierarchy at the following levels: <ul style="list-style-type: none">- top level: procedure type- second level: batch job- third level: date / timestamp

Running Batch Jobs Manually

To run a batch job manually:

1. Select the Batch Job to run
2. Execute the Batch Job

Selecting a Batch Job

To select a batch job to run:

1. Start the Batch Viewer tool.

The Batch Viewer tree displays a list of batch jobs. The list is grouped by procedure type.
2. Expand the tree to display the batch job that you want to run, and then click it to select it.

The Batch Viewer displays a screen for the selected batch job with properties and command buttons.

Batch Job Properties

You can view the batch job properties in the properties pane of the batch viewer tool.

The **Identify** table displays the following properties:

Name

The name of the batch job.

Description

The description of the batch job.

The **Status** table displays the following property:

Current Status

The current status of the batch job. The batch job can have one of the following statuses:

- Executing
- Incomplete
- Completed
- Not Executing
- <Batch Job> Successful
- Description of the batch job failure

Options to Set Before Executing Batch Jobs

Certain types of batch jobs have additional fields that you can configure before running the batch job.

Field	Only For	Description
Re-generate All Match Tokens	Generate Match Token Jobs	Controls the scope of match tokens generation: tokenizes the entire base object (checked) or tokenizes only those records that are flagged in the base object as requiring re-tokenization (un-checked).
Force Update	Load Jobs	If selected, the Load job forces a refresh and loads records from the staging table to the base object regardless of whether the records have already been loaded.
Match Set	Match Jobs	Enables you to choose which match rule set to use for this match job.

Command Buttons for Batch Jobs

After you have selected a batch job, you can click the following command buttons.

Button	Description
Execute Batch	Executes the selected batch job.
Clear History	Clears the job execution history in the Batch Viewer.
Set Status to Incomplete	Sets the status of the currently-executing batch job to Incomplete.
Refresh Status	Refreshes the status display of the currently-executing batch job.

Executing a Batch Job

Note: You must have the application server running for the duration of an executing batch job.

To execute a batch job in the Batch Viewer:

1. In the Batch Viewer, select the batch job that you want to run.
2. In the right panel, click **Execute Batch** (or right-click on the job in the left panel and select **Execute** from the pop-up menu).

If the current status of the job is Executing, then the **Execute Batch** button is disabled. You must wait for the batch job to finish before you can run it again.

Refreshing the Status

While a batch job is running, you can click **Refresh Status** to check if the status has changed.

Setting the Job Status to Incomplete

In very rare circumstances, you might want to change the status of a running job by clicking Set Status to Incomplete and execute the job again. Only do this if the batch job has stopped executing (due to an error, such as a server reboot or crash) but Informatica MDM Hub has not detected that the job has stopped due to a job application lock in the metadata. You will know this is a problem if the current status is **Executing** but the database, application server, and logs show no activity. If this occurs, click this button to clear the job application lock so that you can run the batch job again; otherwise, you will not be able to execute the batch job. Setting the status to Incomplete updates the status of the batch job—it does not abort the job. You must also stop the associated database process after setting the job status to Incomplete.






Note: This option is available only if your user ID has Informatica Administrator rights.

Viewing Job Execution Logs

Informatica MDM Hub creates a job execution log each time that it executes a batch job.

Job Execution Status

Each job execution log entry has one of the following status values:

Icon	Description
	Batch job is currently running.
	Batch job completed successfully.
	Batch job completed successfully, but additional information is available. For example, for Stage and Load, this can indicate that some records were rejected. For Match jobs, this can indicate that the base object is empty or that there are no more records to match.
	Batch job failed.
	Batch job status was manually changed from "Executing" to "Incomplete."

Viewing the Job Execution Log for a Batch Job

To view the job execution log for a batch job:

1. Start the Batch Viewer tool.
2. Expand the tree to display the job execution log that you want to view, and then click it.

The Batch Viewer displays a screen for the selected job execution log.

Job Execution Log Entry Properties

For each job execution log entry, the Batch Viewer displays the following information:

Field	Description
Identity	Identification information for this batch job. Stored in the C_REPOS_TABLE_OBJECT_V table
Name	Name of this job execution log. Date / time when the batch job started.
Description	Description for this batch job in the format: <i>JobName</i> for / from <i>BaseObjectName</i> Examples: <ul style="list-style-type: none">- Load from Consumer_Credit_Stg- Match for Address
Source system	One of the following: <ul style="list-style-type: none">- source system of the processed data- Admin
Source table	Source table of the processed data.
Status	<i>Status information for this batch job</i>
Current Status	Current status of this batch job. If an error occurred, displays information about the error.
Metrics	<i>Metrics for this batch job</i>
[Various]	Statistics collected during the execution of the batch job (if applicable): <ul style="list-style-type: none">- Batch Job Metrics- Auto Match and Merge Metrics- Automerge Metrics- Load Job Metrics- Match Job Metrics- Match Analyze Job Metrics- Stage Job Metrics- Promote Job Metrics
Time	Timestamp for this batch job
Start	Date / time when this batch job started.
Stop	Date / time when this batch job ended.
Elapsed time	Elapsed time for the execution of this batch job.

About Batch Job Metrics

Informatica MDM Hub collects various statistics during the execution of a batch job. The actual metrics returned depends on the specific batch job. When a batch job has completed, it registers its statistics in C_REPOS_JOB_METRIC_TYPE. There can be multiple statistics for each job. The possible job metrics include:

Metric Name	Description
Total records	Total number of records processed by the batch job.
Inserted	Number of records inserted by the batch job into the target object.
Updated	Number of records updated by the batch job in the target object.
No action	Number of records on which no action was taken (the records already existed in the base object).
Matched records	Number of records that were matched by the batch job.
Average matches	Number of average matches.
Updated XREF	Number of records that updated the cross-reference table for the base object. If you load a record during an incremental load, the record is already consolidated. The record exists only in the cross-reference table and not in the base object.
Records tokenized	Number of records tokenized by the batch job. Applicable only when the Generate Match Tokens on Load check box is selected in the Schema tool.
Records flagged for match	Number of records flagged for match.
Automerged records	Number of records that were merged by the Automerge batch job.
Rejected records	Number of records rejected by the batch job.
Unmerged source records	Number of source records that were not merged by the batch job.
Merge contributor XREF records	
Accepted as unique records	Number of records that are accepted as unique records by the batch job. Applicable only if the base object has the Accept All Unmatched Rows as Unique option enabled in the Match / Merge Setup configuration.
Queued for automerge	Number of records that are queued for automerge by a Match job that was executed by the Auto Match and Merge job.
Queued for manual merge	Number of records that are queued for manual merge. Use the Merge Manager in the Hub Console to process these records. For more information, see the <i>Multidomain MDM Data Steward Guide</i> .
Backfill trust records	
Missing lookup / Invalid rowid_object records	Number of source records with missing lookup information or invalid rowid_object records.
Records moved to Hold status	Number of records placed on Hold status.

Metric Name	Description
Records analyzed (to be matched)	Number of records to be matched.
Match comparisons required	Number of match comparisons.
Total cleansed records	Number of cleansed records.
Total landing records	Number of records placed in the landing table.
Invalid supplied rowid_object records	Number of records with invalid rowid_object.
Auto-linked records	Number of auto-linked records.
The best version of the truth snapshot	Snapshot of the best version of the truth (BVT).
Duplicate matched records	Number of duplicate matched records.
Links removed	Number of links removed.
Revalidated records	Number of records revalidated.
Base object records reset to New status	Number of base object records reset to "new" status.
Links converted to matches	Number of links converted to matches.
Auto-promoted records	Number of auto-promoted records.
Deleted XREF records	Number of cross-reference records deleted.
Deleted record	Number of records deleted.
Invalid records	Number of invalid records.
Not promoted active records	Number of active records not promoted.
Not promoted protected records	Number of protected records not promoted.
Deleted base object records	Number of base object records deleted.
Link Records Inserted	
Link Records Unlinked	
Link Records Merged	
Groups Created	
Groups Merged	
Match Records Processed	

Metric Name	Description
Link Management Records Processed	
Link Management Records Rejected	
Records failed to be processed	Number of records that were not processed.
Records failed optimistic lock validation	Number of records that were modified by other processes when the batch job was run.
Reindexed search data	Number of records in the search data that were reindexed.
Removed from search data	Number of records that were deleted from the search data.
Failed to be deleted/added from/into search data	Number of records that were not deleted from the search data or the number of records that were not added to the search data.
Total BO rows indexed	Total number of base object rows that were indexed.

Viewing Rejected Records

For Stage jobs, if the batch job resulted in records being written to the rejects table, then the job execution log displays a View Rejects button.

Note: Records are rejected if the HUB_STATE_IND value is not valid.

To view the rejected records and the reason why each was rejected:

1. Click the **View Rejects** button.
The Batch Viewer displays a table of rejected records.
2. Click **Close**.

Handling the Failed Execution of a Batch Job

If executing a batch job failed, perform the following steps:

- Display the execution log entry for this batch job.
- Read the error text in the Current Status field for diagnostic information.
- Take corrective action as necessary.

Copying the Current Status to the Windows Clipboard

To copy the current status of a batch to the Windows Clipboard (to paste into a document or e-mail, for example):

- Click the  button.

Deleting Job Execution Log Entries

To delete the selected job execution log:

- Click the **Delete** button in the top right hand corner of the job properties page.

Clearing the Job Execution History

After running batch jobs over time, the list of executed jobs can become very large. You should periodically remove the extraneous job execution logs from this list.

Note: The actual procedure steps to clear job history will be slightly different depending on the view (By Table, By Date, or By Procedure Type); the following procedure assumes you are using the By Table view.

To clear the job history:

1. Start the Batch Viewer tool.
2. In the Batch Viewer, expand the tree underneath your base object.
3. Expand the tree under the type of batch job.
4. Select the job for which you want to clear the history.
5. Click **Clear History**.
6. Click **Yes** to confirm that you want to delete all the execution history for this batch job.

Running Batch Jobs Using the Batch Group Tool

This section describes how to use the Batch Group tool in the Hub Console to run batch jobs in groups. To run batch jobs individually, see [“Running Batch Jobs Using the Batch Viewer Tool” on page 535](#).

About Batch Groups

A batch group is a collection of individual batch jobs (for example, Stage, Load, and Match jobs) that can be executed with a single command. Each batch job in a batch group can be executed sequentially or in parallel with other jobs. You use the Batch Group tool to configure and run batch groups.

For more information about developing custom batch jobs and batch groups that can be made available in the Batch Group tool, see [“Batch Jobs Reference ” on page 552](#).

Note: If you delete an object from the Hub Console (for example, if you delete a mapping), the Batch Group tool highlights any batch jobs that depend on that object (for example, a stage job) in red. You must resolve this issue prior to re-executing the batch group.

Sequential and Parallel Execution

Batch jobs can be executed in the following ways:

Execution Approach	Description
sequentially	Only one batch job in the batch group is executed at one time.
parallel	Multiple batch jobs in the batch group are executed concurrently and in parallel.

Execution Paths

An *execution path* is the sequence in which batch jobs are executed when the entire batch group is executed.

The execution path begins with the Start node and ends with the End node. The Batch Group tool does not validate the execution sequence for you—it is up to you to ensure that the execution sequence is correct.

For example, the Batch Group tool would not notify you of an error if you incorrectly specified the Load job for a base object ahead of its Stage job.

Levels

In a batch group, the execution path consists of a series of one or more levels that are executed in sequence.

A level is a collection of one or more batch jobs.

- If a level contains multiple batch jobs, then these batch jobs are executed in parallel.
- If a level contains only a single batch job, then this batch job is executed singly.

All batch jobs in the level must complete before the batch group proceeds to the next task in the sequence.

Note: Because all of the batch jobs in a level are executed in parallel, none of the batch jobs in the same level should have any dependencies. For example, the Stage and Load jobs for a base object should be in separate levels that are executed in the proper sequence.

Starting the Batch Group Tool

To start the Batch Group tool:

- In the Hub Console, expand the Utilities workbench, and then click **Batch Group**.

The Hub Console displays the Batch Group tool.

The Batch Group tool consist of the following areas:

Area	Description
Navigation Tree	Hierarchical list of batch groups and execution logs.
Properties Pane	Properties and command

Configuring Batch Groups

This section describes how to add, edit, and delete batch groups.

Adding Batch Groups

To add a batch group:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. Right-click the Batch Groups node in the Batch Group tree and choose **Add Batch Group** from the pop-up menu.

The Batch Group tool adds a “New Batch Group” to the Batch Group tree.

Note the empty execution sequence. You will configure this after adding the new batch group.

- Specify the following information:

Field	Description
Name	Specify a unique, descriptive name for this batch group.
Description	Enter a description for this batch group.

- Click the **Save** button to save your changes.

The Batch Group tool saves your changes and updates the navigation tree.

To add batch jobs to the new batch group, see [“Assigning Batch Jobs to Batch Group Levels” on page 547](#).

Editing Batch Group Properties

To edit batch group properties:

- Start the Batch Group tool.
- Acquire a write lock.
- In the navigation tree, expand the Batch Group node to show the batch group that you want to edit.
- Specify a different batch group name, if you want.
- Specify a different description, if you want.
- Click the **Save** button to save your changes.

Deleting Batch Groups

To delete a batch group:

- Start the Batch Group tool.
- Acquire a write lock.
- In the navigation tree, expand the Batch Group node to show the batch group that you want to delete.
- Right-click the batch group that you want to delete, and then click **Delete Batch Group**.
The Batch Group tool prompts you to confirm deletion.
- Click **Yes**.

The Batch Group tool removes the deleted batch group from the navigation tree.

Configuring Levels for Batch Groups

A batch group contains one or more levels that are executed in sequence. This section describes how to specify the execution sequence by configuring the levels in a batch group.

Adding Levels to a Batch Group

To add a level to a batch group:

- Start the Batch Group tool.
- Acquire a write lock.
- In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.

4. In the batch groups tree, right click on any level, and choose one of the following options:

Command	Description
Add Level Above	Add a level to this batch group above the selected item.
Add Level Below	Add a level to this batch group below the selected item.
Move Level Up	Move this batch group level above the prior level.
Move Level Down	Move this batch group level below the next level.
Remove this Level	Remove this batch group level.

The Batch Group tool displays the Choose Jobs to Add to Batch Group dialog.

5. Expand the base object(s) for the job(s) that you want to add.
6. Select the job(s) that you want to add.

To select jobs that you want to execute in parallel, hold down the CTRL key and click each job that you want to select.

7. Click **OK**. The Batch Group tool adds the selected job(s) to the batch group.
8. Click the **Save** button to save your changes.

Removing Levels From a Batch Group

To remove a level from a batch group:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch group, right click on the level that you want to delete, and choose **Remove this Level**.

The Hub Console displays the delete confirmation dialog.

5. Click **Yes**.

The Batch Group tool removes the deleted level from the batch group.

To Move a Level Up Within a Batch Group

To move a level up within a batch group:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch groups tree, right click on the level you want to move up, and choose **Move Level Up**.

The Batch Group tool moves the level up within the batch group.

To Move a Level Down Within a Batch Group

To move a level down within a batch group:

1. Start the Batch Group tool.

2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch groups tree, right click on the level you want to move down, and choose **Move Level Down**.

The Batch Group tool moves the level down within the batch group.

Assigning Batch Jobs to Batch Group Levels

In the Batch Group tool, a job is a Informatica MDM Hub batch job. Each level contains one or more batch jobs. If a level contains multiple batch jobs, then all of those batch jobs are executed in parallel.

Adding a Batch Job to a Batch Group Level

To add a batch job to a batch group:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch groups tree, right click on the level to which you want to add jobs, and choose **Add jobs to this level...**

The Batch Group tool displays the Choose Jobs to Add to Batch Group dialog.

5. Expand the base object(s) for the job(s) that you want to add.
6. Select the job(s) that you want to add.
7. To select multiple jobs at once (to execute them in parallel), hold down the CTRL key while clicking jobs.
8. Click **OK**.
9. Save your changes.

The Batch Group tool adds the selected jobs to the target level box. Informatica MDM Hub executes all batch jobs in a group level in parallel.

Configuring Options for Batch Jobs

When configuring a batch group, you can configure job options for certain kinds of batch jobs. For more information about these job options, see ["Options to Set Before Executing Batch Jobs" on page 537](#).

Removing a Batch Job From a Level

To remove a batch job from a level:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch group, right click on the job that you want to delete, and choose **Remove Job**.

The Batch Group tool displays the delete confirmation dialog.

5. Click **Yes** to delete the selected job.

The Batch Group tool removes the deleted job from this level in the batch group.

To Move a Batch Job Up a Level

To move a batch job up a level:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch group, right click on the job that you want to move up, and choose **Move job up**.

The Batch Group tool moves the selected job up one level in the batch group.

To Move a Batch Job Down a Level

To move a batch job down a level:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to configure.
4. In the batch group, right click on the job that you want to move up, and choose **Move job down**.

The Batch Group tool moves the selected job down one level in the batch group.

Refreshing the Batch Groups List

To refresh the batch groups list:

- Right-click anywhere in the navigation pane and choose **Refresh**.

Executing Batch Groups Using the Batch Group Tool

This section describes how to manage batch group execution in the Batch Group tool.

Note: You must have the application server running for the duration of an executing batch group.

Note: If you delete an object from the Hub Console (for example, if you delete a mapping), the Batch Group tool highlights any batch jobs that depend on that object (for example, a stage job) in red. You must resolve this issue prior to re-executing the batch group.

Navigating to the Control & Logs Screen

The Control & Logs screen is where you can control the execution of a batch group and view its execution logs.

To navigate to the Control & Logs screen for a batch group:

1. Start the Batch Group tool.
2. Expand the Batch Group tree to display the batch group that you want to execute.
3. Expand the batch group and click the **Control & Logs** node.

The Batch Group tool displays the Control & Logs screen for this batch group.

Components of the Control & Logs Screen

This screen contains the following components:

Component	Description
Toolbar	Command buttons for managing batch group execution.
Logs for the Batch Group	Execution logs for this batch group.
Logs for Batch Jobs	Execution logs for individual batch jobs in this batch group.

Command Buttons for Batch Groups

Use the following command buttons to manage batch group execution.

Button	Description
Execute	Executes this batch group.
Set to Restart	Sets the execution status of a failed batch group to restart.
Set to Incomplete	Sets the execution status of a running batch group to incomplete.
Clear Selected	Removes the selected group or job execution log.
Clear All	Removes all group and job execution logs.
Refresh	Refreshes the screen for this batch group.

Executing a Batch Group







To execute a batch group:

1. Navigate to the Control & Logs screen for the batch group.
2. Click on the node and then select **Batch Group > Execute**, or click on the **Execute** button.
The Batch Group tool executes the batch group and updates the logs panel with the status of the batch group execution.
3. Click the **Refresh** button to see the execution result.
The Batch Group tool displays progress information.
When finished, the Batch Group tool adds entries to:
 - the group execution log for this batch group
 - the job execution log for individual batch jobs

Note: When you execute a batch group in FAILED status, you are actually re-executing the failed instance, and the status is set to whatever the final outcome is, and the Hub does not generate a new group log. However, in the detailed logs (lower log table), you are not re-executing the failed instance, rather, you are executing the same job in a new instance, and as a result, the Hub generates a new log that is displayed here.

Group Execution Status

Each execution log has one of the following status values:

Icon	Description
	Processing. The batch group is currently running.
	Batch group execution completed successfully.
	Batch group execution completed with additional information. For example, for Stage and Load jobs, this can indicate that some records were rejected. For Match jobs, this can indicate that the base object is empty or that there are no more records to match.
	Batch group execution failed.
	Batch group execution is incomplete.
	Batch group execution has been reset to start over.

Viewing the Group Execution Log for a Batch Group

Each time that it executes a batch group, the Batch Group tool generates a group execution log entry.

Warning: While executing a batch group, if a job fails due to a database connectivity issue, the failed job cannot appear in the job control table. The failure appears in the cmxserver log.

The following table describes the properties for each log entry:

Field	Description
Status	Current status of this batch job. If batch group execution failed, displays a description of the problem.
Start	Date / time when this batch job started.
End	Date / time when this batch job ended.
Message	Any messages regarding batch group execution.

Viewing the Job Execution Log for a Batch Job

Each time that it executes a batch job within a batch group, the Batch Group tool generates a job execution log entry.

Each log entry has the following properties:

Field	Description
Job Name	Name of this batch job.
Status	Current status of this batch job.
Start	Date / time when this batch job started.

Field	Description
End	Date / time when this batch job ended.
Message	Any messages regarding batch group execution.

Note: If you want to view the metrics for a completed batch job, you can use the Batch Viewer.

Restarting a Batch Group That Failed Execution

If batch group execution fails, then you can resolve any problems that may have caused the failure to occur, then restart batch group from the beginning.

To execute the batch group again:

1. In the Logs for My Batch Group list, select the execution log entry for the batch group that failed.
2. Click **Set to Restart**.

The Batch Group tool changes the status of this batch job to Restart.

3. Resolve any problems that may have caused the failure to occur and execute the batch group again.

The Batch Group tool executes the batch group and creates a new execution log entry.

Note: If a batch group fails and you do not click either the **Set to Restart** button or the **Set to Incomplete** button in the Logs for My Batch Group list, Informatica MDM Hub restarts the batch job from the prior failed level.

Handling Incomplete Batch Group Execution

In very rare circumstances, you might want to change the status of a running batch group.

- If the batch group status says it is still executing, you can click Set Status to Incomplete and execute the batch group again. You do this only if the batch group has stopped executing (due to an error, such as a server reboot or crash) but Informatica MDM Hub has not detected that the batch group has stopped due to a job application lock in the metadata.

You will know this is a problem if the current status is *Executing* but the database, application server, and logs show no activity. If this occurs, click this button to clear the job application lock so that you can run the batch group again; otherwise, you will not be able to execute the batch group. Setting the status to Incomplete just updates the status of the batch group (as well as all batch jobs within the batch group)—it does not terminate processing.

Note that, if the job status is Incomplete, you cannot set the job status to Restart.

- If the job status is Failed, you can click Set to Restart. Note that, if the job status is Restart, you cannot set the job status to Incomplete.

Changing the status allows you to continue doing something else while the batch group completes.

To set the status of a running batch group to incomplete:

1. In the Logs for My Batch Group list, select the execution log entry for the running batch group that you want to mark as incomplete.
2. Click **Set to Incomplete**.

The Batch Group tool changes the status of this batch job to Incomplete.

3. Execute the batch group again.

Note: If a batch group fails and you do not click either the Set to Restart button or the Set to Incomplete button in the Logs for My Batch Group list, Informatica MDM Hub restarts the batch job from the prior failed level.

Viewing Rejected Records

If batch group execution resulted in records being written to the rejects table (during the execution of Stage jobs or Load jobs), then the job execution log enables the View Rejects button.

To view rejected records:

1. Click the **View Rejects** button.
The Batch Group tool displays the Rejects window.
2. Navigate and inspect the rejected records as needed.
3. Click **Close**.

Filtering Execution Logs By Status

You can view history logs across all Batch Groups, based on their execution status by clicking on the appropriate node under the **Logs By Status** node.

To filter execution logs by status:

1. Start the Batch Group tool.
2. In the Batch Group tree, expand the Logs by Status node.
The Batch Group tool displays the log status list.
3. Click the particular batch group log entry you want to review in the upper half of the logs panel.
Informatica MDM Hub displays the detailed job execution logs for that batch group in the lower half of the panel.

Note: Batch group logs can be deleted by selecting a batch group log and clicking the **Clear Selected** button. To delete all logs shown in the panel, click the **Clear All** button.

Deleting Batch Groups

To delete a batch group:

1. Start the Batch Group tool.
2. Acquire a write lock.
3. In the navigation tree, expand the Batch Group node to show the batch group that you want to delete.
4. In the batch group, right click on the job that you want to move up, and choose **Delete Batch Group** (or select **Batch Group > Delete Batch Group**).

Batch Jobs Reference

This section describes each Informatica MDM Hub batch job.

Alphabetical List of Batch Jobs

Batch Job	Description
Accept Non-Matched Records As Unique	For records that have undergone the match process but had no matching data, sets the consolidation indicator to 1 (consolidated), meaning that the record was unique and did not require consolidation.
Autolink Jobs	Automatically links records that have qualified for autolinking during the match process and are flagged for autolinking (Automerge_ind=1).
Auto Match and Merge Jobs	Executes a continual cycle of a Match job, followed by an Automerge job, until there are no more records to match, or until the number of matches ready for manual consolidation exceeds the configured threshold. Used with merge-style base objects only.
Automerge Jobs	Automatically merges records that have qualified for automerging during the match process and are flagged for automerging (Automerge_ind = 1). Used with merge-style base objects only.
External Match Jobs	Matches “externally managed/prepared” records with an existing base object, yielding the results based on the current match settings—all without actually modifying the data in the base object.
Generate Match Tokens Jobs	Prepares data for matching by generating match tokens according to the current match settings. Match tokens are strings that encode the columns used to identify candidates for matching.
Hub Delete Jobs	Deletes data from the Hub based on base object / XREF level input.
Initially Index Smart Search Data Jobs	Creates indexes for all the values of the searchable fields in a business entity type. Search uses the index to search for data within the searchable fields.
Key Match Jobs	Matches records from two or more sources when these sources use the same primary key. Compares new records to each other and to existing records, and identifies potential matches based on the comparison of source record keys as defined by the match rules.
Load Jobs	Copies records from a staging table to the corresponding target base object in the Hub Store. During the load process, applies the current trust and validation rules to the records.
Manual Merge Jobs	Shows logs for records that have been manually merged in the Merge Manager tool. Used with merge-style base objects only.
Manual Unmerge Jobs	Shows logs for records that have been manually unmerged in the Merge Manager tool.
Match Jobs	Finds duplicate records in the base object, based on the current match rules.
Match Analyze Jobs	Conducts a search to gather match statistics but does not actually perform the match process. If areas of data with the potential for huge match requirements are discovered, Informatica MDM Hub moves the records to a hold status, which allows a data steward to review the data manually before proceeding with the match process.
Match for Duplicate Data Jobs	For data with a high percentage of duplicate records, compares new records to each other and to existing records, and identifies exact duplicates. The maximum number of exact duplicates is based on the Duplicate Match Threshold setting for this base object.
Multi Merge Jobs	Allows the merge of multiple records in one job.

Batch Job	Description
Promote Jobs	Reads the PROMOTE_IND column from an XREF table and changes to ACTIVE the state on all rows where the column's value is 1.
Recalculate BO Jobs	Recalculates all base objects or base objects you specify with the ROWID_OBJECT_TABLE parameter.
Recalculate BVT Jobs	Recalculates the BVT for the specified ROWID_OBJECT.
Reset Match Table Jobs	Shows logs of the operation where all matched records have been reset to be queued for match.
Revalidate Jobs	Executes the validation logic/rules for records that have been modified since the initial validation during the Load Process.
Stage Jobs	Copies records from a landing table into a staging table. During execution, cleanses the data according to the current cleanse settings.
Synchronize Jobs	Updates metadata for base objects. Used after a base object has been loaded but not yet merged, and subsequent trust configuration changes (such as enabling trust) have been made to columns in that base object. This job must be run before merging data for this base object.

Accept Non-Matched Records As Unique

The Accept Non-Matched Records As Unique batch job changes the status of records without matches. This batch job sets the consolidation indicator to "1" for the records without any matches. A value of "1" indicates that the MDM Hub does not need to consolidate the record. The Manual Merge operation also sets the consolidation indicator to "1" for the target record if it does not have any matches. The Automerge batch job considers records with a consolidation indicator of "1" as unique records.

The MDM Hub creates the Accept Non-Matched Records As Unique batch job after a Merge batch job if Accept All Unmatched Rows as Unique is enabled.

Note: You cannot run the Accept Non-Matched Records As Unique batch job from the Batch Viewer.

Auto Match and Merge Jobs

Auto Match and Merge batch jobs execute a continual cycle of a Match job, followed by an Automerge job, until there are no more records to match, or until the maximum number of records for manual consolidation limit is reached.

The match batch size parameter controls the number of records per cycle that this process goes through to finish the match and merge cycles.

Important: Do not run an Auto Match and Merge job on a base object that is used to define relationships between records in inter-table or intra-table match paths. Doing so will change the relationship data, resulting in the loss of the associations between records.

Second Jobs Shown After Application Server Restart

If you execute an Auto Match and Merge job, it completes successfully with one job shown in the status. However, if you stop and restart the application server and return to the Batch Viewer, you see a second job (listed under Match jobs) with a warning a few seconds later. The second job is to ensure that either the base object is empty or there are no more records to match.

Auto Match and Merge Metrics

After running an Auto Match and Merge job, the Batch Viewer displays the following metrics (if applicable) in the job execution log:

Metric	Description
Matched records	Number of records that were matched by the Auto Match and Merge job.
Records tokenized	Number of records that were tokenized prior to the Auto Match and Merge job.
Automerged records	Number of records that were merged by the Auto Match and Merge job.
Accepted as unique records	Number of records that were accepted as unique records by the Auto Match and Merge job. Applies only if this base object has Accept All Unmatched Rows as Unique enabled (set to Yes) in the Match / Merge Setup configuration.
Queued for automerge	Number of records that were queued for automerge by a Match job that was executed by the Auto Match and Merge job.
Queued for manual merge	Number of records that were queued for manual merge. Use the Merge Manager in the Hub Console to process these records. For more information, see the <i>Multidomain MDM Data Steward Guide</i> .

Automerge Jobs

For merge-style base objects only, after the Match job has been run, you can run the Automerge job to automatically merge any records that qualified for automerging during the match process. When an Automerge job is run, it processes all matches in the MATCH table that are flagged for automerging (Automerge_ind=1).

Note: For state-enabled objects only, records that are PENDING (source and target records) or DELETED are never automerged. When a record is deleted, it is removed from the match table and its consolidation_ind is reset to 4.

Automerge Jobs and Auto Match and Merge

Auto Match and Merge batch jobs execute a continual cycle of a Match job, followed by an Automerge job, until there are no more records to match, or until the maximum number of records for manual consolidation limit is reached .

Automerge Jobs and Trust-Enabled Columns

An Automerge job will fail if there is a large number of trust-enabled columns. The exact number of columns that cause the job to fail is variable and based on the length of the column names and the number of trust-enabled columns. Long column names are at—or close to—the maximum allowable length of 26 characters. To avoid this problem, keep the number of trust-enabled columns below 40 and/or the length of the column names short.

Automerge Metrics

After running an Automerge job, the Batch Viewer displays the following metrics (if applicable) in the job execution log:

Metric	Description
Automerged records	Number of records that were automerged by the Automerge job.
Accepted as unique records	Number of records that were accepted as unique records by the Automerge job. Applies only if this base object has Accept All Unmatched Rows as Unique enabled (set to Yes) in the Match / Merge Setup configuration.

Batch Unmerge

You can unmerge records that were merged by a previous process. Use the ExecuteBatchUnmerge SIF API to batch unmerge records.

You can batch unmerge records that were consolidated by the following operations:

- Automerge batch job.
- Manual merge.
- Manual record edit.
- Load by ROWID_OBJECT.
- Insert or update cross-reference records with the Put SIF API.

The ExecuteBatchUnmerge SIF API does not remove match records for the unmerged object that relate to parent records or related child tables. Use the list of unmerged records in table TEST_OUTPUT_TBL to remove these match records. The MDM Hub differentiates unmerged records from manually added records. By default, the MDM Hub assigns a value of 0 for manually added records. The MDM Hub assigns a value other than 0 for unmerged records in table TEST_OUTPUT_TBL in the EXPLODE_NODE_IND column.

You can also add the EXPLODE_NODE_IND column to the input table. If you set EXPLODE_NODE_IND to 1, the MDM Hub tries to unmerge the entire base object.

For more information about the ExecuteBatchUnmerge SIF API, see the *Multidomain MDM Services Integration Framework Guide*.

BVT Snapshot Jobs

For a base object table, the *best version of the truth* (BVT) is a record that has been consolidated with the best cells of data from the source records.

Note: For state-enabled base objects only, the BVT logic uses the HUB_STATE_IND to ignore the non contributing base objects where the HUB_STATE_IND is -1 or 0 (PENDING or DELETED state). For the online BUILD_BVT call, provide INCLUDE_PENDING_IND parameter.

Possible scenarios include:

1. If this parameter is 0 then include only ACTIVE base object records.
2. If this parameter is 1 then include ACTIVE and PENDING base object records.
3. If this parameter is 2 then calculate based on ACTIVE and PENDING XREF records to provide “what-if” functionality.

4. If this parameter is 3 then calculate based on ACTIVE XREF records to provide current BVT based on XREFs, which may be different than the scenario 1.

External Match Jobs

External match jobs match externally prepared records with a base object, which return results based on the current match settings. External match jobs do not load the data from the input table into the base object or affect base object data in any way. You can use external matching to pretest data, test match rules, and inspect the results before running the standard Match job.

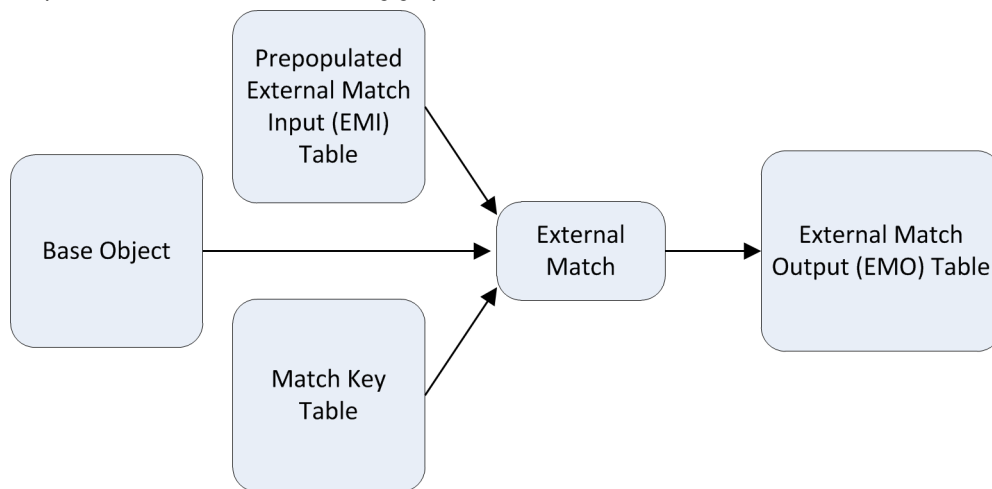
You can use External Match jobs to process both fuzzy-match and exact-match rules and fuzzy-match and exact-match base objects.

Note: Exact-match columns that contain concatenated physical columns require a space at the end of each column. For example, "John" concatenated with "Smith" returns only the match "John Smith".

The External Match job runs as a batch job. There is no corresponding SIF request that external applications can invoke.

Input and Output Tables Used for External Match Jobs

In addition to the base object and its associated match key table, the External Match job uses the input and output tables shown in the following graphic:



External Match Input (EMI) Table

Each base object has an External Match Input (EMI) table for External Match jobs. This table uses the following naming pattern:

`C_BaseObject_EMI`

where *BaseObject* is the name of the base object associated with this External Match job.

When you create a base object, the Schema Manager creates the associated EMI table, and adds the following system columns:

Column Name	Data Type	Size	Not Null	Description
SOURCE_KEY	VARCHAR	50		Used as part of a three-column composite primary key to uniquely identify this record and to map to records in the <i>C_BaseObject_EMO</i> table.
SOURCE_NAME	VARCHAR	50		Used as part of a three-column composite primary key to uniquely identify this record and to map to records in the <i>C_BaseObject_EMO</i> table.
FILE_NAME	VARCHAR	50		Used as part of a three-column composite primary key to uniquely identify this record and to map to records in the <i>C_BaseObject_EMO</i> table.

When you populate the EMI table, at least one of the system columns must contain data. The column names are non-restrictive, which means that they can contain any identifying data if the composite three-column primary key is unique.

In addition, when you configure match rules for a particular column, such as *Person_Name* or *Exact_Cust_ID*, the Schema Manager adds that column to the *C_BaseObject_EMI* table.

Note: To view the columns of an external match table in the Schema Manager, expand the **External Match Table** node.

The records in the EMI table are analogous to the match batch used in Match jobs. The match batch contains the set of records that match with the rest of the base object records. The difference is that, for Match jobs, the match batch records reside in the base object, while for External Match, these records reside in a separate input table.

External Match Output Table

Each base object has an external match output table that contains the output data for external match batch jobs. The MDM Hub drops and recreates the external match output table before the external match batch job runs.

The MDM Hub names the external match output tables *C_<base_object_name>_EMO*, where *base_object_name* is the name of the base object associated with the external match batch job.

The external match batch job populates the external match output table with match pairs. The external match job does not populate the match table. Each row in the external match output table represents a pair of matched records. One record in the pair comes from the external match input table while the other record in the pair comes from the base object. The primary key, which is the *SOURCE_KEY* concatenated with the *SOURCE_NAME* and *FILE_NAME*, identifies the record from the external match input table. The *ROWID_OBJECT_MATCHED* value identifies the record from the base object.

An external match output table contains the following columns:

Column Name	MDM Hub Data Type (Size)	Can Be Null	Description
SOURCE_KEY	VARCHAR (50)	Yes	Maps back to the source record in table C_<base_object_name>_EMI.
SOURCE_NAME	VARCHAR (50)	Yes	Maps back to the source record in table C_<base_object_name>_EMI.
FILE_NAME	VARCHAR (50)	Yes	Maps back to the source record in table C_base_object_name_EMI.
ROWID_OBJECT_MATCHED	CHAR (14)	No	ROWID_OBJECT of the record in the base object that matches the record in the external match input table.
ROWID_MATCH_RULE	CHAR (14)	Yes	Identifies the match rule that the external match batch job used to determine the match.
AUTOMERGE_IND	NUMBER (38)	No	Specifies whether a record qualifies for automatic consolidation during the match process. AUTOMERGE_IND is one of the following values: <ul style="list-style-type: none"> - 0: Record does not qualify for automatic consolidation. - 1: Record qualifies for automatic consolidation. - 2: Records qualify for automatic matching but one or more of the records are in a PENDING state. A value of 2 occurs if you enable state management and you enable "Enable Match on Pending Records." Do not build groups with PENDING records. Leave PENDING records as individual matches. The Automerge batch job processes any records with an AUTOMERGE_IND value of 1.
CREATOR	VARCHAR2 (50)	Yes	User or process responsible for creating the record.
CREATE_DATE	TIMESTAMP	Yes	Date on which the record was created.

