



Informatica® Multidomain MDM
10.5 HotFix 4

設定ガイド

Informatica Multidomain MDM 設定ガイド

10.5 HotFix 4

2025 年 5 月

© 著作権 Informatica LLC 2001, 2025

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複製、写真複製、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

米政府の権利プログラム、ソフトウェア、データベース、および関連文書や技術データは、米国政府の顧客に配信され、「商用コンピュータソフトウェア」または「商業技術データ」は、該当する連邦政府の取得規制と代理店固有の補足規定に基づきます。このように、使用、複製、開示、変更、および適応は、適用される政府の契約に規定されている制限およびライセンス条項に従うものとし、政府契約の条項によって適当な範囲において、FAR 52.227-19、商用コンピュータソフトウェアライセンスの追加権利を規定します。

Informatica、Informatica ロゴ、および ActiveVOS は、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

本ソフトウェアまたはドキュメンテーション（あるいはその両方）の一部は、第三者が保有する著作権の対象となります。必要な第三者の通知は、製品に含まれています。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、infa_documentation@informatica.com までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2025-06-10

目次

序文	25
Informatica のリソース.....	25
Informatica Network.....	25
Informatica ナレッジベース.....	25
Informatica マニュアル.....	26
Informatica 製品可用性マトリックス.....	26
Informatica Velocity.....	26
Informatica Marketplace.....	26
Informatica グローバルカスタマサポート.....	26
 第 I 部 : 概要	27
 第 1 章 : Informatica MDM Hub の管理	28
Informatica MDM Hub の管理の概要.....	28
Informatica MDM Hub の管理の各フェーズ.....	28
スタートアップフェーズ.....	29
設定段階.....	29
プロダクションフェーズ.....	29
 第 2 章 : MDM Hub コンソールの基本操作	30
概要.....	30
MDM Hub コンソールについて.....	30
Hub コンソールの起動.....	30
シングルサインオンによる MDM Hub コンソールの設定.....	32
MDM Hub コンソールでの操作.....	33
プロセスビューとワークベンチビューの切り替え.....	33
ワークベンチビューでのツールの起動.....	33
メタデータを変更するためのロックの取得.....	34
ターゲットデータベースの変更.....	36
別のユーザーとしてのログイン.....	37
ユーザーのパスワードの変更.....	37
ナビゲーションペインでのナビゲーションツリーの使用.....	37
コマンドボタンを使用したオブジェクトの追加、編集、および削除.....	40
MDM Hub コンソールインタフェースのカスタマイズ.....	41
バージョンの詳細の表示.....	41
Informatica MDM Hub ワークベンチおよびツール.....	41
設定ワークベンチのツール.....	42
モデルワークベンチのツール.....	42
セキュリティアクセスマネージャーワークベンチのツール.....	43
データスチュワードワークベンチのツール.....	43

ユーティリティワークベンチのツール.	44
第 3 章 : インターナショナルデータのサポートの設定.	45
インターナショナルデータのサポートの設定の概要.	45
Unicode データベースの設定 (Oracle のみ)	45
US 以外のポピュレーションの一致設定の設定.	46
一致処理に使用するエンコードの設定.	47
1 つのベースオブジェクト内での複数のポピュレーションの使用.	47
Windows レジストリでの ANSI コードページの設定.	48
Unicode 用のクレンジング設定.	48
UTF-8 を使用する場合の UNIX 向けのロケールの推奨事項.	48
破損したデータのトラブルシューティング.	49
Oracle 環境での言語の設定.	49
NLS_LANG の構文.	49
Windows レジストリでの NLS_LANG の設定.	50
環境変数としての NLS_LANG の設定 (Windows)	50
LANG 環境変数とロケールの設定 (UNIX)	50
第 II 部 : ハブコンソールツールの設定.	52
第 4 章 : Hub コンソールのツールへのアクセスの設定.	53
Hub コンソールのツールへのアクセスの設定に関する概要.	53
ユーザー設定.	53
ツールおよびプロセスに対するユーザーアクセス.	54
ツールアクセスツールの起動.	54
ツールおよびプロセスに対するアクセス権のユーザーへの付与.	54
ツールおよびプロセスに対するアクセス権のユーザーからの取り消し.	54
第 5 章 : Hub コンソールツールでのカスタムボタンの実装.	56
概要.	56
Hub コンソールのカスタムボタンについて.	56
ユーザーがカスタムボタンをクリックしたときの動作.	57
Hub コンソールにカスタムボタンを表示する手順.	57
カスタムボタンの追加.	57
カスタム関数の作成.	58
カスタムボタンの外観の制御.	60
カスタムボタンのデプロイ.	60
第 III 部 : データモデルの構築.	62
第 6 章 : Hub Store について	63
概要.	63
Hub Store のデータベース.	63

Hub Store 内の各データベースの関係.	64
Hub ストアデータベースの作成.	64
バージョン要件.	64

第 7 章 : オペレーショナル参照ストアとデータソースの設定. 65

オペレーショナルリファレンスストアとデータソースの設定の概要.	65
作業を開始する前に.	65
データベースツールの概要.	66
データベースツールの起動.	66
オペレーショナルリファレンスストアの設定.	66
Microsoft SQL Server 用のオペレーショナルリファレンスストアの接続プロパティ.	67
オペレーショナルリファレンスストアの Oracle 用接続プロパティ.	67
オペレーショナルリファレンスストアの IBM DB2 用接続プロパティ.	69
オペレーショナル参照ストアの登録.	70
オペレーショナルリファレンスストア登録プロパティの編集.	70
オペレーショナルリファレンスストアのプロパティの編集.	72
オペレーショナルリファレンスストアの接続.	73
パスワードの変更.	74
パスワードの暗号化.	75
オペレーショナル参照のプロダクションモード.	75
オペレーショナルリファレンスストアの登録解除.	76
IBM DB2 でオペレーショナル参照ストアを削除する.	76
データソースの設定.	77
WebLogic でのデータソースの管理.	77
データソースの作成.	77
データソースの削除.	77

第 8 章 : スキーマの構築. 79

概要.	79
作業を開始する前に.	79
スキーマについて.	79
オペレーショナル参照ストアのテーブルのタイプ.	80
スキーマオブジェクトを定義する際の要件.	83
スキーママネージャの起動.	98
ベースオブジェクトの設定.	98
ベースオブジェクトと Hub ストア内のその他のテーブルの間のリレーション.	99
ベースオブジェクトを定義するプロセスの概要.	99
ベースオブジェクトのカラム.	100
相互参照テーブル.	101
履歴テーブル.	105
ベースオブジェクトのプロパティ.	105
ベースオブジェクトの作成.	108
ベースオブジェクトのプロパティの編集.	109

ベースオブジェクトのカスタムインデックス.	109
ベースオブジェクトの影響分析の表示.	111
ベースオブジェクトの削除.	111
テーブルのカラムの設定.	112
カラムについて.	112
カラムエディタへの移動.	116
カラムの追加.	117
別のテーブルからのカラム定義のインポート.	118
カラムのプロパティの編集.	118
カラムの表示順の変更.	119
カラムの削除.	120
ベースオブジェクト間の外部キーリレーションの設定.	120
外部キー関係について.	120
外部キーリレーションを定義するプロセスの概要.	121
外部キー関係の追加.	121
外部キー関係編集.	121
リレーションの詳細.	122
外部キーリレーションのルックアップの設定.	123
外部キー関係の削除.	123
スキーマの表示.	123
スキーマビューアの起動.	123
スキーマ図のズームインとズームアウト.	124
スキーマ図のビューの切り替え.	125
関連するデザインオブジェクトやバッチジョブへの移動.	125
スキーマビューアのオプションの設定.	125
スキーマ図の JPG イメージの保存.	126
スキーマ図の印刷.	126
第 9 章 : クエリとパッケージ.	128
クエリとパッケージの概要.	128
クエリツール.	129
パッケージツール.	129
クエリおよびパッケージのメンテナンス.	130
クエリグループ.	131
クエリグループの追加.	131
クエリグループの編集.	131
クエリグループの削除.	131
汎用クエリ.	132
汎用クエリの追加.	132
汎用クエリの絞り込み.	133
クエリ結果の表示.	138
クエリの影響の表示.	138
クエリの削除.	138

カスタムクエリ.....	139
カスタムクエリ用 SQL 構文.....	139
SQL の検証.....	140
カスタムクエリの追加.....	140
カスタムクエリの編集.....	141
パッケージ.....	141
表示パッケージ.....	141
更新パッケージ.....	142
パッケージの追加.....	142
パッケージの編集.....	143
クエリ変更後のパッケージの更新.....	144
パッケージの削除.....	144
結合クエリの指定.....	144
第 10 章: タイムライン.....	145
概要.....	145
ガイドライン.....	146
例.....	146
レコードバージョン.....	147
レコードバージョンの例.....	148
タイムラインの粒度.....	149
履歴と状態管理.....	150
タイムライン適用ルール.....	150
有効期間の計算.....	150
ルール 1. 有効期間を指定せずにレコードを追加する.....	151
ルール 2. 有効期間を指定してレコードを追加する.....	151
ルール 3. 有効期間のレコードバージョンを追加する.....	152
ルール 4. ある有効期間と重なり、開始日が拡張されたレコードバージョンを追加する.....	152
ルール 5. ある有効期間と重なり、終了日が早まったレコードバージョンを追加する.....	153
ルール 6. 1 つの有効期間内に含まれるレコードバージョンを追加する.....	153
ルール 7. 1 つの有効期間を含むレコードバージョンを追加する.....	154
ルール 8. 有効期間が連続していないレコードバージョンを追加する.....	154
ルール 9. 有効期間内に保留状態でレコードバージョンを追加する.....	155
ルール 10. レコードバージョンがロックされているときにレコードバージョンを追加する.....	155
ルール 11. バージョンが保留状態のときにレコードバージョンを追加する.....	156
ルール 12. 連続したベースオブジェクトでレコードバージョンを削除または更新する.....	157
ルール 13. データを更新する.....	157
ルール 14. 有効期間を更新する.....	158
ルール 15. 有効期間を追加する.....	158
ベースオブジェクトのタイムラインの設定.....	159
手順 1. Hub コンソールでタイムラインを有効にする.....	159
手順 2. プロパティファイルを設定する.....	159
バッチジョブでの複数のレコードバージョンのロード.....	160

複数のレコードバージョンをロードするロードバッチの設定.	160
複数のレコードバージョンのバッチロード例.	161
レコードバージョンの有効期間の編集.	161
レコードバージョンの有効期間の延長.	162
レコードバージョンの有効期間の短縮.	163
レコードバージョンの追加.	164
レコードバージョンの追加例.	164
レコード内のデータの更新.	165
レコードデータ更新の例.	165
リレーションの更新.	165
リレーションレコードのカスタムカラムの更新の例.	165
リレーションレコードのシステムカラムの更新例.	166
リレーションの終了.	168
リレーションの終了の例.	168
リレーション期間の削除.	169
リレーションの削除の例.	169
すべてのリレーション期間の削除.	170
すべてのリレーション期間の削除の例.	170
タイムライン抽出の使用.	170
タイムライン抽出のプロパティの設定.	172

第 11 章 : 状態管理および BPM ワークフローツール. 173

状態管理と BPM ワークフローツールの概要.	173
例.	174
MDM Hub での状態管理.	174
レコードの状態.	174
保留中のレコードの保護.	176
データのロードのルール.	177
レコードの状態およびベースオブジェクトレコード値の存続性.	177
BPM ワークフローツール.	178
ActiveVOS 用に MDM Hub を設定する.	178
状態管理の有効化.	179
ワークフローエンジンの追加.	179
プライマリワークフローアダプタとセカンダリワークフローアダプタの設定.	180
ユーザーロールの設定.	180
保留中のレコードに対する一致の有効化.	183
状態遷移のメッセージトリガ.	184
状態遷移のメッセージトリガの有効化.	184
レコードの昇格.	184
データスチュワードツールでのレコードの昇格.	185
バッチビューアを使用した昇格バッチジョブの設定.	185
バッチグループツールを使用した昇格バッチジョブの設定.	186

第 12 章 : データ暗号化	187
データ暗号化の概要	187
データ暗号化アーキテクチャ	187
データ暗号化の制限	188
データ暗号化ユーティリティ	188
データ暗号化の設定	189
手順 1. DataEncryptor インタフェースの実装	189
手順 2. データ暗号化プロパティファイルの設定	190
手順 3. Hub サーバーのデータ暗号化の設定	190
手順 4. プロセスサーバーのデータ暗号化の設定	191
サービス統合フレームワーク API 要求および応答	191
サンプルのデータ暗号化プロパティファイル	192
第 13 章 : 階層	194
階層の概要	194
階層の例	195
階層の設定について	195
作業を開始する前に	196
設定手順の概要	196
階層マネージャ用のデータの準備	196
階層マネージャ用にデータを準備する方法の例	197
シナリオ	197
方法論	197
HM リポジトリベースオブジェクトの作成	201
デフォルトのエンティティアイコンのアップロード	202
エンティティアイコンの設定	202
エンティティアイコンの追加	202
エンティティアイコンの変更	203
エンティティアイコンの削除	203
エンティティ	203
エンティティベースオブジェクト	203
エンティティベースオブジェクトの例	204
エンティティベースオブジェクトの作成	205
ベースオブジェクトのエンティティベースオブジェクトへの変換	207
エンティティタイプ	207
エンティティタイプの例	209
エンティティタイプの作成	210
エンティティタイプの編集	210
エンティティタイプの削除	211
エンティティの表示オプション	211
エンティティベースオブジェクトをベースオブジェクトに戻す	211
階層タイプ	212

階層の追加.	212
階層の削除.	212
リレーションオブジェクト.	213
リレーションベースオブジェクト.	213
リレーションベースオブジェクトの作成.	215
ベースオブジェクトのリレーションベースオブジェクトへの変換.	215
リレーションベースオブジェクトをベースオブジェクトに戻す.	216
外部キーリレーションベースオブジェクト.	217
外部キーリレーションベースオブジェクトの作成.	217
リレーションタイプ.	217
リレーションタイプの例.	219
リレーションタイプの作成.	220
リレーションタイプの編集.	221
リレーションタイプの削除.	221
パッケージ.	221
階層マネージャデータの設定.	222
エンティティ、リレーション、およびFK リレーションオブジェクトのパッケージの作成.	222
エンティティタイプまたはリレーションタイプへのパッケージの割り当て.	224
プロファイルの概要.	225
プロファイルの追加.	226
プロファイルの編集.	226
プロファイルの検証.	226
プロファイルのコピー.	227
プロファイルの削除.	227
プロファイルからのリレーションタイプの削除.	227
プロファイルからのエンティティタイプの削除.	228
エンティティタイプおよびリレーションタイプへのパッケージの割り当て.	228

第 14 章 : 階層マネージャのチュートリアル. 229

階層の設定の概要.	229
チュートリアル例について.	231
外部キーリレーション.	231
手順 1. Product エンティティベースオブジェクトの作成.	232
手順 2. エンティティタイプの作成.	232
製品エンティティタイプの作成.	235
製品グループのエンティティタイプの作成.	235
手順 3. 製品リレーションオブジェクトの作成.	236
手順 4. リレーションタイプの作成.	237
[製品グループは製品グループの親です] リレーションタイプの作成.	238
[製品グループは製品の親です] リレーションタイプの作成.	239
手順 5. 階層タイプの作成.	240
手順 6. 階層プロファイルへのリレーションタイプの追加.	240
手順 7. パッケージの作成.	241