Input Table Population

Before you run an External Match job, you must populate the EMI table with records to match against the records in the base object. The process of loading data into an EMI table is external to the MDM Hub. Use a data loading tool that works with your database environment.

When you populate the EMI table, you must supply data for at least one of the system columns to help link back from the _EMI table. The C_BaseObject_EMI table must also contain flat records with unique source keys and no foreign keys to other tables.

Converting Data Types in the Input Table

In the MDM Hub, a column with a DATE data type in the base object appears as a VARCHAR data type column in the associated EMI table. This issue occurs because the MDM Hub converts dates to characters for data comparisons. You can avoid this issue if you convert DATE data types in the base object before you populate the EMI table.

1. Identify the records in the base object that you want to convert. Make a note of the DATE values in both the column and the associated rowid_object.

2. With a querying tool, select records of the DATE data type from the STRP table that corresponds with the base object record.

The query returns records with data that resembles a date and time stamp in the SSA_DATA field.

3. Use the date format mask YYYY/MM/DD HH24:MI:SS to convert the DATE value in the base object into a VARCHAR value.

Database	Conversion Command
Oracle and IBM DB2	<code>(to_char(date_column, 'YYYY/MM/DD HH24:MI:SS'))</code>
Microsoft SQL Server	<code>(convert(VARCHAR(10), date_column, 111) + ' ' + convert(VARCHAR(8), date_column, 108))</code>

4. Load data into the EMI table for external matching.
5. Run an External Match job.
The External Match job returns matches on DATE data types.

Running External Match Jobs

1. Populate the data in the C_BaseObject_EMI table with a data loading process that is external to the MDM Hub.
2. In the Hub Console, start either of the following tools:
 - Batch Viewer
 - Batch Group
3. Select the External Match job for the base object.
4. Select the match rule set that you want to use for external match.
5. Run the External Match job.
 - The External Match job matches all records in the C_BaseObject_EMI table against the records in the base object. There is no concept of a consolidation indicator in the input or output tables.
 - The Build Match Group is not run for the results.
6. Inspect the results in the C_BaseObject_EMO table using a data management tool.
7. If you want to save the results, make a backup copy of the data before you run the External Match job again.

Note: The MDM Hub destroys and recreates the C_BaseObject_EMO table every time you run an External Match job.

Generate Match Tokens Jobs

The Generate Match Tokens job runs the tokenize process, which generates match tokens and stores them in a match key table associated with the base object so that they can be used subsequently by the match process to identify candidates for matching. Generate Match Tokens jobs apply to fuzzy-match base objects only—not to exact-match base objects.

The match process depends on the match tokens in the match key table being current. If match tokens need to be updated (for example, if records have been added or updated during the load process), the match

process automatically runs the tokenize process at the start of a match job. To expedite the match process, it is recommended that you run the tokenize process separately—before the match process—either by:

- manually executing the Generate Match Tokens job, or
- configuring the tokenize process to run automatically after the completion of the load process

Tokenize Process for State-Enabled Base Objects

For state-enabled base objects only, the tokenize process skips records that are in the DELETED state.

These records can be tokenized through the Tokenize API, but will be ignored in batch processing. PENDING records can be matched on a per-base object basis by setting the MATCH_PENDING_IND (default off).

Setting the Scope of the Generate Match Tokens Job

Before you run a Generate Match Tokens job, decide if you want to generate tokens for all records or only for the new and changed records that have their ROWID_OBJECT values in the dirty table.

- To generate match tokens for all records in the base object, select the **Re-generate All Match Tokens** check box.
- To generate match tokens for new or updated records in the base object, clear the **Re-generate All Match Tokens** check box.

After the match tokens are generated, you can run the Match job for the base object.

Tip: If the Generate Match Tokens process returns an error that says that the class `ssa.ssaname3.jssan3cl` cannot be initialized, contact your system administrator.

1. Verify that the PATH environment variable includes the path to the following directory, which contains the dynamic linked library (DLL) files for SSA-NAME3: `<MDM installation directory>/hub/cleanse/lib`
2. Verify that Microsoft Visual C++ Redistributable for Visual Studio 2019 is installed on the Process Server that performs search and match for the MDM Hub.
3. If Microsoft Visual C++ Redistributable for Visual Studio 2019 is installed, use a dependency checker, such as Dependency Walker (`depends.exe`), to load `jssan3cl.dll` and confirm that the Visual C++ Redistributable was successfully applied.

Tip: Visual C++ Redistributable for Visual Studio 2019 requires that Windows Server has operating system patches installed. Check the operating system requirements before installing Visual C++ Redistributable. For example, from a baseline version of Windows Server 2012, you must apply around 100 patches (totalling approximately 2 GB) to the operating system before you can successfully install Visual C++ Redistributable.

Initially Index Smart Search Data Jobs

An Initially Index Smart Search Data job creates indexes for all the values of the searchable fields in a business entity. Search uses the index to search for data within the searchable fields.

Run the Initially Index Smart Search Data job to extract the records from a searchable business entity and add them to the index. After you index the data at least once, when you run the Load job, the Load job internally runs the Initially Index Smart Search Data job to index new and updated data.

The Initially Index Smart Search Data job indexes the records asynchronously and reports successful completion after the job queues the indexing request for all the records. A search request returns the expected results only after the successful completion of the indexing requests, which might take a few minutes.

If you add or update any records after indexing all the records, you must index the new or updated business entities. If you delete any base object records, some of the indexes might become outdated and irrelevant. You can run the Initially Index Smart Search Data batch job to reindex the data, remove the outdated indexes, and improve the performance of the search requests.

If any of the integer field values exceed 19 digits, the Initially Index Smart Search Data job does not index the values. When a date and time field value does not contain the time zone detail, the Initially Index Smart Search Data job considers the time in the locale time zone. The job then converts the time into UTC and indexes the UTC time.

Initially Index Smart Search Data Metrics

The following table describes the metrics that the Batch Viewer displays after you successfully run an Initially Index Smart Search Data job:

Metric	Description
Inserted	Indicates the total number of records that are added to the index.
Rejected records	Indicates the total number of records that are rejected. The job rejects a record if the HUB_STATE_IND value is not valid.
Deleted BO records	Indicates the total number of records that are in the deleted state. A record is in the deleted state if the HUB_STATE_IND value is -1.

Key Match Jobs

Used only with primary key match rules, Key Match jobs run the match process on records from two or more source systems when those sources use the same primary key values.

Key Match jobs compare new records to each other and to existing records, and then identify potential matches based on the comparison of source record keys (as defined by the primary key match rules).

A Key Match job is automatically created after a primary key match rule for a base object has been created or changed in the Schema Manager (Match / Merge Setup configuration).

Load Jobs

Load jobs move data from a staging table to the corresponding target base object in the Hub Store. Load jobs also calculate trust values for base objects with defined trusted columns, and they apply validation rules (if defined) to determine the final trust values.

Load Jobs and State-enabled Base Objects

For state-enabled base objects, the load batch process can load records in any state. The state is specified as an input column on the staging table. The input state can be specified in the mapping view a landing table column or it can be derived. If an input state is not specified in the mapping, then the state is assumed to be ACTIVE.

The following table describes how input states affect the states of existing XREFs.

	Existing XREF State:	ACTIVE	PENDING	DELETED	No XREF (Load by rowid)	No Base Object
Incoming XREF State:						
ACTIVE		Update	Update + Promote	Update + Restore	Insert	Insert
PENDING		Pending Update	Pending Update	Pending Update + Restore	Pending Update	Pending Insert
DELETED		Soft Delete	Hard Delete	Hard Delete	Error	Error
Undefined		Treat as Active	Treat as Pending	Treat as Deleted	Treat As Active	Treat As Active

Note: Records are rejected if the HUB_STATE_IND value is not valid.

The following table provides a matrix of how Informatica MDM Hub processes records (for state-enabled base objects) during Load (and Put) for certain operations based on the record state:

	Incoming Record State	Existing Record State	Notes
Update the XREF record when:	ACTIVE	ACTIVE	
	DELETED	ACTIVE	
	PENDING	PENDING	
	ACTIVE	PENDING	
	DELETED	DELETED	
	PENDING	DELETED	
	DELETED		When a base object rowid delete record comes in, Informatica MDM Hub updates the base object and all XREF records (regardless of ROWID_SYSTEM) to DELETED state.
Insert the XREF record when:	PENDING	ACTIVE	The second record for the pair is created.
	ACTIVE	No Record	
	PENDING	No Record	
Delete the XREF record when:	ACTIVE	PENDING (for paired records)	Delete the ACTIVE record in the pair, the PENDING record is then updated. Paired records are two records with the same PKEY_SRC_OBJECT and ROWID_SYSTEM.

	Incoming Record State	Existing Record State	Notes
	DELETED	PENDING	
Informatica MDM Hub displays an error when:	PENDING	ACTIVE (for paired records)	Paired records are two records with the same PKEY_SRC_OBJECT and ROWID_SYSTEM.

Additional Notes:

- If the incoming state is not specified (for a Load update), then the incoming state is assumed to be the same as the current state. For example if the incoming state is null and the existing state of the XREF or base object to update is PENDING, then the incoming state is assumed to be PENDING instead of null.

Rules for Running Load Jobs

The following rules apply to Load jobs:

- Run a Load job only if the Stage job that loads the staging table used by the Load job has completed successfully.
- Run the Load job for a parent table before you run the Load job for a child table.
- If a lookup on the child object is not defined (the lookup table and column were not populated), in order to successfully load data, you must repeat the Stage job on the child object prior to running the Load job.
- Only one Load job at a time can be run for the same base object. Multiple Load jobs for the same base object cannot be run concurrently.
- You cannot run a load job on multiple parent base objects in parallel if the parent base objects share a common child base object.

Forcing Updates in Load Jobs

Before you run a Load job, you can use the **Force Update** check box to configure how the Load job loads data from the staging table to the target base object. By default, Informatica MDM Hub checks the Last Update Date for each record in the staging table to ensure that it has not already loaded the record. To override this behavior, check (select) the **Force Update** check box, which ignores the Last Update Date, forces a refresh, and loads each record regardless of whether it might have already been loaded from the staging table. Use this approach prudently, however. Depending on the volume of data to load, forcing updates can carry a price in processing time.

State Management Override System in Load Jobs

You can configure one source system as a state management override system. The state management override system can have records in the deleted state. The deleted records in the state management override system can override the cell values in the base object records during a load job.

Satisfy the following conditions to use a state management override system to override the cell values in the base object records during a load job:

- The HUB_STATE_IND column in the landing table is mapped to the HUB_STATE_IND column in the staging table.
- The DELETED_IND column in the landing table is mapped to the DELETED_IND column in the staging table.
- The hub state indicator is in the deleted state.

- The value of the DELETED_IND column in the staging table is -999999999.

Generating Match Tokens During Load Jobs

When configuring the advanced properties of a base object in the Schema tool, you can check (select) the **Generate Match Tokens on Load** check box to generate match tokens during Load jobs, after the records have been loaded into the base object.

By default, this check box is unchecked (cleared), and match tokens are generated during the Match process instead.

Load Batch Job on System Columns

The effect of a Load batch job varies for some system columns.

The following table describes the effect that a Load batch job insert operation and update operation have on system columns:

System Column	Type of Load Batch Job Operation	Effect on System Column
CREATE_DATE	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.
CREATE_DATE	Update	The column in the base object retains the original value.
CREATOR	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.
CREATOR	Update	The column in the base object retains the original value.
UPDATE_BY	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.
UPDATE_BY	Update	The column in the base object retains the original value. The Load batch job populates the column in the cross-reference table with data from the staging table.
LAST_UPDATE_DATE	Insert	The Load batch job populates the LAST_UPDATE_DATE column in the base object table and cross-reference table with SYSDATE.
LAST_UPDATE_DATE	Update	The Load batch job populates the LAST_UPDATE_DATE column in the base object table and cross-reference table with SYSDATE.
SRC_LUD	Insert	The Load batch job populates the SRC_LUD column with value from the LAST_UPDATE_DATE column.
SRC_LUD	Update	The Load batch job populates the SRC_LUD column with value from the LAST_UPDATE_DATE column.
DELETED_IND	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.

System Column	Type of Load Batch Job Operation	Effect on System Column
DELETED_IND	Update	The column in the base object retains the original value.
DELETED_BY	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.
DELETED_BY	Update	The column in the base object retains the original value.
DELETED_DATE	Insert	The Load batch job populates these columns with data from the staging table. The Load batch job populates these columns with the column default value if the staging table column has a null value.
DELETED_DATE	Update	The column in the base object retains the original value.

Load Job Metrics

After running a Load job, the Batch Viewer displays the following metrics (if applicable) in the job execution log:

Metric	Description
Total records	Number of records processed by the Load job.
Inserted	Number of records inserted by the Load job into the target object.
Updated	Number of records updated by the Load job in the target object.
No action	Number of records on which no action was taken (the records already existed in the base object).
Updated XREF	Number of records that updated the cross-reference table for this base object. If you are loading a record during an incremental load, that record has already been consolidated (exists only in the XREF and not in the base object).
Records tokenized	Number of records tokenized by the Load job. Applies only if the Generate Match Tokens on Load check box is selected in the Schema tool.
Merge contributor XREF records	Number of updated cross-reference records that have been merged into other rowid_objects. Represents the difference between the total number of updated cross-reference records and the number of updated base object records.
Missing Lookup / Invalid rowid_object records	Number of source records that were missing lookup information or had invalid rowid_object records.

Manual Merge Jobs

After the Match job has been run, data stewards can use the Merge Manager to process records that have been queued by a Match job for manual merge.

Manual Merge jobs are run in the Merge Manager—not in the Batch Viewer. The Batch Viewer only allows you to inspect job execution logs for Manual Merge jobs that were run in the Merge Manager.

Maximum Matches for Manual Consolidation

In the Schema Manager, you can configure the maximum number of matches ready for manual consolidation to prevent data stewards from being overwhelmed with thousands of manual merges for processing. Once this limit is reached, the Match jobs and the Auto Match and Merge jobs will not run until the number of matches has been reduced.

Executing a Manual Merge Job in the Merge Manager

When you start a Manual Merge job, the Merge Manager displays a dialog with a progress indicator. A manual merge can take some time to complete. If problems occur during processing, an error message is displayed on completion. This error also shows up in the job execution log for the Manual Merge job in the Batch Viewer.

In the Merge Manager, the process dialog includes a button labeled **Mark process as incomplete** that updates the status of the Manual Merge job but does not abort the Manual Merge job. If you click this button, the merge process continues in the background. At this point, there will be an entry in the Batch Viewer for this process. When the process completes, the success or failure is reported. For more information about the Merge Manager, see the *Multidomain MDM Data Steward Guide*.

Manual Unmerge Jobs

For merge-style base objects only, after a Manual Merge job has been run, data stewards can use the Data Manager to manually unmerge records that have been manually merged. Manual Unmerge jobs are run in the Data Manager—not in the Batch Viewer. The Batch Viewer only allows you to inspect job execution logs for Manual Unmerge jobs that were run in the Data Manager. For more information about the Data Manager, see the *Multidomain MDM Data Steward Guide*.

Executing a Manual Unmerge Job in the Data Manager

When you start a Manual Unmerge job, the Data Manager displays a dialog with a progress indicator. A manual unmerge can take some time to complete, especially when a record in question is the product of many constituent records. If problems occur during processing, an error message is displayed on completion. This error also shows up in the job execution log for the Manual Unmerge in the Batch Viewer.

In the Data Manager, the process dialog includes a button labeled **Mark process as incomplete** that updates the status of the Manual Unmerge job but does not abort the Manual Unmerge job. If you click this button, the unmerge process continues in the background. At this point, there will be an entry in the Batch Viewer for this process. When the process completes, the success or failure is reported.

Match Jobs

A match job generates search keys for a base object, searches through the data for match candidates, and applies the match rules to the match candidates. The match job then generates the matches and queues the matches for either automatic or manual consolidation.

When you create a new base object in an ORS, the MDM Hub creates its Match job. Each Match job compares new or updated records in a base object with all records in the base object.

After running a Match job, the matched rows are flagged for automatic and manual consolidation. The MDM Hub creates jobs that consolidate the automerge or autolink records. If a record is flagged for manual consolidation, data stewards must use the Merge Manager to perform the manual consolidation. For more information about manual consolidation, see the *Multidomain MDM Data Steward Guide*.

You configure Match jobs in the Match / Merge Setup node in the Schema Manager.

Do not run a Match job on a base object that is used to define relationships between records in intertable or intratable match paths. This changes the relationship data, which results in the loss of the associations between records.

Note: If your environment has an application server that runs Windows and a database server that runs Linux, match jobs can become unresponsive.

Match Tables

When a Informatica MDM Hub Match job runs for a base object, it populates its match table with pairs of matched records.

Match tables are usually named *Base_Object_MTCH*.

Match Jobs and State-enabled Base Objects

The following table describes the details of the match batch process behavior given the incoming states for state-enabled base objects:

Source Base Object State	Target Base Object State	Operation Result
ACTIVE	ACTIVE	The records are analyzed for matching
PENDING	ACTIVE	Whether PENDING records are ignored in Batch Match is a table-level parameter. If set, then batch match will include PENDING records for the specified Base Object. But the PENDING records can only be the source record in a match.
DELETED	Any state	DELETED records are ignored in Batch Match
ANY	PENDING	PENDING records cannot be the target of a match.

Note: For Build Match Group (BMG), do not build groups with PENDING records. PENDING records to be left as individual matches. PENDING matches will have `automerger_ind=2`.

Auto Match and Merge Jobs

For merge-style base objects only, you can run the Auto Match and Merge job for a base object.

Auto Match and Merge batch jobs execute a continual cycle of a Match job, followed by an Automerger job, until there are no more records to match, or until the maximum number of records for manual consolidation limit is reached.

Setting Limits for Batch Jobs

The Match job for a base object does not attempt to match every record in the base object against every other record in the base object.

Instead, you specify (in the Schema tool):

- how many records the job should match each time it runs.
- how many matches are allowed for manual consolidation.

This feature helps to prevent data stewards from being overwhelmed with manual merges for processing. Once this limit is reached, the Match job will not run until the number of matches ready for manual consolidation has been reduced.

Selecting a Match Rule Set

For Match jobs, before executing the job, you can select the match rule set that you want to use for evaluating matches.

The default match rule set for this base object is automatically selected. To choose any other match rule set, click the drop-down list and select any other match rule set that has been defined for this base object.

Match Job Metrics

After running a Match job, the Batch Viewer displays the following metrics (if applicable) in the job execution log:

Metric	Description
Matched records	Number of records that were matched by the Match job.
Records tokenized	Number of records that were tokenized by the Match job.
Queued for automerge	Number of records that were queued for automerge by the Match job. Use the Automerge job to process these records.
Queued for manual merge	Number of records that were queued for manual merge by the Match job. Use the Merge Manager in the Hub Console to process these records.

Match Analyze Jobs

Match Analyze jobs perform a search to gather metrics but do not conduct any actual matching.

If areas of data with the potential for huge match requirements (hot spots) are discovered, Informatica MDM Hub moves these records to an on-hold status to prevent overmatching. Records that are on hold have a consolidation indicator of 9, which allows a data steward to review the data manually in the Data Manager tool before proceeding with the match and consolidation. Match Analyze jobs are typically used to tune match rules or simply to determine whether data for a base object is overly “matchy” or has large intersections of data (“hot spots”) that will result in overmatching.

Dependencies for Match Analyze Jobs

Each Match Analyze job is dependent on new / updated records in the base object that have been tokenized and are thus queued for matching. For base objects that have intertable match enabled, the Match Analyze job is also dependent on the successful completion of the data tokenization jobs for all child tables, which in turn is dependent on successful Load jobs for the child tables.

Limiting the Number of On-Hold Records

You can limit the number of records that the Match Analyze job moves to the on-hold status. By default, no limit is set. To configure a limit, edit the `cmxcleanse.properties` file and add the following setting:

```
cmx.server.match.threshold_to_move_range_to_hold = n
```

where `n` is the maximum number of records that the Match Analyze job can move to the on-hold status. For more information about the `cmxcleanse.properties` file, see the *Multidomain MDM Installation Guide*.

Match Analyze Job Metrics

After running a Match Analyze job, the Batch Viewer displays the following metrics (if applicable) in the job execution log.

Metric	Description
Records tokenized	Number of records that were tokenized by the Match Analyze job.
Records moved to hold status	Number of records that were moved to a "Hold" status (consolidation indicator = 9) to avert overmatching. These records typically represent a hot spot in the data and are not run through the match process. Data stewards can remove the hold status in the Data Manager.
Records analyzed (to be matched)	Number of records that were analyze for matching.
Match comparisons required	Number of actual matches that would be required to process this base object.

Metrics in Execution Log

Metric	Description
Records moved to Hold Status	Number of records moved to Hold
Records analyzed (to be matched)	Number of records analyzed for match
Match comparisons required	Number of actual matches that would be required to process this base object

Statistics

Statistic	Description
Top 10 range count	Top ten number of records in a given search range.
Top 10 range comparison count	Top ten number of match comparison that will need to be performed for a given search range.
Total records moved to hold	Count of the records moved to hold.
Total matches moved to hold	Total number of matches these records moved to hold required.
Total ranges processed	Number of ranges required to process all the matches in base object.
Total candidates	Total number of match candidates required to process all matches for this base object.
Time for analyze	Amount of time required to run the analysis.

Match for Duplicate Data Jobs

Match for Duplicate Data jobs search for exact duplicates to consider them matched.

The maximum number of exact duplicates is based on the base object columns defined in the Duplicate Match Threshold property in the Schema Manager for each base object.

Note: The Match for Duplicate Data job does not display in the Batch Viewer when the duplicate match threshold is set to 1 and non-equal matches are enabled on the base object.

To match for duplicate data:

1. Execute the Match for Duplicate Data job right after the Load job is finished.
2. Once the Match for Duplicate Data job is complete, run the Automerge job to process the duplicates found by the Match for Duplicate Data job.
3. Once the Automerge job is complete, run the regular match and merge process (Match job and then Automerge job, or the Auto Match and Merge job).

Multi Merge Jobs

A Multi Merge job allows the merge of multiple records in a single job—essentially incorporating the entire set of records to be merged as one batch. This batch job is initiated only by external applications that invoke the SIF MultiMergeRequest request. For more information, see the *Multidomain MDM Services Integration Framework Guide*.

Promote Jobs

For state-enabled objects, the Promote job reads the PROMOTE_IND column from an XREF table and changes the system state to ACTIVE for all rows where the column's value is 1.

Informatica MDM Hub resets PROMOTE_IND after the Promote job has run.

Note: The PROMOTE_IND column on a record is not changed to 0 during the promote batch process if the record is not promoted.

Here are the behavior details for the Promote batch job:

XREF State Before Promote	Base Object State Before Promote	Hub Action on XREF	Hub Action on BO	Refresh BVT?	Resulting BO State	Operation Result
PENDING	ACTIVE	Promote	Update	Yes	ACTIVE	Informatica MDM Hub promotes the pending XREF and recalculates the BVT to include the promoted XREF.
PENDING	PENDING	Promote	Promote	Yes	ACTIVE	Informatica MDM Hub promotes the pending XREF and base object. The BVT is then calculated based on the promoted XREF.

XREF State Before Promote	Base Object State Before Promote	Hub Action on XREF	Hub Action on BO	Refresh BVT?	Resulting BO State	Operation Result
DELETED	This operation behaves the same way regardless of the state of the base object record.	None	None	No	The state of the resulting base object record is unchanged by this operation.	Informatica MDM Hub ignores DELETED records in Batch Promote. This scenario can only happen if a record that had been flagged for promotion is deleted prior to running the Promote batch process.
ACTIVE	This operation behaves the same way regardless of the state of the base object record.	None	None	No	The state of the resulting base object record is unchanged by this operation.	Informatica MDM Hub ignores ACTIVE records in Batch Promote. This scenario can only happen if a record that had been flagged for promotion is made ACTIVE prior to running the Promote batch process.

Note: Promote and delete operations will cascade to direct child records.

Running Promote Jobs Using the Hub Console

To run an Promote job:

1. In the Hub Console, start either of the following tools:
 - Batch Viewer.
 - Batch Group.
2. Select the Promote job for the desired base object.
3. Execute the Promote job.
4. Display the results of the Promote job.

Informatica MDM Hub displays the results of the Promote job.

Promote Job Metrics

After running a Promote job, the Batch Viewer displays the following metrics (if applicable) in the job execution log.

Metric	Description
Autopromoted records	Number of records that were promoted by the Promote job.
Deleted XREF records	Number of XREF records that were deleted by the Promote job.
Active records not promoted	Number of ACTIVE records that were not promoted.
Protected records not promoted	Number of protected records that were not promoted.

Once the Promote job has run, you can view these statistics on the job summary page in the Batch Viewer.

Recalculate Base Object Jobs

Recalculates the best version of the truth for all base objects if you do not use the ROWID_OBJECT_TABLE parameter to identify base objects to recalculate.

Run the Recalculate Base Object job after you change trust settings or validation settings. The MDM Hub does not recalculate the best version of the truth after you change the trust setting or validation setting even if it synchronizes metadata. If you do not recalculate the best version of the truth after you change the trust settings or validation settings, the best version of the truth might be outdated.

You can run the Recalculate Base Object job with or without the ROWID_OBJECT_TABLE parameter. If you run the job with the ROWID_OBJECT_TABLE parameter, the MDM Hub recalculates the best version of the truth for all base objects identified by the ROWID_OBJECT column in the table/inline view. Brackets are required around inline view. If you run the job without the ROWID_OBJECT_TABLE parameter, the MDM Hub recalculates the best version of the truth for all records in the base object. The MDM Hub recalculates the records in batch sizes of MATCH_BATCH_SIZE or a quarter of the number of the records in the table, whichever is less.

Recalculate BVT Jobs

Recalculates the best version of the truth for the specified ROWID_OBJECT.

Run the Recalculate BVT job after you change trust settings or validation settings. The MDM Hub does not recalculate the best version of the truth after you change the trust or validation settings even if it synchronizes metadata. If you do not recalculate the base object after you change the trust settings or validation settings, the best version of the truth might be outdated.

Reset Match Table Jobs

The Reset Match Table job is created automatically after you run a match job and the following conditions exist: if records have been updated to consolidation_ind = 2, and if you then change your match rules.

If you change your match rules after matching, you are prompted to reset your matches. When you reset matches, everything in the match table is deleted. In addition, the Reset Match Table job then resets the consolidation_ind=4 where it is =2.

When you save changes to the schema match columns, a message box is displayed that prompts you to reset the existing matches and create a Reset Match Table job in the Batch Viewer. You need to click **Yes**.

Note: If you do not reset the existing matches, your next Match job will take longer to execute because Informatica MDM Hub will need to regenerate the match tokens before running the Match job.

Note: This job cannot be run from the Batch Viewer.

Revalidate Jobs

Revalidate jobs execute the validation logic/rules for records that have been modified since the initial validation during the Load Process. You can run Revalidate if/when records change post the initial Load process's validation step. If no records change, no records are updated. If some records have changed and get caught by the existing validation rules, the metrics will show the results.

Note: Revalidate jobs can only be run if validation is enabled on a column after an initial load and prior to merge on base objects that have validate rules setup.

Revalidate is executed manually using the batch viewer for base objects.

Stage Jobs

Stage jobs move data from a landing table to a staging table, performing any cleansing that has been configured in the Informatica MDM Hub mapping between the tables.

Stage jobs have parallel cleanse jobs that you can run. The stage status indicates which Process Server is hit during a stage.

For state-enabled base objects, records are rejected if the HUB_STATE_IND value is not valid.

Note: If the Stage job is grayed out, then the mapping has become invalid due to changes in the staging table, in a column mapping, or in a cleanse function. Open the specific mapping using the Mappings tool, verify it, and then save it.

Stage Job Guidelines

When executing the Stage job batch job:

- Run the Stage job only if the ETL process responsible for loading the landing table used by the Stage job completes successfully.
- Make sure that there are no dependencies between Stage jobs.
- You can run multiple Stage jobs simultaneously if there are multiple Process Servers set up to run the jobs.

Stage Job Metrics

After running a Stage job, the Batch Viewer displays the following metrics in the job execution log:

Metric	Description
Total records	Number of records processed by the Stage job.
Inserted	Number of records inserted by the Stage job into the target object.
Rejected	Number of records rejected by the Stage job.

Synchronize Jobs

The Synchronize job updates metadata for base objects. Run the Synchronize job after you change schema trust settings and before you run the Load job.

When you save a base object, the MDM Hub asks you to run a Synchronize job if the following conditions are true:

- The base object has data.
- The base object has newly added trust-enabled columns.

To run the Synchronize job, navigate to the Batch Viewer, and find the Synchronize job for the base object that you want to run. Then click the **Execute Batch** button to run the job. After the initial load occurs, the MDM Hub updates the metadata for the base objects that have trust-enabled columns. After you run the Synchronize job, you can run a Load job.

If a base object has too many trust-enabled columns defined or if the column names are too long, the Synchronize job fails. A column name is too long when it is close to the maximum allowable length of 26 characters. To avoid job failure, you must have short column names and fewer than 48 trust-enabled

columns. Alternatively, you can bypass the need to run a Synchronize job if you enable all trust and validation columns before you save the base object.

CHAPTER 28

User Exits

This chapter includes the following topics:

- [User Exits Overview, 576](#)
- [User Exit Processing, 577](#)
- [User Exit JAR Files, 578](#)
- [UserExitContext Class, 579](#)
- [Stage Process User Exits, 580](#)
- [Load Process User Exits, 583](#)
- [Match Process User Exits, 585](#)
- [Merge Process User Exits, 587](#)
- [Unmerge Process User Exits , 588](#)
- [Task Management User Exits, 590](#)
- [Using Services Integration Framework APIs Within User Exits, 591](#)
- [Guidelines for Implementing User Exits, 595](#)

User Exits Overview

A user exit is custom Java code that you develop that runs at specific points in the batch or Services Integration Framework (SIF) API processes to extend the functionality of MDM Hub. MDM Hub exits regular MDM Hub processing to run code that users develop.

For example, you can use a post-landing user exit to perform pre-cleansing on addresses before the delta detection process.

You upload all the user exits to MDM Hub in a single JAR file. You can upload one JAR file to each Operational Reference Store.

MDM Hub supplies input parameter values when it calls a user exit. Follow the guidelines for implementing user exits to ensure that the user exits do not unnecessarily decrease performance.

You can implement user exits in the following MDM Hub processes:

Stage Process

The stage process moves data from a landing table to a staging table associated with a base object. You can run the following user exits during the stage process:

- Post-landing user exit. Use the post-landing user exit to refine data in a landing table after you populate the landing table through an ETL process.
- Pre-stage user exit. Use a pre-stage user exit to perform custom handling of delta processes.
- Post-stage user exit. Use a post-stage user exit to perform custom processing at the end of a stage job.

Load Process

The load process moves data from a staging table to a base object table. You can run the following user exit during the load process:

- Post-load user exit. Use a post-load user exit to perform custom processing after the load process.

Match Process

The match process identifies base object records that are potential duplicates. You can run the following user exits during the match process:

- Pre-match user exit. Use a pre-match user exit to perform custom processing before the match process.
- Post-match user exit. Use a post-match user exit to perform custom processing on the match table.

Merge Process

The merge process consolidates duplicate base object records into a single master base object record. You run the following user exit during the merge process:

- Post-merge user exit. Use a post-merge user exit to perform custom processing after the merge process.

Unmerge Process

The unmerge process unmerges a single master base object record into the individual base object records that existed before the records were merged. You can run the following user exits during the unmerge process:

- Pre-unmerge user exit. Use a pre-unmerge user exit to perform custom processing before the unmerge process.
- Post-unmerge user exit. Use a post-unmerge user exit to perform custom processing after the unmerge process.

User Exit Processing

MDM Hub processes the user exits as part of the block, batch, or SIF API transaction, as applicable. If the user exit generates an exception, MDM Hub rolls back the processing for the block, batch, or SIF API call. The SIF APIs and batch processes call the same user exits.

When user exits run during a batch process, the user exits run before or after the batch job runs, with the exception of the post-load and post-merge user exits. The post-load and post-merge user exits run after MDM Hub processes each block. You configure the block size in the `cmxserver.properties` file when you configure the Process Servers.

User Exit JAR Files

You can develop custom user exit classes based on the user exit interface classes. Implement custom user exit classes in a separate JAR file and then upload the JAR file to the MDM Hub to register the user exits.

The MDM Hub installation includes a JAR file called `mdm-ue.jar` found in `<MDM Hub installation directory>/hub/server/lib`. The file `mdm-ue.jar` contains Java interface classes for each user exit.

Use the Hub Console to upload the user exit JAR file with custom code to the MDM Hub. You can upload one user exit JAR file for each Operational Reference Store. If you want to change the implementation of a user exit in MDM Hub, first remove the user exits from the MDM Hub. Then upload a JAR file that contains the user exits with the latest implementation.

Implementing the User Exit JAR File

To implement the user exit JAR file, create custom Java code, and then package the code in a JAR file.

1. Use a Java development tool to create a class that implements a user exit interface.
2. Export the implemented custom user exit classes as a JAR file. You can do this through a Java development tool or through a Java command, such as the following example:

```
<java_project_folder>\bin>jar -cf <user_specified_JAR_file_name> .\com\userexit\*.class
```

Uploading User Exits to the MDM Hub

To upload user exits to an Operational Reference Store, use the user object registry in the Hub Console.

1. Verify that you are connected to the Operational Reference Store that you want to upload user exits to.
2. In the **Utilities** workbench, select **User Object Registry**.
3. Acquire a write lock.
4. Select **User Exits** in the navigation pane.
5. Click the **Add** button.
6. From the **Add User Exit** window, click **Browse**.
7. From the **Open** window, browse to the JAR file that contains the user exits. Click **Open**.
8. From the **Add User Exit** window, optionally enter a description, and then click **OK**.

The user exits appear in the user exit table in the properties pane.

Removing User Exits from the MDM Hub

To delete all user exits from the MDM Hub, remove the user exits from the user object registry. You cannot remove individual user exits.

1. In the **Utilities** workbench, select **User Object Registry**.
2. Acquire a write lock.
3. Select **User Exits** in the navigation pane.
4. Select the user exit table in the properties pane, and then click the **Remove** button. Click **OK**.

UserExitContext Class

The UserExitContext class contains the parameters that MDM Hub to the user exits.

The UserExitContext class passes the following parameters to the user exits:

batchJobRowid

Job ID for the batch job. The UserExitContext class passes the BatchJobRowid parameter to all user exits during batch processes.

connection

Database connection that the MDM Hub process uses.

stagingTableName

Source table for the load job. The UserExitContext class passes the stagingTableName parameter during the load and stage process.

tableName

Name of the table that the MDM Hub process uses.

The following code shows the UserExitContext class:

```
package com.informatica.mdm.userexit;

import java.sql.Connection;

/**
 * Represents the hub context that is passed to the user exit interfaces.
 * This is a placeholder for data that is applicable to all user exits.
 */
public class UserExitContext {
    String jobRowid;
    String tableName;
    String stagingTablName;
    Connection conn;

    /**
     * See the corresponding {@link #setBatchJobRowid(String) setter} method for details.
     *
     * @return the rowid of the batch job
     */
    public String getBatchJobRowid() {
        return this.jobRowid;
    }

    /**
     * See the corresponding {@link #setTableName(String) setter} method for details.
     *
     * @return the name of the table used in the hub process
     */
    public String getTableName() {
        return this.tablName;
    }

    /**
     * See the corresponding {@link #setDBConnection(String) setter} method for details.
     *
     * @return the database connection used in the hub process
     */
    public Connection getDBConnection() {
        return this.conn;
    }

    /**
     * Set the rowid of the batch job in the context. This is applicable to batch jobs

```

```

only.
*
* @param batchJobRowid the rowid of the batch job
*/
public void setBatchJobRowid(String batchJobRowid) {
    this.jobRowid = batchJobRowid;
}

/**
* Set the name of the table used in the hub process.
*
* @param tableName the name of the table
*/
public void setTableName(String tableName) {
    this.tablName = tableName;
}

/**
* See the corresponding {@link #setStagingTableName(String) setter} method for
details.
*
* @return the name of the staging table used in the hub load and stage process
*/
public String getStagingTableName() {
    return this.stagingTablName;
}

/**
* Set the name of the staging table used in the context. This is applicable to load
and stage process.
*
* @param stagingTableName the name of the staging table
*/
public void setStagingTableName(String stagingTableName) {
    this.stagingTablName = stagingTableName;
}

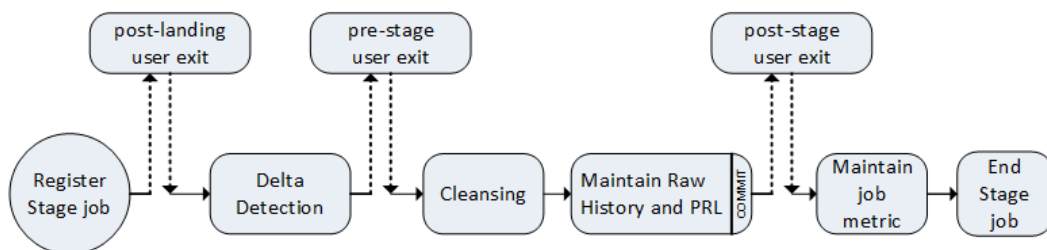
/**
* Set the database connection in the context.
*
* @param connection the database connection used in the hub process
*/
public void setDBConnection(Connection connection) {
    this.conn = connection;
}
}

```

Stage Process User Exits

The stage process can call the post-landing, pre-stage, and post-stage user exits.

The following image shows the stage process and the user exits that the stage process can call:



The user exits run within the stage process in the following sequence:

1. The MDM Hub registers the stage job.
2. The post-landing user exit runs.
3. The MDM Hub performs delta detection.
4. The pre-stage user exit runs.
5. The MDM Hub performs data cleansing.
6. The MDM Hub populates the stage table.
7. If you enable audit trails, the MDM Hub populates the raw table.
8. The MDM Hub commits the stage table changes.
9. The post-stage user exit runs.
10. The stage job ends.