製品エンティティオブジェクトのパッケージの作成.	241
製品リレーションのパッケージの作成.	242
手順 8. パッケージの割り当て.	243
[PKG Product] パッケージを製品エンティティタイプに割り当てる.	243
[PKG Product] パッケージを製品グループのエンティティタイプに割り当てる.	244
[PKG 製品リレーション] パッケージを [製品グループは製品グループの親です] リレーシ ョンタイプに割り当てる.	244
[PKG 製品リレーション] パッケージを [製品グループは製品の親です] リレーションに割 り当てる.	246
手順 9. 階層マネージャにおけるデータの表示項目の設定.	246
製品エンティティタイプのラベルの設定.	247
製品グループエンティティタイプのラベルの設定.	248
製品エンティティタイプのツールチップテキストの設定.	248
製品グループエンティティタイプのツールチップテキストの設定.	249
[製品グループは製品グループの親です] リレーションタイプのツールチップテキストの設定.	250
[製品グループは製品の親です] リレーションタイプのツールチップテキストの設定.	251
エンティティのリストの設定.	251
リレーションのリストの設定.	252
エンティティ検索フィールドの設定.	253
リレーション検索フィールドの設定.	254
エンティティ検索結果の設定.	256
リレーション検索結果の設定.	257
Hub コンソール階層マネージャのエンティティの詳細の設定.	258
IDD 階層マネージャのエンティティの詳細の設定.	259
リレーションの詳細の設定.	259
編集可能なエンティティフィールドの設定.	260
リレーションフィールドの編集について.	261
エンティティ作成フィールドの設定.	261
リレーション作成フィールドの設定.	262
階層管理.	264

第 IV 部 : データフローの設定. 265

第 15 章 : MDM Hub のプロセス. 266

MDM Hub のプロセスの概要.	266
Informatica MDM Hub のプロセスについて.	266
バッチプロセスの全体的なデータフロー.	266
ベースオブジェクトのレコードの統合ステータス.	267
セルデータの存続性と優先順位.	268
ROWID_OBJECT の存続性.	269
ランディングプロセス.	269
ランディングプロセスについて.	269
ランディングプロセスの管理.	270

ステージプロセス.....	271
ロードプロセス.....	272
ロードプロセスについて.....	272
ロードプロセスに関連付けられているテーブル.....	272
初期データロードおよび増分ロード.....	273
信頼設定および検証ルール.....	273
ロードプロセスのランタイム実行フロー.....	274
ロードプロセスに関するその他の考慮事項.....	278
トークン化プロセス.....	279
一致トークンおよび一致キー.....	279
一致キーテーブル.....	279
一致キーの例.....	280
あいまい一致ベースオブジェクトにのみ適用されるトークン化プロセス.....	280
トークン化プロセスの主要な概念.....	281
トークン化およびマージプロセスのパフォーマンスの最適化.....	282
一致キーテーブルのクリーンアップ.....	283
一致プロセス.....	286
一致ルール.....	287
完全一致およびあいまい一致ベースオブジェクト.....	287
一致プロセスで使用されるサポートテーブル.....	288
ポピュレーションセット.....	288
重複データの一致.....	288
一致グループの構築と遷移一致.....	289
手動統合の最大一致数.....	289
外部一致ジョブ.....	289
分散型プロセスサーバー.....	289
アプリケーションサーバーまたはデータベースサーバーの障害の処理.....	289
統合プロセス.....	290
統合プロセスについて.....	290
統合オプション.....	291
統合とワークフロー統合.....	292
パブリッシュプロセス.....	292
Informatica MDM Hub でサポートされている JMS モデル.....	292
調整済みデータをアウトバウンド配信するためのパブリッシュプロセス.....	292
パブリッシュプロセスのメッセージトリガ.....	293
送信 JMS メッセージキュー.....	293
ORS 固有の XML メッセージスキーマ.....	293
パブリッシュプロセスのランタイムフロー.....	294
第 16 章: ランディングプロセスの設定.....	295
ランディングプロセスの設定の概要.....	295
ソースシステムの設定.....	295
ソースシステムの概要.....	295

システムと信頼ツールの起動.	296
ソースシステムのプロパティ.	297
ソースシステムの追加.	297
ソースシステムのプロパティの編集.	298
ソースシステムの削除.	298
ランディングテーブルの設定.	298
ランディングテーブルについて.	299
ランディングテーブルのカラム.	299
ランディングテーブルのプロパティ.	299
ランディングテーブルの追加.	300
ランディングテーブルのプロパティの編集.	300
ランディングテーブルの削除.	301
第 17 章 : MDM Hub ステージング.	302
MDM Hub ステージングの概要.	302
MDM Hub ステージングテーブル.	303
MDM Hub ステージングプロセステーブル.	303
ステージングテーブルのプロパティ.	304
マッピングのプロパティ.	305
MDM Hub ステージングの要件.	305
ステージングテーブルを追加する.	305
ランディングテーブルとステージングテーブル間のカラムのマッピング.	308
クレンジングされたデータとパススルーされた（変更なし）データ.	309
マッピングツールの起動.	310
マッピングの作成.	311
プライマリキーのマッピング.	312
マッピングカラム.	314
マッピングのレコードのフィルタリング.	315
クレンジング関数を使用するマッピングの保持.	316
行 ID によるロード.	316
監査証跡と差分検出を設定する.	317
ステージングテーブルの監査証跡の設定.	317
ステージングテーブルの差分検出の設定.	318
ステージングテーブルの管理.	321
ステージングテーブルのプロパティの変更.	321
ステージングテーブルのソースシステムへの移動.	321
ステージングテーブルの削除.	322
マッピングの管理.	322
マッピングのプロパティの編集.	322
マッピングのコピー.	322
スキーマへの移動.	323
マッピングのテスト.	323
マッピングの削除.	323

第 18 章 : 物理削除の検出	324
物理削除の検出の概要	324
物理削除の検出のタイプ	325
ベースオブジェクトの削除フラグ値	325
信頼および検証ルール	325
物理削除の検出テーブル	326
物理削除の検出の設定	327
物理削除の検出用のプライマリキーカラムの指定	330
直接削除	331
直接削除フラグ設定用の物理削除の検出設定	331
終了日による直接削除用の物理削除の検出の設定	332
終了日コードのある直接削除の例	332
直接削除用の物理削除の検出の例	332
合意削除	334
合意削除フラグ設定用の物理削除の検出設定	334
終了日による合意削除用の物理削除の検出の設定	335
終了日コードのある合意削除の例	336
合意削除用の物理削除の検出の例	336
ユーザーイグジット内での物理削除の検出の使用	339
 第 19 章 : データクレンジングの設定	 341
データクレンジングの設定の概要	341
MDM Hub でデータクレンジングを設定する	341
プロセスサーバーでデータクレンジングを設定する	342
クレンジング操作のモード	342
分散型データクレンジング	342
クレンジング要求	343
プロセスサーバーツールの起動	343
プロセスサーバーのプロパティ	343
プロセスサーバーの追加	345
プロセスサーバーの保護された通信の有効化	345
プロセスサーバープロパティの編集	346
プロセスサーバーの削除	346
プロセスサーバーの設定のテスト	346
クレンジング関数の設定	347
クレンジング関数ツールの起動	347
クレンジング関数のタイプ	348
クレンジング関数のプロパティ	348
クレンジング関数の設定の概要	348
ユーザークレンジングライブラリの設定	348
Java クレンジングライブラリの設定	349
正規表現関数の追加	350

グラフ関数を設定する.....	350
関数のテスト.....	355
クレンジング関数での条件の使用.....	355
条件付き実行コンポーネントについて.....	356
条件付き実行コンポーネントを使用する状況.....	356
条件付き実行コンポーネントの追加.....	356
クレンジングリストの設定.....	356
クレンジングリストについて.....	357
クレンジングリストの追加.....	357
クレンジングリストのプロパティ.....	357
クレンジングリストのプロパティの編集.....	359

第 20 章 : ロードプロセスの設定..... 362

概要.....	362
作業を開始する前に.....	362
データのロードに関する設定タスク.....	363
ステージングテーブルの設定.....	363
ステージングテーブルのカラム.....	363
ソースシステムキーを保持する.....	365
最高予約キーの指定.....	365
最高予約キーの例.....	365
セルの更新の有効化.....	366
ステージングテーブルのカラムのプロパティ.....	366
ステージングテーブルのプロパティの変更.....	368
外部キーカラムのルックアップ.....	369
初期データロードの設定.....	370
ソースシステムの信頼の設定.....	370
信頼について.....	371
信頼のプロパティ.....	372
信頼の値の設定に関する考慮事項.....	373
カラムの信頼.....	374
検証ルールの設定.....	376
検証ルールの概要.....	376
カラムに対する検証ルールの有効化.....	377
[検証ルール] ノードへの移動.....	378
検証ルールのプロパティ.....	378
検証ルールの追加.....	381
検証ルールのプロパティの編集.....	382
検証ルールの順序の変更.....	382
検証ルールの削除.....	382

第 21 章 : 一致プロセスの設定..... 384

作業を開始する前に.....	384
----------------	-----

一致プロセスの設定タスク.....	384
データの把握.....	384
一致プロセスに関連するベースオブジェクトのプロパティ.....	385
一致ルールを定義するための設定手順.....	385
インターナショナルデータを含むベースオブジェクトの設定.....	385
分散一致設定.....	385
データロードの設定.....	385
[一致/マージ設定の詳細] ダイアログへの移動.....	386
ベースオブジェクトの一致プロパティの設定.....	387
一致プロパティの設定.....	387
一致プロパティ.....	387
長い ROWID_OBJECT 値のサポート.....	391
関連レコードの一致パスの設定.....	391
一致パス.....	391
外部キーのリレーションとフィルタ.....	391
リレーションベースオブジェクト.....	392
テーブル間パス.....	392
テーブル間パスのベースオブジェクトの例.....	392
ベースオブジェクトの例のカラム.....	393
設定手順の例.....	394
テーブル内パス.....	394
テーブル内パスのベースオブジェクトの例.....	394
ベースオブジェクトの例のカラム.....	395
リレーションベースオブジェクトの作成.....	395
設定手順の例.....	395
[パス] タブへの移動.....	396
一致パスのフィルタの設定.....	396
パスコンポーネントの設定.....	398
表示名.....	398
物理名.....	399
子レコードの欠如を許可する.....	399
制約.....	400
パスコンポーネントの追加.....	400
パスコンポーネントの編集.....	400
パスコンポーネントの削除.....	400
一致カラムの設定.....	401
一致カラムについて.....	401
あいまい一致ベースオブジェクトの一致カラムの設定.....	404
完全一致ベースオブジェクトの一致カラムの設定.....	407
一致ルールセット.....	409
一致ルールセットのプロパティ.....	410
[一致ルールセット] タブへの移動.....	412

一致ルールセットの追加.	413
一致ルールセットのプロパティの編集.	413
一致ルールセットの名前の変更.	413
一致ルールセットの削除.	414
一致ルールセットの一致カラムルールの設定.	414
一致カラムルールを設定するための前提条件.	414
完全一致ベースオブジェクトとあいまい一致ベースオブジェクトの一致カラムルールの違い.	414
一致レコードの統合オプションの指定.	415
あいまい一致ベースオブジェクトのみに適用される一致ルールのプロパティ.	415
一致ルールの一致カラムのプロパティ.	423
一致カラムルールの完全一致カラムに対する要件.	430
カラムの一致ルールを設定するためのコマンドボタン.	430
一致カラムルールの追加.	431
一致カラムルールの編集.	432
一致カラムルールの削除.	433
一致カラムルールの実行シーケンスの変更.	433
一致カラムルールの統合オプションの指定.	434
カラムの一致ウェイトの設定.	434
カラムに対するセグメント一致の設定.	434
プライマリキーの一致ルールの設定.	435
プライマリキーの一致ルールについて.	435
プライマリキーの一致ルールの追加.	436
プライマリキーの一致ルールの編集.	436
プライマリキーの一致ルールの削除.	437
一致キーの分布の調査.	437
一致キーの分布について.	437
[一致キーの分布] タブへの移動.	438
[一致キーの分布] タブのコンポーネント.	438
一致キーのフィルタリング.	439
マッチプロセスからのレコードの除外.	440
近接検索.	440
近接検索の設定.	441
軽量一致.	442
第 22 章 : 一致ルールの設定例.	444
一致ルールの設定例の概要.	444
一致ルールの設定シナリオ.	445
一致ルールの設定.	446
手順 1. データの確認.	446
手順 2. 一致ジョブのベースオブジェクトの特定.	447
手順 3. 一致プロパティの設定.	447
一致プロパティの設定.	447
手順 4. 一致パスの定義.	448

一致パスコンポーネントの追加.	448
手順 5.一致カラムの定義.	453
一致カラムの定義.	453
手順 6.一致ルールセットの定義.	458
一致ルールセットの定義.	459
手順 7.一致ルールの追加.	460
一致ルールの追加.	460
手順 8.一致ルールのマージオプションの設定.	463
一致ルールを自動マージ一致ルールとして設定.	463
手順 9.一致プロパティの確認.	464
一致プロパティの確認.	465
手順 10.一致ルールのテスト.	466
第 23 章 : Elasticsearch による検索.	468
Elasticsearch による検索の概要.	468
Elasticsearch アーキテクチャによる検索.	468
検索のインストールおよび設定.	469
手順 1.Elasticsearch のインストールとセットアップ.	470
インストール前のタスクの完了.	470
Elasticsearch のインストール.	471
Elasticsearch Java 仮想マシン (JVM) の設定.	471
Elasticsearch プロパティファイルの設定.	472
Elasticsearch の保護 (オプション)	473
分析プラグインのインストール.	473
ストップワード、シノニム、文字マッピングの設定.	474
Elasticsearch の起動.	475
アップグレード Elasticsearch.	475
手順 2. 検索のための MDM Hub プロパティの設定.	475
Hub サーバーのプロパティの設定.	475
プロセスサーバーのプロパティの設定.	477
手順 3. プロビジョニングツールを使用した検索の設定.	479
Elasticsearch クラスタの設定.	479
エラスティック検索のカスタムインデックス設定の作成 (オプション)	481
検索可能なフィールドの設定.	483
検索またはクエリ結果の表示の設定.	487
類似レコードを表示するレイアウトの設定 (オプション)	488
手順 4.オペレーショナル参照ストアの検証.	490
手順 5. 検索データのインデックス処理.	490
キースタ、トラストストア、および証明書の作成 (オプション)	491
第 24 章 : 統合プロセスの設定.	492
統合プロセスの設定の概要.	492
統合設定.	492

不変行 ID オブジェクト.....	492
個別システム.....	493
親のマージ解除時に子をマージ解除（カスケードマージ解除）.....	493
統合設定の変更.....	496