Post-landing User Exit

MDM Hub calls the post-landing user exit after MDM Hub registers the stage job.

Use the post-landing user exit to refine data in a landing table after you populate the landing table through an ETL process. You can use the post-landing user exit to perform custom processing on the landing table before delta detection. For example, you might perform hard delete detection, replace control characters with printable characters, or perform pre-cleansing processing on addresses.

Post-landing User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-landing user exit.

Interface name

The post-landing user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostLandingUserExit
```

Methods

The post-landing user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String
landingTableName,
String previousLandingTableName) throws Exception;
```

If you use the consensus type Hard Delete Detection, add the following logic to the method:

```
ConsensusFlagUpdate consensusProcess = new
ConsensusFlagUpdate(userExitContext.getBatchJobRowid(), stagingTableName);
consensusProcess.startConsensusFlagUpdate(userExitContext.getDBConnection());
```

Parameters

The post-landing user exit uses the following parameters:

landingTableName

Source table for the stage job.

previousLandingTableName

The previous landing table name that contains a copy of the source data mapped to the staging table from the previous time the stage job ran.

stagingTableName

Target table for the stage job.

userExitContext

Passes parameters to the user exit.

Pre-stage User Exit

MDM Hub calls the pre-stage user exit before it loads data into a staging table.

Use a pre-stage user exit to perform custom handling of delta processes. You might use a pre-stage user exit to determine whether delta volumes exceed predefined allowable limits. For example, you might use the user exit to stop the stage process if the number of deltas from the source system is greater than 500,000.

Pre-stage User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-stage user exit.

Interface Name

The pre-stage user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreStageUserExit
```

Methods

The pre-stage user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String  
landingTableName,  
String deltaTableName) throws Exception;
```

Parameters

The pre-stage user exit uses the following parameters:

deltaTableName

The delta table name. The delta table contains the records that MDM Hub identifies as deltas.

landingTableName

Source table for the stage job.

stagingTableName

Target table for the stage job.

userExitContext

Passes parameters to the user exit.

Post-stage User Exit

MDM Hub calls the post-stage user exit after MDM Hub loads data into a staging table.

Use a post-stage user exit to perform custom processing at the end of a stage job. You might use a post-stage user exit to process rejected records from the stage job. For example, you might configure the user exit to delete records that MDM Hub rejects for known, noncritical conditions.

Post-stage User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-stage user exit.

Interface Name

The post-stage user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostStageUserExit
```

Methods

The post-stage user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String  
landingTableName,  
String previousLandingTableName) throws Exception;
```

If you use Hard Delete Detection, add the following logic to the method:

```
HardDeleteDetection hdd = new HardDeleteDetection(rowidJob, stagingTableName);  
hdd.startHardDeleteDetection(userExitContext.getDBConnection());
```

Parameters

The post-stage user exit uses the following parameters:

landingTableName

Source table for the stage job.

previousLandingTableName

The previous landing table name that contains the copy of the source data mapped to the staging table from the previous time the stage job ran.

stagingTableName

Target table for the stage job.

userExitContext

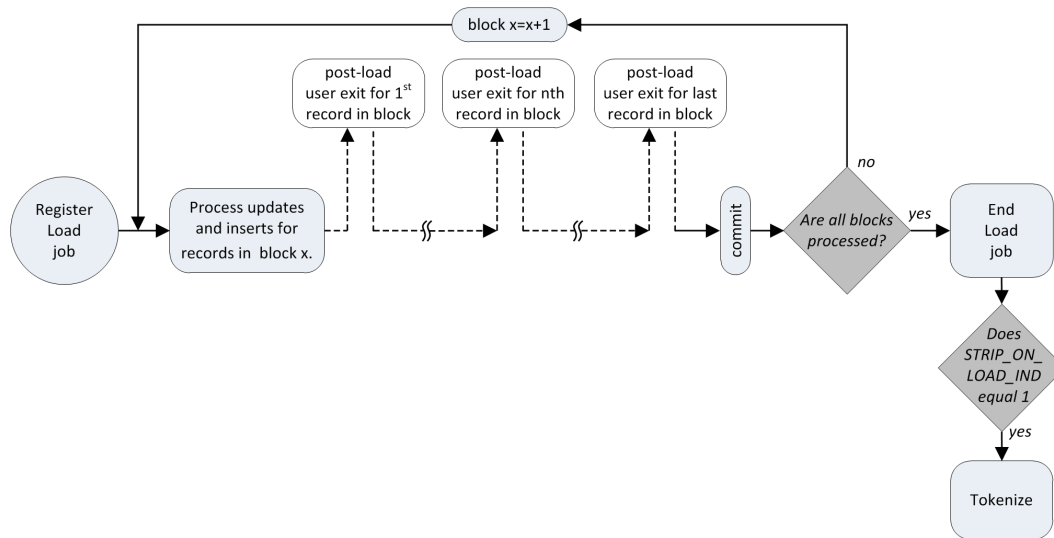
Passes parameters to the user exit.

Load Process User Exits

The load process can call the post-load user exit.

The post-load user exit is called if the loaded count of the job is greater than 0. The post-load user exit runs after the MDM Hub processes each block instead of at the end of the batch process. The post-load user exit runs for each record in the block after the MDM Hub processes the updates and inserts for all records in the block.

The following image shows the load process and the user exits that the load process can call:



The post-load user exit runs within the load process in the following sequence:

1. The MDM Hub registers the load job.
2. The MDM Hub updates or inserts the records for the first block.
3. The post-load user exit runs for each record in the block.
4. The MDM Hub commits the changes.
5. If the MDM Hub has more blocks to load, the process returns to step 2 to process the next block.
6. The load job ends after the MDM Hub loads all blocks.
7. If strip_on_load_ind equals 1, the tokenize job generates match tokens required to perform fuzzy matching.

Post-load User Exit

The MDM Hub calls the post-load user exit after the load process.

Use a post-load user exit to perform custom processing after the load process.

Note: To avoid recursive post-load user exit calls, do not use the MultiMerge SIF API with the post-load user exit. When you use the Put API with the post-load user exit, set the Put API BypassPostLoadUE parameter to `true` to prevent recursive post-load user exit calls.

Post-load User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-load user exit.

Interface Name

The post-load user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostLoadUserExit
```

Methods

The post-load user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, ActionType actionType,
    Map<String, Object> baseObjectDataMap, Map<String, Object> xrefDataMap,
    List<Map<String, Object>> xrefDataMapList) throws Exception;
```

Parameters

The post-load user exit uses the following parameters:

actionType

Indicates whether the load process inserted the record or updated the record.

baseObjectDataMap

Contains the data from the base object that the load process updated or inserted.

userExitContext

Passes parameters to the user exit.

xrefDataMap

Contains the data from the cross-reference record that contributed to the update or insert.

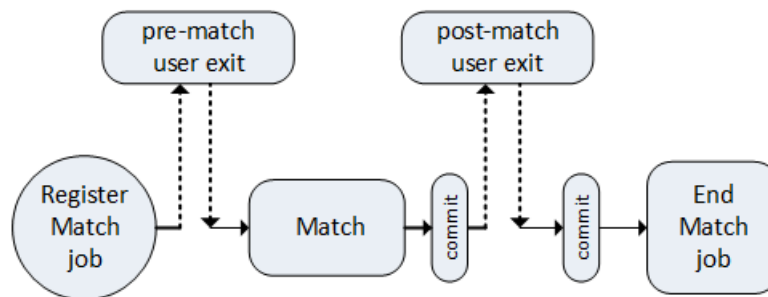
xrefDataMapList

Contains the data from the cross-reference records that the MDM Hub modified because the effective periods overlap. The MDM Hub populates the xrefDataMapList parameter for timeline-enabled base objects.

Match Process User Exits

The match process can call the pre-match and post-match user exits.

The following image shows the match process and the user exits the match process can call:



The user exits run within the match process in the following sequence:

1. The MDM Hub registers the match job.
2. The pre-match user exit runs.
3. The MDM Hub runs the match job.
4. The MDM Hub commits the match changes.
5. The post-match user exit runs.
6. The MDM Hub commits the match changes.
7. The match job ends.

Pre-match User Exit

MDM Hub calls the pre-match user exit before the match process.

Use a pre-match user exit to perform custom processing before the match process.

Pre-match User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-match user exit.

Interface Name

The pre-match user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreMatchUserExit
```

Methods

The pre-match user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws  
Exception;
```

Parameters

The pre-match user exit uses the following parameters:

matchSetName

The name of the match rule set.

userExitContext

Passes parameters to the user exit.

Post-match User Exit

MDM Hub calls the post-match user exit after the match process.

Use a post-match user exit to perform custom processing on the match table. For example, you might use a post-match user exit to manipulate matches in the match queue.

Post-match User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-match user exit.

Interface

The post-match user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostMatchUserExit
```

Methods

The post-match user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws  
Exception;
```

Parameters

The post-match user exit uses the following parameters:

matchSetName

The name of the match rule set that MDM Hub used to find the match.

userExitContext

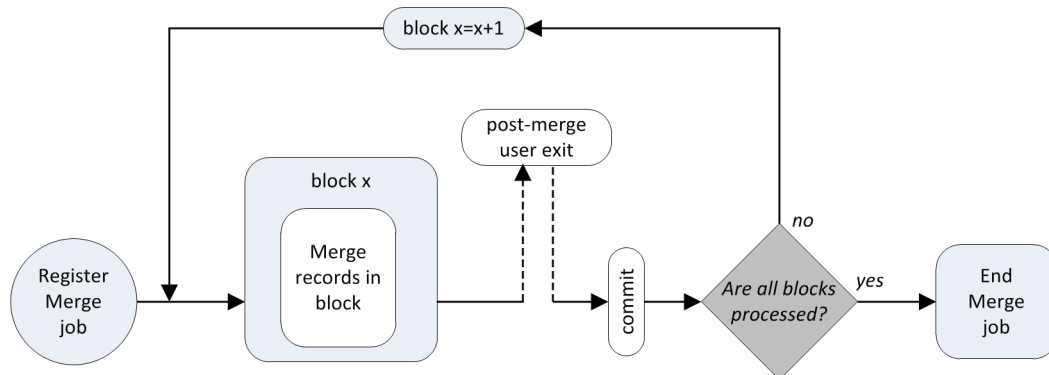
Passes parameters to the user exit.

Merge Process User Exits

The merge process can call the post-merge user exit.

The Merge SIF API and the MultiMerge SIF API call the post-merge user exit after the merge process completes.

The following image shows the batch merge process and the user exits the merge process can call:



The user exits run within the batch merge process in the following sequence:

1. The MDM Hub registers the merge job.
2. The MDM Hub merges the matched records in a block.
3. The post-merge user exit runs for all affected records in the block.
4. The MDM Hub commits the changes.
5. The MDM Hub repeats steps 2 through 4 for the remaining blocks.
6. The merge job ends after the MDM Hub processes all blocks.

Post-merge User Exit

The MDM Hub calls the post-merge user exit after the merge process.

Use a post-merge user exit to perform custom processing after the merge process. For example, you might use a post-merge user exit to match and merge child records affected by the match and merge of a parent record.

Note: To avoid recursive post-merge user exit calls, do not use the MultiMerge SIF API with the post-merge user exit.

Post-merge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-merge user exit.

Interface Name

The post-merge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostMergeUserExit
```

Methods

The post-merge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, Map<String, List<String>>  
baseObjectRowIds) throws Exception;
```

Parameters

The post-merge user exit uses the following parameters:

baseObjectRowIds

List of base object row IDs involved in the merge. The first entry is the target base object record. The remaining entries in the list are the source base objects that merged into the target.

Map<String, List<String>>

Map with the target row ID as a key.

userExitContext

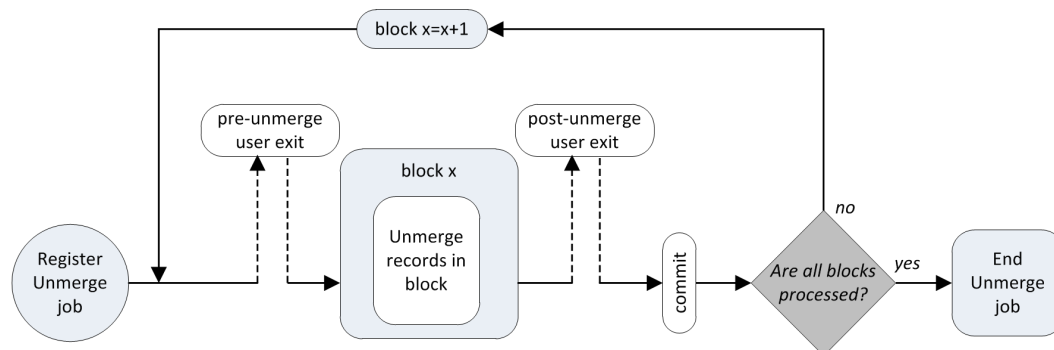
Passes parameters to the user exit.

Unmerge Process User Exits

The unmerge process can call the pre-unmerge and post-unmerge user exits.

The Unmerge SIF API calls the pre-unmerge user exit before the unmerge process starts. The Unmerge SIF API calls the post-unmerge user exit after the unmerge process completes.

The following image shows the batch unmerge process and the user exits that the batch unmerge process can call:



The user exits run within the batch unmerge process in the following sequence:

1. MDM Hub registers the unmerge job.
2. The pre-unmerge user exit runs.

3. MDM Hub unmerges the records in the first block.
4. The post-unmerge user exit runs.
5. MDM Hub commits the changes.
6. MDM Hub repeats steps 2 through 5 for the remaining blocks.
7. The unmerge job ends after MDM Hub processes all blocks.

Pre-unmerge User Exit

The MDM Hub calls the pre-unmerge user exit before the unmerge process.

Use a pre-unmerge user exit to perform custom processing before the unmerge process.

Note: To avoid recursive pre-unmerge user exit calls, do not use the Unmerge SIF API with the post-unmerge user exit.

Pre-unmerge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-unmerge user exit.

Interface Name

The pre-unmerge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreUnmergeUserExit
```

Methods

The pre-unmerge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, Set<UnmergeKey> unmergeKeys)  
throws Exception;
```

Parameters

The pre-unmerge user exit uses the following parameters:

pkeySourceObject

Primary key of the source object. Passed by `UnmergeKeys` for each record in the block.

rowidSystem

System row ID of the cross-reference record for the base object that the unmerge processes. Passed by `UnmergeKeys` for each record in the block.

userExitContext

Passes parameters to the user exit.

Post-unmerge User Exit

The MDM Hub calls the post-unmerge user exit after the unmerge process.

Use a post-unmerge user exit to perform custom processing after the unmerge process.

Post-unmerge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-unmerge user exit.

Interface Name

The post-unmerge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostUnmergeUserExit
```

Methods

The post-unmerge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, Set<PostUnmergeResponse>
responses) throws Exception;
```

Parameters

The post-unmerge user exit uses the following parameters:

boTableName

Name of the base object table that contains the unmerged records. Passed by *PostUnmergeResponse* for each record in the block.

rowidObject

Row ID of the reinstated base object records. Passed by *PostUnmergeResponse* for each record in the block.

userExitContext

Passes parameters to the user exit.

Task Management User Exits

You can use the *AssignTasks* user exit and the *GetAssignableUsersForTask* user exit for task management.

AssignTasks User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the *AssignTasks* user exit.

Interface Name

The *AssignTasks* user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.AssignTasksUserExit
```

Methods

The *AssignTasks* user exit uses the following methods:

```
void processAssignTasks(UserExitContext userExitContext, int maxTasks)
```

Parameters

The *AssignTasks* user exit uses the following parameters:

userExitContext

Passes parameters to the user exit.

maxTasks

Maximum number of tasks to assign to each user.

GetAssignableUsersForTask User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the GetAssignableUsersForTask user exit.

Interface Name

The GetAssignableUsersForTask user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.GetAssignableUsersForTaskUserExit
```

Methods

The GetAssignableUsersForTask user exit uses the following methods:

```
public List<String> processGetAssignableUsersForTask(String taskType, String  
rowidSubjectArea, UserExitContext userExitContext)
```

Parameters

The GetAssignableUsersForTask user exit uses the following parameters:

rowidSubjectArea

The Row ID of the subject area.

taskType

The type of task.

userExitContext

Passes parameters to the user exit.

Using Services Integration Framework APIs Within User Exits

You can create user exits to call Services Integration Framework (SIF) APIs. User exits use a SIF client that calls SIF APIs that you implement in the user exit code.

To create a SIF client, you can use the UserExitSifClient Java class included in the `mdm-ue.jar` file. You can use the SIF client only within a user exit implementation to call SIF APIs. Use the database connection information as an input parameter.

User exits and SIF API calls are part of the same transaction because they use the same database connection. When a user exit calls a SIF API, the process is transactional. You can view changes that are not yet committed. If an error occurs the entire process can roll back.

Creating a User Exit to Call a Services Integration Framework API

You can add custom code to create user exits that call Services Integration Framework (SIF) APIs. You need to add the code to create a SIF client that can call a SIF API.

1. Create a Java project in a Java Integrated Development Environment, such as Eclipse.
2. Add the following MDM Hub JAR files to the Java project:
 - `mdm-ue.jar`
 - `siperian-api.jar`
 - `log4j-<version number>.jar`The JAR files are in the following directory:
 - On UNIX. `<MDM Hub installation directory>/hub/server/lib`
 - On Windows. `<MDM Hub installation directory>\hub\server\lib`
3. Create a Java class for a user exit that implements the user exit interface in the `mdm-ue.jar` file.
4. Add custom code to call the SIF API.
 - a. Use the `UserExitSifClient` class file in the `mdm-ue.jar` file to create a user exit client.
 - b. Define the SIF API request that you must call.
 - c. Specify a code that utilizes the user exit SIF client to call the SIF API.
5. Compile the Java class and package the class files in a custom user exit JAR file.
6. Use the **User Object Registry** tool in the Hub Console to upload the custom user exit JAR file to the MDM Hub.

User Exit Example

Your organization needs to perform a fuzzy match on a record. You need to generate match tokens and store them in a match key table associated with the base object before you can perform a fuzzy match.

To generate match tokens you need to call the Tokenize API. You can configure a user exit that calls the Tokenize API to generate match tokens for the record on which you need to perform a fuzzy match.

The following sample user exit code uses a SIF client to call the Tokenize API to generate match tokens for a record:

```
private String ORS_ID = "orclmain-MDM_SAMPLE";
private UserExitSifClient sifClient;

@Override
public void processUserExit(UserExitContext arg0, ActionType arg1,
    Map<String, Object> arg2, Map<String, Object> arg3,
    List<Map<String, Object>> arg4) throws Exception {

    // Begin custom user exit code ...
    log.info("##### - Starting PostLoad User Exit");

    // Get the ROWID_OBJECT value of the record that was loaded.
    String rowidObject = (String) arg3.get("ROWID_OBJECT");

    // Initialize user exit SIF Client.
    sifClient = new UserExitSifClient(arg0.getDBConnection(), ORS_ID);

    // Tokenize the record that was loaded.
    TokenizeRequest r = new TokenizeRequest();

    // Specify a user that should call the SIF API
    r.setUsername("userExitSifUser");
```

```

r.setOrsId(ORS_ID);

// Specify the base object that must be tokenized.
r.setSiperianObjectUid(SiperianObjectType.BASE_OBJECT.makeUid("C_PARTY"));

// Specify the record that must be tokenized.
RecordKey rkey=new RecordKey();
rkey.setRowid(rowidObject);
r.setRecordKey(rkey);
r.setActionType("UPDATE");

// Call Tokenize SIF API.
TokenizeResponse response = (TokenizeResponse)sifClient.process(r);

// Print out response message
log.info("TokenizeReponse=" + response.getMessage());

// When making subsequent SIF API requests, SIF client can be reused.
// It does not need to be initialized again.

} // End processUserExit

```

Services Integration Framework APIs

You can use the UserExitSifClient Java class included in the user exit JAR file to call Services Integration Framework (SIF) APIs.

The following table describes SIF APIs that user exits can call:

SIF API	Description
CanUnmergeRecords	Determines whether the cross-reference record that you specify can be unmerged from the consolidated base object record.
CleansePut	Inserts or updates the base object with a single record that a key identifies.
Cleanse	Transforms an input record that you specify in the request to the output format that is specified by the MDM Hub cleanse function that you select.
CleanTable	Removes data from an Operational Reference Store and all its associated tables.
CreateTask	Creates a task.
Delete	Removes records from a base object. If you specify the deleteBORRecord flag, the Delete API deletes the base object record even if you specified only a source key and system name.
FlagForAutomerge	Flags a record for automerge in the match table. If the value of the automerge indicator of a record in the match table is 1, the record merges during the next automerge process. If the record does not exist, the FlagForAutomerge API creates the record in the match table and sets the automerge indicator to 1.
GetAssignableUsersForTasks	Retrieves a list of users to whom you can assign tasks.
GetEffectivePeriods	Retrieves the aggregate effective period for a base object record.
GetMatchedRecords	Retrieves the match candidates for a record.

SIF API	Description
GetMergeHistory	Retrieves a tree that represents the merge history of a base object record.
Get	Uses a known key to retrieve a single record from a package.
GetSearchResults	Retrieves additional data when the number of records found by search queries exceeds the number of records that you want returned by the search queries.
GetTasks	Retrieves lists of tasks and task details.
GetTrustScore	Retrieves the current trust score for a column in a base object record.
GetXrefForEffectiveDate	Retrieves multiple cross-reference records for an effective Date.
Merge	Merges two base object records that represent the same object.
MultiMerge	Merges multiple base object records that represent the same object. You can specify the field level overrides for the merged record. Important: If you use this API in Post Load and Post Merge user exits, the MDM Hub generates recursive calls. Use other user exits to call the MultiMerge API.
PreviewBvt	Creates a preview of what a base object record would look like when a specified set of records merge or pending updates occur.
PromotePendingXrefs	Promotes or flags cross-reference records for promotion.
Put	Inserts or updates a single record identified by a key into a base object. Important: If you use this API in a Post Load user exit, the Informatica MDM Hub generates recursive calls. Set the <code>BypassPostLoadUE</code> parameter of the PUT API to <code>true</code> to bypass the Post Load user exit.
RemoveMatchedRecords	Removes matches associated with a base object record from the match table.
Restore	Restores cross-reference records.
SearchHmQuery	Searches Hierarchy Manager entities and relationships.
SearchMatch	Searches for records in a package based on match columns and rule definitions.
SearchQuery	Retrieves a set of records from a package that satisfies the criteria you specify.
SetRecordState	Sets the consolidation indicator for base object records that are identified by the keys you specify.
Tokenize	Generates match tokens for a base object record that the MDM Hub updates or inserts.
Unmerge	Unmerges base object records. Important: If you use this API in a PreUnmerge user exit, the MDM Hub generates recursive calls. Use another user exit to call the Unmerge API.
UpdateMatchRecord	Creates or updates a record in a match table.
UpdateTask	Updates a task.

Guidelines for Implementing User Exits

Implement user exits in a way that does not decrease MDM Hub performance unnecessarily.

Consider the following guidelines when you implement user exits:

Consider how user exits affects performance.

When you implement user exits, you affect MDM Hub performance. Consider if you need to use a user exit to accomplish what you want to do, or if you can do the same thing asynchronously without relying on user exits.

For example, when you load data, you might want to create tasks for the pending records. Although it is possible to create tasks with a user exit, it is unnecessary. You create a bottleneck in the process and unnecessarily decrease performance. You can create the tasks later because these tasks are not critical at that point in the process.

Use the database connection to query or update the database.

The Java user exit context provides a database connection. Use this database connection to query the database or update the database through direct JDBC calls or SIF API calls.

Consider how SIF API calls from a user exits affects batch performance.

When you call SIF APIs from a user exit, consider how these calls affect batch processes.

Use SIF APIs when possible.

Do not use the MDM Hub tables directly if you can use an equivalent SIF API.

Do not explicitly commit or roll back changes made in a user exit.

If you use the database connection to make SIF API calls or direct JDBC calls, the calls participate in the same transaction as the user exit caller. The user exit caller maintains the changes made in the transaction. Do not explicitly call commit or rollback on the connection passed to the user exits.

The MDM Hub does not authenticate or authorize user exits.

User exits run with full access to MDM Hub objects. User exits run with full access to MDM Hub Object regardless of the context. You can use the user exit SIF client to call a subset of supported SIF APIs. Only the supported SIF APIs can participate in the same transaction as the user exit caller. You cannot use the user exit SIF client to access unsupported SIF APIs.

Part VI: Configuring Application Access

This part contains the following chapters:

- [ORS-specific APIs, 597](#)
- [ORS-specific Message Schemas, 602](#)
- [Viewing Registered Custom Code, 606](#)

CHAPTER 29

ORS-specific APIs

This chapter includes the following topics:

- [ORS-specific APIs Overview, 597](#)
- [Performance Considerations, 598](#)
- [Supported Repository Objects, 598](#)
- [Archive Table, 599](#)
- [Generating and Deploying an ORS-Specific SIF API, 600](#)
- [Renaming an ORS-specific SIF API, 600](#)
- [Downloading an ORS-Specific Client JAR File, 600](#)
- [Using ORS-Specific Client JAR Files with SIF SDK, 601](#)
- [Removing an ORS-Specific SIF API, 601](#)

ORS-specific APIs Overview

You can generate APIs for specific objects such as base objects, packages, and cleanse functions in an Operational Reference Store. Use the SIF Manager tool of the Hub Console to generate ORS-specific APIs.

Configure the base objects and packages of the Operational Reference Store before you generate ORS-specific APIs. The ORS-specific APIs are available with SiperianClient through the client JAR file and as a web service. When you generate ORS-specific APIs the MDM Hub associates a version ID with the ORS-specific APIs client JAR file.

ORS-specific APIs act on specific Operational Reference Store objects. For example, a generic API might place data in the database record that you specify. An ORS-specific API identifies the same data as a name and an email address and places those fields into a customer record, as defined in the Operational Reference Store.

Use the ORS-specific APIs through the Services Integration Framework (SIF) SDK or as a SOAP web service. If you use the SIF SDK, you need to install the Java Development Kit and the Apache Jakarta Ant build system.

Performance Considerations

The performance of ORS-specific APIs generation depends on the number of objects that you select. For optimal performance, select only those objects for which you need to generate APIs.

Note: To prevent running out of heap space for the associated SIF API Javadocs, you might need to increase the heap size. The default heap size is 256M. You can also override this default by setting the `sif.jvm.heap.size` parameter in the `cmxserver.properties` file.

Supported Repository Objects

You can generate SIF API for specific repository objects that are secure. To secure an object, use the Secure Resources tool in the Hub Console.

You can generate ORS-specific SIF API for the following repository objects:

- Base objects
- Packages
- Mappings
- Cleanse functions
- Match columns
- Match rule sets

Note: When you generate APIs for match columns and match rule sets, ensure that you select the associated packages. If you do not select the associated packages, the MDM Hub does not generate the ORS-specific SIF API for the match columns and match rule sets.

ORS-Specific SIF API Properties

Use the SIF Manager utility in the Hub Console to configure the properties of an ORS-specific SIF API.

You can configure the following properties of an ORS-specific SIF API:

Logical Name

Logical name of the ORS.

You can edit the logical name and ensure that the edited name is unique. After you edit the logical name of an ORS-specific SIF API, regenerate and deploy the ORS-specific SIF API.

Java Name

Java name of the ORS.

The client JAR file name of the ORS-specific SIF API includes the Java name. Edit the logical name to change the Java name. Ensure that the edited logical name is unique.

WSDL URL

URL of the WSDL file that the MDM Hub generates when it deploys the ORS-specific SIF API.

API Generation Time

The date and time when you generate the ORS-specific SIF API. The format is `mm/dd/yy hh:mm tt`.

Version ID

Unique ID of the ORS-specific SIF API that the MDM Hub generates and deploys.

The MDM Hub uses the version ID in the following elements:

- Properties on the **Environment Report** and **ORS databases** tabs of the Enterprise Manager tool.
- Name of the client JAR file.
- The `MANIFEST.MF` file in the client JAR.

Repository Objects Statuses

The status of a repository object determines whether the MDM Hub can generate and deploy an ORS-specific SIF API for the repository object.

In the SIF Manager utility of the Hub Console, the **Selected and Out of Sync Objects** table displays the status of a repository object in the **Status** column. After you update an object, you can refresh the status of the object. To refresh the status of an object, on the **SIF API Manager** tab in the SIF Manager utility, click **Refresh Objects Status**.

A repository object can have one of the following statuses:

New

Indicates that the object is new and has no SIF API generated and deployed for it. If you generate and deploy an ORS-specific SIF API for the object, the status changes to `Up to date`.

Up to date

Indicates that the object has not changed after the SIF API generation and the object is up to date.

Out of sync

Indicates that the object has changed after the SIF API generation and the object is out of sync. Regenerate the SIF API to change the status to `Up to date`.

Not secure

Indicates that the object is not secure and you cannot generate SIF API for it. In the Secure Resources tool of the Hub Console, objects with the `Not secure` status appear as a private resource.

Deleted

Indicates that the object is deleted and you cannot generate API for it. In the Hub Console, if you use any tool in the Model workbench to delete an object, the status of the object becomes `Deleted`. When you generate and deploy an ORS-specific SIF API, the objects with the `Deleted` status are removed.

Archive Table

You can archive all the ORS-specific SIF APIs that you generate in the `C_REPAR_SIF_ORS_CONFIG` table stored in the `CMX_DATA` tablespace. Use the version ID of an ORS-specific SIF API to identify the archives.

The records in the archive table contain blob data that can be larger than the character-based records and can build up over time. The database administrator can archive or purge the archive table periodically to clean the database.

Generating and Deploying an ORS-Specific SIF API

Use the SIF Manager utility in the Hub Console to generate and deploy ORS-specific SIF API for a secure repository object. You can select specific repository objects to generate ORS-specific SIF API.

To generate and deploy ORS-specific SIF API, the MDM Hub requires access to a Java compiler on the computer that has application server installed. Ensure that you configure the base objects and packages of the ORS before you generate and deploy an ORS-specific SIF API.

1. Start the Hub Console, and connect to an ORS.
2. Expand the Utilities workbench, and click **SIF Manager**.
The SIF Manager utility appears.
3. To acquire a write lock, on the **Write Lock** menu, click **Acquire Lock**.
4. To update the status of the repository objects, on the **SIF API Manager** tab, click **Refresh Objects Status**.
The status of the repository objects is updated in the **Selected and Out of Sync Objects** table.
5. Select the repository objects for which you want to generate and deploy SIF API.
6. Click **Generate and Deploy ORS-Specific SIF APIs**.
The ORS-specific client JAR file and a WSDL file are generated.

Renaming an ORS-specific SIF API

Use the SIF Manager utility in the Hub Console to rename an ORS-specific SIF API.

1. In the SIF Manager utility, on the **Write Lock** menu, click **Acquire Lock**.
2. On the **SIF API Manager** tab, in the **Logical Name** box, click the **Edit** button, and edit the logical name.
3. Click the **Accept** button.
The logical name is saved, and the Java name is updated to match the logical name.
4. Click **Generate and Deploy ORS-Specific SIF APIs**.
The ORS-specific client JAR file and the WSDL file are regenerated.

Downloading an ORS-Specific Client JAR File

After you generate an ORS-specific SIF API for the specific repository objects, download the client JAR file that contains the SiperianClient classes and SIF API reference documentation.

1. In the SIF Manager utility, on the **SIF API Manager** tab, click **Download Client JAR File**.
The **Select the directory in which to save the client JAR file** dialog box appears.
2. Select the directory in which you want to save the client JAR file, and click **Save**.
The `<Java Name>Client_<Version ID>.jar` file is downloaded and saved in the selected directory.

Using ORS-Specific Client JAR Files with SIF SDK

You can use the ORS-specific client JAR file with SIF SDK.

1. If you use an integrated development environment (IDE) and have a project file to build web services, add the downloaded client JAR file to the build class path.
2. Open the `build.xml` file in the following directory:
 - On Windows. `<Resource Kit Installation Directory>\hub\resourcekit\sdk\sifsdk`
 - On UNIX. `<Resource Kit Installation Directory>/hub/resourcekit/sdk/sifsdk`
3. Customize the `build.xml` file so that the `build_war` macro includes the downloaded client JAR file.
4. Save and close the `build.xml` file.

Removing an ORS-Specific SIF API

Use the SIF Manager utility in the Hub Console to remove an ORS-specific SIF API.

1. In the SIF Manager utility, on the **Write Lock** menu, click **Acquire Lock**.
2. On the **SIF API Manager** tab, click **Remove ORS-Specific APIs**.

The ORS-specific client JAR file and the WSDL file are removed.

CHAPTER 30

ORS-specific Message Schemas

This chapter includes the following topics:

- [ORS-specific Message Schemas Overview, 602](#)
- [About the JMS Event Schema Manager Tool, 602](#)
- [Starting the JMS Event Schema Manager Tool, 603](#)
- [Starting the SIF Manager Tool, 603](#)
- [Generating and Deploying ORS-specific Schemas, 604](#)

ORS-specific Message Schemas Overview

You can use the SIF Manager tool of the Hub Console to generate ORS-specific JMS event message schemas. The Out of Sync Objects section in the SIF Manager tool displays objects for which you can generate message schemas.

The performance of ORS-specific JMS event message schema generation depends on the number of objects that the MDM Hub uses to generate and deploy ORS-specific JMS event message schemas.

The ORS-specific JMS event message schema is available as an XSD file that you can download or access through an URL. If you want to use legacy event XML message schema, you do not need to generate ORS-specific JMS event message schemas. If you use the SIF SDK, you need to install the Java Development Kit and the Apache Jakarta Ant build system.

About the JMS Event Schema Manager Tool

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 to be defined.

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 will need to change the logical name before generating the schema.

Additionally, 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:

Object Name	Description
[packageName]Event	Complex type containing elements of type EventMetadata and [packageName].
[packageName]Record	Complex type representing a package and its fields. Also includes an element of type SipMetadata. 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.

Starting the JMS Event Schema Manager Tool

To start the JMS Event Schema Manager tool:

1. In the Hub Console, connect to an Operational Reference Store (Operational Reference Store).
2. Expand the Informatica Utilities workbench and then click **SIF Manager**.
3. Click the **JMS Event Schema Manager** tab.

The Hub Console displays the JMS Event Schema Manager tool.

The JMS Event Schema Manager tool displays the following areas:

Area	Description
JMS ORS-specific Event Message Schema	<p>Shows the event message schema for the ORS.</p> <p>Use this function to generate and deploy ORS-specific JMS Event Messages for the current ORS. The logical name is used to name the components of the deployment. The schema can be downloaded or accessed using a URL.</p> <p>Note: If legacy XML event message objects are to be used, ORS-specific message object generation is not required.</p>
Out of Sync Objects	Shows the database objects in the schema that are out of sync. with the generated API.

Starting the SIF Manager Tool

You can start the SIF Manager tool in the Utilities workbench of the Hub Console.

1. Start the Hub Console, and connect to an Operational Reference Store (ORS).
2. Expand the Utilities workbench and then click **SIF Manager**.

The SIF Manager tool appears.

Generating and Deploying ORS-specific Schemas

The Java software development kit (SDK) includes a compiler in `tools.jar`.

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. If the SDK is not available, you will need to add the `tools.jar` file to the classpath of the application server.

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 will need to change the logical name before generating the schema.

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 schemas.

Also, the JMS Event Schema generation requires at least one secure package or remote package.

To generate and deploy ORS-specific schemas:

1. Start the JMS Event Schema Manager.
The Hub Console displays the JMS Event Schema Manager tool.
2. Enter a value in the Logical Name field for the event schema.
In order to make any changes to the schema, you must have a write lock.
3. Click **Generate and Deploy ORS-specific Schemas**.

Note: There must be at least one secure package or remote package configured to generate the schema. If there are no secure objects to generate, the Informatica MDM Hub generates a runtime error message.

Downloading an XSD File

An XSD file defines the structure of an XML file and can also be used to validate the XML file.

For example, if an XML file contains a reference to an XSD, an XML validation tool can be used to verify that the tags in the XML conform to the definitions defined in the XSD.

To download an XSD file:

1. Start the JMS Event Schema Manager.
The Hub Console displays the JMS Event Schema Manager tool.
2. Acquire a write lock.
In order to make any changes to the schema, you must have a write lock.
3. Click **Download XSD File**.
Alternatively, you can use the URL specified in the Schema URL to access the XSD file.

Finding Out-of-Sync Objects

You use Find Out Of Sync Objects to determine if the event schema needs to be re-generated to reflect changes in the system.

The JMS Event Schema Manager displays a list of packages and remote packages that have changed since the last schema generation.

Note: The Out of Sync Objects function compares the generated APIs to the database objects in the schema so both must be present to find the out-of-sync objects.

To find the out-of-sync objects:

1. Start the JMS Event Schema Manager.

The Hub Console displays the JMS Event Schema Manager tool.

2. Acquire a write lock.

In order to make any changes to the schema, you must have a write lock.

3. Click **FindOut of Sync Objects**.

The JMS Event Schema Manager displays all out of sync objects in the lower panel.

Note: Once you have evaluated the impact of the out-of-sync objects, you can then decide whether or not to re-generate the schema (typically, external components which interact with the Hub are written to work with a specific version of the generated schema). If you regenerate the schema, these external components may no longer work.

If the JMS Event Schema Manager returns any out-of-sync objects, click Generate and Deploy ORS-specific Schema to re-generate the event schema.

Auto-searching for Out-of-Sync Objects

You can configure Informatica MDM Hub to periodically search for out-of-sync objects and re-generate the schema as needed.

This auto-poll feature operates within the data change monitoring thread which automatically engages a specified number of milliseconds between polls. You specify this time frame using the Message Check Interval in the Message Queues tool. When the monitoring thread is active, this automatic service first checks if the out-of-sync interval has elapsed and if so, performs the out-of-sync check and then re-generates the event schema as needed.

To configure the Hub to periodically search for out-of-sync objects:

1. Set the logical name of the schema to be generated in the JMS Event Schema Manager.

Note: If you bypass this step, the Hub issues a warning in the server log asking you configure the schema generation.