第 25 章：保留中の制御テーブル..... 497

第 26 章：パブリッシュプロセスの設定..... 498

パブリッシュプロセスの概要.....	498
パブリッシュプロセスの設定手順.....	499
メッセージキューツールの起動.....	499
グローバルメッセージキュー設定の設定.....	499
メッセージキューサーバーの設定.....	500
メッセージキューサーバーのプロパティ.....	500
メッセージキューサーバーの追加.....	501
メッセージキューサーバーのプロパティの編集.....	501
メッセージキューサーバーの削除.....	502
送信メッセージキューの設定.....	502
メッセージキューについて.....	502
メッセージキューのプロパティ.....	502
メッセージキューサーバーへのメッセージキューの追加.....	502
メッセージキューのプロパティの編集.....	503
メッセージキューの削除.....	503
JMS メッセージの並行処理の設定.....	504
JMS セキュリティの設定.....	504
メッセージトリガの設定.....	504
メッセージトリガのイベントのタイプ.....	505
メッセージトリガのベストプラクティス.....	506
メッセージトリガシステムのプロパティ.....	507
メッセージトリガの追加.....	507
メッセージトリガの編集.....	508
メッセージトリガの削除.....	508
メッセージキューのポーリングの無効化.....	509
JMS メッセージの XML リファレンス.....	509
ORS 固有の XML メッセージスキーマの生成.....	509
XML メッセージの要素.....	510
メッセージのフィルタリング.....	511
XML メッセージの例.....	511
従来の JMS メッセージの XML リファレンス.....	523
従来の XML のメッセージフィールド.....	523
従来の XML のメッセージのフィルタリング.....	524
従来の XML のメッセージの例.....	524

第 V 部 : Informatica MDM Hub のプロセスの実行	536
第 27 章 : バッチジョブの使用	537
バッチジョブの使用の概要	537
バッチジョブのスレッド設定	538
マルチスレッドバッチジョブのプロセス	538
マルチスレッドバッチジョブの例	538
マルチスレッドバッチのパフォーマンス	539
マルチスレッドバッチジョブのプロパティ	539
バッチジョブの起動	539
バッチジョブが使用するサポートテーブル	540
バッチジョブの順次実行	540
バッチジョブの実行前のランディングテーブルの生成	540
一致ジョブおよび後続の統合ジョブ	540
最初に親テーブルのデータをロードする	541
外部キーリレーションを持つオブジェクトへのデータのロード	541
バッチジョブの作業に関するベストプラクティス	541
Oracle 環境での統計収集の並列度の制限	541
バッチジョブの作成	542
自動的に作成されるバッチジョブ	542
変更の発生時に作成されるバッチジョブ	543
情報提供用バッチジョブ (Hub コンソールで実行されないバッチジョブ)	543
プロセスサーバーの設定	544
バッチビューアツールを使用したバッチジョブの実行	544
バッチビューアツール	544
バッチビューアツールの起動	544
テーブル、データ、またはプロシージャタイプ別のグループ化	544
バッチジョブの手動実行	545
ジョブ実行ログの表示	547
ジョブ実行履歴のクリア	552
バッチグループツールを使用したバッチジョブの実行	552
バッチグループについて	552
バッチグループツールの開始	553
バッチグループの設定	553
バッチグループリストの更新	557
バッチグループツールを使用したバッチグループの実行	557
ステータスによる実行ログのフィルタリング	561
バッチグループの削除	561
バッチジョブのリファレンス	561
バッチジョブのアルファベット順リスト	562
一致しないレコードを一意とする	563
自動一致とマージジョブ	563

自動マージジョブ	564
バッチマージ解除	565
BVT スナップショットジョブ	565
外部一致ジョブ	566
一致トークンの生成ジョブ	569
スマート検索データジョブの初期インデックス処理	571
キー一致ジョブ	571
ロードジョブ	572
手動マージジョブ	576
手動マージ解除ジョブ	577
一致ジョブ	577
一致分析ジョブ	579
重複データの一致ジョブ	581
複数マージジョブ	581
昇格ジョブ	581
ベースオブジェクトの再計算ジョブ	583
BVT の再計算ジョブ	583
一致テーブルのリセットジョブ	583
再検証ジョブ	583
ステージジョブ	584
同期ジョブ	584
第 28 章 : ユーザー出口	586
ユーザーイグジットの概要	586
ユーザー出口の処理	587
ユーザー出口 JAR ファイル	588
ユーザーイグジット JAR ファイルの実装	588
MDM Hub へのユーザーイグジットのアップロード	588
MDM Hub からのユーザーイグジットの削除	589
UserExitContext クラス	589
ステージプロセスユーザーイグジット	591
ランディング後ユーザー出口	591
ステージング前ユーザー出口	592
ステージング後ユーザー出口	593
ロードプロセスユーザーイグジット	594
ロード後ユーザーイグジット	594
一致プロセスユーザーイグジット	595
マッチ前ユーザー出口	596
マッチ後ユーザー出口	596
マージプロセスユーザーイグジット	597
マージ後ユーザーイグジット	598
アンマージプロセスユーザー出口	598
アンマージ前ユーザーイグジット	599

アンマージ後ユーザーイグジット.	600
タスク管理ユーザー出口.	601
AssignTasks ユーザー出口インタフェース.	601
GetAssignableUsersForTask ユーザー出口インタフェース.	601
ユーザー出口内でのサービス統合フレームワークの API の使用.	602
サービス統合フレームワークの API を呼び出すためのユーザーイグジットの作成.	602
ユーザーイグジットの例.	603
サービス統合フレームワークの API.	604
ユーザーイグジットの実装のガイドライン.	605

第 VI 部 : アプリケーションアクセスの設定. 607

第 29 章 : ORS 固有の API. 608

ORS 固有の API の概要.	608
パフォーマンスに関する考慮事項.	609
サポートされているリポジトリオブジェクト.	609
ORS 固有の SIF API プロパティ.	609
リポジトリオブジェクトのステータス.	610
アーカイブテーブル.	610
ORS 固有の SIF API の生成とデプロイ.	611
ORS 固有の SIF API の名前変更.	611
ORS 固有のクライアント JAR ファイルのダウンロード.	611
ORS 固有のクライアント JAR ファイルと SIF SDK の併用.	612
ORS 固有の SIF API の削除.	612

第 30 章 : ORS 固有のメッセージスキーマ. 613

ORS 固有のメッセージスキーマの概要.	613
JMS イベントスキーママネージャツールについて.	613
JMS イベントスキーママネージャツールの起動.	614
SIF マネージャツールの起動.	614
ORS 固有のスキーマの生成およびデプロイ.	615
XSD ファイルのダウンロード.	615
非同期オブジェクトの検出.	615
非同期オブジェクトの自動検索.	616

第 31 章 : 登録済みのカスタムコードの表示. 617

概要.	617
ユーザーオブジェクト.	617
ユーザーオブジェクトレジストリツールの起動.	618
ユーザー出口の表示.	618
ユーザー出口について.	618
ユーザー出口の表示.	618
カスタム Java クレンジング関数の表示.	618

カスタム Java クレンジング関数について.	619
カスタム Java クレンジング関数の登録方法.	619
登録されているカスタム Java クレンジング関数の表示.	619
カスタムボタン関数の表示.	619
カスタムボタン関数について.	619
カスタムボタン関数の登録方法.	619
登録されているカスタムボタン関数の表示.	620

付録 A : MDM Hub のプロパティ. 621

MDM Hub のプロパティの概要.	621
Hub サーバーのプロパティ.	621
Hub サーバーのプロパティファイルのサンプル.	641
プロセスサーバーのプロパティ.	644
プロセスサーバーのプロパティファイルのサンプル.	652
オペレーショナルリファレンスストアのプロパティ.	653

付録 B : 設定の詳細の表示. 655

設定の詳細の表示の概要.	655
エンタープライズマネージャの開始.	655
エンタープライズマネージャのプロパティ.	656
C_REPOS_DB_RELEASE テーブル.	656
環境レポート.	658
MDM Hub 環境レポートの保存.	658
エンタープライズマネージャでのバージョン履歴の表示.	658
アプリケーションサーバーのログの使用.	659
アプリケーションサーバーのログレベル.	659
ログファイルのローリング.	659
アプリケーションサーバーのログの設定.	660
クライアントに対する Hub コンソールログの使用.	661

付録 C : 行レベルのロック. 662

行レベルのロックの概要.	662
行レベルのロックについて.	662
デフォルトの動作.	662
ロックのタイプ.	663
行レベルのロックの使用に関する考慮事項.	663
行レベルのロックの設定.	663
ORS での行レベルのロックの有効化.	663
ロック待機時間の設定.	664
SIF 要求とバッチプロセスの間のロックの相互作用.	664
API とバッチの相互運用性がある場合の対話.	664
API とバッチの相互運用性が無効になっている場合の対話.	665

付録 D : MDM Hub ログイン	666
MDM Hub のログインの概要	666
ログインの設定	667
Hub コンソールのログ	667
Hub Server ログ	667
プロセスサーバーログ	668
エンティティ 360 ログ	668
プロビジョニングツールのログ	668
付録 E : テーブルのパーティション化	670
テーブルのパーティション化のサポート	670
付録 F : 製品使用ツールキットを使用した MDM 環境情報の収集	671
製品使用ツールキットを使用した MDM 環境情報の収集の概要	671
システム設定情報	672
MDM Hub 環境情報	672
Hub サーバーでの MDM Hub データ収集の有効化	673
プロセスサーバーでの MDM Hub データ収集の有効化	673
Hub サーバーでの MDM Hub データ収集の無効化	674
プロセスサーバーでの MDM Hub データ収集の無効化	674
付録 G : 用語集	675
索引	705

序文

Informatica MDM Hub システムを設定するには、Informatica^(R) *Multidomain MDM* の設定ガイドの指示に従います。データモデルを構築、データフローを設定、プロセスを実行およびアプリケーションアクセスを設定するために、MDM Hub コンソールツールの使用方法を学習します。

このガイドは、『*Multidomain MDM の概要ガイド*』をすでに読み、MDM Hub のアーキテクチャと主な概念を基本的に理解していることを前提として作成されています。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica Network

Informatica Network は、Informatica ナレッジベースや Informatica グローバルカスタマサポートなど、多くのリソースへの入口です。Informatica Network を利用するには、<https://network.informatica.com> にアクセスしてください。

Informatica Network メンバーは、次のオプションを利用できます。

- ナレッジベースで製品リソースを検索できます。
- 製品の提供情報を表示できます。
- サポートケースを作成して確認できます。
- 最寄りの Informatica ユーザーグループネットワークを検索して、他のユーザーと共同作業を行えます。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、<https://search.informatica.com> にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム (KB_Feedback@informatica.com) です。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム (infa_documentation@informatica.com) までご連絡ください。

Informatica 製品可用性マトリックス

製品可用性マトリックス (PAM) には、製品リリースでサポートされるオペレーティングシステム、データベースなどのデータソースおよびターゲットが示されています。Informatica PAM は、<https://network.informatica.com/community/informatica-network/product-availability-matrices> で参照できます。

Informatica Velocity

Informatica Velocity は、Informatica プロフェッショナルサービスが開発したヒントとベストプラクティスのコレクションで、多数のデータ管理プロジェクトから得た実体験に基づいています。Informatica Velocity には、世界中の組織と連携してデータ管理ソリューションを計画、開発、デプロイ、管理する Informatica コンサルタントによる集合知を表しています。

Informatica Velocity リソースには、<http://velocity.informatica.com> からアクセスしてください。Informatica Velocity についての質問、コメント、またはアイデアがある場合は、ips@informatica.com から Informatica プロフェッショナルサービスにお問い合わせください。

Informatica Marketplace

Informatica Marketplace は、お使いの Informatica 製品を拡張したり強化したりするソリューションを検索できるフォーラムです。Marketplace で、Informatica デベロッパーやパートナーからの多数のソリューションを活用すれば、生産性を向上したり、プロジェクトでの実装時間を短縮したりできます。Informatica Marketplace は、<https://marketplace.informatica.com> からアクセスしてください。

Informatica グローバルカスタマサポート

電話または Informatica Network を介してグローバルカスタマサポートに連絡できます。

各地域の Informatica グローバルカスタマサポートの電話番号は、Informatica Web サイト (<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>) を参照してください。

Informatica Network のオンラインサポートリソースについては、<https://network.informatica.com> のサポートオプションをご確認ください。

パート I: 概要

この部には、以下の章があります。

- [Informatica MDM Hub の管理, 28 ページ](#)
- [MDM Hub コンソールの基本操作, 30 ページ](#)
- [国際化データのサポートの設定, 45 ページ](#)

第 1 章

Informatica MDM Hub の管理

この章では、以下の項目について説明します。

- [Informatica MDM Hub の管理の概要, 28 ページ](#)
- [Informatica MDM Hub の管理の各フェーズ, 28 ページ](#)

Informatica MDM Hub の管理の概要

Informatica MDM Hub 管理者は、Informatica MDM Hub システムの設定に関する主な責任を担います。

管理者は、Informatica MDM Hub に Hub コンソールからアクセスします。Hub コンソールには、Informatica MDM Hub 実装を管理するための一連のツールが用意されています。

Informatica MDM Hub 管理者は Hub コンソールを使用して、以下の作業を実行します。

- Hub ストア内のデータモデルやその他のオブジェクトを構築する。
- Informatica MDM Hub のデータ管理プロセスを設定および実行する。
- Informatica MDM Hub の機能やリソースに対する外部アプリケーションからのアクセスを設定する。
- 実行中の操作を監視する。
- Informatica MDM Hub をトラブルシューティングするためにグローバルカスタマサポートで必要なログを保持する。

Informatica MDM Hub の管理の各フェーズ

管理のフェーズは、お客様の組織の方法に基づいた Informatica MDM Hub の実装によって異なる場合があります。

MDM Hub 管理には、次のフェーズを含めることができます。

1. スタートアップ。MDM Hub のインストールおよび設定。
2. 設定。MDM Hub の機能のビルドおよびテスト。
3. プロダクション。環境のデプロイ、調整、および維持。

スタートアップフェーズ

スタートアップフェーズには次のようなコアの Informatica MDM Hub コンポーネントのインストールと設定が含まれます。Hub ストア、Hub サーバー、プロセスサーバー、およびクレンジングアダプタ。

Hub ストア、Hub サーバー、およびプロセスサーバーのインストール手順については、ご使用のアプリケーションサーバーに対応した『*Multidomain MDM のインストールガイド*』を参照してください。クレンジングアダプタを設定して、サポートされている外部クレンジングエンジンと MDM Hub を統合する手順については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

注: この章の手順は、スタートアップフェーズを完了しており、Informatica MDM Hub 実装の設定を開始できる状態にあることを前提としています。

設定段階

Informatica MDM Hub をインストールしてセットアップしたら、Informatica MDM Hub の機能の設定とテストを開始できます。設定する項目には、Hub ストア内のデータモデルやその他のオブジェクト、データ管理プロセス、外部アプリケーションによるアクセスなどが含まれます。

この段階では、Informatica MDM Hub の機能を構築およびテストするプロセスを、組織で規定された要件に合うまで動的に繰り返し行います。この章では、その大部分で、設定段階に関連するタスクについて説明します。

スキーマの構築と Informatica MDM Hub の設定が適切に完了すると、開発者が Informatica MDM Hub の機能やリソースにアクセスする外部アプリケーションを構築できるようになります。外部アプリケーションの開発手順については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

プロダクションフェーズ

Informatica MDM Hub 実装の設定とテストを適切に行った後、管理者は Informatica MDM Hub をプロダクション環境にデプロイします。