2. Enable the Queue Status for Data Changes Monitoring message.

3. Select the root node Message Queues and set the Out of sync check interval (milliseconds).

Since the out-of-sync auto-poll feature effectively depends on the Message check interval, you should set the Out-of-sync check interval to a value greater than or equal to that of the Message check interval.

Note: You can disable the out-of-sync check by setting the out-of-sync check interval to 0.

CHAPTER 31

Viewing Registered Custom Code

This chapter includes the following topics:

- [Overview, 606](#)
- [User Objects, 606](#)
- [Starting the User Object Registry Tool, 607](#)
- [Viewing User Exits, 607](#)
- [Viewing Custom Java Cleanse Functions, 607](#)
- [Viewing Custom Button Functions, 608](#)

Overview

This chapter describes how to use the User Object Registry tool to view registered custom code.

User Objects

User objects are custom functions that you can register with Informatica MDM Hub to extend Hub functionality.

The User Object Registry Tool is a read-only tool that keeps track of user objects that you develop for and then register in MDM Hub. You can access the following user objects in the User Object Registry:

User Exits

User-customized, unencrypted Java code that includes a set of fixed, predefined parameters. User exits run at a specific point during a batch job. You can configure user exits differently for each base object.

Custom Java Cleanse Functions

Java cleanse functions that uses custom logic to supplement the standard cleanse libraries. Custom Java cleanse functions are JAR files that Informatica MDM Hub stores as binary large objects in the database.

Custom Button Functions

User interface functions that supply additional icons and logic to Data Manager, Merge Manager, and Hierarchy Manager.

Starting the User Object Registry Tool

To start the User Object Registry tool:

1. In the Hub Console, connect to an Operational Reference Store (ORS).
2. Expand the Informatica Utilities workbench and then click **User Object Registry**.

The Hub Console displays the User Object Registry tool.

The User Object Registry tool displays the following areas:

Column	Description
Registered User Object Types	Hierarchical tree of user objects registered in the selected ORS, organized by the following categories: <ul style="list-style-type: none">- User Exits.- Custom Java Cleanse Functions.- Custom Button Functions.
User Object Properties	Properties for the selected user object.

Viewing User Exits

This section describes how to view user exits in the User Object Registry tool.

About User Exits

User exits consists of Java code that runs at specific points in the batch or SIF API processes to extend the functionality of MDM Hub.

User exits are triggered by the Informatica MDM Hub back-end processes that provide a mechanism to integrate custom operations with Hub Server processes such as post-load, post-merge, and post-match.

Viewing User Exits

To view the Informatica MDM Hub user exits in the User Object Registry tool:

1. Start the User Object Registry tool.
2. In the list of user objects, select **User Exits**.

The User Object Registry tool displays the user exits.

Viewing Custom Java Cleanse Functions

This section describes how to view registered custom Java cleanse functions in the User Object Registry tool.

About Custom Java Cleanse Functions

The User Object Registry exposes the details of custom cleanse functions that have been added to Java libraries (not user libraries).

In Informatica MDM Hub, you can build and execute cleanse functions that cleanse data. A cleanse function is a function that is applied to a data value in a record to standardize or verify it. For example, if your data has a column for salutation, you could use a cleanse function to standardize all instances of “Doctor” to “Dr.” You can apply cleanse functions successively, or simply assign the output value to a column in the staging table.

How Custom Java Cleanse Functions Are Registered

Cleanse functions are configured using the Cleanse Functions tool in the Hub Console.

Viewing Registered Custom Java Cleanse Functions

To view the registered custom Java cleanse functions in the User Object Registry tool:

1. Start the User Object Registry tool.
2. In the list of user objects, select **Custom Java Cleanse Functions**.

The User Object Registry tool displays the registered custom Java cleanse functions.

Viewing Custom Button Functions

This section describes how to view registered custom button functions in the User Object Registry tool.

About Custom Button Functions

In your Informatica MDM Hub implementation, you can provide Hub Console users with custom buttons that can be used to extend your Informatica MDM Hub implementation. Custom buttons can give users the ability to invoke a particular external service (such as retrieving data or computing results), perform a specialized operation (such as launching a workflow), and other tasks. Custom buttons can be added to any of the following tools in the Hub Console: Merge Manager, Data Manager, and Hierarchy Manager.

Server and client-based custom functions are visible in the User Object Registry.

How Custom Button Functions Are Registered

To add a custom button to the Hub Console in your Informatica MDM Hub implementation, complete the following tasks:

1. Determine the details of the external service that you want to invoke, such as the format and parameters for request and response messages.
2. Write and package the business logic that the custom button will execute.
3. Deploy the package so that it appears in the applicable tool(s) in the Hub Console.

Viewing Registered Custom Button Functions

To view the registered custom button functions in the User Object Registry tool:

1. Start the User Object Registry tool.
2. Select **Custom Button Functions**.

The User Object Registry tool displays the registered custom button functions.

APPENDIX A

MDM Hub Properties

This appendix includes the following topics:

- [MDM Hub Properties Overview, 610](#)
- [Hub Server Properties, 610](#)
- [Sample Hub Server Properties File, 628](#)
- [Process Server Properties, 630](#)
- [Sample Process Server Properties File, 638](#)
- [Operational Reference Store Properties, 639](#)

MDM Hub Properties Overview

The MDM Hub properties files contain the Hub Server and Process Server configuration settings.

You can use a text editor to view or edit the properties files. When you first install the Hub Server and the Process Server, the installer sets the values for some of the properties in the properties files. After installation, you can edit the properties to change the behavior of the MDM Hub. For example, to configure data encryption for the Hub Server, you need to specify the path and file name of the data encryption JAR in the `encryption.plugin.jar` property.

Some properties are optional, which means you must add them manually to the MDM Hub properties files. For example, to enable clustering with WebSphere, you must add the `cluster.flag` property to the Hub Server properties, and then set its value to `true`.

After you edit the properties files, restart the application server for the change to take effect.

Hub Server Properties

You configure the Hub Server properties in the `cmxserver.properties` file.

The `cmxserver.properties` file is in the following directory:

<MDM Hub installation directory>/hub/server/resources

Business Entity Download Properties

The following properties change the default limitations for Business Entity download:

pdf_num_of_children

Optional. Must be added manually. Specifies the maximum number of children you can export for a parent business entity. The default value is 1000.

pdf-depth

Optional. Must be added manually. Specifies the maximum depth of child records you can export for a business entity. The default value is 20.

Task Export Properties

To change the default limitations for task download, configure the following Hub Server property:

cmx.e360.taskdata.export.limit

Number of tasks that you export into an Excel file up to a maximum of 10000 records. Default is 1000.

Selective Filtering Properties

The following property allows to enable or disable filtering on the child or grand child records for a parent business entity:

cmx.e360.legacysearch.filterchildoutput

Enable the property to apply filtering on the child or grandchild records for a parent business entity. Default is false.

Hub Server Environment Properties

The following properties set the location of the Hub Server and connection details for the application server and the database:

cmx.home

Installation directory of the Hub Server. This property is set during the Hub Server installation.

cmx.appserver.hostname

Name of the EJB cluster to which you want to deploy the MDM Hub. Specify host names of the cluster servers in the following format:

`<host_name>.<domain>.com`

For more information about deploying the Hub Server in a clustered environment, see the *Multidomain MDM Installation Guide*.

cmx.appserver.naming.protocol

Naming protocol for the application server type. The default values are `iiop` for Websphere, `jnp` for JBoss 5, `remote` for JBoss 7, and `t3` for WebLogic. This property is set during the Hub Server installation.

cmx.appserver.rmi.port

Application server port. The default values are `2809` for Websphere, `7001` for WebLogic, `1099` for JBoss 5, and `4447` for JBoss 7. For more information about deploying the Hub Server in a clustered environment, see the *Multidomain MDM Installation Guide*.

cmx.appserver.type

Type of application server. This property can have one of the following values: `JBoss`, `WebSphere`, or `WebLogic`. This property is set during the Hub Server installation.

cmx.server.attachment.temp.ttl_minutes

Number of minutes after a file is created in the TEMP storage before it expires. Set to `0` to prevent files from expiring. Default is `60`.

cmx.server.masterdatabase.type

Type of the MDM Hub Master Database. This property can have one of the following values: `DB2`, `Oracle`, or `MSSQL`. This property is set during the Hub Server installation.

cmx.server.masterdatabase.schemaname

Required for the IBM DB2 environments only. Use the property to specify the name of the MDM Hub Master Database when the name is other than `cmx_system`. Default is `cmx_system`.

cookie-secure

Secures Data Director session cookies. To enable secure IDD session cookies, uncomment the cookie-secure flag and set the value to `true`. Default is `false`.

Restart the Hub Console for the change to take effect.

http-only

Secures Data Director session cookies for HTTP only. To enable session cookies, uncomment the http-only flag and set the value to `true`. Default is `false`.

Restart the Hub Console for the change to take effect.

locale

Locale for the Hub Server and Hub Console. The value of this property is set when you first install the Hub Server.

Application Server Properties for JBoss

The Hub Server uses the following properties when the Hub Server is running in a JBoss application server:

cmx.appserver.version

Version of JBoss on application server. This property can have one of the following values: `5` or `7`. This property is set during the Hub Server installation.

cmx.jboss7.management.port

JBoss native management port. Default is `9990` for JBoss. This property is set during the Hub Server installation.

cmx.jboss7.security.enabled

Enables JBoss EJB security. Set to `true` to enable JBoss EJB security. Set to `false` to disable JBoss EJB security. Default is `true`. For information about configuring JBoss security, see the *Multidomain MDM Installation Guide*.

cmx.server.ejb3

For JBoss 7 only. Set to `true` to enable application server EJB3 lookup. Default is `false`. This property is set during the Hub Server installation.

ejb-client-version

Optional. Must be added manually. Specifies the EJB client version. If you do not want to use the default JBoss EJB client, use `ejb-client-version` to specify a different EJB client. For information about configuring an EJB client for the Hub Console, see the *Multidomain MDM Installation Guide*.

jboss.cluster

For JBoss 7 only. Specifies whether EJB servers are clustered for the Hub Server. Set to `true` to enable EJB clustering. Set to `false` if you do not have clustered EJB servers. Default is `false`.

Application Server Properties for WebSphere

The Hub Server uses the following properties when the Hub Server is running in a WebSphere application server:

cluster.flag

Optional. Must be added manually. For WebSphere only. Specifies whether clustering is enabled. Set to `true` to enable clustering. Set to `false` to disable clustering. Default is `false`.

cmx.appserver.password

Password of the WebSphere administrator. This property becomes available when WebSphere administrative security is enabled.

cmx.appserver.username

User name of the WebSphere administrator. This property becomes available when WebSphere administrative security is enabled.

cmx.appserver.soap.connector.port

For WebSphere only. SOAP connector port. Default value is 8880 for WebSphere. For more information about deploying the Hub Server in a clustered environment, see the *Multidomain MDM Installation Guide* for WebSphere.

cmx.websphere.security.enabled

Specifies if WebSphere security is enabled. Set to `true` or `yes` to enable WebSphere administrative security. Default is `no`. For information about enabling WebSphere administrative security, see the *Multidomain MDM Upgrade Guide*.

cmx.websphere.security.sas.config.name

For WebSphere only. Custom name of the `sas.client.props` file. Use for environments that use secure EJB lookups.

Default is `sas.client.props`

cmx.websphere.security.sas.config.url

For WebSphere only. Location of the `sas.client.props` file. Use for environments that use secure EJB lookups.

Default is `https://yourdomain.com:9443/cmx/filesx/Security/WebSphere/sas.client.props`.

cmx.websphere.security.ssl.config.name

For WebSphere only. Custom name of the `ssl.client.props` file. For environments that use secure EJB lookups.

Default is `ssl.client.props`

cmx.websphere.security.ssl.config.url

For WebSphere only. Location of the `ssl.client.props` file. Use for environments that use secure EJB lookups.

Default is `https://yourdomain.com:9443/cmx/filesx/Security/WebSphere/ssl.client.props`.

was.jms.log.dir

Optional. Must be added manually. For WebSphere only. Specifies the location of the log directory for SIB, which is a WebSphere resource.

was.jms.permanent_store.dir

Optional. Must be added manually. For WebSphere only. Specifies the location of the permanent store directory for SIB, which is a WebSphere resource.

was.jms.temp_store.dir

Optional. Must be added manually. For WebSphere only. Specifies the location of the temporary store directory for SIB, which is a WebSphere resource.

Database Properties

The following properties can be set for databases:

cmx.server.loadWorker.max.joins.optimization

Optional. Must be added manually. For IBM DB2 only. Specifies the maximum number of joins used in a query. Set to 20 when queries, with more than 12 lookup tables in a load job, take an excessively long time to run in DB2. Default is 30.

General Properties

The following properties set the behavior of the Hub Server processes:

cmx.outbound.bypass.multixref.insystem

Optional. Must be set manually. Set to `true` to bypass the creation of messages on the Hub Server when a SIF API updates a base object with multiple cross-reference records. Default is `false`.

cmx.server.datalayer.cleanser.execution

Specifies where the cleanse jobs run. Set to `LOCAL` to run cleanse jobs on the application server. Set to `DATABASE` to run cleanse jobs on the database server. Default is `LOCAL`. For information about integrating cleanse engines, see the *Multidomain MDM Cleanser Adapter Guide*.

cmx.server.datalayer.cleanser.working_files

Specifies whether the temporary files created during cleanse jobs are stored or not. You can use the temporary files for troubleshooting or auditing purposes. Set to `FALSE` to delete the temporary working files. Set to `KEEP` to store the temporary working files. Default is `KEEP`. For information about integrating cleanse engines, see the *Multidomain MDM Cleanser Adapter Guide*.

cmx.server.datalayer.cleanser.working_files.location

Location of the cleanse job working files. MDM Hub uses this property during the Hub Server initialization routine. This property is set during the Hub Server installation. Do not modify this property. For information about integrating cleanse engines, see the *Multidomain MDM Cleanser Adapter Guide*.

cmx.server.encryptionMethod

Optional. Must be added manually. Set to `SSL` to enable SSL encryption.

cmx.server.load.nonsmos.sourcesystem.enddate.like.smos

Sets the relationship end-date of a system that is not a State Management Override System (SMOS) to be the same as that of an SMOS. Set to `true` to enable the relationship end-date to be the same as an SMOS.

cmx.server.met.max_send_size

Maximum file size, in bytes, that the Repository Manager can send. Default is 9000000.

cmx.server.met.promotion_snapshot

Optional. Must be added manually. Set to `true` to enable generation of `.meta` files. Set to `false` to disable generation of `.meta` files. Default is `true`.

cmx.server.multi_data_set_schema

Optional. Must be added manually. Set to `true` to enable message trigger XML messages to contain the parent record and all corresponding child records. Set to `false` to disable message trigger XML messages to contain the parent record and all corresponding child records. Default is `false`.

cmx.server.poller.monitor.interval

Number of seconds between polls to all servers. Set to 0 to disable server polling. Default is 30.

cmx.server.put.autopopulate.missing.user.columns.bo.list

Must be added manually. Set the property when the Nullable property is disabled for the base object columns.

When the Nullable property is disabled for columns, if a record is updated in Data Director or during a SIF PUT operation, a cross-reference record is created with values only for the updated fields. All the other cross-reference record fields, including the fields that must not be null, have null values. During the best version of the truth (BVT) calculation for the record, null values can be the winner for fields that must not be null, and an error occurs.

To ensure that the BVT calculation takes into account the fields that must not be null, set the property value to a comma-separated list of the names of base objects that have columns with the Nullable property disabled. When the property is set, the MDM Hub updates the null values in the cross-reference record with values from the associated base object. This ensures that during BVT calculation, null values are not the winner for fields that must not be null.

cmx.server.selective.bvt.enabled

Optional. Must be added manually. Specifies that the MDM Hub only applies BVT calculation to fields that are part of the SIF request. Set to `true` so that the MDM Hub updates only the fields specified in a SIF request. Default is `false`.

cmx.server.validateServerCertificate

Optional. Must be added manually. Set to `false` to turn off server certificate validation. Default is `true`.

com.informatica.mdm.message.queue.max.bytes.threshold

Optional. Must be added manually. Specifies a maximum limit, in bytes, for messages sent to the message queue. If a message exceeds the specified size, the message will not be sent and the message status is set to Failed.

com.informatica.mdm.sifapi.xref.edit.sys0.only

Optional. Must be added manually. Set to `true` to create edit cross-reference records only through the Admin source system. Set to `false` to create edit cross-reference records through all source systems. Default is `true`.

Important: You must set the property for both the Hub Server and the Process Server and the property value must be the same in both the properties files.

<connection factory name>.qcf.password

Optional. Must be added manually. Configures the MDM Hub to use the password set in the application server to configure JMS security.

<connection factory name>.qcf.username

Optional. Must be added manually. Configures the MDM Hub to use the user name set in the application server to configure JMS security. For more information about securing message queues, see [“Configuring JMS Security” on page 496](#).

databaseld.password

Optional. Must be added manually. Specifies the encrypted password to use with Password Encryption tool. For more information about using the Password Encryption tool, see the *Multidomain MDM Resource Kit Guide*.

databaseld.username

Optional. Must be added manually. Specifies user name to use with Password Encryption tool. For more information about using the Password Encryption tool, see the *Multidomain MDM Resource Kit Guide*.

encryption.plugin.jar

Path and file name of the data encryption JAR file. For information about configuring data encryption for the Hub Server, see [“Step 3. Configure Data Encryption for the Hub Server” on page 185](#).

mq.data.change.monitor.thread.start

In a multinode environment, specifies whether there is message queue polling for individual nodes. To disable message queue polling, set to `false`. Default is `true` on all Java virtual machines where an MDM Hub EAR file is deployed.

searchQuery.buildBvtTemp.MaxRowCount

Optional. Must be added manually. Specifies the maximum number of records the GetOneHop API uses during BVT calculation. Default is 5000. For more information about GetOneHop, see the *Multidomain MDM Services Integration Framework Guide*.

sif.api.hm.flyover.max.record.count

Optional. Must be added manually. Sets the maximum record count to limit the number of relationship records that the Hierarchy View relationship table displays. Default is 10000.

After you update the server properties, you must validate the schema and then redeploy the Data Director application. For information about Hierarchy View relationship table records, see the *Multidomain MDM Data Director Implementation Guide*.

sif.jvm.heap.size

Optional. Must be added manually. Sets the default heap size, in megabytes, for APIs. Default is 256.

sif.search.result.query.temptableTimeToLive.seconds

Optional. Must be added manually. For GetOneHop. Specifies the number of seconds data in temporary tables exist during a search query. Default is 30. For more information about GetOneHop, see the *Multidomain MDM Services Integration Framework Guide*.

sip.hm.entity.font.size

Optional. Must be added manually. Sets the font size in Hierarchy Manager. Values can range from 6 to 100. For information about setting Hierarchy Manager properties, see the *Multidomain MDM Data Steward Guide*.

sip.hm.entity.max.width

Optional. Must be added manually. Sets the maximum entity box width in Hierarchy Manager. Values can range from 20 to 5000. For information about setting Hierarchy Manager properties, see the *Multidomain MDM Data Steward Guide*.

sip.lookup.dropdown.limit

Number of entries that appear in a menu in the Data Manager tool and the Merge Manager tool. This property does not have a minimum or maximum limit. Default is 100.

cmx.match.training.confidence.threshold

Optional. Minimum match confidence score required to create a match rule set in the Provisioning tool. Default is 85.

cmx.match.training.data.encoding

Optional. Configures encoding for match training in the Provisioning tool. Set to 1 to enable encoding for match training. Default is 0.

Ensure that you use the same value that the `cmx.server.match.server_encoding` property uses.

Batch Process Properties

The following properties affect batch processes:

cmx.server.automerge.block_size

Block size for the automerge batch process. Default is 250.

cmx.server.automerge.threads_per_job

Number of threads for the automerge batch process. The recommended maximum value to enter is $n-1$, where n is the number of CPUs available to the Hub Server. Default is 1.

cmx.server.batch.acceptunmatchedrecordsasunique.block_size

Maximum number of records to process in each block for the Accept Non-Matched Records as Unique batch job. Default is 250.

cmx.server.batch.acceptunmatchedrecordsasunique.threads_per_job

Number of threads that the MDM Hub uses to process the Accept Non-Matched Records as Unique batch job. Default is 20.

cmx.server.batch.batchunmerge.block_size

Block size for the batch unmerge process. Default is 250.

cmx.server.batch.load.block_size

Maximum number of records to process in each block for the load job. Default is 250.

cmx.server.batch.recalculate.block_size

Maximum number of records to process in each block for the recalculate BVT and revalidate jobs. Default is 250.

cmx.server.batch.threads_per_job

Number of threads that the MDM Hub uses to process the load, recalculate BVT, and revalidate batch jobs except automerge batch jobs. Also specifies the number of threads that the MDM Hub uses for the batch unmerge process.

The recommended maximum value to enter is $n-1$, where n is the number of CPUs available to the Hub Server. Default is 10.

cmx.server.batch.max_concurrent_job_groups

Maximum number of concurrent import job groups to process. Default is 10.

cmx.server.jobControl.noOfDays

Optional. Must be added manually. Number of days of history to process for a batch group job log in the Batch Group tool in the Hub Console. Default is -1, which indicates that a log includes all history details.

cmx.server.strp_clean.execution_mode

Optional. Must be added manually. Configures the scope of operation of the background cleanup process on the match key table.

Specify one of the following values for the scope of operation:

- ALL. Deletes match tokens that have `invalid_ind=1` from all the match key tables on all the registered Operational Reference Stores.
- CONFIGURED_ORS. Deletes match tokens that have `invalid_ind=1` from all the match key tables on the Operational Reference Stores that you specify. If you set the scope of operation to `CONFIGURED_ORS`, add the `cmx.server.strp_clean.ors` property to the `cmxserver.properties` file.
- CONFIGURED_STRP. Deletes match tokens that have `invalid_ind=1` from the match key tables of specific base objects in specific Operational Reference Stores. If you set the scope of operation to `CONFIGURED_STRP`, add the `cmx.server.strp_clean.strp` property to the `cmxserver.properties` file.

cmx.server.strp_clean.ors

Optional. Must be added manually. Specifies the names of the Operational Reference Stores on which the background cleanup process must run to delete match tokens that are not valid. For example, to delete match tokens that have `invalid_ind=1` from all the match key tables in `cmx_ors1` and `cmx_ors2`, add `cmx.server.strp_clean.ors=cmx_ors1,cmx_ors2`.

cmx.server.strp_clean.strp

Optional. Must be added manually. Specifies the Operational Reference Store and base object combinations for which the background cleanup process must run to clean up match key tables. For example, to delete match tokens that have `invalid_ind=1` from the match key tables for BO1 in `cmx_ors1` and BO2 in `cmx_ors2`, add `cmx.server.strp_clean.strp=cmx_ors1.C_BO1,cmx_ors2.C_BO2`.

cmx.server.strp_clean.delete_records_count

Optional. Must be added manually. Specifies the number of records to clean up from the match key table.

cmx.server.strp_clean.retry_sec

Optional. Must be added manually. Specifies the time duration in seconds for which you want the MDM Hub to search for records with match tokens that are not valid in the match key table. Default is 60.

cmx.server.strp_clean.threads_count

Optional. Must be added manually. Specifies the number of threads that the MDM Hub uses when it searches for records with match tokens that are not valid in the match key table. Default is 20.

mq.data.change.threads

Number of threads to use to process JMS messages during the publish process. Default is 1.

mq.data.change.batch.size

Number of JMS messages to process in each batch for the publish process. Default is 500.

mq.data.change.timeout

Amount of time in seconds that is allowed to process the JMS messages. Default is 120.

Security Manager Properties

The following properties affect the Security Manager:

cmx.server.clock.tick_interval

Number of milliseconds for 1 clock tick. Default is 60000.

cmx.server.provider.userprofile.cacheable

Specifies if data can be cached. Caching data helps improve performance. Set to `true` to enable data caching. Set to `false` to disable data caching. Default is `true`.

cmx.server.provider.userprofile.expiration

Number of milliseconds that cached data survives before it expires. Default is 60000.

cmx.server.provider.userprofile.lifespan

Number of milliseconds that cached data survives before it expires. Default is 60000.

cmx.server.sam.cache.resources.refresh_interval

Number of clock ticks between intervals when the Security Access Manager (SAM) resource data is reloaded from the database. Default is 5. For information about changing the refresh interval, see the *Multidomain MDM Security Guide*. To specify the number of milliseconds for 1 clock tick, use the `cmx.server.clock.tick_interval` property.

cmx.server.sam.cache.user_profile.refresh_interval

Number of clock ticks between intervals when the SAM resource data for a user profile is reloaded from the database. Default is 30. To specify the number of milliseconds for 1 clock tick, use the `cmx.server.clock.tick_interval` property.

Oracle Database User Exits

The following properties can be used with Oracle databases. To use the properties, add the properties to both the `cmxserver.properties` file and the `cmxcleanse.properties` file.

cmx.server.dbuserexit.load.PostLoadUserExit

Optional. Specifies whether the MDM Hub calls a database postload user exit after the load process. Set to `true` to enable this property. Default is `false`. For information about enabling PL/SQL user exits, see the *Multidomain MDM Upgrade Guide* for your environment.

cmx.server.dbuserexit.put.PostLoadUserExit

Optional. Specifies whether MDM Hub calls a database user exit after performing a Put request. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PostMergeUserExit

Optional. Specifies whether MDM Hub calls a database user exit after performing a Merge request or an Automerge batch. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PreUnmergeUserExit

Optional. Specifies whether MDM Hub calls a database user exit before performing an Unmerge request or an Unmerge batch. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PostUnmergeUserExit

Optional. Specifies whether MDM Hub calls a database user exit after performing an Unmerge request or an Unmerge batch. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PreUserMergeAssignment

Optional. Specifies whether MDM Hub calls a database user exit before assigning unmerged records for review. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.AssignTask

Optional. Specifies whether MDM Hub calls a database user exit before assigning tasks to users. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.GetAssignableUserForTask

Optional. Specifies whether MDM Hub calls a database user exit before assigning tasks to users. Set to `true` to enable this property. Default is `false`.

Data Director General Properties

The following properties affect the behavior of Data Director.

Note: After you update any of the following server properties, you must validate the schema and then redeploy the IDD application.

com.siperian.dsapp.mde.common.idd2cccs.Many2ManyChild.name.version

Must be added manually. When MDM administrators generate the business entity schema, the uppercase characters in some child-level subject area names change to lowercase. To preserve the case, set the property to `10.2`.

cmx.server.sa2be.sort

Optional. Must be added manually. Specifies whether to retain the order of fields from the subject area layout in a business entity when you generate the business entity definition for the subject area. Set to `true` to preserve sort order from the original layout. Default is `false`.

case.insensitive.search

If set to `true`, you can enable the case insensitiveness attribute for individual columns in the base object, which enables case insensitive query search in Data Director. A new index is created for each column on which this setting is enabled. As index management has its own performance overhead, use this property with caution. Default is `false`.

cmx.bdd.redirect_to_login_after_logout

Optional. Must be added manually. For Google SSO authentication in Data Director only. Set to `true` to configure Data Director to return to the login screen when you log out. Set to `false` to configure Data Director to redirect to the default logout screen when you log out. Default is `false`.

cmx.bdd.server.traffic.compression_enabled

Specifies if Data Director server traffic compression is enabled. Compressing traffic helps improve performance. Set to `true` to enable compression. Set to `false` to disable compression. Default is `true`.

cmx.dataview.enabled

When MDM administrators implement the subject area model, IDD users use the **Data** tab to search, edit, and manage records. This option specifies whether the **Data** tab and related elements appear in IDD applications.

In a new installation, the default is `false`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `true`.

When `cmx.dataview.enabled=true`, the following user interface elements appear in IDD applications:

- **New** tab, which opens the **New** window with subject areas
- **Data** tab with the following temporary interfaces:
 - Data workspace tab with subject area record views to edit and manage records
 - Search tabs with search queries and search query results
 - Task tabs to manage tasks
- Links to the **Data** view from menus on other views
- Custom tabs, if configured

When the `cmx.dataview.enabled` and `cmx.e360.view.enabled` property are set to `true`, and you want to enable the Cross-reference view, the History view, and the Matching Records view related to the Data Director with subject areas, set the `cmx.e360.match_xref.view.enabled` property to `false`.

If a MDM administrator implements the Entity 360 framework, Data Director users use the **Search** box to find records and entity tabs to edit and manage master data. In this case, you can hide the **Data** tab and related elements to avoid confusing users with similar functionality. For example, if you set `cmx.e360.view.enabled=true`, set `cmx.dataview.enabled=false`.

cmx.bdd.enable_url_authentication

Optional. Must be added manually. Enables authentication for URLs in Data Director. When enabled, when users log in, they pass their user name and password to the URL in Data Director. Set to `true` to enable authentication. Set to `false` to turn off authentication. Default is `false`.

cmx.bdd.password_blowfish_encrypted

Optional. Must be added manually. Enables Blowfish encryption for user passwords when authentication is enabled for URLs in Data Director. When enabled, passwords are not exposed in the URL in Data Director. Set to `true` to enable encryption. Set to `false` to turn off encryption. Default is `false`.

cmx.display.deployed.invalid.met.app

When the metadata for an Operational Reference Store is not valid, Data Director does not display the list of deployed applications. Applications that use a different, valid Operational Reference Store are also unavailable. To show the list of deployed applications, add this property and set it to `true`.

cmx.e360.view.enabled

When MDM administrators implement the Entity 360 framework, IDD users use the **Search** box to find records and a entity tabs to edit and manage records. In a new installation, the default is `true`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `false`.

When `cmx.e360.view.enabled=true`, the following user interface elements appear in the Data Director applications:

- **New** tab, which opens the **New** window with business entities.
- **Task Manager** tab to manage tasks.
- **Search** tab with search results.
- Entity tabs to edit and manage business entity records. Entity tabs appear when you add a new business entity record or open a business entity record from search results. The label on the tab is dynamic, based on the action that opens the workspace.
- Links to the **Business Entity** view from menus on other views.
- Custom tabs, if configured

cmx.e360.submit_all

Optional. Must be added manually. Set to `true` so that Data Director submits all fields, including unedited fields, to the server to perform a validate and save operation when you edit the root or child field of a business entity record. Default is `false`.

For example, a record has the person business entity as the root field and the address and email as the child fields. If you edit the address field, the person, address and email child fields are sent to the server to validate and save the record.

cmx.dataview.taskmanager.enabled

Optional. Applicable only if the `cmx.e360.view.enabled` property is set to `false`. Indicates whether to display the Task Manager in a Data Director application that uses subject areas. Set to `true` to display the Task Manager. Default is `false`.

If the `cmx.dataview.taskmanager.enabled` is set to `false` and you do not create a Home page in the Provisioning tool for a Data Director application, the application displays the legacy Start page.

cmx.e360.match_xref.view.enabled

Specifies whether to enable the Cross-reference view and the Matching Records view for the Data Director with business entities. To enable the views, set to `true`. Default is `true`.

When the `cmx.dataview.enabled` and `cmx.e360.view.enabled` property are set to `true`, and you want to enable the Cross-reference view, the History view, and the Matching Records view related to the Data Director with subject areas, set the `cmx.e360.match_xref.view.enabled` property to `false`.

cmx.server.override_orstitle

Optional. Must be added manually. Specifies a preferred default title for current tasks when you log in to Data Director. Set the `cmx.server.override_orstitle` property to a preferred title in the `cmxserver.properties` file.

For example, if you set the property to `All Subject Areas`, the title appears as **Tasks for All Subject Areas** onscreen.

cmx.server.be-import.task-limit

Specifies the maximum number of records users can import for the task approval workflow to trigger. Set the `cmx.server.be-import.task-limit` value to the preferred maximum number. For example, set `cmx.server.be-import.task-limit=10000` for users to import up to 10000 records and the task approval workflow to trigger. If a user attempts to import more than 10000 records, the task approval workflow does not trigger and displays an error..

cmx.server.find-replace.record-limit

Specifies the maximum number of records users can replace in a bulk operation. Set the `cmx.server.find-replace.record-limit` property to the preferred maximum number. For example, if you set the property to `cmx.server.find-replace.record-limit=10000` users can find and replace up to a maximum of 10000 records.

cmx.server.find-replace.task-limit

Specifies the maximum number of replaced records that trigger the task approval workflow. Set the `cmx.server.find-replace.task-limit` property to the preferred maximum number. For example, if you set the property `cmx.server.find-replace.task-limit=500` a task approval workflow triggers when a user replaces up to 500 hundred records. If a user attempts to replace more than 500 records, an error displays.

cmx.server.find-replace.entity-record-limit

Specifies the maximum number of records users can copy from the Smart search screen to the Find and Replace screen. Set the `cmx.server.find-replace.entity-record-limit` property to the preferred maximum number. For example, if you set the property `cmx.server.find-replace.entity-record-limit=1000` users can copy up to 1000 records from the Smart search screen to the Find and Replace screen. If a user attempts to copy more than 1000 records, an error displays.

cmx.file_import.job_group.ttl

Specifies the maximum time a file import job group is stored by the MDM Hub before it is deleted. Default is `180day`. You must add a suffix after the value. The following are the suffix options: `day`, `hour`,

min or sec. For example, if you set the property `cmx.file_import.job_group.ttl=180day` the import job group is saved in MDM Hub for one hundred and eighty days.

cmx.file_import.job_group_control.ttl

Specifies the maximum time a file import job group control log is stored in the MDM Hub before it is deleted. Default is `30day`. You must add a suffix after the value. The following are the suffix options: day, hour, min or sec. For example, if you set the property `cmx.file_import.job_group_control.ttl=30day` the import job group control log is saved in MDM Hub for thirty days.

cmx.file_import.mapping.temp.ttl

Specifies the maximum time to save a file import mapping in the MDM Hub temporary storage before it is deleted. Default is `20min`. You must add a suffix after the value. The following are the suffix options: day, hour, min or sec. For example, if you set the property `cmx.file_import.mapping.temp.ttl=20min` the import job mapping is saved in MDM Hub for twenty minutes.

cmx.file_import.mapping.system.ttl

Specifies the maximum time to save a file import mapping in the MDM Hub permanent storage before it is deleted. Default is `20day`. You must add a suffix after the value. The following are the suffix options: day, hour, min or sec. For example, if you set the property `cmx.file_import.mapping.system.ttl=20day` the import job mapping is saved in MDM Hub for twenty days.

cmx.file.allowed_file_extensions

Lists the extensions of files that you can attach to a record or task in the Data Director application. By default, you can attach .pdf and .jpg files. When you specify multiple extensions, separate each value by a comma.

For example, `cmx.file.allowed_file_extensions=pdf,jpg,png,txt,zip,exe`.

cmx.file.max_file_size_mb

Specifies the size limit of files that you can attach in the Data Director application.

Note: Data Director applications that use the subject area data model have a static size limit of 20 MB. If you specify a size limit greater than 20 MB, Data Director applications that use the subject area data model uphold the static size limit of 20 MB. Data Director applications that use the business entity data model uphold the size limit defined in the `cmx.file.max_file_size_mb` property.

Data Director Search Properties

The following properties affect the behavior of search in Data Director:

ex.max.conn.per.host

Sets the maximum number of Elasticsearch nodes that you want to connect to the host. Set to the number of Elasticsearch cluster nodes on the host.

ex.max.threads

Sets the maximum number of threads that you want the Apache asynchronous non-blocking receiver to use for each node in the Elasticsearch cluster. Default is 1.

Change the value only when suggested by Informatica Global Customer Support.

es.index.refresh.interval

Sets the interval, in seconds, for Elasticsearch to commit the changes to the data after an Initially Index Smart Search Data batch job is run. The data is available for search after this time interval. Default is 30.

This property impacts the high indexing volume encountered during initial indexing. Change the value only when suggested by Informatica Global Customer Support.

cmx.task.search.records.return

Controls the Elasticsearch pagination when users search for tasks in the Task Manager of Data Director with business entities. Default is 1000.

Change the value only when suggested by Informatica Global Customer Support.

cmx.server.security.sql.injection.check

Optional. Must be added manually. Configures the SQL Injection check feature. To disable this feature, set the value to `false`, in the `cmxserver.properties` file, and then restart the server. Default is `true`.

cmx.server.batch.smartsearch.initial.block_size

Maximum number of records that the Initially Index Smart Search Data batch job can process in each block. Default is 250. When you index a large data set, increase the number of records. The recommended maximum value is 1000.

cmx.server.match.max_time_searcher

Optional. Must be added manually. Specifies the maximum duration allowed for a search to run. Default is 99999999 seconds.

cmx.server.remove_duplicates_in_search_query_results

Specifies if duplicate records appear in the search query results. Set to `true` to show duplicate records in search query results. Set to `false` to hide duplicate records in search query results. Default is `false`.

cmx.server.enrichcopager.thread_pool

Manually add the property. Sets the number of threads the EnrichCoPager property uses from the thread pool to perform parallel ReadCO operations. Default is 30. If you set the number of threads to 1, the property is disabled.

cmx.server.enrichcopager.min_rec_for_multithreading

Sets the minimum number of records to return before the EnrichCoPager property uses multithreading. Default is 2.

cmx.ss.enabled

Indicates whether to enable search. In a new installation, the default is `true`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `false`.

search.provisioning.numshards

Optional. Number of shards to create on your Elasticsearch environment. The value depends on the maximum number of shards and the total number of nodes. For example, if the maximum number of shards is 1 and the number of nodes is 3, you can create 3 shards. Default is the total number of Hub Servers.

search.provisioning.numreplicas

Optional. Number of copies of the Elasticsearch engine documents that you want to create on different nodes. Use the replication factor to create multiple copies of the documents in the shards of different nodes. You require multiple copies of the documents to achieve high availability if one or more nodes shut down unexpectedly. For example, if the replication factor is 2, you get two copies of the documents in two nodes. For Elasticsearch, the default is 0.

sif.search.result.drop.batch.interval.milliseconds

Optional. Must be added manually. Specifies the interval, in milliseconds, that the SearchResultManager daemon pauses after each batch of search results cleansing is processed. Default is 0.

sif.search.result.drop.batch.record.count

Optional. Must be added manually. Specifies the number of cached searches that the SearchResultManager daemon cleans up at one time. Default is 200.

sif.search.result.query.timeToLive.seconds

Optional. Must be added manually. Specifies the number of seconds that an unused query remains cached. Default is 900.

sif.search.result.refresh.interval.seconds

Optional. Must be added manually. Specifies the interval, in seconds, the SearchResultManager daemon pauses after running the cleanup process for cached searches. Default is 1. For information about configuring SIF Search APIs, see the *Multidomain MDM Services Integration Framework Guide*.

ssl.keyStore

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Absolute path and file name of the keystore file.

ssl.keyStore.password

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Plain text password for the keystore file.

ssl.trustStore

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Absolute path and file name of the truststore file.

ssl.trustStore.password

Required if you use the HTTPS port of the application server to configure the Hub Server. Manually add the property. Plain text password for the truststore file.