このフェーズでは、Informatica MDM Hub 運用の継続的な管理に加え、実際のビジネスデータの処理を最適化するためのパフォーマンスの調整も行います。

注: タイムリーなトラブルシューティングを可能にするために、MDM Hub 管理者は必要なすべてのログファイルへのアクセス権を提供する必要があります。

第 2 章

MDM Hub コンソールの基本操作

この章では、以下の項目について説明します。

- [概要, 30 ページ](#)
- [MDM Hub コンソールについて, 30 ページ](#)
- [Hub コンソールの起動, 30 ページ](#)
- [シングルサインオンによる MDM Hub コンソールの設定, 32 ページ](#)
- [MDM Hub コンソールでの操作, 33 ページ](#)
- [Informatica MDM Hub ワークベンチおよびツール, 41 ページ](#)

概要

この章では、MDM Hub コンソールを紹介し、Informatica MDM Hub 実装の設定に関連するツールの概要を説明します。

MDM Hub コンソールについて

システム管理者とデータスチュワードは、Hub コンソールという Informatica MDM Hub のユーザーインターフェースを使用して、Informatica MDM Hub の各機能を操作できます。Hub コンソールは各種のツールで構成されています。各ツールを使用して、特定の操作や連携する操作を行うことができます。

注: Hub コンソールで使用できるツールは、それぞれの Informatica ライセンス契約によって異なります。

Hub コンソールの起動

MDM Hub にアクセスするには、HTTP または HTTPS 接続を使用して Hub コンソールを起動します。

Hub コンソールを開始する前に、次の情報があることを確認してください。

- ホスト名と URL のポート番号
- ユーザー名とパスワード

- HTTPS 接続を介して Hub コンソールにアクセスする場合のクライアントマシンの SSL 証明書

1. ブラウザウィンドウを開いて、以下の URL を入力します。

`http://<MDM Hub host: MDM Hub ホスト>:<port number: ポート番号>/cmx/`

[ハブコンソールの起動] ページが表示されます。

2. ユーザー名とパスワードを入力し、**[ダウンロード]** をクリックします。

Hub コンソールのダウンロードを起動するために必要な MDM Hub アプリケーション JAR ファイル。

注: MDM Hub アプリケーション JAR ファイルをダウンロードできない場合は、MDM 管理者に連絡してください。管理者は、次のディレクトリから JAR ファイルを配布できます。 <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/resources/hub

3. アプリケーション JAR ファイルを実行します。

注: クライアントマシンに SSL 証明書がなく、HTTPS 接続を介して Hub コンソールにアクセスする場合は、SSL 証明書をインストールする必要があります。SSL 証明書をインストールするには、次の手順を実行します。

- 次のコマンドを実行して、ローカルのクライアントマシンの Java キーストアに証明書をインポートします。

```
keytool -import -trustcacerts -alias <certificate alias name> -file <certificate alias file> -keystore <local java cacerts keystore location>
```

- 次のコマンドを実行して、証明書を含むトラストストアファイルの場所とパスワードを渡します。

```
java -Djavax.net.ssl.trustStore=<truststore file location> -Djavax.net.ssl.trustStorePassword=<truststore_password> -jar hubConsole.jar
```

デフォルトの cacert ファイルではなく、すべてのカスタム信頼証明書を含む別のトラストストアを使用します。アプリケーションサーバーを管理するチームから証明書を取得します。サーバーでは、自己署名証明書またはセキュリティ証明書が使用されている可能性があります。サーバーでバージョンを変更した場合にのみ、.jar ファイルをダウンロードしてください。jar ファイルをダウンロードをした場合は、毎回同じコマンドを使用してこのファイルを起動します。

4. アプリケーションの最大メモリ割り当てプールを指定するには、次のコマンドを実行します。

```
java -Xmx<n>G -jar hubConsole.jar
```

<n>は、GB 単位の最大メモリ割り当てです。

[Informatica MDM Hub ログイン] ダイアログボックスが表示されます。

5. ユーザー名とパスワードを入力します。

6. 特定の Hub サーバーノードに接続する場合、またはロードバランサあるいはリバースプロキシサーバーを使用する場合は、[接続プロパティ] フィールドで事前設定された接続パラメータを上書きします。

このパラメータは次の形式で入力します。

`<host name>:<port name>`

ここで、ホスト名とポート名は、Hub サーバーのホスト名とポート名、または使用するロードバランサあるいはリバースプロキシサーバーのホスト名とポート名を使用します。

7. **[OK]** をクリックします。

[データベースの変更] ダイアログボックスが表示されます。

8. ターゲットデータベースを選択します。

ターゲットデータベースは MDM Hub マスターデータベースです。

9. リストから言語を選択して、**[接続]** をクリックします。

Hub コンソールのユーザーインターフェースが、選択した言語で表示されます。Hub コンソールユーザーインターフェースを表示する言語を変更する場合は、言語を選択して Hub コンソールを再起動します。

シングルサインオンによる MDM Hub コンソールの設定

MDM Hub コンソールで、SAML を使用したシングルサインオンを設定できます。

Hub Console のシングルサインオン（SSO）設定を開始する前に、次の情報があることを確認してください。

- Hub Console の IDP URL。
IT チームに IDP URL をリクエストします。
注: IDPMetadata.xml には必要な情報が含まれています。
- アサーションコンシューマサービス（ACS）URL のホスト名とポート番号。
コンソールにアクセスするには、次の形式を使用します。
`http://<host>:<port>/cmx/sso/hub-console/`
シングルサインオン（SSO）操作では、ACS URL を使用して、Security Assertion Markup Language（SAML）応答をサービスプロバイダに送信します。認証が成功すると、SAML アサーションを交換するための ACS URL にリダイレクトされます。

注: MDM Hub コンソールの名前 ID が MDM 外部ユーザーのユーザー名と一致していることを確認します。

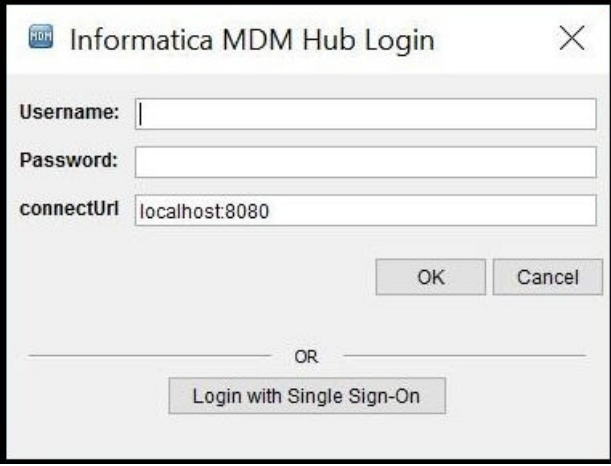
Hub Console でシングルサインオンを有効にし、アプリケーション JAR を実行すると、IDP ログインページにデフォルトで使用可能な SSO オプションが表示されます。

シングルサインオンは、MDM Hub コンソールのログインページで設定することができます。

MDM Hub コンソールで SSO を使用してユーザーを認証するには、`cmxserver.properties` ファイルで次のプロパティを設定します。

```
hubconsole.show.login.with.sso=true
```

次の画像は、SSO オプションを使用した MDM Hub コンソールのログインページを示しています。



MDM と Azure Active Directory を IDP として使用した SAML シングルサインオンの設定の詳細については、次の記事を参照してください。

https://knowledge.informatica.com/s/article/Using-customloginutility-to-Configure-SAML-SSO?language=en_US&type=external

MDM Hub コンソールでの操作

Hub コンソールは、Informatica MDM Hub 実装の設定および管理に使用できるツールの集まりです。
各ツールで、Informatica MDM Hub 実装の特定の領域に焦点を合わせることができます。

プロセスビューとワークベンチビューの切り替え

Informatica MDM Hub では、ツールは 2 つの方法でグループ化されます。

ビュー	説明
ワークベンチ別	類似するツールをワークベンチ別にグループ化します。ワークベンチとは、関連するツールの論理的な集合です。
プロセス別	各ツールを、タスクを完了するために必要なツールおよび手順に従って論理ワークフロー別にグループ化します。