Hierarchy REST API Properties

The following properties affect hierarchy REST APIs for business entity services:

cmx.server.hierarchy.max-search-depth

Maximum depth searched when you use hierarchy REST APIs to find a hierarchy path. Default value is 100.

cmx.server.hierarchy.max-search-width

Maximum width of the searched hierarchy to include when you use the hierarchy REST APIs to export. Default value is 1000000.

com.informatica.mdm.bulk.relationship.changes.limit

Maximum number of changes in a request when you use the bulk task administration REST APIs. Default value is 1000.

Data Director Task, Workflow, and Reports Properties

The following properties affect tasks, review process workflows, and reports:

activevos.jndi

Optional. Must be added manually. Specifies the JNDI lookup string to connect report services to ActiveVOS. If you edited the ActiveVOS EAR files to customize the JNDI lookup string, use this property to specify the custom JNDI lookup string. Default JNDI lookup string is `java:/jdbc/ActiveVOS`.

activevos.merge.workflow.service.name

You must specify the name of the MDM service calls to Informatica ActiveVOS. By default, this property is not defined. If the property is not defined, an automatic merge task is not created.

activevos.workflow.startup.timeout.seconds

Number of seconds to wait for Informatica ActiveVOS to create a task and return a task ID. Default is 10.

cmx.e360.BPMProcess.view.enabled

Optional. Must be added manually. Indicates whether to display the workflow diagram associated with the tasks in the Task Manager for the users with the ActiveVOS abAdmin role assigned. Set to `true` to display the workflow diagram. Default is `false`.

cmx.e360.BPMProcess.view.autologout.seconds

Optional. Must be added manually. Number of seconds for an ActiveVOS session to remain active when you access the workflow diagram in the Task Manager. The session ends after the specified time period. Default is 30.

cmx.server.task.grouppotentialmergebyruleid

Optional. Must be added manually. Specifies that a manual match task that generates multiple matches creates multiple task entries with the same ROWID. Set to `false` to create one task for each match entry. Default is `true`.

sip.task.assignment.interval

Number of minutes between automatic task assignments. Set to 0 to disable automatic task assignments. Default is 0.

sip.task.assignment.start.delay

The time in minutes that automatic task assignment waits to start after the MDM Hub initializes. If you do not configure a delay, an error can occur when a user creates tasks. Default is 10 minutes.

sip.task.digest.interval

Number of hours between task notifications. Set to 0 to disable task notifications. Default is 0.

sip.task.maximum.assignment

Number of tasks automatically assigned to each user when automatic task assignment is enabled. Default is 25.

task.creation.batch.size

Optional. Must be added manually. Sets the maximum number of records to process for each match table for each iteration of the automatic tasks assignment process. Default is 1000.

For information about configuring merge task properties, see the *Multidomain MDM Business Process Manager Adapter SDK Implementation Guide*.

task.creation.maximum

Optional. Must be added manually. Sets the maximum number of tasks that the MDM Hub creates for each match table. Default is 50. If the number of tasks exceeds this value, then no more merge tasks for records in the match table can be created until the number of tasks associated with the match table are closed.

Siperian Workflow Engine Mail Server Properties

When you use the Siperian workflow engine, the following properties affect the behavior of the mail server that is used for task notifications:

mail.smtp.auth

Determines whether the specified mail server requires authentication for outgoing messages. If you use the MDM Hub mail server, set `mail.smtp.auth` to `true`. Default is `false`.

For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

mail.smtp.host

Name of the mail server for task notification emails. After you update the server properties, you must validate the schema and then redeploy the Data Director application. For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

mail.smtp.password

Password for the specified `mail.smtp.user`. If `mail.smtp.auth` is `true`, set a value for `mail.smtp.password`. For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

mail.smtp.port

Port number of the mail server. Default is 25. For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

mail.smtp.sender

Specifies the email address of the task notification email sender. For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

mail.smtp.user

User name for the outgoing mail server. If `mail.smtp.auth` is `true`, set a value for `mail.smtp.user`. For information about configuring task notification emails, see the *Multidomain MDM Data Director Implementation Guide*.

Password Hashing and Custom Hashing Properties

The following properties affect password hashing and custom hashing algorithms:

password.security.hash.algo

Determines the hashing algorithm (ALGO_NAME) that is used to encrypt passwords in the MDM Hub. This property is set during the Hub Server installation. Set to `SHA3` to use the SHA3 hashing algorithm. Set to any other name, without special characters or spaces, for a custom hashing algorithm.

For more information about configuring hashing algorithms, see the *Multidomain MDM Security Guide*.

password.security.hash.algo.<ALGO_NAME>.class

Contains the underlying implementation class of the hashing algorithm specified in the `password.security.hash.algo` property. This property is set during the Hub Server installation.

password.security.hash.algo.property.<param-name>

Optional. Specifies any custom property of a configured hashing algorithm. By default, specifies the size property of the SHA3 hashing algorithm. Set to one of the following values: 224, 256, 384, or 512. Default is 512.

com.informatica.mdm.security.certificate.provider.class

Set to `com.siperian.sam.security.certificate.PKIUtilDeafultImpl` for the default certificate provider in the MDM Hub. This property is set during the Hub Server installation.

informatica.security.unique.id

Customer hashing key that is used for password hashing. Informatica recommends to use a hashing key that contains a sequence of up to 32 hexadecimal characters with no delimiters. For more information about using a customer hashing key, see the *Multidomain MDM Security Guide*.

Important: Protect the secrecy of your customer hashing key. If the value of the customer hashing key is lost, you must reset all passwords.

Security Configuration Utility Properties

To use the security configuration utility, set the following properties:

mdm.mail.server.host

Set to the SMTP server host of the email client that is used by the MDM administrator. For example, `smtp.gmail.com`. If you reset a password, the security configuration utility sends a new temporary password to the associated email address of the user account.

mdm.mail.server.port

Set to the port that is used by the email client that is used by the MDM administrator.

mdm.mail.server.user

Set to the email address of the MDM administrator. For example, `MDM_Hub_admin@gmail.com`.

mdm.mail.server.password

Enter the password for the email address of MDM administrator.

mdm.mail.server.smtpauth

Set to `true` to enable SMTP authentication. Required to connect to Gmail SMTP server.

mdm.mail.server.ttls

Set to `true` to enable TTLS authentication. Required to connect to Gmail SMTP server.

Sample Hub Server Properties File

The Hub Server properties file is called `cmxserver.properties`.

The following example shows the contents of a typical `cmxserver.properties` file:

```
# Installation directory
cmx.home=C:/infamdm/Hub_971_DB2/server

# Master database settings
cmx.server.masterdatabase.type=DB2

# Server settings
# Application server type: jboss, websphere or weblogic
cmx.appserver.type=jboss
cmx.appserver.version=7
#Should application server use ejb3 lookup (Jboss7 supports only ejb3 lookup mechanism )
cmx.server.ejb3=true

# Application server hostname. Optional property to be used for deploying MDM into EJB
cluster
#cmx.appserver.hostname=clustername

# The following port number depends on appserver type
# default setting: 2809 for websphere, 1099 for jboss5, 4447 for jboss7 7001 for weblogic
cmx.appserver.rmi.port=4447
```



```

# default setting: iiop for websphere, jnp for jboss5, remote for jboss7, t3 for weblogic
cmx.appserver.naming.protocol=remote
# default setting: 8880 for websphere only (this is not being used in jboss and weblogic
cmx.appserver.soap.connector.port=
# default setting: 'No' for websphere only (this is not being used in jboss and weblogic
cmx.websphere.security.enabled=
## You can customize location of sas.client.props and ssl.client.props which are used
for secured ejb lookup
#cmx.websphere.security.sas.config.url=https://yourdomain.com:9443/cmx/filesx/Security/
Websphere/sas.client.props
#cmx.websphere.security.ssl.config.url=https://yourdomain.com:9443/cmx/filesx/Security/
Websphere/ssl.client.props
## Or you can just customize file names (default values are sas.client.props and
ssl.client.props)
#cmx.websphere.security.sas.config.name=sas.client.props
#cmx.websphere.security.ssl.config.name=ssl.client.props

# enable JBoss EJB security support
#cmx.jboss7.security.enabled=true

# DO NOT EDIT SETTINGS BELOW
cmx.server.datalayer.cleanse.execution=SERVER
cmx.server.datalayer.cleanse.working_files.location=C:/infamdm/Hub_971_DB2/server/logs
cmx.server.datalayer.cleanse.working_files=LOCAL

# SAM properties
cmx.server.sam.cache.resources.refresh_interval=5
cmx.server.sam.cache.user_profile.refresh_interval=1
cmx.server.clock.tick_interval=60000
cmx.server.provider.userprofile.cacheable=true
cmx.server.provider.userprofile.expiration=60000
cmx.server.provider.userprofile.lifespan=60000

# Setting for dropdown limit
sip.lookup.dropdown.limit=100

#
# Task settings
#
# Number of Hours between task notifications. 0 means that notifications are disabled.
sip.task.digest.interval=0
# Number of Minutes between automated task assignments. 0 means that assignment is
disabled.
sip.task.assignment.interval=0
# Maximum number of tasks automatically assigned to each user
sip.task.maximum.assignment=25

#
# Mail server settings for task notification emails
#
mail.smtp.host=
mail.smtp.port=25
mail.smtp.auth=false
mail.smtp.sender=siperian_task_notification@siperian.com
# Use the following if your smtp server requires authentication.
#mail.smtp.user=
#mail.smtp.password=

# interval sleeping between polling all servers in seconds, default=10, 0 will disable
cmx.server.poller.monitor.interval=30

#MET properties
cmx.server.met.max_send_size=9000000

# BDD traffic compression option
cmx.bdd.server.traffic.compression_enabled=true

# Sif property to remove duplicates from the search query results
cmx.server.remove_duplicates_in_search_query_results=false

```

```
# The Case Insensitive Search feature can be disabled by setting this property to false.
case.insensitive.search=false

# Locale for hub server and hub console
locale=en

# cookie secure flag and cookie httpOnly flag
# In JBoss, both of these flags will be used.
# In WebLogic, cookie-http-only flag is set to true by default, so only cookie-secure
flag will be used here.
# in WebLogic, setting httpOnly will have no effect.
# in webSphere, these setting should be done thorough websphere console under Session
Management
# in deployed siperian-mrm.ear.
#cookie-secure=false
#http-only=false

#Property for batch job processing. The number of threads will be used to distribute
blocks of a batch job to batch servers.
cmx.server.batch.threads_per_job=10

#Block size for Load job.
cmx.server.batch.load.block_size=250
#Block size for Recalculate and Revalidate job.
cmx.server.batch.recalculate.block_size=250

#Properties for Automerge batch job (number of threads to use and block size)
cmx.server.automerge.threads_per_job=1
cmx.server.automerge.block_size=250

#Properties for Active VOS BPM integration
# Name of the merge operation in active vos
activevos.merge.workflow.operation.name=start
# Name of the service for all mdm service calls to ActiveVOS
activevos.merge.workflow.service.name=Merge
#The wait time for ActiveVOS to create task for the process and return task ID
activevos.workflow.startup.timeout.seconds=10
encryption.plugin.jar=C:\Temp\informatica_dataencryption.jar
```

Process Server Properties

You can configure the Process Server properties in the `cmxcleanse.properties` file.

The `cmxcleanse.properties` file is in the following directory:

```
<MDM Hub installation directory>/hub/cleanse/resources
```

cmx.server.datalayer.cleanse.working_files.location

Location of custom Java library files, which by default is set to the location of the Process Server installation directory. If you specify a different location, ensure that you restart the application server after the change. For information about integrating cleanse engines, see the *Multidomain MDM Cleanse Adapter Guide*.

cmx.server.datalayer.cleanse.working_files

Specifies whether the temporary files created during cleanse jobs are stored or not. You can use the temporary files for troubleshooting or auditing purposes. Set to `FALSE` to delete the temporary working files. Set to `KEEP` to store the temporary working files. Default is `KEEP`. For information about integrating cleanse engines, see the *Multidomain MDM Cleanse Adapter Guide*.

cmx.server.datalayer.cleanser.execution

Specifies where the cleanser jobs run. Set to `LOCAL` to run cleanser jobs on the application server. Set to `DATABASE` to run cleanser jobs on the database server. Default is `LOCAL`. For information about integrating cleanser engines, see the *Multidomain MDM Cleanser Adapter Guide*.

cmx.home

Installation directory of the Process Server. This property is set during the Process Server installation.

cmx.appserver.type

Type of application server. This property can have one of the following values: `JBoss`, `WebSphere`, or `WebLogic`. This property is set during the Process Server installation.

cmx.appserver.version

Version of JBoss on application server. This property can have one of the following values: `5` or `7`. This property is set during the Process Server installation.

cmx.appserver.soap.connector.port

For WebSphere only. SOAP connector port. Default value is `8880` for WebSphere.

cmx.websphere.security.enabled

Specifies if WebSphere security is enabled. Set to `true` or `yes` to enable WebSphere administrative security. Default value is `No`.

cmx.jboss7.management.port

JBoss management port. Default is `9990` for JBoss. This property is set during the Process Server installation.

cmx.server.load.nonsmos.sourcesystem.enddate.like.smos

Sets the relationship end-date of a system that is not a State Management Override System (SMOS) to be the same as that of an SMOS. Set to `true` to enable the relationship end-date to be the same as an SMOS.

cmx.server.match.lwm

Optional. Must be added manually. Controls the lightweight matching feature. To enable the lightweight matching feature with full scoring on the matching records, set to `Y`. To enable the lightweight matching feature without full scoring on the matching records, set to `ONLY`. Default is `N`.

Use this property with the `cmx.server.match.lwm_param` and `cmx.server.match.stats` properties.

cmx.server.match.lwm_param

Optional. Must be added manually. Requires that the `cmx.server.match.lwm` property is set to `Y` or `ONLY`. Set the property value to the SSA-NAME3 controls in the following format:

```
LWM=Y LWM_FIELDS=<field1>,<weight1>[, ..., <fieldn>,<weightn>]  
LWM_LIMIT=<Reject>[, <Accept>]
```

cmx.server.match.stats

Optional. Must be added manually. Requires that the `cmx.server.match.lwm` property is set to `Y` or `ONLY`.

cmx.server.match.server_encoding

Configures encoding for match processing. Set to `1` to enable encoding for match processing. Default is `0`.

cmx.server.match.max_records_per_ranger_node

Number of records per match ranger node. More records per match uses more memory. The optimal number of records for each match depends on the memory and processing power available to the Process Server. Default is 3000.

cmx.server.match.disable_subtype_match_child

Specifies whether the Find Duplicates process might use a less strict child validation strategy. As a result, the search might return some incorrect matches at the child record level. Set to 1 if you want to allow such matches. Default is 0.

cmx.server.match.max_return_records_searcher

Sets a limit on the number of candidate records that are scored for a search thread during a fuzzy search operation. Must be added manually. Default is -1.

Set the property when the fuzzy search operations are time sensitive or CPU intensive. The MDM Hub considers the value of the `GETLIST Limit` property for the Operational Reference Store (ORS) to determine when a search thread stops. You configure the `GETLIST Limit` property by using the Databases tool in the Hub Console.

If you set a value for the `cmx.server.match.max_return_records_searcher` property, the fuzzy search operations might complete faster. The search threads stop when one of the following conditions is met:

- The number of candidate records that are scored reach the value set for the `cmx.server.match.max_return_records_searcher` property.
- The number of matched records reach the value set for the `GETLIST Limit` property.

If you do not set the property or use the default value of -1, the fuzzy search operations ignore the `cmx.server.match.max_return_records_searcher` property and are based on the `GETLIST Limit` property. The search threads stop when one of the following conditions is met:

- The number of matched records reach the value set for the `GETLIST Limit` property.
- No candidate records are left to score.

com.informatica.mdm.sifapi.xref.edit.sys0.only

Optional. Must be added manually. Set to `true` to create edit cross-reference records only through the Admin source system. Set to `false` to create edit cross-reference records through all source systems. Default is `true`.

Important: You must set the property for both the Hub Server and the Process Server and the property value must be the same in both the properties files.

cmx.ss.enabled

Indicates whether to enable search. In a new installation, the default is `true`. When upgrading, if this property is set, the value remains set to the pre-upgrade value. If this property is not set, the default is `false`.

JBoss 6.4.0 only. When you enable search in an environment that uses JBoss 6.4.0, you must set `cmx.server.match.file_load` to `false`. This setting forces the Process Server to use the JDBC uploader instead of native database utilities for matches.

cleanse.library.addressDoctor.property.SetConfigFile

Informatica Address Verification configuration file path. For example, `C:/infamdm/Hub/cleanse/resources/AddressDoctor/5/SetConfig.xml`. For information about integrating cleanse engines, see the *Multidomain MDM Cleanse Adapter Guide*.

cleanse.library.addressDoctor.property.ParametersFile

Informatica Address Verification parameters file path. For example, C:/infamdm/Hub/cleanse/resources/AddressDoctor/5/Parameters.xml.

cleanse.library.addressDoctor.property.DefaultCorrectionType

Informatica Address Verification correction type, which must be set to PARAMETERS_DEFAULT.

cleanse.library.trilliumDir.property.config.file.1

Trillium Director cleanse library configuration file 1 file path. For example, C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td_default_config_Global.txt. For information about integrating cleanse engines, see the *Multidomain MDM Cleanse Adapter Guide*.

cleanse.library.trilliumDir.property.config.file.2

Trillium Director cleanse library configuration file 2 file path. For example, C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td11_default_config_US_detail.txt.

cleanse.library.trilliumDir.property.config.file.3

Trillium Director cleanse library configuration file 3 file path. For example, C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td11_default_config_US_summary.txt.

cleanse.library.trilliumDir11.property.config.file.1

Trillium Director 11 cleanse library configuration file 1 file path. For example, C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_Global.txt. For information about integrating cleanse engines, see the *Multidomain MDM Cleanse Adapter Guide*.

cleanse.library.trilliumDir11.property.config.file.2

Trillium Director 11 cleanse library configuration file 2 file path. For example, C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_US_detail.txt.

cleanse.library.trilliumDir11.property.config.file.3

Trillium Director 11 cleanse library configuration file 3 file path. For example, C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_US_summary.txt.

cleanse.library.trilliumDir.property.set_maximum_retry_count

Optional. Sets the maximum number of times the MDM Hub attempts to connect to the Trillium server to process a record. Default is 5. For information about increasing the number of network connection retries, see the *Multidomain MDM Cleanse Adapter Guide*.

cleanse.library.group1EntServer.property.config.file

Group1Software Enterprise Server configuration file. This property is set during the Process Server installation.

cleanse.library.group1CDQ.property.config.file

Group1Software CDQ Server configuration file. This property is set during the Process Server installation.

cleanse.library.firstLogicDirect.property.config.file

FirstLogicDirect configuration file. This property is set during the Process Server installation.

cmx.server.match.distributed_match

Optional. Must be added manually. Specifies whether the Process Server participates in a distributed processing environment for cleanse operations and fuzzy match processes. Default is 1, which is enabled. To disable distributed processing, set to 0.

For more information about configuring multiple Process Servers, see the *Multidomain MDM Installation Guide*.

cmx.server.cleansize.min_size_for_distribution

Optional. Must be added manually. Specifies the size at which a job can be distributed among the Process Servers. Default is 1000.

cmx.server.java_jdbc_loader

Optional. Must be added manually. Specifies whether to use the JDBC file loader instead of an SQL loader. Set to `true` to use the JDBC file loader. Default is `false`.

cmx.server.tokenize.file_load

Optional. Must be added manually. Specifies whether to use an intermediate file to load data into the database for tokenization. Set to `true` to use an intermediate file to load data. Set to `false` for direct data load. Default is `true` for Oracle and IBM DB2 environments, where using intermediate files improves performance. Default is `false` for Microsoft SQL Server environments.

cmx.server.tokenize.loader_batch_size

Optional. Must be added manually. Maximum number of insert statements to send to the database during direct load of the tokenization process. Default is 1000.

cmx.server.match.file_load

Optional. Must be added manually. Specifies whether to use an intermediate file to load data into the database for matching. Set to `true` to use an intermediate file to load data. Set to `false` for direct data load. Default is `true` for Oracle and IBM DB2 environments. Default is `false` for Microsoft SQL Server environments and IBM DB2 environments configured for external match.

Note: When the `cmx.server.match.file_load` property is set to `false`, the number of matches in the cleanse log might differ from the Batch Viewer. If the number of matches differ, refer to the number of matches listed in the Batch Viewer.

cmx.server.match.loader_batch_size

Optional. Must be added manually. Maximum number of insert statements to send to the database during direct load of the match process. Default is 1000.

cmx.server.match.exact_match_fuzzy_bo_api

Optional. Must be added manually. Set to 1 to perform exact matches on fuzzy base objects. Set to 0 to disable exact matches on fuzzy base objects. Default is 0.

Restart the application server for the change to take effect. For information about configuring exact matches on fuzzy base objects, see the *Multidomain MDM Services Integration Framework Guide*.

encryption.plugin.jar

Optional. Must be added manually. Path and file name of the data encryption JAR file. For information about configuring data encryption, see [“Step 3. Configure Data Encryption for the Hub Server” on page 185](#).

cmx.server.bmg.use longs

Optional. Must be added manually. Set to 1 to enable the Process Server to use long ROWID_OBJECT values. Set to 0 to disable the Process Server from using long ROWID_OBJECT values. Default is 0.

cmx.server.match.threshold_to_move_range_to_hold

Optional. Must be added manually. Sets the upper limit of records that the Match Analyze job can move to the on-hold status. Default is 1000000.

cmx.server.dbuserexit.load.PostLoadUserExit

Optional. Must be added manually to both the `cmxserver.properties` file and the `cmxcleanse.properties` file. For Oracle only. Specifies whether the MDM Hub calls a database postload user exit after the load process. Set to `true` to enable this property. Default is `false`. For information about enabling PL/SQL user exits, see the *Multidomain MDM Upgrade Guide* for your environment.

cmx.server.dbuserexit.PostLandingUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a post-landing user exit. Set to `true` to enable this property. Default is `false`.

For information about enabling PL/SQL user exits, see the *Multidomain MDM Upgrade Guide*.

cmx.server.dbuserexit.PreStageUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a database user exit before performing a Stage request. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PostStageUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a database user exit after performing a Stage request. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PreMatchUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a database user exit before performing a Match request. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PostMatchUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a database user exit after performing a Match request. Set to `true` to enable this property. Default is `false`.

cmx.server.dbuserexit.PostMergeUserExit

Optional. Must be added manually. For Oracle only. Specifies whether MDM Hub calls a database user exit after performing a Merge request. Set to `true` to enable this property. Default is `false`.

cluster.flag

Optional. Must be added manually. For WebSphere only. Specifies whether clustering is enabled. Set to `true` to enable clustering. Set to `false` to disable clustering. Default is `false`.

cmx.server.cleanser.number_of_recs_batch

Optional. Must be added manually. Sets the maximum number of records for cleansing included in a batch. Default is 50.

For information about configuring run-time behaviour in the Process Server, see the *Multidomain MDM Cleanser Adapter Guide*.

cmx.server.match.searcher_search_level

Optional. Must be added manually. Sets the search level for the Extended Search in Data Director. Value can be `Narrow`, `Typical`, `Exhaustive`, or `Extreme`. Default is `Narrow`.

After you update the server properties, you must validate the schema and then redeploy the Data Director application. For information about search levels, see [“Search Levels” on page 401](#). For information about configuring Extended Search, see the *Multidomain MDM Data Director Implementation Guide*.

cmx.server.match.searcher.database.worker.multithreaded

Optional. Must be added manually. When set to `true`, multiple parallel threads are used to process the search ranges and optimize the performance of the SearchMatch API. By default, multi-threaded range processing is disabled.

If you set the `cmx.server.match.searcher.database.worker.multithreaded` property, ensure that you also set the thread count by configuring the `cmx.server.match.searcher_thread_count` property.

cmx.server.match.searcher.dbfiltered.max.key.size

Optional. Specifies the DBFILTERED threshold to optimize the performance of the SearchMatch API. The DBFILETRED feature is invoked when the SearchMatch record has a SSA_KEY that is less than or equal to the value of the `cmx.server.match.searcher.dbfiltered.max.key.size` property.

cmx.server.match.searcher.resultset.size

Specifies the resultset size of a SearchMatch database query.

cmx.server.match.searcher_thread_count

Optional. Must be added manually. Configures the thread count for the SearchMatch API. Default is 1. Set to 1 to use one thread for the SearchMatch API.

If you set the `cmx.server.match.searcher_thread_count` property to a value other than the default value, ensure that you set the `cmx.server.match.searcher.database.worker.multithreaded` property to `true`.

For information on optimizing performance of the SearchMatch API, see the following H2Ls on the Informatica My Support Portal:

- *Multidomain MDM for IBM DB2 Performance Tuning* at <https://mysupport.informatica.com/docs/DOC-11208>
- *Multidomain MDM for Oracle Performance Tuning* at <https://mysupport.informatica.com/docs/DOC-11207>
- *Multidomain MDM for Microsoft SQL Server Performance Tuning* at <https://mysupport.informatica.com/docs/DOC-11149>

ex.max.conn.per.host

Sets the maximum number of Elasticsearch nodes that you want to connect to the host. Set to the number of Elasticsearch cluster nodes on the host.

ex.max.threads

Sets the maximum number of threads that you want the Apache asynchronous non-blocking receiver to use for each node in the Elasticsearch cluster. Default is 1.

Change the value only when suggested by Informatica Global Customer Support.

search.provisioning.numshards

Optional. Number of shards to create on your Elasticsearch environment. The value depends on the maximum number of shards and the total number of nodes. For example, if the maximum number of shards is 1 and the number of nodes is 3, you can create 3 shards. Default is the total number of Process Servers on which you enable search.

search.provisioning.numreplicas

Optional. Number of copies of the Elasticsearch engine documents that you want to create on different nodes. Use the replication factor to create multiple copies of the documents in the shards of different nodes. You require multiple copies of the documents to achieve high availability if one or more nodes shut down unexpectedly. For example, if the replication factor is 2, you get two copies of the documents in two nodes. For Elasticsearch, the default is 0.

MAX_INITIAL_RESULT_SIZE_TO_CONSIDER

Optional. Manually add the property. Total number of search results to display in the Data Director application. The recommended maximum value is 250. Default is 130. Any value higher than 130 affects the performance of the Data Director application.

mdm.smartsearch.cache.ttl

Optional. Manually add the property. Number of milliseconds for the cached search results of a Search Business Entity web service request to survive before the cached results expire. Default is 60000.

min_rec_for_multithreading

Minimum batch size for the MDM Hub to apply multi-threaded batch operations to a batch job. Applies to the following types of batch job: Automerge, Unmerge, Load, Initially Index Smart Search Data, Stage, Distributed Match, and the tokenization process. Default is 1000.

mq.data.change.monitor.thread.start

In a multinode environment, specifies whether there is message queue polling for individual nodes. To disable message queue polling, set to `false`. Default is `true` on all Java virtual machines where an MDM Hub EAR file is deployed.

ssl.keyStore

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Absolute path and file name of the keystore file.

ssl.keyStore.password

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Plain text password for the keystore file.

ssl.trustStore

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Absolute path and file name of the truststore file.

ssl.trustStore.password

Required if you use the HTTPS port of the application server to configure the Process Server. Manually add the property. Plain text password for the truststore file.

cmx.websphere.security.ssl.config.url

Required if you use the HTTPS port of the application server to configure the Process Server. For WebSphere only. Manually add the property. Absolute path of the `ssl.client.props` file with the file name.

cmx.outbound.bypass.multixref.insystem

Optional. Must be set manually. Set to `true` to bypass the creation of messages on the Process Server when a batch job updates a base object with multiple cross-reference records. Default is `false`.

cmx.server.stage.sqlldr.charset

Optional. If you upload data by using SQL*Loader and the uploaded data is corrupted, set this property to the character set that matches your data, for example, AL32UTF8. When you run a stage job, the stage job generates a control file for SQL*Loader with the specified character set. You can then reload the data. Default is `UTF8`.

cmx.server.stripDML.blockSize

Number of records that the MDM Hub processes in each block. Default is 100.

cmx.server.stripDML.noOfThreadsForDelete

Number of threads that the MDM Hub uses to delete records from the match key tables. Default is 30.

cmx.server.stripDML.noOfThreadsForInsert

Number of threads that the MDM Hub uses to insert records into the match key tables. Default is 50.

cmx.server.stripDML.noOfThreadsForUpdate

Number of threads that the MDM Hub uses to update records in the match key tables. Default is 30.

cmx.server.stripDML.useDeleteInsertLock

Optional. Must be set manually. Set to `true` to ensure MDM Hub runs tokenization during a match job or a tokenization API call on a base object with a large number of records. Default is `false`.

cmx.server.stripDML.useUpdate

Optional. Must be set manually. For IBM DB2 only. Set to `true` to improve performance in IBM DB2 environments during retokenization. Default is `false`.

com.siperian.common.xml.sdo.sdo_resolver_pool_size

Optional. Must be added manually. Specifies the number of thread pools to use for SDO generation. The value can range from 1 to 20. If the provided value is outside the range, the default value is used. Default is 3.

Sample Process Server Properties File

The Process Server properties file is called `cmxcleanse.properties`.

The following example shows the contents of a typical `cmxcleanse.properties` file:

```
# Cleanse properties
#
cmx.server.datalayer.cleanse.working_files.location=C:/infamdm/Hub_971_DB2/cleanse/tmp
cmx.server.datalayer.cleanse.working_files=KEEP
cmx.server.datalayer.cleanse.execution=LOCAL

# Server settings
# Installation directory
cmx.home=C:/infamdm/Hub_971_DB2/cleanse
# Application server type: jboss, tomcat, websphere or weblogic
cmx.appserver.type=jboss
cmx.appserver.version=7

# default setting: 8880 for websphere only (this is not being used in jboss and weblogic
cmx.appserver.soap.connector.port=
cmx.websphere.security.enabled=

# Match Properties
cmx.server.match.server_encoding=0

# limit memory usage by managing the number of records per match ranger node
cmx.server.match.max_records_per_ranger_node=3000

# Cleanse Properties

# Informatica Address Verification Properties
cleanse.library.addressDoctor.property.SetConfigFile=C:/infamdm/Hub_971_DB2/cleanse/
resources/AddressDoctor/5/SetConfig.xml
cleanse.library.addressDoctor.property.ParametersFile=C:/infamdm/Hub_971_DB2/cleanse/
resources/AddressDoctor/5/Parameters.xml
cleanse.library.addressDoctor.property.DefaultCorrectionType=PARAMETERS_DEFAULT

# Trillium Director Properties
cleanse.library.trilliumDir.property.config.file.1=
cleanse.library.trilliumDir.property.config.file.2=
cleanse.library.trilliumDir.property.config.file.3=

# Trillium Director 11+ Properties
cleanse.library.trilliumDir11.property.config.file.1=C:/infamdm/Hub_971_DB2/cleanse/
```

```
resources/TrilliumDirector11/samples/director/td11_default_config_Global.txt
cleanse.library.trilliumDir11.property.config.file.2=C:/infamdm/Hub_971_DB2/cleanse/
resources/TrilliumDirector11/samples/director/td11_default_config_US_detail.txt
cleanse.library.trilliumDir11.property.config.file.3=C:/infamdm/Hub_971_DB2/cleanse/
resources/TrilliumDirector11/samples/director/td11_default_config_US_summary.txt

# Group1Software Enterprise Server Properties
cleanse.library.group1EntServer.property.config.file=

# Group1Software CDQ Server Properties
cleanse.library.group1CDQ.property.config.file=

#FirstLogicDirect Properties
cleanse.library.firstLogicDirect.property.config.file=
encryption.plugin.jar=C:\Temp\informatica_dataencryption.jar
```

Operational Reference Store Properties

Properties of the Operational Reference Store databases include database vendor and version, as well as information from the C_REPOS_DB_RELEASE table.

The following list describes the Operational Reference Store properties that appear in the Enterprise Manager tool:

Database Product Name

Name of your database system.

Database Product Version

Database version used.

Environment ID

The environment ID.

TNS Name

TNS name of the Operational Reference Store database.

Connect URL

URL to connect to the Operational Reference Store database.

Database ID

The database ID.

Time difference between systems

Delta-detection value, in seconds, which determines if the incoming data is in the future.

Column length in bytes

For Oracle environments.

Used by SQL*Loader to determine if the database it is loading into is a UTF-8 database.

The default value is 1, which means that the database is UTF-8.

Load template

For IBM DB2 environments. The path on the database server for log files that the db2 load command generates when you are in logging mode.

MTIP regeneration required

If set to `true`, it indicates that the MTIP views will be regenerated before the match/ merge process.

The default value is `false`, which means that views will not be regenerated.

The Regenerate MTIPs button of the Enterprise Manager tool lets you regenerate MTIP views.

Note: You must not regenerate MTIP views when the ORS is in production mode. Switch off production mode setting before trying to regenerate MTIPs from the Enterprise Manager tool.

Global no logging

For Oracle and IBM DB2 environments. This is used when tables are created to enable logging for database recovery. The default value is `true`, which means no logging.

Model Repository Service URL

Connection parameters that the MDM Hub requires to connect to the Model Repository Service. You specify the connection parameters in the Enterprise Manager tool.

APPENDIX B

Viewing Configuration Details

This appendix includes the following topics:

- [Viewing Configuration Details Overview, 641](#)
- [Starting the Enterprise Manager, 641](#)
- [Properties in Enterprise Manager, 642](#)
- [Environment Report, 644](#)
- [Viewing Version History in Enterprise Manager, 644](#)
- [Using Application Server Logs, 644](#)
- [Using Hub Console Log For the Client, 646](#)

Viewing Configuration Details Overview

You can use the Enterprise Manager tool in the Hub Console to configure and view the configuration details of an Informatica MDM Hub implementation.

Use the Enterprise Manager tool to view properties, version histories, and environment reports for the Hub Server, the Process Servers, the ORS databases, and the MDM Hub Master Database. You can also use Enterprise Manager to configure and view the database logs for your ORS databases.

Starting the Enterprise Manager

To start the Enterprise Manager tool in the Hub Console, expand the Configuration workbench, and then click **Enterprise Manager**. You can also click the Enterprise Manager tool quick launch button in the Hub Console toolbar.

The Enterprise Manager tool then appears in the Hub Console.

Properties in Enterprise Manager

You use the Enterprise Manager tool to display the properties for the Hub Server, Process Server, the MDM Hub Master Database, or the ORS database. Before you can choose servers or databases to view, you must first start Enterprise Manager.

Use the Enterprise Manager tool to view the following properties:

Hub Server

When you select the Hub Server tab, Enterprise Manager displays the Hub Server properties. For more information about these properties, refer to the `cmxserver.properties` file.

Process Servers

When you select the Process Servers tab, Enterprise Manager displays a list of the Process Servers.

When you select a specific Process Server, Enterprise Manager displays its properties. For more information about these properties, refer to the `cmxcleanse.properties` file.

Master database

When you select the Master Database tab, Enterprise Manager displays the MDM Hub Master Database properties. The only database properties displayed are the database vendor and version.

ORS databases

When you select the Operational Reference Store Databases tab, Enterprise Manager displays a list of the Operational Reference Store databases. When you select a Operational Reference Store database, Enterprise Manager displays the properties for that Operational Reference Store.

The top panel contains a list of Operational Reference Store databases that are registered with the MDM Hub Master Database. The bottom panel displays the properties and the version history of the Operational Reference Store database that is selected in the top panel. Properties of Operational Reference Store include database vendor and version, as well as information from the `C_REPOS_DB_RELEASE` table. Version history is also kept in the `C_REPOS_DB_VERSION` table.

Note: Enterprise Manager displays only Operational Reference Store databases that are valid for the current version of MDM Hub. If Enterprise Manager cannot obtain database information for an Operational Reference Store database, Enterprise Manager displays a message explaining why the Operational Reference Store database is not included in the list.

C_REPOS_DB_RELEASE Table

The `C_REPOS_DB_RELEASE` table contains the Operational Reference Store properties.

The following list describes the `C_REPOS_DB_RELEASE` properties that Enterprise Manager displays for the Operational Reference Store databases, depending on your preferences:

DEBUG_LEVEL

Debug level for the ORS database. The debug level can be one of the following integer values:

- 100 = error level debugging
- 200 = warning level debugging
- 300 = information level debugging
- 400 = debug level debugging
- 500 = debugging for all levels

ENVIRONMENT_ID

The environment ID.

DEBUG_FILE_PATH

Path to the location of the ORS database debug log.

DEBUG_FILE_NAME

Name of the ORS database debug log.

DEBUG_IND

Flag that indicates whether debug is enabled or not.

0 = debug is not enabled

1 = debug is enabled

DEBUG_LOG_FILE_SIZE

For Oracle environments. Size of database log file (in MB); default is 5.

DEBUG_LOG_FILE_NUMBER

For Oracle environments. Number of log files used for log rolling; default is 5.

TNSNAME

TNS name of the ORS database.

CONNECTION_PORT

Port on which the ORS database listens.

ORACLE_SID

Oracle database ID.

DATABASE_HOST

Host on which the database is installed.

INTER_SYSTEM_TIME_DELTA_SEC

The number of seconds required to make up the difference between the database server system time and the system time of another machine.

COLUMN_LENGTH_IN_BYTES_IND

For Oracle environments. Flag that the SQLLoader uses to determine if the database it is loading into is a UTF-8 database. A default value of 1 means that the database is UTF-8.

LOAD_TEMPLATE

For IBM DB2 environments. The path on the database server for log files that the db2 load command generates when you are in logging mode.

MTIP_REGENERATION_REQUIRED_IND

Flag that indicates that the MTIP views will be regenerated before the match/merge process. The default value of 0 (zero) means that views will not be regenerated.

GLOBAL_NOLOGGING_IND

For Oracle and IBM DB2 environments. This is used when tables are created to enable logging for DB recovery. The default of 1 means no logging.

Environment Report

When you select the Environment Report tab, Enterprise Manager displays a summary of the properties of all the hub component tabs, in addition to any associated error messages. The report lists the properties in the following order:

- Hub Server
- Process Servers
- Master database
- ORS databases

Saving the MDM Hub Environment Report

To save the MDM Hub environment report, use the Enterprise Manager tool in the Hub Console.

1. From the **Configuration** workbench in the Hub Console, select the **Enterprise Manager** tool.
2. From the **Enterprise Manager** tool, select the **Environment Report** tab.
3. Click **Save**.
4. From the **Save Hub Environment Report** dialog box, navigate to the directory where you want to save the environment report.
5. Click **Save**.

Viewing Version History in Enterprise Manager

You use the Enterprise Manager tool to display the version history for the Hub Server, Process Server, the MDM Hub Master Database, or ORS database. Before you can choose servers or databases to view, you must first start Enterprise Manager.

1. In the Enterprise Manager screen, select the tab for the type of information you want to view.
 - Hub Server
 - Process Servers
 - Master database
 - ORS databases

Enterprise Manager displays version history that is specific for your choice.

2. Select the **Version History** tab.

Enterprise Manager displays version information for that specific component. Version history is sorted in descending order of install start time.

Using Application Server Logs

Use the `log4j.xml` file to configure the application server log files for Hub Server or Process Server. You cannot use Enterprise Manager to configure the application server logs.

Application Server Log Levels

The `log4j.xml` configuration file for application server logs includes a set of log levels to debug and retrieve information.

The following table describes the application server log levels:

Name	ORS Metadata Table Value	Description
ALL	500	Logs all associated information.
DEBUG	400	Used for debugging. (default)
INFO	300	Application server log information.
WARNING	200	Warning messages.
ERROR	100	Error messages.

Log File Rolling

When an application server log file reaches the maximum size defined in the `MaxFileSize` parameter in the `log4j.xml` configuration, the Enterprise Manager performs a log rolling procedure to archive the existing log information and to prevent log files from being overridden with new application server information:

1. Define the following application server log configuration settings in the `log4j.xml` file for Hub Server and Process Server located at `<infamdm_install_dir>\hub\server\conf` and `<infamdm_install_dir>\cleanse\server\conf` respectively:
 - **File.** Name of the log file such as `C:\infamdm\hub\server\logs\cmxserver.log`.
On WebLogic and WebSphere, use the following file names as a best practice to gather logs in the same location while making them distinguishable from each other:
Hub Server: `C:\<infamdm_install_dir>\hub\server\logs\cmxserver.log`
Process Server: `C:\<infamdm_install_dir>\hub\server\logs\cmxcleanse.log`
 - **MaxFileSize.** Maximum file size of the application server log file.
 - **MaxBackupIndex.** Maximum number of files.
 - **Threshold.** The threshold logging level, which overrides any logging level that is higher than the threshold level.
2. The log files used for the Hub Server and Process Server appends the various application server messages to the application server log files.
3. When the physical log file size exceeds the `MaxFileSize`, the Hub activates the log rolling procedure:
 - a. The active log file is renamed to `<filename>.hold`.
 - b. For every file named `<filename>.(n)`, the file is renamed to `<filename>.(n+1)`.
 - c. If `n+1` is larger than `MaxBackupIndex`, the file is deleted.
 - d. When rolling over goes beyond the maximum number of application server files, the MDM Hub renames the original log file. For example, the MDM Hub renames `cmxserver.log` to `cmxserver.log.1`, and `cmxserver.log.1` to `cmxserver.log.2`. The Hub then overwrites `cmxserver.log` with the new log information.
 - e. The `<filename>.hold` file is renamed to `<filename>.1`.

Note: The Hub also creates a `log.rolling` file prior to executing a rollover. If your log files are not rolling as expected, check your log file directory and remove the `log.rolling` file so that the log rolling can continue.

Configuring Application Server Logs

You must set the application server settings for debugging.

Application Server logs are maintained at the application server level. The log entries in an application server log file are not ORS-specific, but specific to the application server. The Hub Server log file writes to the `cmxserver.log` file that is specific to the application server. If you use multiple application servers, then each application server has its own configuration and log files. Similarly, the Process Server installed on WebSphere and WebLogic write to the log files on their respective application servers. In the case of MDM Hub installations on JBoss, the Hub Server and Process Servers are on the same JBoss instance and write to the same shared application server log file.

Before you can choose servers or databases to view, you must start Enterprise Manager. To configure application server logs, perform the following steps:

1. Open the `log4j.xml` file for editing.
The `log4j.xml` file is in `<infamdm_install_dir>\hub\server\conf`.
2. Edit the `log4j.xml` file.
 - Change default value to `DEBUG`.
 - Set the logging category names in the `log4j.xml` file to `com.siperian`, `com.informatica`, and `com.delos`.
 - Set the logging category `siperian.performance` to `"OFF"`. You must set this logging category to `"ON"` only when Informatica requires it of you.

The following sample shows the settings for the `log4j.xml` file:

```
<category name="com.delos">
  <priority value="DEBUG"/>
</category>

<category name="com.siperian">
  <priority value="DEBUG"/>
</category>

<category name="com.informatica">
  <priority value="DEBUG"/>
</category>

<category name="siperian.performance" additivity="false">
  <priority value="OFF"/>
  <appender-ref ref="FILE"/>
</category>
```

Using Hub Console Log For the Client

The console log contains messages and errors resulting from console events, and communication between the console and server. The console log is located on the client machine, in the following directory:

```
<Windows_users_home_dir>\Siperian\console.log
```

You can use the `log4j.properties` file located in the same folder to manage logging level, log file size, and the number of previous logs kept. You must set the last line of the `log4j.properties` file as follows:

```
log4j.rootCategory=DEBUG, FileLog, ConsoleLog
```

APPENDIX C

Row-level Locking

This appendix includes the following topics:

- [Row-level Locking Overview, 647](#)
- [About Row-level Locking, 647](#)
- [Configuring Row-level Locking, 648](#)
- [Locking Interactions Between SIF Requests and Batch Processes, 649](#)

Row-level Locking Overview

This appendix describes how to enable and use row-level locking to protect data integrity when asynchronously running batch and API processes (including SIF requests) or API/API processes.

Note: You can skip this section if batch jobs and API calls are *not* run concurrently in your Informatica MDM Hub implementation.

About Row-level Locking

Informatica MDM Hub uses row-level locking to manage data integrity when batch and SIF operations are executing concurrently on records in a base object. Row-level locking:

- enables concurrent updates of base object data (the same base object but different records) from both batch and SIF processes
- eliminates conflicts between online and batch processes
- provides a high degree concurrent access to data, and
- avoids duplication of the hardware necessary for a mirrored environment.

Row-level locking should be enabled if batch / API or API/API processes will be run asynchronously in your Informatica MDM Hub implementation. Row-level locking applies to asynchronous SIF and batch processing. Synchronous batch processing is restricted due to existing application-level locking.

Default Behavior

Row-level locking is disabled by default. While disabled, API processes (including SIF requests) and batch processes cannot be executed asynchronously on the same base object at the same time. If you explicitly

enable row-level locking for an Operational Reference Store, then Informatica MDM Hub uses the row-level locking mechanism to manage concurrent updates for tokenize, match, and merge processes.

Types of Locks

Data management in Informatica MDM Hub involves the following types of locks:

exclusive lock

Prohibits all other jobs (API or batch processes) from processing on the locked base object.

shared lock

Prohibits only certain jobs from running. For example, a batch job could issue a non-exclusive on a base object and, when interoperability is enabled (on), this shared lock would prohibit other batch jobs but would allow API jobs to attempt to process on the base object.

row-level lock

Shared lock that locks the affected base object rows.

Considerations for Using Row-level Locking

When using row-level locking in a Informatica MDM Hub implementation, consider the following issues:

- A batch process can lock individual records for a significant period of time, which can interfere with SIF access over the web.
- There can be small windows of time during which SIF requests will be locked out. However, these should be few and limited to less than ten minutes.
- The Hub Store needs to be sufficiently sized to support the combined resource demand load of executing batch processes and SIF requests concurrently.

Note: Interoperability must be enabled if Batch jobs are being run together. If, you have multiple parents that attempt access to the same child (or parent) records when running different Batch jobs, one job will fail if it attempts to lock records being processed by the other batch and it holds them longer than the batch wait time. The maximum wait time is defined in C_REPOS_TABLE.

Configuring Row-level Locking

This section describes how to configure row-level locking.

Enabling Row-level Locking on an ORS

By default, row-level locking is not enabled. To enable row-level locking for an Operational Reference Store:

1. In the Hub Console, start the Databases tool.
2. Select an Operational Reference Store to configure.
3. Edit the Operational Reference Store properties.
4. Select (check) the **Batch API Lock Interoperability** check box.
5. Save your changes.

Note: Once you enable row-level locking, the Tokenize on Put property cannot be enabled.

Configuring Lock Wait Times

Once enabled, you can use the Schema Manager to configure SIF and batch lock wait times for base objects in the Operational Reference Store. If the wait time is exceeded, Informatica MDM Hub displays an error message.

Locking Interactions Between SIF Requests and Batch Processes

This section describes the locking interactions between SIF requests and batch processes.

Interactions When API Batch Interoperability is Enabled

The following table shows the interactions between the API and batch processes when row-level locking is enabled.

Existing Lock / New Incoming Call	Batch – Exclusive Lock	Batch – Shared Lock	API Row-level Lock
Batch – Exclusive Lock	Immediately display an error message	Immediately display An error message	Wait for Batch_Lock_Wait_Seconds checking the lock existence. Display an error message if the lock is not cleared by the wait time. Called for each table to be locked.
Batch – Shared Lock	Immediately display an error message	Immediately display An error message	Wait for Batch_Lock_Wait_Seconds to apply a row-level lock using FOR UPDATE SELECT. If the table does not manage the lock, display an error message. Called for each table to be locked.
API - Row level lock	Immediately display an error message	Wait for API_Lock_Wait_Seconds to apply a row-level lock using FOR UPDATE SELECT. If table does not manage the lock, display An error message.	Wait for API_Lock_Wait_Seconds to apply a row-level lock using FOR UPDATE SELECT. If the table does not manage the lock, display an error message. Called for each table to be locked.

Note: When executing Approve tasks, custom index on INTERACTION_ID column of XREF table helps improve the performance on the service side, but the trade off is that it has a negative performance impact on large batch processes, particularly Load and Auto Merge.

Interactions When API Batch Interoperability is Disabled

The following table shows the interactions between the API and batch processes when API batch interoperability is disabled. In this scenario, batch processes will issue an exclusive lock, while SIF requests will check for an exclusive lock but will not issue any locks.

Existing Lock / New Incoming Call	Batch	API
Batch – Exclusive Lock	Immediately display an error message	See “Default Behavior” on page 647..
API	Immediately display an error message	See “Default Behavior” on page 647.

APPENDIX D

MDM Hub Logging

This appendix includes the following topics:

- [MDM Hub Logging Overview, 651](#)
- [Configuring the Logging Settings, 652](#)
- [Hub Console Log, 652](#)
- [Hub Server Log, 652](#)
- [Process Server Log, 653](#)
- [Entity 360 Log, 653](#)
- [Provisioning Tool Log, 653](#)

MDM Hub Logging Overview

All messages and errors that occur in the MDM Hub are stored in log files. The MDM Hub creates the log files during an installation or upgrade of the MDM Hub.

The following table describes the log files for the MDM Hub components:

MDM Hub Component	Log File
Hub Console	console.log
Hub Server	cmxserver.log
Process Server	cmxserver.log
Entity 360	entity360view.log
Provisioning Tool	provisioning.log

Configuring the Logging Settings

You can configure the Hub Server for logging. Specify the configuration settings for logging in the `log4j.xml` file.

1. Open `log4j.xml` in the following directory:

`<MDM Hub installation directory>/hub/server/conf`

2. Set the logging level in the `<priority name>` element for the following category names:

- `com.siperian`
- `com.delos`
- `com.informatica`

You can set the logging level to one of the following values:

- `DEBUG`. Most detailed logging.
- `INFO`. Less detailed logging.
- `ERROR`. Least detailed logging.

Default is `INFO`.

3. Set the `Threshold` parameters to `DEBUG`.

Hub Console Log

The MDM Hub creates the `console.log` file in the operating system environment in which you launch the Hub Console. If you launch the Hub Console in a Windows environment, the Hub Console log is created in the `C:\Documents and Settings\<user_home>\siperian` directory. If you launch the Hub Console in an UNIX environment, the Hub Console log is created in the `/<user_home>/siperian` directory.

The Hub Console log contains the log messages from the Hub Console. Any error that occurs when the MDM Hub communicates with the application server or error messages from the application server or console error messages are logged to this log file. By default, the MDM Hub creates `console.log`.

The `console.log` file is a rolling log file. After it reaches 5 MB, the logs are copied to `console.log.1` and resumes. Hub Server does this indefinitely, potentially creating many log files. You must periodically delete the old files or transfer them to another storage area.

Hub Server Log

The Hub Server generates `cmxserver.log`, which is the log file for the application server.

The Hub Server log appears in the following directory:

`<MDM Hub installation directory>/hub/server/logs`

The Hub Server log contains the logging and debugging information from the application server. By default, the Hub Server creates the `cmxserver.log` file. The `cmxserver.log` file is a rolling log file and once it reaches 5 MB, the Hub Server copies it to `cmxserver.log.1` and resumes. Hub Server does this indefinitely, potentially creating many files. You must periodically delete the old files or transfer them to another storage area.

Process Server Log

The Process Server generates `cmxserver.log` for the CLEANSE, TOKENIZATION, and SIMULATION functions.

The Process Server log is the log file for the application server. The `cmxserver.log` file is in the following directory:

```
<MDM Hub installation directory>/hub/cleanse/logs
```

The `cmxserver.log` file contains debugging and error messages for the cleanse process. By default, the Process Server creates the `cmxserver.log` file. The `cmxserver.log` file is a rolling log file and once it reaches the maximum file size, the Process Server copies it to `cmxserver.log.1` and resumes.

By default the maximum size of the Process Server log file is 5 MB. However, you can configure the maximum size in the following directory:

```
<MDM Hub installation directory>/hub/cleanse/conf/log4j.xml
```

Entity 360 Log

The MDM Hub generates `entity360view.log`, which stores all messages, errors, and full stack traces for the Entity 360 framework.

The `entity360view.log` file is in the following directory:

```
<MDM Hub installation directory>/hub/server/logs
```

The `log4j-entity360view.xml` file stores configuration information for the Entity 360 framework, but does not contain configuration information for the Hub Server.

The `log4j-entity360view.xml` file is in the following directory:

```
<MDM Hub installation directory>/hub/server/conf
```

Provisioning Tool Log

The MDM Hub generates `provisioning.log`, which stores configuration log messages for the provisioning tool.

The `provisioning.log` file appears in the following directory:

```
<MDM Hub installation directory>/hub/server/logs
```

The `log4j-provisioning.xml` file stores configuration information for the provisioning tool, but does not contain configuration information for the Hub Server.

The `log4j-provisioning.xml` file is in the following directory:

```
<MDM Hub installation directory>/hub/server/conf
```

APPENDIX E

Table Partitioning

This appendix includes the following topic:

- [Support for Table Partitioning, 654](#)

Support for Table Partitioning

Multidomain MDM supports and is expected to function normally on partitioned tables though it has not been tested by the MDM Engineering team. Informatica expects that table partitioning will improve performance for some real-time access to MDM, but this may also have an impact on match and merge performance with records that cross partition boundaries. Therefore, Informatica recommends that customers test and measure the overall performance impact as part of the normal development process.

APPENDIX F

Collecting MDM Environment Information with the Product Usage Toolkit

This appendix includes the following topics:

- [Collecting MDM Environment Information with the Product Usage Toolkit Overview, 655](#)
- [Enabling MDM Hub Data Collection on the Hub Server, 657](#)
- [Enabling MDM Hub Data Collection on the Process Server, 657](#)
- [Disabling MDM Hub Data Collection on the Hub Server, 658](#)
- [Disabling MDM Hub Data Collection on the Process Server, 658](#)

Collecting MDM Environment Information with the Product Usage Toolkit Overview

You can enable or disable the MDM Hub from sending information about the MDM Hub environment to Informatica. The Product Usage Toolkit can send information about the system components for the Hub Servers, Process Servers, and information about the MDM Hub environment.

The information sent by the Product Usage Toolkit acts as an advanced health check that provides valuable insight into the MDM Hub implementation. Experts at Informatica can tailor best practice recommendations to your environment and can give suggestions to accelerate project implementation.

When you enable data collection, the Product Usage Toolkit sends information to Informatica 10 minutes after you start the application server. Thereafter, the Product Usage Toolkit sends the information every 30 days.

Important: When you install or upgrade the MDM Hub, data collection is enabled by default. After you install or upgrade, you can disable data collection at any time.

System Configuration Information

The following table describes the information about the Hub Server components and Process Server components that is collected when you enable data collection:

Information Collected	Description
Memory	Total physical memory, available physical memory, maximum virtual memory, available virtual memory, and virtual memory in use
CPU	CPU name, maximum clock speed, and load percentage
Environment variables	Name and value of the environment variables
Disk drive	Drive letter ID, drive size, and amount of free space
Operating system information	Operating system name, version, manufacturer, configuration, and build type
Operating system information about patches	List of operating system patches
Services	Name and status of each service
Network configuration	Details about the network configuration
Virtualization	Node ID, version, and serial number

MDM Hub Environment Information

The following table describes the information about the MDM Hub environment that is collected when you enable data collection:

Information Collected	Description
Host type	Type of host
Build number	MDM Hub build number
Application server	Application server properties
Java	Java properties
Javopts	Java options
Version history	Product version, component version, server name, version number, installation start time, installation end time, and installation status
License	License status of each MDM Hub component, license type, license restrictions, issue date, expiry date, and license key
Log4j server	Log4j server properties
MDM Hub Master Database	MDM Hub Master Database properties

Information Collected	Description
Database	Database URL, database version, database type, and database user
Operational Reference Store	Database version, workflow engine name, package information, subject area names, SSA population information, repository table information, Informatica Data Director user count, and system names

Enabling MDM Hub Data Collection on the Hub Server

To enable MDM Hub data collection on the Hub Server, edit the `mdmsupport.properties` file.

1. Navigate to the following location:
 - On UNIX. `<MDM Hub installation directory>/hub/server/support`
 - On Windows. `<MDM Hub installation directory>\hub\server\support`
2. Open the file `mdmsupport.properties` in a text editor.
3. Set `phonehome.send_csm_files_to_informatica` to 1.
4. Save and close the file.
5. Restart the application server.

Enabling MDM Hub Data Collection on the Process Server

To enable MDM Hub data collection on the Process Server, edit the `mdmsupport.properties` file.

1. Navigate to the following location:
 - On UNIX. `<MDM Hub installation directory>/hub/cleanse/support`
 - On Windows. `<MDM Hub installation directory>\hub\cleanse\support`
2. Open the file `mdmsupport.properties` in a text editor.
3. Set `phonehome.send_csm_files_to_informatica` to 1.
4. Save and close the file.
5. Restart the application server.

Disabling MDM Hub Data Collection on the Hub Server

To disable MDM Hub data collection on the Hub Server, edit the `mdmsupport.properties` file.

1. Navigate to the following location:
 - On UNIX. `<MDM Hub installation directory>/hub/server/support`
 - On Windows. `<MDM Hub installation directory>\hub\server\support`
2. Open the file `mdmsupport.properties` in a text editor.
3. Set `phonehome.send_csm_files_to_informatica` to 0.
4. Save and close the file.
5. Restart the application server.

Disabling MDM Hub Data Collection on the Process Server

To disable MDM Hub data collection on the Process Server, edit the `mdmsupport.properties` file.

1. Navigate to the following location:
 - On UNIX. `<MDM Hub installation directory>/hub/cleanse/support`
 - On Windows. `<MDM Hub installation directory>\hub\cleanse\support`
2. Open the file `mdmsupport.properties` in a text editor.
3. Set `phonehome.send_csm_files_to_informatica` to 0.
4. Save and close the file.
5. Restart the application server.

APPENDIX G

Glossary

accept limit

A number that determines the acceptability of a match. The accept limit is defined by Informatica within a population in accordance with its match purpose.

active state (records)

A state associated with a base object or cross reference record. A base object record is active if at least one of its cross reference records is active. A cross reference record contributes to the consolidated base object only if it is active.

Active records participate in MDM Hub processes by default. These are the records that are available to participate in any operation. If records are required to go through an approval process, then these records have been through that process and have been approved.

Admin source system

Default source system. Used for manual trust overrides and data edits from the Data Manager or Merge Manager tools. See [source system on page 683](#).

auditable events

Activities in MDM Hub that can be tracked by the internal audit mechanism. MDM Hub stores audit information in the audit log table C_REPOS_AUDIT.

authentication

Process of verifying the identity of a user to ensure that they are who they claim to be. In Informatica MDM Hub, users are authenticated based on their supplied credentials—user name / password, security payload, or a combination of both. Informatica MDM Hub provides an internal authentication mechanism and also supports user authentication using third-party authentication providers.

authorization

Process of determining whether a user has sufficient privileges to access a requested Informatica MDM Hub resource. In Informatica MDM Hub, resource privileges are allocated to roles. Users and user groups are assigned to roles. A user's resource privileges are determined by the roles to which they are assigned, as well as by the roles assigned to the user group(s) to which the user belongs.

automerger

Process of merging records automatically. For merge-style base objects only. Match rules can result in automatic merging or manual merging. A match rule that instructs Informatica MDM Hub to perform an automerger will combine two or more records of a base object table automatically, without manual intervention.

base object

A table that contains information about an entity that is relevant to your business, such as customer or account.

batch group

A collection of individual batch jobs (for example, Stage, Load, and Match jobs) that can be executed with a single command. Each batch job in a group can be executed sequentially or in parallel to other jobs.

batch job

A sequence of commands processed in batch mode. This means that a sequence of commands is listed in a file, called a batch file, and submitted to run as a single process.

batch mode

Way of interacting with MDM Hub through batch jobs, which can be executed in the Hub Console or using third-party management tools to schedule and execute batch jobs.

best version of the truth (BVT)

A record that has been consolidated with the best cells of data from the source records.

For merge-style base objects, the base object record is the BVT record, and is built by consolidating the most-trustworthy cell values from the corresponding source records.

BI vendor

A company that produces Business Intelligence software products.

bulk merge

See [automerge on page 659](#).

business entity

A nested structure of base objects. Use the Entity 360 framework in Informatica Data Director to view all the information that relates to the root base object of a business entity. Perform a search in Informatica Data Director to find data within a business entity.

business entity service

A business entity service is a set of operations that run MDM Hub code to create, update, delete, and search for base object records in a business entity.

business process

A business process is a workflow that achieves an organizational goal and implements a business function. A business process contains the activities that are required to achieve the goal and defines paths of execution through the activities. Multidomain MDM ships with predefined Informatica ActiveVOS business processes that are managed by the ActiveVOS Server. The organizational goal of these processes is to ensure that authorized personnel, such as business managers or data stewards, review all updates to master data.

business process management (BPM)

Business process management focuses on adapting an organization's processes. Informatica MDM ships with an embedded business process management engine. Using this engine, you can automate the review-and-approval processes for master data.

BVT

See [best version of the truth \(BVT\) on page 660](#).

cascade delete

An operation that deletes associated records in a child object when a record in the parent object is deleted. To enable a cascade delete operation, set the CASCADE_DELETE_IND parameter to 1.

cascade unmerge

Feature that specifies what happens if records in the parent base object are unmerged. If the feature is enabled, when records in a parent object are unmerged, the MDM Hub also unmerges affected records in the child base object.

cell

Intersection of a column and a record in a table. A cell contains a data value or null.

change list

List of changes to make to a target repository. A *change* is an operation in the change list—such as adding a base object or updating properties in a match rule—that is executed against the target repository. Change lists represent the list of differences between Hub repositories.

cleanse

See [data cleansing on page 663](#).

cleanse engine

A cleanse engine is a third party product used to perform data cleansing with the Informatica MDM Hub.

cleanse function

Code changes the incoming data during Stage jobs, converting each input string to an output string. Typically, these functions are used to standardize data and thereby optimize the match process. By combining multiple cleanse functions, you can perform complex filtering and standardization.

cleanse list

A logical grouping of rules for replacing parts of an input string during the cleanse process.

column

In a table, a set of data values of a particular type, one for each row of the table. See [system column on page 684](#), [user-defined column on page 686](#).

comparison change list

A change list that is the result of comparing the contents of two repositories and generating the list of changes to make to the target repository. Comparison change lists are used in Repository Manager when promoting or importing design objects.

complete match tracking

The display of the complete or original match chain that caused two records to be matched through intermediate records.

conditional mapping

A mapping between a column in a landing table and a staging table that uses a SQL WHERE clause to conditionally select only those records in the landing table that meet the filter condition.

Configuration workbench

Includes tools for configuring a variety of MDM Hub objects, including the Operational Reference Store, users, security, message queues, and metadata validation.

consolidated record

See [master record on page 672](#).

consolidation indicator

Represents the consolidation state of a record in a base object. Stored in the CONSOLIDATION_IND column.

consolidation process

Process of merging or linking duplicate records into a single record. The goal in Informatica MDM Hub is to identify and eliminate all duplicate data and to merge or link them together into a single, consolidated record while maintaining full traceability.

constraint table

A database table where a unique or foreign key constraint is defined.

content metadata

Data that describes the business data that has been processed by Informatica MDM Hub. Content metadata is stored in support tables for a base object, including cross-reference tables, history tables, and others. Content metadata is used to help determine where the data in the base object came from, and how the data changed over time.

contiguous effective period

The effective period of a record version that is contiguous with the effective period of other record versions such that there is no break in the effective period of the record.

control table

A type of system table in an Operational Reference Store that Informatica MDM Hub automatically creates for a base object. Control tables are used in support of the load, merge, and unmerge processes. For each trust-enabled column in a base object, Informatica MDM Hub maintains a record (the last update date and an identifier of the source system) in a corresponding control table.

creation change list

A change list that is the result of exporting the contents of a repository. Creation change lists are used in Repository Manager for importing design objects.

cross-reference table

A type of system table in an Operational Reference Store that Informatica MDM Hub automatically creates for a base object. For each record of the base object, the cross-reference table contains zero to n (0-n) records per source system. This record contains the primary key from the source system and the most recent value that the source system has provided for each cell in the base object table.

Data Access Services

These application server level capabilities enable Informatica MDM Hub to support multiple modes of data access and expose numerous Informatica MDM Hub data services via the InformaticaServices Integration Framework (SIF). This facilitates both real-time synchronous integration, as well as asynchronous integration.

database

Organized collection of data in the Hub Store. Informatica MDM Hub supports two types of databases: a Master Database and an Operational Reference Store (ORS).

data change event

A change to data that is effective for a time period.

data cleansing

The process of standardizing data content and layout, decomposing and parsing text values into identifiable elements, verifying identifiable values (such as zip codes) against data libraries, and replacing incorrect values with correct values from data libraries.

Data Integration Service

An application service that performs data integration jobs for Informatica Developer. Data integration jobs include previewing data and running mappings.

data integration staging

The process of reading data directly from a source system to cleanse the data by using the Data Quality transformations, and moving the cleansed data into the corresponding staging table in the MDM Hub.

Data Manager

Tool used to review the results of all merges—including automatic merges—and to correct data content if necessary. It provides you with a view of the data lineage for each base object record. The Data Manager also allows you to unmerge previously merged records, and to view different types of history on each consolidated record.

Use the Data Manager tool to search for records, view their cross-references, unmerge records, unlink records, view history records, create new records, edit records, and override trust settings. The Data Manager displays all records that meet the search criteria you define.

datasource

In the application server environment, a datasource is a JDBC resource that identifies information about a database, such as the location of the database server, the database name, the database user ID and password, and so on. Informatica MDM Hub needs this information to communicate with an Operational Reference Store.

data steward

Informatica MDM Hub user who has the primary responsibility for data quality. Data stewards access Informatica MDM Hub through the Hub Console, and use Informatica MDM Hub tools to configure the objects in the Hub Store.

Data Steward workbench

Part of the Informatica MDM Hub UI used to review consolidated data as well as matched data queued for exception handling by data analysts or stewards who understand the data semantics and are guardians of data reliability in an organization.

Includes tools for using the Data Manager, Merge Manager, and Hierarchy Manager.

data type

Defines the characteristics of permitted values in a table column—characters, numbers, dates, binary data, and so on. Informatica MDM Hub uses a common set of data types for columns that map directly data types for the database platform used in your Informatica MDM Hub implementation.

decay curve

Visually shows the way that trust decays over time. Its shape is determined by the configured decay type and decay period.

decay period

The amount of time (days, weeks, months, quarters, and years) that it takes for the trust level to decay from the maximum trust level to the minimum trust level.

decay type

The way that the trust level decreases during the decay period.

deleted state (records)

Deleted records are records that are no longer desired to be part of the Hub's data. These records are not used in process (unless specifically requested). Records can only be deleted explicitly and once deleted can be restored if desired. When a record that is Pending is deleted, it is permanently deleted and cannot be restored.

delta detection

During the stage process, the MDM Hub processes new or changed records when this feature is enabled. Delta detection can be done either by comparing entire records or through a date column.

design object

Parts of the metadata used to define the schema and other configuration settings for an implementation. Design objects include instances of the following types of Informatica MDM Hub objects: base objects and columns, landing and staging tables, columns, indexes, relationships, mappings, cleanse functions, queries

and packages, trust settings, validation and match rules, Security Access Manager definitions, Hierarchy Manager definitions, and other settings.

distinct mapping

A mapping between a column in a landing table and a staging table that selects only the distinct records from the landing table. Using distinct mapping is useful in situations in which you have a single landing table feeding multiple staging tables and the landing table is denormalized. For example, a landing table might contain both customer and address data.

distinct source system

A source system that provides data that gets inserted into the base object without being consolidated.

distribution

Process of distributing the master record data to other applications or databases after the best version of the truth has been established via reconciliation.

downgrade

Operation that occurs when inserting or updating data using the load process or using cleansePut & Put APIs when a validation rule reduces the trust for a record by a percentage.

duplicate

One or more records in which the data in certain columns (such as name, address, or organization data) is identical or nearly identical. Match rules executed during the match process determine whether two records are sufficiently similar to be considered duplicates for consolidation purposes.

Dynamic Data Masking

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

effective period

The period of time for which a record is effective. The effective period is defined by a start date and an end date.

entity

In Hierarchy Manager, an entity is a typed object that can be related to other entities. Examples of entities include individual, organization, product, and household.

entity base object

An entity base object is a base object used to store information about Hierarchy Manager entities.

entity type

In Hierarchy Manager, entity types define the kinds of objects that can be related using Hierarchy Manager. Examples are individual, organization, product, and household. All entities with the same entity type are stored in the same entity base object. In the HM Configuration tool, entity types are displayed in the navigation tree under the Entity Object with which the Type is associated.

exact match

A match / search strategy that matches only records that are identical. If you specify an exact match, you can define only exact match columns for this base object (exact-match base objects cannot have fuzzy match columns). A base object that uses the exact match / search strategy is called an exact-match base object.

exclusive lock

In the Hub Console, a lock that is required in order to make exclusive changes to the underlying schema. An exclusive lock prevents all other Hub Console users from making changes to the target database at the same time. An exclusive lock must be released by the user with the exclusive lock; it cannot be cleared by another user.

execution path

The sequence in which batch jobs are executed when the entire batch group is executed in the Informatica MDM Hub. The execution path begins with the Start node and ends with the End node. The Batch Group tool does not validate the execution sequence for you—it is up to you to ensure that the execution sequence is correct.

export process

In Repository Manager, the process of exporting metadata in a repository to a portable change list XML file, which can then be used to import design objects into another repository or to save it in a source control system for archival purposes. The export process copies all supported design objects to the change list XML file.

external application user

MDM Hub user who accesses MDM Hub data indirectly through third-party applications.

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

external cleanse

The process of cleansing data prior to populating the landing tables. External cleansing is typically performed outside of Informatica MDM Hub using an extract-transform-load (ETL) tool or some other data cleansing utility.

external match

Process that allows you to match new data (stored in a separate input table) with existing data in a fuzzy-match base object, test for matches, and inspect the results—all without actually changing data in the base object in any way, or changing the match table associated with the base object.

extract-transform-load (ETL) tool

A software tool (external to Informatica MDM Hub) that extracts data from a source system, transforms the data (using rules, lookup tables, and other functionality) to convert it to the desired state, and then loads (writes) the data to a target database. For Informatica MDM Hub implementations, ETL tools are used to extract data from source systems and populate the landing tables.

farming deployment

A type of deployment to deploy applications into a JBoss cluster. You deploy the application archive file in the farm directory of any of the cluster member and the application is automatically duplicated across all nodes in the same cluster.

foreign key

In a relational database, a column (or set of columns) whose value corresponds to a primary key value in another table (or, in rare cases, the same table). The foreign key acts as a pointer to the other table. For example, the Department_Number column in the Employee table would be a foreign key that points to the primary key of the Department table.

function

See [cleanse function on page 661](#).

fuzzy match

A match / search strategy that uses probabilistic matching, which takes into account spelling variations, possible misspellings, and other differences that can make matching records non-identical. If selected, Informatica MDM Hub adds a special column (Fuzzy Match Key) to the base object. A base object that uses the fuzzy match / search strategy is called a fuzzy-match base object. Using fuzzy match requires a selected population.

fuzzy match key

Special column in the base object that the Schema Manager adds if a match column uses the fuzzy match / search strategy. This column is the primary field used during searching and matching to generate match candidates for this base object. All fuzzy base objects have one and only one Fuzzy Match Key.

geocode

An address of a location based on geographic coordinates such as latitude, longitude, and optionally elevation. You need geocodes to perform a proximity search.

global business identifier (GBID)

A column that contains common identifiers (key values) that allow you to uniquely and globally identify a record based on your business needs. Examples include:

- identifiers defined by applications external to Informatica MDM Hub, such as ERP or CRM systems.
- Identifiers defined by external organizations, such as industry-specific codes (AMA numbers, DEA numbers, and so on), or government-issued identifiers (social security number, tax ID number, driver's license number, and so on).

hard delete

A base object or cross-reference record is physically removed from the database.

Hierarchies Tool

Informatica MDM Hub administrators use the design-time Hierarchies tool (was previously the "Hierarchy Manager Configuration Tool") to set up the structures required to view and manipulate data relationships in Hierarchy Manager. Use the Hierarchies tool to define Hierarchy Manager components—such as entity types, hierarchies, relationships types, packages, and profiles—for your Informatica MDM Hub implementation. The Hierarchies tool is accessible via the Model workbench.

hierarchy

In Hierarchy Manager, a set of relationship types. These relationship types are not ranked based on the place of the entities of the hierarchy, nor are they necessarily related to each other. They are merely relationship types that are grouped together for ease of classification and identification.

Hierarchy Manager

The Hierarchy Manager allows users to manage hierarchy data that is associated with the records managed in the MDM Hub. For more information, see the *Multidomain MDM Configuration Guide*.

hierarchy type

In Hierarchy Manager, a logical classification of hierarchies. The hierarchy type is the general class of hierarchy under which a particular relationship falls. See [hierarchy on page 667](#).

history table

A type of table in an Operational Reference Store that contains historical information about changes to an associated table. History tables provide detailed change-tracking options, including merge and unmerge history, history of the pre-cleansed data, history of the base object, and history of the cross-reference.

HM package

A Hierarchy Manager package represents a subset of an MDM package and contains the metadata needed by Hierarchy Manager.

hotspot

In business data, a group of records representing overmatched data—a large intersection of matches.

Hub Console

Informatica MDM Hub user interface that comprises a set of tools for administrators and data stewards. Each tool allows users to perform a specific action, or a set of related actions, such as building the data model, running batch jobs, configuring the data flow, running batch jobs, configuring external application access to Informatica MDM Hub resources, and other system configuration and operation tasks.

hub object

A generic term for various types of objects defined in the Hub that contain information about your business entities. Some examples include: base objects, cross reference tables, and any object in the hub that you can associate with reporting metrics.

Hub Server

A run-time component in the middle tier (application server) used for core and common services, including access, security, and session management.

Hub Store

In a Informatica MDM Hub implementation, the database that contains the Master Database and one or more Operational Reference Store (ORS) database.

immutable source

A data source that always provides the best, final version of the truth for a base object. Records from an immutable source will be accepted as unique and, once a record from that source has been fully consolidated, it will not be changed—even in the event of a merge. Immutable sources are also distinct systems. For all source records from an immutable source system, the consolidation indicator for Load and PUT is always 1 (consolidated record).

implementer

Informatica MDM Hub user who has the primary responsibility for designing, developing, testing, and deploying Informatica MDM Hub according to the requirements of an organization. Tasks include (but are not limited to) creating design objects, building the schema, defining match rules, performance tuning, and other activities.

import process

In Repository Manager, the process of adding design objects from a library or change list to a repository. The design object does not already exist in the target repository.

incremental load

Any load process that occurs after a base object has undergone its initial data load. Called incremental loading because only new or updated data is loaded into the base object. Duplicate data is ignored.

indirect match

See [transitive match on page 685](#).

initial data load

The very first time that you data is loaded into an empty base object. During the initial data load, all records in the staging table are inserted into the base object as new records.

internal cleanse

The process of cleansing data during the stage process, when data is copied from landing tables to the appropriate staging tables. Internal cleansing occurs inside Informatica MDM Hub using configured cleanse functions that are executed by the Process Server in conjunction with a supported cleanse engine.

job execution log

In the Batch Viewer and Batch Group tools, a log that shows job completion status with any associated messages, such as success, failure, or warning.

key match job

An Informatica MDM Hub batch job that matches records from two or more sources when these sources use the same primary key. Key Match jobs compare new records to each other and to existing records, and then identify potential matches based on the comparison of source record keys as defined by the primary key match rules. Run a Key Match batch job after the primary key match rules are defined.

key type

Identifies important characteristics about the match key to help Informatica MDM Hub generate keys correctly and conduct better searches. Informatica MDM Hub provides the following match key types: Person_Name, Organization_Name, and Address_Part1.

key width

During match, determines how fast searches are during match, the number of possible match candidates returned, and how much disk space the keys consume. Key width options are Standard, Extended, Limited, and Preferred. Key widths apply to fuzzy match objects only.

landing table

A table where a source system puts data that will be processed by Informatica MDM Hub.

land process

Process of populating landing tables from a source system.

lineage

Which systems, and which records from those systems, contributed to consolidated records in the Hub Store.

linear decay

The trust level decreases in a straight line from the maximum trust to the minimum trust.

linear unmerge

A base object record is unmerged and taken out of the existing merge tree structure. Only the unmerged base object record itself will come out the merge tree structure, and all base object records below it in the merge tree will stay in the original merge tree.

load insert

When records are inserted into the target base object. During the load process, if a record in the staging table does not already exist in the target table, then Informatica MDM Hub inserts the record into the target table.

load process

Process of loading data from a staging table into the corresponding base object in the Hub Store. If the new data overlaps with existing data in the Hub Store, Informatica MDM Hub uses trust settings and validation rules to determine which value is more reliable. See [trust on page 686](#), [validation rule on page 687](#), [load insert on page 670](#), [load update on page 670](#).

load update

When records are inserted into the target base object. During the load process, if a record in the staging table does not already exist in the target table, then Informatica MDM Hub inserts the record into the target table.

lock

See [write lock on page 687](#), [exclusive lock on page 666](#).

logical data object

An object that describes a logical entity in an organization. It has attributes and keys, and it describes relationships between attributes.

logical data object mapping

A mapping that links a logical data object to one or more physical data objects. It can include transformation logic.

logical data object model

A data model that describes data in an organization and the relationship between the data. It contains logical data objects and defines relationships between them.

logical data object read mapping

A mapping that provides a view of data through a logical data object. It contains one or more physical data objects as sources and a logical data object as the mapping output.

logical data object write mapping

A mapping that writes data to targets using a logical data object as input. It contains one or more logical data objects as input and a physical data object as the target.

lookup

Process of retrieving a data value from a parent table during Load jobs. In the MDM Hub, when you configure a staging table associated with a base object, if a foreign key column in the staging table (as the child table) is related to the primary key in a parent table, you can configure a lookup to retrieve data from that parent table.

manual merge

Process of merging records manually. Match rules can result in automatic merging or manual merging. A match rule that instructs Informatica MDM Hub to perform a manual merge identifies records that have enough points of similarity to warrant attention from a data steward, but not enough points of similarity to allow the system to automatically merge the records.

manual unmerge

Process of unmerging records manually.

mapping

Defines a set of transformations that are applied to source data. Mappings are used during the stage process (or using the SiperianClient CleansePut API request) to transfer data from a landing table to a staging table. A mapping identifies the source column in the landing table and the target column to populate in the staging table, along with any intermediate cleanse functions used to clean the data. See [conditional mapping on page 662](#), [distinct mapping on page 665](#).

mapplet

A set of transformations that you build in the Mapplet Designer. Create a mapplet when you want to reuse the logic in multiple mappings.

master data

A collection of common, core entities—along with their attributes and their values—that are considered critical to a company's business, and that are required for use in two or more systems or business processes. Examples of master data include customer, product, employee, supplier, and location data.

Master Database

Database that contains the Informatica MDM Hub environment configuration settings—user accounts, security configuration, ORS registry, message queue settings, and so on. A given Informatica MDM Hub environment can have only one Master Database. The default name of the Master Database is CMX_SYSTEM. See also [Operational Reference Store \(ORS\) on page 676](#).

Master Data Management

The controlled process by which the master data is created and maintained as the system of record for the enterprise. MDM is implemented in order to ensure that the master data is validated as correct, consistent, and complete, and—optionally—circulated in context for consumption by internal or external business processes, applications, or users.

master record

Single record in the base object that represents the “best version of the truth” for a given entity (such as a specific organization or person). The master record represents the fully-consolidated data for the entity.

match

The process of determining whether two records should be automatically merged or should be candidates for manual merge because the two records have identical or similar values in the specified columns.

match candidate

For fuzzy-match base objects only, any record in the base object that is a possible match.

match column

A column that is used in a match rule for comparison purposes. Each match column is based on one or more columns from the base object.

match column rule

Match rule that is used to match records based on the values in columns you have defined as match columns, such as last name, first name, address1, and address2.

match key

Encoded strings that represent the data in the fuzzy match key column of the base object. Match keys consist of fixed-length, compressed, and encoded values built from a combination of the words and numbers in a name or address such that relevant variations have the same match key value. Match keys are one part of the match tokens that are generated during the tokenize process, stored in the match key table, and then used during the match process to identify candidates for matching.

match key table

System table that stores the match tokens (match keys + unencoded, raw data) that are generated during the tokenize process. This data is used during the match process to identify candidates for matching, comparing the match keys according to the match rules that have been defined to determine which records are duplicates.

match list

Define custom-built standardization lists. Functions are pre-defined functions that provide access to specialized cleansing functionality such as address verification or address decomposition.

Match Path

Allows you to traverse the hierarchy between records—whether that hierarchy exists between base objects (*inter-table paths*) or within a single base object (*intra-table paths*). Match paths are used for configuring match column rules involving related records in either separate tables or in the same table.

match process

Process of comparing two records for points of similarity. If sufficient points of similarity are found to indicate that two records probably are duplicates of each other, Informatica MDM Hub flags those records for merging.

match purpose

For fuzzy-match base objects, defines the primary goal behind a match rule. For example, if you're trying to identify matches for people where address is an important part of determining whether two records are for the same person, then you would use the Match Purpose called Resident. Each match purpose contains knowledge about how best to compare two records to achieve the purpose of the match. Informatica MDM Hub uses the selected match purpose as a basis for applying the match rules to determine matched records. The behavior of the rules is dependent on the selected purpose.

match rule

Defines the criteria by which Informatica MDM Hub determines whether records might be duplicates. Match columns are combined into match rules to determine the conditions under which two records are regarded as being similar enough to merge. Each match rule tells Informatica MDM Hub the combination of match columns it needs to examine for points of similarity.

match rule set

A logical collection of match rules that allow users to execute different sets of rules at different stages in the match process. Match rule sets include a search level that dictates the search strategy, any number of automatic and manual match rules, and optionally, a filter that allows you to selectively include or exclude records during the match process. Match rules sets are used to execute to match column rules but not primary key match rules.

match search strategy

Specifies the reliability of the match versus the performance you require: fuzzy or exact. An exact match / search strategy is faster, but an exact match will miss some matches if the data is imperfect. See [fuzzy match on page 667](#), [exact match on page 666](#)., [match process on page 673](#).

match search strategy

Specifies the reliability of the match versus the performance you require: fuzzy or exact. An exact match / search strategy is faster, but an exact match will miss some matches if the data is imperfect. See [fuzzy match on page 667](#), [exact match on page 666](#)., [match process on page 673](#).

match subtype

Used with base objects that containing different types of data, such as an Organization base object containing customer, vendor, and partner records. Using match subtyping, you can apply match rules to specific types of data within the same base object. For each match rule, you specify an exact match column that will serve as the "subtyping" column to filter out the records that you want to ignore for that match rule.

match table

Type of system table, associated with a base object, that supports the match process. During the execution of a Match job for a base object, Informatica MDM Hub populates its associated match table with the ROWID_OBJECT values for each pair of matched records, as well as the identifier for the match rule that resulted in the match, and an automerger indicator.

match token

Strings that represent both encoded (match key) and unencoded (raw) values in the match columns of the base object. Match tokens are generated during the tokenize process, stored in the match key table, and then used during the match process to identify candidates for matching.

match type

Each match column has a match type that determines how the match column will be tokenized in preparation for the match comparison.

maximum trust

The trust level that a data value will have if it has just been changed. For example, if source system A changes a phone number field from 555-1234 to 555-4321, the new value will be given system A's maximum trust level for the phone number field. By setting the maximum trust level relatively high, you can ensure that changes in the source systems will usually be applied to the base object.

Merge Manager

Tool used to review and take action on the records that are queued for manual merging.

merge process

Process of combining two or more records of a base object table because they have the same value (or very similar values) in the specified match columns. See [consolidation process on page 662](#), [automerge on page 659](#), [manual merge on page 671](#).

merge-style base object

Type of base object that is used with Informatica MDM Hub's match and merge capabilities.

message

In Informatica MDM Hub, refers to a Java Message Service (JMS) message. A message queue server handles two types of JMS messages:

- inbound messages are used for the asynchronous processing of Informatica MDM Hub service invocations
- outbound messages provide a communication channel to distribute data changes via JMS to source systems or other systems.

message queue

A mechanism for transmitting data from one process to another (for example, from Informatica MDM Hub to an external application).

message queue rule

A mechanism for identifying base object events and transferring the effected records to the internal system for update. Message queue rules are supported for updates, merges, and records accepted as unique.

message queue server

In Informatica MDM Hub, a Java Message Service (JMS) server, defined in your application server environment, that Informatica MDM Hub uses to manage incoming and outgoing JMS messages.

message trigger

A rule that gets fired when which a particular action occurs within Informatica MDM Hub. When an action occurs for which a rule is defined, a JMS message is placed in the outbound message queue. A message trigger identifies the conditions which cause the message to be generated (what action on which object) and the queue on which messages are placed.

metadata

Data that is used to describe other data. In Informatica MDM Hub, metadata is used to describe the schema (data model) that is used in your Informatica MDM Hub implementation, along with related configuration settings.

metadata validation

See [validation process on page 687](#).

minimum trust

The trust level that a data value will have when it is “old” (after the decay period has elapsed). This value must be less than or equal to the maximum trust. If the maximum and minimum trust are equal, the decay curve is a flat line and the decay period and decay type have no effect. See also [decay period on page 664](#).

Model repository

A relational database that stores the metadata for projects and folders that you can access through the Developer tool.

Model Repository Service

An application service in the Informatica domain that runs and manages the Model repository. The Model repository stores metadata created by Informatica products in a relational database to enable collaboration among the products.

Model workbench

Part of the Informatica MDM Hub UI used to configure the solution during deployment by the implementers, and for on-going configuration by data architects of the various types of metadata and rules in response to changing business needs.

Includes tools for creating query groups, defining packages and other schema objects, and viewing the current schema.

noncontiguous effective period

The effective period of a record version that is not contiguous with the effective period of other record versions such that there are breaks in the effective period of the record.

non-contributing cross reference

A cross-reference (XREF) record that does not contribute to the BVT (best version of the truth) of the base object record. As a consequence, the values in the cross-reference record will never show up in the base object record. Note that this is for state-enabled records only.

non-equal matching

When configuring match rules, prevents equal values in a column from matching each other. Non-equal matching applies only to exact match columns.

operation

Deprecated term. See [request on page 680](#).

Operational Reference Store (ORS)

A database that contains master data and the rules that act on the master data. Rules include the rules for processing the master data, the rules for managing the set of master data objects, and the processing rules and auxiliary logic that the MDM Hub uses to define the best version of the truth. An MDM Hub configuration can have one or more Operational Reference Stores. The default name of an ORS is CMX_ORS.

overmatching

For fuzzy-match base objects only, a match that results in too many matches, including matches that are not relevant. When configuring match, the goal is to find the optimal number of matches for your data.

package

A *package* is a public view of one or more underlying tables in Informatica MDM Hub. Packages represent subsets of the columns in those tables, along with any other tables that are joined to the tables. A package is based on a query. The underlying query can select a subset of records from the table or from another package.

password policy

Specifies password characteristics for Informatica MDM Hub user accounts, such as the password length, expiration, login settings, password re-use, and other requirements. You can define a global password policy for all user accounts in a Informatica MDM Hub implementation, and you can override these settings for individual users.

path

See [Match Path on page 672](#).

pending state (records)

Pending records are records that have not yet been approved for general usage in the Hub. These records can have most operations performed on them, but operations have to specifically request Pending records. If records are required to go through an approval process, then these records have not yet been approved and are in the midst of an approval process.

period end date

The time when the effective period of a record version ends.

period start date

The time when the effective period of a record version starts.

physical data object

A physical representation of data that is used to read from, look up, or write to resources.

policy decision points (PDPs)

Specific security check points that authenticate user identity and authorize user access to MDM Hub resources.

policy enforcement points (PEPs)

Specific security check points that enforce, at run time, security policies for authentication and authorization requests.

population

Defines certain characteristics about data in the records that you are matching. By default, Informatica MDM Hub comes with the US population, but Informatica provides a standard population per country. Populations account for the inevitable variations and errors that are likely to exist in name, address, and other identification data; specify how Informatica MDM Hub builds match tokens; and specify how search strategies and match purposes operate on the population of data to be matched. Used only with the Fuzzy match/search strategy.

primary key

In a relational database table, a column (or set of columns) whose value uniquely identifies a record. For example, the Department_Number column would be the primary key of the Department table.

primary key match rule

Match rule that is used to match records from two systems that use the same primary keys for records. See also [match column rule on page 672](#).

private resource

A Informatica MDM Hub resource that is hidden from the Roles tool, preventing its access through Services Integration Framework (SIF) operations. When you add a new resource in Hub Console (such as a new base object), it is designated a PRIVATE resource by default.

privilege

Permission to access an MDM Hub resource. With MDM Hub internal authorization, each role is assigned one of the following privileges.

Privilege	Allows the User To....
READ	View data.
CREATE	Create data records in the Hub Store.
UPDATE	Update data records in the Hub Store.
MERGE	Merge and unmerge data.
EXECUTE	Execute cleanse functions and batch groups.
DELETE	Delete data records from the Hub Store.

Privileges determine the access that external application users have to MDM Hub resources. For example, a role might be configured to have READ, CREATE, UPDATE, and MERGE privileges on particular packages and package columns. These privileges are not enforced when using the Hub Console, although the settings still affect the use of Hub Console to some degree.

process

See [business process on page 660](#).

Process Server

A server that performs cleanse, match, and batch jobs. The Process Server is deployed in an application server environment. The Process Server interfaces with cleanse engines, such as Trillium Director to standardize the data. The Process Server is multi-threaded so that each instance can process multiple requests concurrently.

profile

In Hierarchy Manager, describes what fields and records an HM user may display, edit, or add. For example, one profile can allow full read/write access to all entities and relationships, while another profile can be read-only (no add or edit operations allowed).

promotion process

Meaning depends on the context:

- **Repository Manager:** Process of copying changes in design objects from one repository to another. Promotion is used to copy incremental changes between repositories.
- **State Management:** Process of changing the system state of individual records in Informatica MDM Hub (for example from PENDING state to ACTIVE state).

provider

See [security provider on page 682](#).

provider property

A name-value pair that a security provider might require in order to access for the service(s) that they provide.

proximity search

A process that matches records that are within the geocode radius that you specify. Proximity search is performed on base objects for which columns are populated with latitude, longitude, and optionally elevation.

publish

Process of submitting a Informatica MDM Hub message to a message queue for distribution to other applications, databases, and so on.

query

A request to retrieve data from the Hub Store. Informatica MDM Hub allows administrators to specify the criteria used to retrieve that data. Queries can be configured to return selected columns, filter the result set with a WHERE clause, use complex query syntax (such as GROUP BY, ORDER BY, and HAVING clauses), and use aggregate functions (such as SUM, COUNT, and AVG).

query group

A logical group of queries. A query group is simply a mechanism for organizing queries. See [query on page 678](#).

raw table

A table that archives data from a landing table.

real-time mode

Way of interacting with Informatica MDM Hub using third-party applications, which invoke Informatica MDM Hub operations via the Services Integration Framework (SIF) interface. SIF provides operations for various services, such as reading, cleansing, matching, inserting, and updating records. See also [batch mode on page 660](#), [Services Integration Framework \(SIF\) on page 682](#).

reconciliation

For a given entity, Informatica MDM Hub obtains data from one or more source systems, then reconciles “multiple versions of the truth” to arrive at the master record—the best version of the truth—for that entity. Reconciliation can involve cleansing the data beforehand to optimize the process of matching and consolidating records for a base object. See [distribution on page 665](#).

record

A row in a table that represents an instance of an object. For example, in an Address table, a record contains a single address. See also [source record on page 683](#), [consolidated record on page 662](#).

record version

The version of a record that is effective or valid for a specific period of time. A record can have multiple versions, and each version can be effective for a different time period.

referential integrity

Enforcement of parent-child relationship rules among tables based on configured foreign key relationship.

regular expression

A computational expression that is used to match and manipulate text data according to commonly-used syntactic conventions and symbolic patterns. In Informatica MDM Hub, a regular expression function allows you to use regular expressions for cleanse operations. To learn more about regular expressions, including syntax and patterns, refer to the Javadoc for `java.util.regex.Pattern`.

reject table

A table that contains records that Informatica MDM Hub could not insert into a target table, such as:

- staging table (stage process) after performing the specified cleansing on a record of the specified landing table
- Hub store table (load process)

A record could be rejected because the value of a cell is too long, or because the record’s update date is later than the current date.

relationship

In Hierarchy Manager, describes the affiliation between two specific entities. Hierarchy Manager relationships are defined by specifying the relationship type, hierarchy type, attributes of the relationship, and dates for when the relationship is active. See [relationship type on page 680](#), [hierarchy on page 667](#).

relationship base object

A relationship base object is a base object used to store information about Hierarchy Manager relationships.

relationship type

Describes general classes of relationships. The relationship type defines:

- the types of entities that a relationship of this type can include
- the direction of the relationship (if any)
- how the relationship is displayed in the Hub Console

See [relationship on page 679](#), [hierarchy on page 667](#).

repository

An Operational Reference Store (Operational Reference Store). The Operational Reference Store stores metadata about its own schema and related property settings. In Repository Manager, when copying metadata between repositories, there is always a *source repository* that contains the design object to copy, and the *target repository* that is destination for the design object.

Repository Manager

The Repository Manager tool in the Hub Console is used to validate metadata for a repository, promote design objects from one repository to another, import design objects into a repository, and export a repository to a change list. .

request

Informatica MDM Hub request (API) that allows external applications to access specific Informatica MDM Hub functionality using the Services Integration Framework (SIF), a request/response API model.

resource

Any Informatica MDM Hub object that is used in your Informatica MDM Hub implementation. Certain resources can be configured as secure resources: base objects, mappings, packages, remote packages, cleanse functions, HM profiles, the audit table, and the users table. In addition, you can configure secure resources that are accessible by SIF operations, including content metadata, match rule sets, metadata, batch groups, the audit table, and the users table.

resource group

A collection of secure resources that simplify privilege assignment, allowing you to assign privileges to multiple resources at once, such as easily assigning resource groups to a role. See [resource on page 680](#), [privilege on page 677](#).

Resource Kit

The Informatica MDM Hub Resource Kit is a set of utilities, examples, and libraries that provide examples of Informatica MDM Hub functionality that can be expanded on and implemented.

RISL decay

Rapid Initial Slow Later decay puts most of the decrease at the beginning of the decay period. The trust level follows a concave parabolic curve. If a source system has this decay type, a new value from the system will probably be trusted but this value will soon become much more likely to be overridden.

role

Defines a set of privileges to access secure Informatica MDM Hub resources.

row

See [record on page 679](#).

rule

A statement that defines valid behavior or that defines calculations and comparisons. MDM Hub manages rules and applies rules to master data. See also [match rule on page 673](#), [message queue rule on page 674](#), [state transition rules on page 683](#), [timeline rules on page 685](#), and [validation rule on page 687](#).

rule set

See [match rule set on page 673](#).

rule set filtering

Ability to exclude records from being processed by a match rule set. For example, if you had an Organization base object that contained multiple types of organizations (customers, vendors, prospects, partners, and so on), you could define a match rule set that selectively processed only vendors.

schema

The data model that is used in a customer's Informatica MDM Hub implementation. Informatica MDM Hub does not impose or require any particular schema. The schema is independent of the source systems.

Schema Manager

The Schema Manager is a design-time component in the Hub Console used to define the schema, as well as define the staging and landing tables. The Schema Manager is also used to define rules for match and merge, validation, and message queues.

Schema Viewer Tool

The Schema Viewer tool is a design-time component in the Hub Console used to visualize the schema configured for your Informatica MDM Hub implementation. The Schema Viewer is particularly helpful for visualizing a complex schema.

search levels

Defines how stringently Informatica MDM Hub searches for matches: narrow, typical, exhaustive, or extreme. The goal is to find the optimal number of matches for your data—not too few (undermatching), which misses significant matches, or too many (overmatching), which generates too many matches, including insignificant ones.

secure resource

A protected Informatica MDM Hub resource that is exposed to the Roles tool, allowing the resource to be added to roles with specific privileges. When a user account is assigned to a specific role, then that user account is authorized to access the secure resources using SIF according to the privileges associated with that role. In order for external applications to access a Informatica MDM Hub resource using SIF operations, that resource must be configured as SECURE. Because all Informatica MDM Hub resources are PRIVATE by

default, you must explicitly make a resource SECURE after the resource has been added. See also [private resource on page 677](#), [resource on page 680](#).

Status Setting	Description
SECURE	Exposes this Informatica MDM Hub resource to the Roles tool, allowing the resource to be added to roles with specific privileges. When a user account is assigned to a specific role, then that user account is authorized to access the secure resources using SIF requests according to the privileges associated with that role.
PRIVATE	Hides this Informatica MDM Hub resource from the Roles tool. Default. Prevents its access via Services Integration Framework (SIF) operations. When you add a new resource in Hub Console (such as a new base object), it is designated a PRIVATE resource by default.

security

The ability to protect information privacy, confidentiality, and data integrity by guarding against unauthorized access to, or tampering with, data and other resources in your Informatica MDM Hub implementation.

Security Access Manager (SAM)

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

Security Access Manager workbench

Includes tools for managing users, groups, resources, and roles.

security payload

Raw binary data supplied to an MDM Hub operation request that can contain supplemental data required for further authentication or authorization.

security provider

A third-party application that provides security services (authentication, authorization, and user profile services) for users accessing Informatica MDM Hub.

segment matching

Way of limiting match rules to specific subsets of data. For example, you could define different match rules for customers in different countries by using segment matching to limit certain rules to specific country codes. Segment matching is configured on a per-rule basis and applies to both exact-match and fuzzy-match base objects.

Services Integration Framework (SIF)

The part of Informatica MDM Hub that interfaces with client programs. Logically, it serves as a middle tier in the client/server model. It enables you to implement the request/response interactions using any of the following architectural variations:

- Loosely coupled Web services using the SOAP protocol.
- Tightly coupled Java remote procedure calls based on Enterprise JavaBeans (EJBs) or XML.

- Asynchronous Java Message Service (JMS)-based messages.
- XML documents going back and forth via Hypertext Transfer Protocol (HTTP).

SIRL decay

Slow Initial Rapid Later decay puts most of the decrease at the end of the decay period. The trust level follows a convex parabolic curve. If a source system has this decay type, it will be relatively unlikely for any other system to override the value that it sets until the value is near the end of its decay period.

soft delete

A base object or a cross-reference record is marked as deleted in a user attribute or in the HUB_STATE_IND.

source record

A raw record from a source system.

source system

An external system that provides data to Informatica MDM Hub.

stage process

Process of reading the data from the landing table, performing any configured cleansing, and moving the cleansed data into the corresponding staging table. If you enable delta detection, Informatica MDM Hub only processes new or changed records.

staging table

A table where cleansed data is temporarily stored before being loaded into base objects via load jobs.

state-enabled base object

A base object for which state management is enabled.

state management

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

State management is limited to the following states: ACTIVE, PENDING, and DELETED.

state transition rules

Rules that determine whether and when a record can change from one state to another. State transition rules differ for base object and cross-reference records.

stripping

Deprecated term.

strip table

Deprecated term.

surviving cell data

When evaluating cells to merge from two records, Informatica MDM Hub determines which cell data should survive and which one should be discarded. The *surviving cell data* (or *winning cell*) is considered to represent the *better* version of the truth between the two cells. Ultimately, a single, consolidated record contains the best surviving cell data and represents the *best* version of the truth.

survivorship

Determination made by Informatica MDM Hub when evaluating cells to merge from two records. Informatica MDM Hub determines which cell data should survive and which one should be discarded. Survivorship applies to both trust-enabled columns and columns that are not trust enabled. When comparing cells from two different records, Informatica MDM Hub determines survivorship based on properties of the data. For example, if the two columns are trust-enabled, then the cell with the highest trust score wins. If the trust scores are equal, then the cell with the most recent LAST_UPDATE_DATE wins. If the LAST_UPDATE_DATE is equal, Informatica MDM Hub uses other criteria for determining survivorship.

system column

A column in a table that Informatica MDM Hub automatically creates and maintains. System columns contain metadata. Common system columns for a base object include ROWID_OBJECT, CONSOLIDATION_IND, and LAST_UPDATE_DATE.

Systems and Trust Tool

Systems and Trust tool is a design-time tool used to name the source systems that can provide data for consolidation in Informatica MDM Hub. You use this tool to define the trust settings associated with each source system for each trust-enabled column in a base object.

system state

Describes how base object records are supported by Informatica MDM Hub. The following states are supported: ACTIVE, PENDING, and DELETED.

table

In a database, a collection of data that is organized in rows (records) and columns. A table can be seen as a two-dimensional set of values corresponding to an object. The columns of a table represent characteristics of the object, and the rows represent instances of the object. In the Hub Store, the Master Database and each Operational Reference Store (Operational Reference Store) represents a collection of tables. Base objects are stored as tables in an Operational Reference Store.

target database

In the Hub Console, the Master Database or an Operational Reference Store (Operational Reference Store) that is the target of the current tool. Tools that manage data stored in the Master Database, such as the Users tool, require that your target database is the Master Database. Tools that manage data stored in an Operational Reference Store require that you specify which Operational Reference Store to

timeline

The data change events of business entities and their relationships over a period of time. You define data change events through effective periods.

timeline action

The action to perform for entities for which you track data change events. You can perform actions such as add a record, edit a record, and edit effective period.

timeline granularity

The time measurement that you want to use to define effective periods for versions of records. For example, you can choose the effective periods to be in years, months, or seconds.

timeline rules

Pre-defined rules that the MDM Hub enforces to track data change events. Based on the timeline rules, the data change events are captured in record versions and their effective periods.

tokenize process

Specialized form of data standardization that is performed before the match comparisons are done. For the most basic match types, tokenizing simply removes “noise” characters like spaces and punctuation. The more complex match types result in the generation of sophisticated match codes—strings of characters representing the contents of the data to be compared—based on the degree of similarity required.

token table

Deprecated term.

traceability

The maintenance of data so that you can determine which systems—and which records from those systems—contributed to consolidated records.

transactional data

Represents the actions performed by an application, typically captured or generated by an application as part of its normal operation. It is usually maintained by only one system of record, and tends to be accurate and reliable in that context. For example, your bank probably has only one application for managing transactional data resulting from withdrawals, deposits, and transfers made on your checking account.

transitive match

During the Build Match Group (BMG) process, a match that is made *indirectly* due to the behavior of other matches. For example, if record 1 matches to record 2, record 2 matches to record 3, and record 3 matches to record 4, after the BMG process removes redundant matches, it might produce results in which records 2, 3, and 4 match to record 1. In this example, there was no explicit rule that matched record 4 to record 1. Instead, the match was made indirectly.

tree unmerge

Unmerge a tree of merged base object records as an intact sub-structure. A sub-tree having unmerged base object records as root will come out from the original merge tree structure. (For example, merge a1 and a2 into a, then merge b1 and b2 into b, and then finally merge a and b into c. If you then perform a tree unmerge on a, and then unmerge a from a1, a2 is a sub tree and will come out from the original tree c. As a result, a is the root of the tree after the unmerge.)

trust

Mechanism for measuring the confidence factor associated with each cell based on its source system, change history, and other business rules. Trust takes into account the age of data, how much its reliability has decayed over time, and the validity of the data.

trust level

For a source system that provides records to Informatica MDM Hub, a number between 0 and 100 that assigns a level of confidence and reliability to that source system, relative to other source systems. The trust level has meaning only when compared with the trust level of another source system.

trust score

The current level of confidence in a given record. During load jobs, Informatica MDM Hub calculates the trust score for each records. If validation rules are defined for the base object, then the Load job applies these validation rules to the data, which might further downgrade trust scores. During the consolidation process, when two records are candidates for merge or link, the values in the record with the higher trust score wins. Data stewards can manually override trust scores in the Merge Manager tool.

undermatching

For fuzzy-match base objects only, a match that results in too few matches, which misses relevant matches. When configuring match, the goal is to find the optimal number of matches for your data.

unmerge

Process of unmerging previously-merged records. For merge-style base objects only.

user

An individual (person or application) who can access Informatica MDM Hub resources. Users are represented in Informatica MDM Hub by *user accounts*, which are defined in the Master Database.

user-defined column

Any column in a table that is not a system column. User-defined columns are added in the Schema Manager and usually contain business data.

user exit

User exits consists of Java code that runs at specific points in the batch or SIF API processes to extend the functionality of MDM Hub.

Developers can extend Informatica MDM Hub batch processes by adding custom code to the appropriate user exit for pre- and post-batch job processing.

user group

A logical collection of user accounts.

user object

User-defined functions that are registered with the MDM Hub to extend its functionality.

The MDM Hub has the following types of user objects:

User Object	Description
User Exits	Java code that includes a set of fixed, pre-defined parameters. The user exit is configured, on a per-base object basis, to execute at a specific point during a Informatica MDM Hub batch process run.
Custom Java Cleanse Functions	Java cleanse functions that supplement the standard cleanse libraries with customer logic. These functions are basically Jar files and stored as BLOBs in the database.
Custom Button Functions	Custom UI functions that supply additional icons and logic in Data Manager, Merge Manager and Hierarchy Manager.

Utilities workbench

Includes tools for auditing application event, configuring and running batch groups, and generating the SIF APIs.

validation process

Process of verifying the completeness and integrity of the metadata that describes a repository. The validation process compares the logical model of a repository with its physical schema. If any issues arise, the Repository Manager generates a list of issues requiring attention.

validation rule

Rule that tells Informatica MDM Hub the condition under which a data value is not valid. When data meets the criteria specified by the validation rule, the trust value for that data is downgraded by the percentage specified in the validation rule. If the Reserve Minimum Trust flag is set for the column, then the trust cannot be downgraded below the column's minimum trust.

workbench

In the Hub Console, a mechanism for grouping similar tools. A workbench is a logical collection of related tools. For example, the Model workbench contains tools for modelling data such as Schema, Queries, Packages, and Mappings.

workflow

In Informatica Multidomain MDM, a workflow represents a business process within an organization. See [business process on page 660](#).

write lock

In the Hub Console, a lock that is required in order to make changes to the underlying schema. All non-data steward tools (except the Operational Reference Store security tools) are in read-only mode unless you acquire a write lock. Write locks allow multiple, concurrent users to make changes to the schema.

INDEX

A

- Accept Non-Matched Records As Unique jobs [554](#)
- ACTIVE state, about [170](#)
- Add record version
 - example [161](#)
- Address match purpose [409](#)
- Address_Part1 key type [395](#)
- Admin source system
 - about the Admin source system [288](#)
 - renaming [290](#)
- allow null foreign key
 - in staging table column [358](#)
- allow null update
 - on staging table columns [358](#)
- analyzer
 - character filter
 - token filter [471](#)
 - tokenizer [471](#)
- application server logs
 - about [644](#)
 - configuring [646](#)
 - levels [645](#)
 - log file rolling [645](#)
- Apply Null Values
 - column property [109](#)
- AssignTasks user exit
 - interface [590](#)
- audit trails, configuring [309](#)
- Auto Match and Merge jobs
 - metrics [555](#)
- Automerge jobs
 - Auto Match and Merge jobs [555](#)
 - metrics [556](#)
 - trust-enabled columns [555](#)
- AUTOMERGE_IND
 - external match output table column [558](#)

B

- base object properties
 - match process behavior [376](#)
- base object style [102](#)
- base objects
 - adding columns [78](#)
 - converting to entity base objects [201](#)
 - creating [104](#)
 - defining [95](#)
 - deleting [107](#)
 - described [76](#)
 - editing [105](#)
 - entity base objects [198](#)
 - exact match base objects [279](#)
 - fuzzy match base objects [279](#)
 - history table [101](#)

- base objects (*continued*)
 - impact analysis [107](#)
 - load inserts [268](#)
 - load updates [269](#)
 - overview of [95](#)
 - record survivorship, state management [173](#)
 - relationship base objects [383](#)
 - reserved special characters [78](#)
 - reserved suffixes [78](#)
 - reverting from relationship base objects [210](#)
 - style [102](#)
 - system columns [96](#)
- batch groups
 - adding [544](#)
 - deleting [545](#)
 - editing [545](#)
 - executing [548](#)
 - levels, configuring [545](#)
- batch interoperability [649](#)
- batch jobs
 - about [528](#)
 - Accept Non-Matched Records As Unique [554](#)
 - Auto Match and Merge jobs [554](#)
 - automatically-created batch jobs [533](#)
 - Automerge jobs [555](#)
 - Batch Unmerge jobs [556](#)
 - BVT Snapshot jobs [556](#)
 - clearing history [543](#)
 - command buttons [537](#)
 - configurable options [537](#)
 - configuring [528](#)
 - design considerations [532](#)
 - executing [537](#)
 - External Match jobs [557](#)
 - foreign key relationships and [532](#)
 - Generate Match Tokens jobs [560](#)
 - job execution logs [538](#)
 - job execution status [538](#)
 - Key Match jobs [562](#)
 - launching [530](#)
 - Load jobs [562](#)
 - Manual Merge jobs [566](#)
 - Manual Unmerge jobs [567](#)
 - Match Analyze jobs [569](#)
 - Match for Duplicate Data jobs [571](#)
 - Match jobs [567](#)
 - Multi Merge jobs [571](#)
 - multi-threaded batch jobs [529](#)
 - Promote jobs [571](#)
 - properties [536](#)
 - refreshing the status [538](#)
 - rejected records [542](#)
 - Reset Match Table jobs [573](#)
 - Revalidate jobs [573](#)
 - running manually [536](#)
 - selecting [536](#)

- batch jobs (*continued*)
 - sequencing batch jobs [531](#)
 - setting job status to incomplete [538](#)
 - Stage jobs [574](#)
 - status, setting [538](#), [551](#)
 - supporting tables [531](#)
 - Synchronize jobs [367](#), [574](#)
 - when changes occur [534](#)
- Batch Viewer tool
 - about [535](#)
- best version of the truth (BVT)
 - about [284](#)
- BPM workflow tool [174](#)
- build match groups (BMGs) [281](#)
- build_war macro [600](#)
- BVT Snapshot jobs [556](#)

C

- C_REPOS_DB_RELEASE table
 - columns [642](#)
- C_REPOS_EXT_HARD_DEL_DETECT
 - primary key source columns [322](#)
- C_REPOS_SYSTEM table
 - referenced by ROWID_SYSTEM in cross-reference table [99](#)
- cascade unmerge [484](#)
- cell update [358](#)
- child nodes in Navigation pane
 - hiding [36](#)
 - showing [36](#)
- cleanse functions
 - about cleanse functions [338](#)
 - aggregation [301](#)
 - availability of [338](#)
 - Cleanse Functions tool [339](#)
 - cleanse lists [348](#)
 - conditional execution components [347](#)
 - configuration overview [340](#)
 - constants [345](#)
 - decomposition [301](#)
 - function modes [344](#)
 - graph functions [342](#)
 - inputs [345](#)
 - Java cleanse libraries [341](#)
 - libraries [338](#)
 - logging [344](#)
 - mappings [301](#)
 - outputs [346](#)
 - properties of [340](#)
 - secure resources [338](#)
 - testing [347](#)
 - types [339](#)
 - types of [338](#)
 - user cleanse libraries [340](#)
 - workspace buttons [345](#)
- Cleanse Functions tool
 - starting [339](#)
 - workspace buttons [345](#)
- cleanse lists
 - about cleanse lists [348](#)
 - adding [348](#)
 - defaultValue property [349](#)
 - editing [351](#)
 - exact match [351](#)
 - Input string property [349](#)
 - match output strings, importing [352](#)
 - match strings, importing [352](#)
- cleanse lists (*continued*)
 - matched property [350](#)
 - matchFlag property [350](#)
 - output property [350](#)
 - properties of [349](#)
 - regular expression match [351](#)
 - replaceAllOccurrences property [349](#)
 - searchType property [349](#)
 - SQL match [351](#)
 - stopOnHit property [349](#)
 - string matches [351](#)
 - Strip property [349](#)
- cleansing data
 - about cleansing data [333](#)
 - in the MDM Hub [333](#)
 - setup tasks [333](#)
 - Unicode settings [46](#)
- clearing history of batch jobs [543](#)
- CM_DIRTY_IND
 - base object column [96](#)
- cmxcleanse.properties
 - about [630](#)
- cmxserver.log
 - description [652](#), [653](#)
- cmxserver.properties
 - about [610](#)
- columns
 - adding to tables [107](#)
 - data types [108](#)
 - properties [109](#)
 - reserved names [78](#)
- command buttons
 - adding objects [38](#), [39](#)
 - editing object properties [38](#), [39](#)
 - removing objects [38](#), [39](#)
- complete tokenize ratio [102](#)
- conditional execution components
 - about conditional execution components [347](#)
 - adding [348](#)
 - when to use [347](#)
- conditional mapping [307](#)
- configuration workbench
 - about [40](#)
- Configuration workbench [51](#), [52](#)
- console.log
 - description [652](#)
- consolidate process
 - about [483](#)
- consolidated record [94](#)
- consolidation
 - best version of the truth [283](#)
- consolidation indicator
 - about the consolidation indicator [260](#)
 - sequence [261](#)
 - values [260](#)
- consolidation process
 - options [283](#)
 - overview [282](#)
- CONSOLIDATION_IND
 - base object column [96](#)
- constants [345](#)
- constraints
 - disabling [102](#)
- Contact match purpose [409](#)
- control tables [363](#)
- Corporate_Entity match purpose [409](#)
- corrupted data
 - troubleshooting [46](#)

- CREATE_DATE
 - base object column [96](#)
 - cross-reference table column [99](#)
 - external match output table column [558](#)
 - staging table column [355](#)
- CREATOR
 - base object column [96](#)
 - cross-reference table column [99](#)
 - external match output table column [558](#)
 - staging table column [355](#)
- cross-reference table
 - load process [265](#)
- cross-reference tables
 - about cross-reference tables [97](#)
 - columns [99](#)
 - defined [76](#)
 - described [76](#)
 - enabling history of promotions [175](#)
 - history table [101](#)
 - relationship to base objects [98](#)
 - ROWID_XREF column [99](#)
- custom button functions
 - about [606](#)
 - registering [608](#)
 - viewing [609](#)
- custom buttons
 - about custom buttons [53](#)
 - adding [57](#)
 - appearance of [54](#)
 - clicking [54](#)
 - custom functions, writing [55](#)
 - deploying [57](#)
 - examples of [55](#)
 - icons [57](#)
 - listing [57](#)
 - properties file [57](#)
 - text labels [57](#)
 - type change [57](#)
 - updating [57](#)
- custom functions
 - client-based [55](#)
 - deleting [57](#)
 - server-based [55](#)
 - writing [55](#)
- custom indexes
 - about [105](#)
 - creating [106](#)
 - creating outside the MDM Hub [107](#)
 - deleting [106](#)
 - editing [106](#)
 - navigating to the node [106](#)
- custom java cleanse functions
 - about [606](#)
- custom Java cleanse functions
 - viewing [607](#), [608](#)
- custom queries
 - about [135](#)
 - adding [136](#)
 - deleting [134](#)
 - editing [137](#)
 - Queries tool [125](#)
 - SQL syntax [135](#)
 - SQL validation [136](#)

D

- data
 - corrupted, troubleshooting [46](#)
 - understanding [375](#)
- data access
 - Hierarchy Manager [216](#)
- data cleansing
 - about data cleansing [333](#)
 - Process Servers [333](#)
- data collection
 - about [655](#)
 - Hub Server [657](#), [658](#)
 - MDM Hub environment information [656](#)
 - Process Server [657](#), [658](#)
 - system configuration information [656](#)
- data encryption
 - architecture [182](#)
 - concept [182](#)
 - configuring [184](#)
 - configuring the Hub Server [185](#)
 - configuring the Process Server [186](#)
 - overview [182](#)
 - properties file [185](#)
 - restrictions [183](#)
 - sample properties file [187](#)
 - supported API requests [186](#)
 - utilities [183](#)
- data sources
 - about data sources [73](#)
 - JDBC data sources [73](#)
 - removing [74](#)
- data steward workbench
 - about [42](#)
- Data Type
 - column property [109](#)
- data types
 - about [108](#)
- database object name, constraints [78](#)
- database properties
 - Operational Reference Store [642](#)
- databases
 - database ID [69](#)
 - target database [30](#)
 - Unicode, configuring [44](#)
- Databases tool
 - about the Databases tool [63](#)
- DataEncryptor
 - implementation [184](#)
 - interface [184](#)
- decay curve [364](#)
- decay types
 - linear [364](#)
 - RISL [364](#)
 - SIRL [364](#)
- decrease effective period
 - example [159](#)
- Default
 - column property [109](#)
- defining [287](#)
- DELETED state, about [170](#)
- DELETED_BY
 - base object column [96](#)
 - cross-reference table column [99](#)
 - staging table column [355](#)
- DELETED_DATE
 - base object column [96](#)
 - cross-reference table column [99](#)

- DELETED_DATE (*continued*)
 - staging table column [355](#)
- DELETED_IND
 - cbase object column [96](#)
 - cross-reference table column [99](#)
 - staging table column [355](#)
- delta detection
 - configuring [310](#)
 - considerations for using [312](#)
 - how handled [311](#)
 - landing table configuration [291](#)
- Display Name
 - column property [109](#)
- distinct mapping [307](#)
- distinct source systems [484](#)
- Division match purpose [409](#)
- duplicate data
 - match for [280](#)
- duplicate match threshold [102](#)

E

- Elasticsearch
 - high availability [461](#)
 - index settings [471](#)
- elasticsearch archive
 - extracting [462](#)
- Elasticsearch installation
 - pre-installation tasks [461](#)
 - prerequisites [461](#)
- EMO table
 - external match output table [558](#)
- encrypting passwords [72](#)
- Enterprise Manager
 - Operational Reference Store properties [642](#)
- entities
 - about [198](#)
 - display options [205](#)
- Entity 360 view
 - entity360view.log [653](#)
- entity base objects
 - about [198](#)
 - converting from base objects [201](#)
 - creating [200](#)
 - reverting to base objects [206](#)
- entity icons
 - adding [197](#)
 - configuring [197](#)
 - deleting [198](#)
 - editing [197](#)
 - uploading default [197](#)
- entity types
 - about [202](#)
 - assigning HM packages to [218](#)
 - creating [204](#)
 - deleting [205](#)
 - editing [205](#)
 - example [203](#)
- entity360view.log
 - description [653](#)
- environment report
 - saving [644](#)
- exact match rule [279](#)
- exact matches
 - exact match / search strategy [406](#)
 - exact match base objects [279](#)
 - exact match columns [392](#), [421](#)

- exact matches (*continued*)
 - exact match strategy [380](#)
 - filtered match [406](#)
- exclusive locks
 - about [33](#)
 - acquiring [35](#)
- exhaustive search level [401](#)
- extended key widths [396](#)
- external match jobs
 - output table [558](#)
- External Match jobs
 - about External Match jobs [557](#)
 - input table [557](#)
 - running [560](#)
- external match output table
 - about [558](#)
 - columns [558](#)
- external match tables
 - system columns [557](#)
- extreme search level [401](#)

F

- Family match purpose [409](#)
- Fields match purpose [409](#)
- FILE_NAME
 - external match output table column [558](#)
- filtered match rule [279](#), [406](#)
- filters
 - about filters [388](#)
 - adding [388](#)
 - deleting [389](#)
 - editing [389](#)
 - properties of [388](#)
- foreign key
 - lookups [361](#)
- foreign key relationship base objects
 - about [211](#)
 - creating [211](#)
- foreign key relationships
 - about [116](#)
 - adding [117](#)
 - creating [116](#)
 - defined [116](#)
 - deleting [119](#)
 - editing [117](#)
 - supported [532](#)
 - virtual relationships [117](#)
- foreign keys
 - lookups [270](#)
- fuzzy match rule [279](#)
- fuzzy matches
 - fuzzy match / search strategy [406](#)
 - fuzzy match base objects [279](#), [395](#)
 - fuzzy match columns [392](#)
 - fuzzy match strategy [380](#)

G

- GBID
 - column property [109](#)
- GBID columns [111](#)
- Generate Match Tokens jobs [560](#)
- Generate Match Tokens on Load [565](#)
- generating
 - match tokens on load lrequeue on parent merge [102](#)

- generic queries
 - about [128](#)
 - adding [128](#)
 - building blocks
 - columns, selecting [130](#)
 - comparison conditions, defining [132](#)
 - constants, defining [131](#)
 - functions, defining [131](#)
 - sort order, defining [133](#)
 - tables, selecting [130](#)
 - deleting [134](#)
 - editing [129](#)
 - Queries tool [125](#)
 - refining query criteria [129](#)
 - SQL statement, viewing [134](#)
- geographic coordinates
 - elevation [392](#)
 - latitude [392](#)
 - longitude [392](#)
- GetAssignableUsersForTask user exit
 - interface [591](#)
- Global Identifier (GBID) columns [111](#)
- glossary [659](#)
- GOV
 - cross-reference table column [99](#)
- graph functions
 - adding [343](#)
 - adding functions to [343](#)
 - conditional execution components [347](#)
 - inputs [342](#)
 - outputs [342](#)
- group execution logs
 - status values [550](#)
 - viewing [550](#)

H

- hard delete detection
 - primary key source columns [322](#)
- Hard Delete Detection
 - C_REPOS_EXT_HARD_DEL_DETECT table [318](#)
 - consensus delete [326](#)
 - delete flag values [317](#)
 - direct delete [323](#), [324](#), [327](#)
 - end date [324](#), [327](#)
 - hard delete detection table [318](#)
 - overview [316](#)
 - stage process [316](#)
 - trust and validation rules [317](#)
 - types [317](#)
 - within user exits [331](#)
- Has Default
 - column property [109](#)
- hierarchies
 - about [189](#), [206](#)
 - adding [207](#)
 - deleting [207](#)
 - example [190](#)
 - hierarchy node [206](#)
- Hierarchies tool
 - configuration overview [191](#)
- Hierarchy Manager
 - configuration overview [191](#)
 - data configuration [216](#)
 - entity icons, uploading [197](#)
 - prerequisites [191](#)
 - repository base object tables [196](#)

- Hierarchy Manager (*continued*)
 - sandboxes [220](#)
- highest reserved key
 - example [358](#)
- history
 - enabling [101](#)
- history of cross-reference promotion, enabling [175](#)
- history tables
 - base object history tables [101](#)
 - cross-reference history tables [101](#)
 - defined [76](#)
 - enabling [105](#)
- HM packages
 - assigning to entity types [218](#)
 - deleting [216](#)
 - editing [216](#)
- Household match purpose [409](#)
- HPCT table *See pending control table*
- HTTPS
 - for Process Servers [337](#)
- Hub Console
 - starting [30](#)
- Hub Server
 - cmxserver.log [652](#)
 - disabling data collection [658](#)
 - enabling data collection [657](#)
 - logging settings [652](#)
 - properties [610](#)
- Hub states
 - record states [170](#), [173](#)
- Hub Store
 - Master Database [60](#)
 - Operational Record Store (ORS) [60](#)
 - properties of [69](#)
 - schema [75](#)
 - table types [76](#)
- HUB_STATE_IND
 - cross-reference table column [99](#)
 - staging table column [355](#)
- HUB_STATE_IND column
 - about [171](#)

I

- IBM DB2
 - data types [108](#)
- immutable rowid object [483](#)
- importing table column definitions [113](#)
- increase effective period
 - example [159](#)
- incremental loads [266](#)
- index settings
 - analyzer [471](#)
- indexes
 - custom indexes [105](#)
- Individual match purpose [409](#)
- Informatica Data Director basic search
 - allow missing child records [390](#)
- Informatica Data Quality
 - batch jobs and mappings [301](#)
- initial data loads (IDLs) [266](#)
- inputs [345](#)
- inter-table paths [383](#)
- INTERACTION_ID
 - cross-reference table column [99](#)
- INTERACTION_ID column
 - about [172](#)

intertable matching
described [421](#)
intra-table paths [385](#)

J

JAR file
for user exits [578](#)
implementing the user exit JAR file [578](#)
JAR files
ORS-Specific, downloading [600](#)
ORS-Specific, using [601](#)
Java archive (JAR) files
tools.jar [604](#)
Java user exits
about [576](#)
JMS Event Schema Manager
about [602](#)
auto-searching for out-of-sync objects [605](#)
finding out-of-sync objects [604](#)
starting [603](#)

K

key
primary [305](#)
primary, from multiple columns [305](#)
primary, from single column [305](#)
Key Match jobs [562](#)
key types [395](#)
key widths [396](#)

L

land process
C_REPOS_SYSTEM table [288](#)
external batch process [263](#)
extract-transform-load (ETL) tool [263](#)
landing tables [262](#)
managing [263](#)
overview [262](#)
real-time processing (API calls) [263](#)
source systems [262](#)
ways to populate landing tables [263](#)
landing table
about [295](#)
landing tables
about landing tables [291](#)
adding [292](#)
columns [291](#)
defined [76](#)
editing [292](#)
properties of [291](#)
removing [293](#)
Language
configuring for Oracle environments [47](#)
LAST_ROWID_SYSTEM
base object column [96](#)
LAST_UPDATE_DATE
base object column [96](#)
cross-reference table column [99](#)
staging table column [355](#)
limited key widths [396](#)
linear decay [364](#)

load batch
example [157](#)
for multiple record versions [156](#)
Load jobs
forced updates, about [564](#)
Generate Match Tokens on Load [565](#)
load batch size [102](#)
rejected records [542](#)
load process
load inserts [267](#)
overview [265](#)
steps for managing data [267](#)
user exits [583](#)
loading by row ID [308](#)
loading data
incremental loads [266](#)
initial data loads (IDLs) [266](#)
locks
lock expiration [34](#)
acquiring [33–35](#)
clearing [35](#)
exclusive locks [33, 35](#)
non-exclusive locks [33](#)
releasing [35](#)
server caching [34](#)
types of locks [648](#)
write locks [33, 34](#)
log file rolling
about [645](#)
log4j-entity360view.xml
description [653](#)
logs, application server
configuring [646](#)
levels [645](#)
logs, application server database
about [644](#)
lookups
about lookups [361](#)
configuring [361](#)
for foreign keys [361](#)

M

manual
batch jobs [536](#)
Manual Merge jobs [566](#)
Manual Unmerge jobs [567](#)
mapping
management of [314](#)
mappings
adding [303](#)
between staging and landing tables [76](#)
cleansed [301](#)
column [306](#)
conditional mapping [307](#)
configuring [300](#)
diagrams [302](#)
distinct mapping [307](#)
editing [314](#)
Informatica Data Quality batch jobs [301](#)
jumping to a schema [315](#)
loading by row ID [308](#)
passed through [301](#)
properties of [297](#)
query parameters [308](#)
removing [315](#)
testing [315](#)

- Mappings tool [302, 574](#)
- Master Database
 - changing the password [71](#)
 - creating [61](#)
- match
 - accept all unmatched rows as unique [380](#)
 - allow missing child records [390](#)
 - child records [421](#)
 - dynamic match analysis threshold setting [381](#)
 - fuzzy population [380](#)
 - Match Analyze jobs [569](#)
 - Match for Duplicate Data jobs [571](#)
 - match minutes, maximum elapsed [102](#)
 - match only once setting [381](#)
 - match only previous rowid objects setting [381](#)
 - match output strings, importing [352](#)
 - match strings, importing [352](#)
 - match subtype [414](#)
 - match table, resetting [573](#)
 - match tables [568](#)
 - match tokens, generate on PUT [102](#)
 - match/search strategy [380](#)
 - maximum matches for manual consolidation [379](#)
 - non-equal matching [418](#)
 - NULL matching [415](#)
 - pending records [175](#)
 - populations [44](#)
 - populations for fuzzy matches [380](#)
 - properties
 - about match properties [378](#)
 - segment matching [420](#)
 - strategy
 - exact matches [380](#)
 - fuzzy matches [380](#)
 - string matches in cleanse lists [351](#)
- match column rules
 - adding [422](#)
 - deleting [424](#)
 - editing [423](#)
- match columns
 - about match columns [392](#)
 - exact match base objects [398](#)
 - exact match columns [392](#)
 - fuzzy match base objects [395](#)
 - fuzzy match columns [392](#)
 - key widths [396](#)
 - match key types [395](#)
 - missing child records [390](#)
- Match jobs
 - state-enabled BOs [568](#)
- match key tables
 - defined [76](#)
- match paths
 - inter-table paths [383](#)
 - intra-table paths [385](#)
 - relationship base objects [383](#)
- match process
 - build match groups (BMGs) [281](#)
 - exact match base objects [279](#)
 - fuzzy match base objects [279](#)
 - match key table [280](#)
 - match rules [279](#)
 - match tables [280](#)
 - overview [277](#)
 - populations [280](#)
 - related base object properties [376](#)
 - support tables [280](#)
 - transitive matches [281](#)
- match process (*continued*)
 - user exits [585](#)
- match purposes
 - field names [393](#)
 - field types [393](#)
- match rule properties
 - geocode radius [414](#)
- match rule sets
 - about match rule sets [400](#)
 - adding [404](#)
 - deleting [405](#)
 - editing [404](#)
 - editing the name [404](#)
 - filters [403](#)
 - properties of [401](#)
 - search levels [401](#)
- match rules
 - about match rules [279](#)
 - accept limit [414](#)
 - defining [376, 405](#)
 - exact match columns [421](#)
 - match / search strategy [406](#)
 - match levels [413](#)
 - match purposes
 - about [407](#)
 - Address match purpose [409](#)
 - Contact match purpose [409](#)
 - Corporate_Entity match purpose [409](#)
 - Division match purpose [409](#)
 - Family match purpose [409](#)
 - Fields match purpose [409](#)
 - Household match purpose [409](#)
 - Individual match purpose [409](#)
 - Organization match purpose [409](#)
 - Person_Name match purpose [409](#)
 - Resident match purpose [409](#)
 - Wide_Contact match purpose [409](#)
 - Wide_Household match purpose [409](#)
 - primary key match rules
 - adding [427](#)
 - editing [427, 428](#)
- Reset Match Table jobs [573](#)
- types of [279](#)

- match subtype [414](#)
- matching
- duplicate data [280](#)
- maximum trust [364](#)
- MDM Hub
- logging [651](#)
- MDM Hub Console
- about [30](#)
- logging in [35](#)
- MDM Hub Console interface
- customizing [39](#)
- General tab [39](#)
- Quick Launch tab [39](#)
- toolbar [39](#)
- window sizes and positions [39](#)
- wizard welcome screens [39](#)
- MDM Hub staging
- overview [264, 294](#)
- merge
- Batch Unmerge [556](#)
- Manual Merge jobs [566](#)
- Manual Unmerge jobs [567](#)
- Merge Manager tool [282](#)
- merge package [137](#)

- merge process
 - user exits [587](#)
- merge process process
 - about [483](#)
- message queue servers
 - about message queue servers [492](#)
 - adding [493](#)
 - deleting [494](#)
 - editing [493](#)
- message queues
 - about message queues [285](#), [494](#)
 - adding [494](#)
 - deleting [495](#)
 - editing [495](#)
 - message check interval [491](#)
 - Message Queues tool [491](#)
 - properties of [494](#)
 - receive batch size [491](#)
 - receive timeout [491](#)
 - status of [491](#)
- message schema
 - ORS-specific, generating and deploying [604](#)
- message triggers
 - about [284](#)
 - about message triggers [496](#)
 - adding [499](#)
 - deleting [500](#)
 - editing [500](#)
 - enabling for pending updates [180](#)
 - enabling for state changes [179](#)
 - types of [497](#)
- messages
 - elements in [501](#)
 - examples
 - accept as unique message [503](#)
 - AmRule message [504](#)
 - BoDelete message [504](#)
 - BoSetToDelete message [505](#)
 - insert message [507](#)
 - merge message [507](#), [519](#)
 - merge update message [508](#)
 - no action message [509](#)
 - PendingInsert message [509](#)
 - PendingUpdate message [510](#)
 - PendingUpdateXref message [511](#)
 - unmerge message [511](#)
 - update message [512](#)
 - update XREF message [513](#)
 - XRefDelete message [513](#)
 - XRefSetToDelete message [514](#)
 - examples (legacy)
 - accept as unique message [516](#)
 - bo delete message [516](#)
 - bo set to delete message [517](#)
 - delete message [506](#), [518](#)
 - insert message [519](#)
 - merge update message [520](#)
 - pending insert message [521](#)
 - pending update message [521](#)
 - pending update XREF message [522](#)
 - unmerge message [524](#)
 - update message [523](#)
 - update XREF message [524](#)
 - XREF delete message [525](#)
 - XREF set to Delete message [526](#)
 - filtering [503](#), [515](#)
 - message fields [515](#)

- metadata
 - synchronizing [114](#)
 - trust [114](#)
- Microsoft SQL Server
 - custom index limitations [106](#)
 - data types [108](#)
 - row width limitation [108](#)
- minimum trust [364](#)
- missing child records
 - about [390](#)
- model workbench
 - about [41](#)
- Multi Merge jobs [571](#)
- Multidomain MDM
 - installation details [39](#)

N

- narrow search level [401](#)
- Navigation pane
 - about [36](#)
- Navigation pane child nodes
 - hiding [36](#)
 - showing [36](#)
- navigation tree
 - changing the item view [37](#)
 - child nodes [36](#)
 - filtering items [37](#)
 - filtering options [37](#)
 - parent nodes [36](#)
 - running commands [38](#)
 - searching for items [38](#)
 - sorting by display name [37](#)
- New Query Wizard [136](#)
- non-equal matching [418](#)
- NULL matching [415](#)
- null values
 - allowing null values in a column [109](#)
- Nullable
 - column property [109](#)

O

- oad batch
 - setup for timeline [156](#)
- Operation Reference Store
 - registering [64](#), [66](#), [67](#)
- Operational Reference Store
 - database properties [642](#)
- Operational Reference Stores (ORS)
 - about ORSs [60](#)
 - connection testing [70](#)
 - creating [61](#)
 - editing [69](#)
 - GETLIST limit (rows) [69](#)
 - JNDI data source name [69](#)
 - password, changing [71](#)
 - unregistering [73](#)
- Oracle
 - data types [108](#)
- Organization match purpose [409](#)
- Organization_Name key type [395](#)
- ORIG_ROWID_OBJECT
 - cross-reference table column [99](#)
- ORS-specific API
 - properties [598](#)

- ORS-Specific API
 - API archive table
 - maintenance [599](#)
 - archive table [599](#)
 - build_war macro [601](#)
 - downloading client JAR [600](#)
 - using [601](#)
- ORS-specific APis
 - repository object statuses [599](#)
- ORS-specific APIs
 - overview [597](#)
 - performance [598](#)
 - repository objects [598](#)
- ORS-specific message schemas
 - overview [602](#)
- ORS-Specific SIF API
 - generating and deploying [600](#)
- outputs [346](#)

P

- packages
 - about [137](#), [215](#)
 - adding [138](#)
 - deleting [140](#)
 - display packages, about [137](#)
 - editing [139](#)
 - join queries [140](#)
 - merge [137](#)
 - overview [124](#)
 - Packages tool [125](#)
 - PUT-enabled [137](#)
 - putable [137](#)
 - refreshing after query change [139](#)
 - schema changes [126](#)
 - update packages, about [137](#)
- Packages tool [125](#)
- parallel degree [102](#)
- passwords
 - changing [36](#)
 - encrypting [72](#)
- path components
 - adding [391](#)
 - deleting [391](#)
 - editing [391](#)
- PCTL table *See pending control table*
- pending control table
 - about [489](#)
- pending records, enabling match [175](#)
- PENDING state, about [170](#)
- PERIOD_END_DATE
 - cross-reference table column [99](#)
 - staging table column [355](#)
- PERIOD_START_DATE
 - cross-reference table column [99](#)
 - staging table column [355](#)
- Person_Name key type [395](#)
- Person_Name match purpose [409](#)
- Physical Name
 - column property [109](#)
- PKEY_SRC_OBJECT
 - cross-reference table column [99](#)
 - staging table column [355](#)
- populations
 - configuring [44](#)
 - multiple populations [45](#)
 - non-US populations [44](#)
- populations (*continued*)
 - selecting [380](#)
- post-landing user exit
 - about [581](#)
 - interface [581](#)
- post-load user exit
 - about [584](#)
 - interface [584](#)
- post-match user exit
 - about [586](#)
 - interface [586](#)
- post-merge user exit
 - about [587](#)
 - interface [588](#)
- post-stage user exit
 - about [582](#)
 - interface [583](#)
- post-unmerge user exit
 - about [589](#)
- Post-unmerge user exit
 - interface [590](#)
- pre-match user exit
 - interface [586](#)
- pre-merge user exit
 - about [586](#)
- pre-stage user exit
 - about [582](#)
 - interface [582](#)
- pre-unmerge user exit
 - interface [589](#)
- preferred key widths [396](#)
- preserving source system keys [357](#)
- primary key
 - from multiple columns [305](#)
 - from single column [305](#)
 - hard delete detection [322](#)
- primary key match rules
 - about [426](#)
 - adding [427](#)
 - deleting [428](#)
 - editing [427](#)
- Process Server
 - cmxserver.log [653](#)
 - disabling data collection [658](#)
 - enabling data collection [657](#)
 - properties [630](#)
- Process Servers
 - about [333](#)
 - adding [337](#)
 - batch jobs [335](#)
 - cleanse requests [335](#)
 - configuring [333](#)
 - deleting [338](#)
 - distributed [334](#)
 - editing [338](#)
 - HTTPS, enabling [337](#)
 - modes [334](#)
 - on-line operations [335](#)
 - Process Server tool [335](#)
 - properties of [335](#)
 - testing [338](#)
- Processes view
 - about [32](#)
- Product Usage Toolkit
 - about [655](#)
 - MDM Hub environment information [656](#)
 - system configuration information [656](#)

- profiles
 - about profiles [219](#)
 - adding [219](#)
 - copying [221](#)
 - deleting [221](#)
 - validating [220](#)
- Promote jobs [571](#), [649](#)
- PROMOTE_IND
 - cross-reference table column [99](#)
- Provisioning Tool
 - provisioning.log [653](#)
- provisioning.log
 - description [653](#)
- proximity search
 - configuring [432](#)
 - overview [431](#)
- publish process
 - about [284](#)
 - distribution flow [284](#)
 - message queues [285](#)
 - message triggers [284](#)
 - ORS-specific schema file [285](#)
 - run-time flow [286](#)
 - XSD file [285](#)
- purposes, match [407](#)
- PUT_UPDATE_MERGE_IND
 - cross-reference table column [99](#)
- PUT-enabled package [137](#)
- Putable
 - column property [109](#)
- putable package [137](#)

Q

- queries
 - impact analysis, viewing [134](#)
 - join queries [140](#)
 - organizing [127](#)
 - overview [124](#)
 - package dependencies, analyzing [134](#)
 - results, viewing [134](#)
 - schema changes [126](#)
- Queries tool [125](#)
- query groups
 - about [127](#)
 - adding [127](#)
 - deleting [127](#)
 - editing [127](#)

R

- rapid slow initial later (RISL) decay [364](#)
- raw table
 - about [295](#)
- record
 - edit [161](#)
 - update [161](#)
- record state [169](#)
- record states
 - about [170](#)
- record version
 - add [160](#)
 - edit [161](#)
 - update [161](#)
- records
 - match pending [175](#)

- regular expression functions
 - adding [342](#)
- reject table
 - about [295](#)
- relationship
 - details [118](#)
- relationship base object
 - delete a relationship period [165](#)
 - delete all relationship periods [166](#)
 - end a relationship [164](#)
 - example of deleting [165](#), [166](#)
 - example of ending [164](#)
 - example of updating [162](#), [163](#)
 - update a relationship [162](#)
- relationship base objects
 - about [208](#)
 - converting to [209](#)
 - creating [209](#)
 - creating foreign key relationship base objects [211](#)
 - foreign key relationship base objects [211](#)
 - reverting to base objects [210](#)
- relationship objects
 - about [207](#)
- relationship types
 - about [212](#)
 - creating [214](#)
 - deleting [215](#)
 - editing [215](#)
 - example [213](#)
- relationships
 - foreign key relationships [116](#)
- repository base object (RBO) tables [196](#)
- requeue on parent merge [102](#)
- Reset Match Table jobs [573](#)
- Resident match purpose [409](#)
- Revalidate jobs [573](#)
- row-level locking
 - about row-level locking [647](#)
 - configuring [648](#)
 - considerations for using [648](#)
 - default behavior [647](#)
 - enabling [648](#)
 - wait times [649](#)
- ROWID_MATCH_RULE
 - external match output table column [558](#)
- ROWID_OBJECT
 - base object column [96](#)
 - cross-reference table column [99](#)
 - staging table column [355](#)
- ROWID_OBJECT_MATCH
 - external match output table column [558](#)
- ROWID_SYSTEM
 - cross-reference table column [99](#)
- ROWID_XREF
 - cross-reference table column [99](#)

S

- S_
 - cross-reference table column [99](#)
- sample applications
 - for data encryption [183](#)
- sandboxes [220](#)
- Schema Manager
 - adding columns to tables [107](#)
 - base objects [94](#)
 - foreign key relationships [116](#)

Schema Manager (*continued*)

- starting [93](#)
- schema match columns [573](#)
- schema objects [78](#)
- schema trust columns [574](#)
- Schema Viewer
 - column names [121](#)
 - command buttons [119](#)
 - context menu [121](#)
 - Diagram pane [119](#)
 - hierarchic view [121](#)
 - options [121](#)
 - orientation [121](#)
 - orthogonal view [121](#)
 - Overview pane [119](#)
 - panes [119](#)
 - printing [122](#)
 - saving as JPG [122](#)
 - starting [119](#)
 - tooggling views [121](#)
 - zooming all [120](#)
 - zooming in [120](#)
 - zooming out [120](#)
- schemas
 - about schemas [75](#)
 - changes affect queries and packages [126](#)
- search
 - architecture with Elasticsearch [460](#)
 - stop words [464](#)
 - stopwords [465](#)
 - stopwords.txt file [464](#), [465](#)
 - synonyms [464](#), [465](#)
 - synonyms.txt file [464](#), [465](#)
 - with Elasticsearch [459](#)
 - words to ignore [465](#)
- search levels for match rule sets [401](#)
- secure communications
 - enabling, in the Process Server [337](#)
- security access manager workbench
 - about [41](#)
- segment matching [420](#)
- sequencing batch jobs [531](#)
- SIF API
 - ORS-Specific, removing [601](#)
 - ORS-Specific, renaming [600](#)
 - supported with user exits [593](#)
- SMOS
 - state management override system [288](#)
- source system
 - properties of [289](#)
- source systems
 - about source systems [287](#)
 - adding [289](#)
 - Admin source system [288](#)
 - distinct source systems [484](#)
 - highest reserved key [357](#)
 - immutable source systems [483](#)
 - preserving keys [357](#)
 - removing [290](#)
 - renaming [290](#)
 - system repository table (C_REPOS_SYSTEM) [288](#)
 - Systems and Trust tool, starting [288](#)
- SOURCE_KEY
 - external match output table column [558](#)
- SOURCE_NAME
 - external match output table column [558](#)
- SQL
 - custom queries [135](#)

- SQL*Loader
 - corrupted data [46](#)
- SRC_LUD
 - cross-reference table column [99](#)
- SRC_ROWID
 - staging table column [355](#)
- Stage jobs
 - rejected records [542](#)
- stage process
 - staging tables [355](#)
 - tables [295](#)
 - user exits [580](#)
- staging table
 - about [295](#)
 - load process [265](#)
 - management of [313](#)
- staging table columns
 - allow null update property [358](#)
 - lookup column property [358](#)
 - lookup table property [358](#)
- staging tables
 - about [355](#)
 - allow null foreign key [358](#)
 - cell update [358](#)
 - column properties [358](#)
 - columns [112](#)
 - columns, creating [112](#)
 - configuring [355](#)
 - defined [76](#)
 - editing [313](#), [360](#)
 - highest reserved key [357](#)
 - jumping to source system [313](#)
 - preserve source system keys [357](#)
 - properties of [296](#)
 - removing [314](#)
 - system columns [355](#)
 - user-defined columns [355](#)
- standard key widths [396](#)
- state management
 - about [169](#), [170](#)
 - base object record survivorship [173](#)
 - enabling [175](#)
 - HUB_STATE_IND column [171](#)
 - INTERACTION_ID column [172](#)
 - Load jobs [562](#)
 - Match jobs [568](#)
 - message triggers, enabling [179](#)
 - promoting records [180](#)
 - record promotion [180](#)
 - record states
 - state transitions [172](#)
 - rules for loading data [173](#)
 - state transition rules, about [172](#)
 - timeline [146](#)
- state management override system
 - load job [564](#)
- state management override system (SMOS)
 - about [288](#)
 - states, record [170](#)
 - status, setting [538](#)
- STG_ROWID_TABLE
 - cross-reference table column [99](#)
- stopwords.txt file
 - configuring [464](#)
 - customizing [465](#)
- survivorship [261](#), [262](#)
- Synchronize jobs [367](#), [574](#)
- synchronizing metadata [114](#)

- synonyms.txt file
 - configuring [464](#)
 - customizing [465](#)
- system columns
 - base objects [96](#)
 - described [116](#)
 - external match tables [557](#)
 - staging tables [355](#)
- system repository table [288](#)
- system states
 - record states [170](#)
- Systems and Trust tool [288](#)

T

- table columns
 - about table columns [108](#)
 - adding [113](#)
 - deleting [115](#)
 - editing [114](#)
 - Global Identifier (GBID) columns [111](#)
 - importing from another table [113](#)
 - staging tables [112](#)
- tables
 - adding columns to [107](#)
 - base objects [76](#)
 - control tables [363](#)
 - cross-reference tables [76](#)
 - history tables [76](#)
 - Hub Store [76](#)
 - landing tables [76](#)
 - match key tables [76](#), [272](#)
 - staging tables [76](#)
 - supporting tables used by batch process [531](#)
 - system repository table (C_REPOS_SYSTEM) [288](#)
- target database
 - changing [35](#)
 - selecting [30](#)
- task management
 - user exits [590](#)
- tasks
 - BPM workflow tool [174](#)
- timeline
 - configuring [156](#)
 - enabling [101](#), [105](#), [155](#)
 - example [142](#)
 - granularity [145](#)
 - guidelines [142](#)
 - load batch [156](#)
 - overview [141](#)
 - rules [146](#), [147](#)
 - state management [146](#)
- timeline record versions
 - example [144](#)
- timeline rules
 - contiguity [147](#)
 - overlapping effective periods [147](#)
- token filter
 - built-in [473](#)
- tokenize process
 - about the tokenize process [271](#)
 - dirty table [273](#)
 - key concepts [273](#)
 - match key tables [272](#)
 - match keys [272](#)
 - match tokens [272](#)

- tokenizer
 - built-in [473](#)
- Tool Access tool [52](#)
- tools
 - Batch Viewer tool [535](#)
 - Cleanse Functions tool [339](#)
 - Databases tool [63](#)
 - Mappings tool [574](#)
 - Merge Manager tool [282](#)
 - Schema Manager [93](#)
 - Tool Access tool [52](#)
 - user access to [51](#)
 - Users tool [51](#)
- traceability [282](#)
- troubleshooting
 - cmxserver.log file [652](#)
- trust
 - about trust [362](#)
 - calculations [363](#)
 - columns [366](#)
 - considerations for setting [365](#)
 - decay curve [364](#)
 - decay graph types [364](#)
 - decay periods [363](#)
 - enabling for a column [366](#)
 - levels [363](#)
 - maximum trust [364](#)
 - minimum trust [364](#)
 - properties of [364](#)
 - slow initial rapid later (SIRL) decay [364](#)
 - state-enabled base objects [369](#)
 - synchronizing trust settings [574](#)
 - Systems and Trust tool [288](#)
 - trust levels, defined [363](#)
- Trust
 - column property [109](#)
- typical search level [401](#)

U

- Unicode
 - cleanse settings [46](#)
 - NLS_LANG [47](#)
 - Unix and locale recommendations [46](#)
- Unique
 - column property [109](#)
- unique key [305](#)
- unmerge
 - batch unmerge [556](#)
 - cascade unmerge [484](#)
 - manual unmerge [567](#)
 - unmerge child when parent unmerges [484](#)
- unmerge process
 - user exits [588](#)
- UPDATED_BY
 - base object column [96](#)
 - column in cross-reference table [99](#)
 - staging table column [355](#)
- user exit
 - creating [592](#)
 - example [592](#)
 - supported SIF APIs [593](#)
 - to call SIF API [591](#), [592](#)
- user exit for SIF API
 - creating [592](#)
 - example [592](#)

- user exits
 - about [576, 606, 607](#)
 - AssignTasks interface [590](#)
 - GetAssignableUsersForTask interface [591](#)
 - guidelines [595](#)
 - implementing the JAR file [578](#)
 - JAR files [578](#)
 - load process [583](#)
 - match process [585](#)
 - merge process [587](#)
 - post-landing [581](#)
 - post-landing user exit interface [581](#)
 - post-load [584](#)
 - post-load interface [584](#)
 - post-load user exit parameters [584](#)
 - post-match [586](#)
 - post-match interface [586](#)
 - post-merge [587](#)
 - post-merge interface [588](#)
 - post-stage [582](#)
 - post-stage interface [583](#)
 - post-unmerge [589](#)
 - post-unmerge interface [590](#)
 - pre-match interface [586](#)
 - pre-merge [582](#)
 - pre-stage [582](#)
 - pre-stage interface [582](#)
 - pre-unmerge interface [589](#)
 - processing [577](#)
 - removing [578](#)
 - rollback [577](#)
 - stage process [580](#)
 - task management [590](#)
 - unmerge process [588](#)
 - uploading [578](#)
 - user exits
 - post-load parameters [584](#)
 - UserExitContext class [579](#)
 - viewing [607](#)
- User Object Registry
 - about [606](#)
 - custom button functions, viewing [609](#)
 - custom Java cleanse functions, viewing [608](#)
 - starting [607](#)
 - user exits, viewing [607](#)
- user objects
 - about [606](#)
- user-defined columns
 - staging tables [355](#)
- UserExitContext
 - about [579](#)
- users
 - tool access [51](#)
- Users [51](#)
- utilities workbench
 - about [42](#)

V

- Validate
 - column property [109](#)
- validation checks [368](#)
- validation rule
 - custom, example [372](#)
- validation rules
 - about validation rules [368](#)
 - adding [373](#)
 - custom validation rules [370, 372](#)
 - defined [368](#)
 - defining [368](#)
 - domain checks [370](#)
 - downgrade percentage [371](#)
 - editing [373, 374](#)
 - enabling columns for validation [369](#)
 - examples of [371](#)
 - execution sequence [369](#)
 - existence checks [370](#)
 - pattern validation [370](#)
 - properties of [370](#)
 - referential integrity [370](#)
 - removing [374](#)
 - required columns [368](#)
 - reserve minimum trust [371](#)
 - rule column properties [371](#)
 - rule name [370](#)
 - rule SQL [371](#)
 - rule types [370](#)
 - state-enabled base objects [369](#)
 - validation checks [368](#)
- VERSION_SEQ
 - staging table column [355](#)

W

- Wide_Contact match purpose [409](#)
- Wide_Household match purpose [409](#)
- Workbenches view
 - about [32](#)
 - tools [32](#)
- workflow engines
 - adding [175](#)
- workflow tool [174](#)
- workflows
 - record states [170](#)
- write locks
 - about [33](#)
 - acquiring [34](#)

X

- XSD file
 - downloading [604](#)