Hub コンソールウィンドウの左側にあるタブをクリックすると、**プロセスビュー**と**ワークベンチビュー**を切り替えることができます。

注: Informatica MDM Hub にログインすると、Informatica MDM Hub セキュリティ管理者が使用を許可したツールが含まれているワークベンチおよびプロセスのみが表示されます。

ワークベンチビュー

ワークベンチ別にツールを表示する手順

- Hub コンソールウィンドウの左側にある **【ワークベンチ】** タブをクリックします。
Hub コンソールでは、**【ワークベンチ】** タブに、使用可能なワークベンチのリストが表示されます。**【ワークベンチ】** ビューには、類似する機能別に Hub コンソールのツールが表示されます。
ワークベンチ名およびツールの説明は、ツールのグループ化方法と同様に、メタデータに基づきます。ツールのグループ化はカスタマイズできます。

プロセスビュー

プロセス別にツールを表示する手順

- Hub コンソールウィンドウの左側にある **【プロセス】** タブをクリックします。
Hub コンソールでは、**【プロセス】** タブに、使用可能なワークベンチのリストが表示されます。ツールは、一般的なシーケンスまたはプロセスに整理されます。
プロセスは、特定のタスクを完了するためのツールの論理シーケンスを示します。同じツールが複数のプロセスに表示されたり、1 つのプロセスに複数回表示されたりすることもあります。

ワークベンチビューでのツールの起動

ワークベンチビューから Hub コンソールを開始します。

- ワークベンチビューで、開始するツールを含んでいるワークベンチを展開します。
- 必要に応じて、ワークベンチのノードを展開してそのワークベンチに関連付けられているツールを表示します。

3. ツールをクリックします。

別のデータベースが必要なツールを選択した場合、Hub コンソールではそれを選択するように求めるメッセージが表示されます。

設定ワークベンチ、データベース、ユーザー、セキュリティプロバイダ、ツールアクセス、メッセージキュー、リポジトリマネージャ、エンタープライズマネージャ、および Workflow Manager のすべてのツールは、MDM Hub マスターデータベースへの接続が必要です。その他のすべてのツールは、オペレーショナル参照ストアへの接続が必要です。

Hub コンソールに選択したツールが表示されます。

メタデータを変更するためのロックの取得

Hub ストア内のメタデータを変更するには、Hub コンソールを介して、リポジトリテーブルに対するロックを取得する必要があります。

Hub コンソール経由でアクセスするツールは、データスチュワードツールを除き、すべて読み取り専用モードになります。これらのツールを使用して Hub ストア内のメタデータを変更するには、リポジトリテーブルに対するロックを取得する必要があります。

Hub ストアに複数の変更を同時に加えるには、排他ユーザーまたは複数ユーザーのロックを取得します。他のユーザーが保持している書き込みロックまたは排他ロックを強制的に解除することができます。

ロックのタイプ

【書き込みロック】 メニューには 2 種類のロックが用意されています。

以下の表に、Hub コンソールでアクセスできるロックのタイプを示します。

ロックのタイプ	説明
排他ロック	排他ロックが有効になっている間は、1 ユーザーだけが基になるオペレーショナル参照ストアを変更でき、その他のユーザーはオペレーショナル参照ストアを変更できません。
書き込みロック	複数のユーザーが基になるメタデータを同時に変更できます。書き込みロックは、MDM Hub マスターデータベースまたはオペレーショナル参照ストアで取得できます。

注: プロダクションモードのオペレーショナル参照ストアではロックを取得できません。オペレーショナル参照ストアがプロダクションモードの場合に書き込みロックを取得しようとすると、ロックを取得できないことを示すメッセージが表示されます。

ロックを必要とするツール

データベースの設定を変更する場合は、MDM Hub マスターデータベースとオペレーショナル参照ストアでツールのロックを取得する必要があります。

MDM Hub マスターデータベースの設定を変更するには、以下のツールのロックを取得する必要があります。

- データベース
- リポジトリマネージャ
- メッセージキュー
- セキュリティプロバイダ
- ツールアクセス

- ユーザー

オペレーショナル参照ストアの設定を変更するには、以下のツールのロックを取得する必要があります。

- バッチグループ
- クレンジング関数
- 階層
- 階層マネージャ
- マッピング
- パッケージ
- プロセスサーバー
- クエリ
- ロール
- スキーママネージャ
- スキーマビューア
- セキュアリソース
- SIF マネージャ
- システムと信頼
- ユーザーとグループ

注: データマネージャ、マージマネージャ、および階層マネージャでは、書き込みロックは必要ありません。これらのツールの詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。監査マネージャでも書き込みロックは必要ありません。

自動ロックの有効期限

Hub コンソールでは、現在の接続に対するロックが 60 秒ごとに更新されます。ユーザーは手動でロックを解除することができます。ユーザーがロックを保持したまま別のデータベースに切り替えると、ロックは自動的に解除されます。Hub コンソールを終了した場合、ロックは 1 分後に期限切れになります。

サーバーキャッシュおよび MDM Hub コンソールのロック

Hub コンソールでロックが有効になっていない場合、Hub サーバーは、パフォーマンスの理由から、メタデータおよびその他の構成設定をキャッシュします。Hub コンソールユーザーが書き込みロックまたは排他ロックを取得すると、キャッシュは無効になります。キャッシュが空になると、Informatica MDM Hub は代わりにこの情報をデータベースから取得します。すべてのロックが解放されると、キャッシュは再び有効になります。

複数の Hub コンソールが同じオペレーショナル参照ストアを使用している場合は、Hub サーバーに対して書き込みロックを行っても、他の Hub Server のキャッシュは無効になりません。

書き込みロックの取得

書き込みロックを取得すれば、同時に複数のユーザーが Hub コンソールでデータの編集を行うことができます。

ただし、書き込みロックでは、それらのユーザーにより同時に同じデータが編集される可能性があります。その場合は、最後に保存された変更が優先されます。

1. **【書き込みロック】** > **【ロックの取得】** の順にクリックします。
 - 他のユーザーがすでにロックを取得している場合は、そのユーザーのログイン名とマシンアドレスが表示されます。
 - オペレーショナル参照ストアがプロダクションモードの場合は、ロックを取得できないことを示すメッセージが表示されます。
 - 正常にロックを取得すると、ツールは読み/書きモードになります。オペレーショナル参照ストアごと、または MDM Hub マスターデータベースで、複数のユーザーが書き込みロックを取得できます。
2. 作業が完了したら、**【書き込みロック】** > **【ロックの解除】** の順にクリックします。

排他ロックの取得

Hub コンソールでは排他ロックを取得できます。

1. **【書き込みロック】** > **【ロックのクリア】** の順にクリックして、他のユーザーが保持している書き込みロックをクリアします。
2. **【書き込みロック】** > **【排他ロックの取得】** の順にクリックします。

オペレーショナル参照ストアがプロダクションモードの場合は、排他ロックを取得できないことを示すメッセージが表示されます。
3. 作業が完了したら、**【書き込みロック】** > **【ロックの解除】** の順にクリックします。

ロックの解除

Hub コンソールではロックを解除できます。

- ▶ **【書き込みロック】** > **【ロックの解除】** の順にクリックします。

ロックのクリア

Hub コンソールではロックをクリアできます。書き込みロックが解除される前に、他のユーザーに対して変更を保存するように警告するメッセージは表示されないため、ロックをクリアするのは必要な場合だけにしてください。

- ▶ **【書き込みロック】** > **【ロックのクリア】** の順にクリックします。

Hub コンソールで、オペレーショナル参照ストアに対するすべてのロックが解除されます。

ターゲットデータベースの変更

Hub コンソールウィンドウの下部にあるステータスバーには、接続しているターゲットデータベースの名前とログインに使用したユーザー名が表示されます。

1. ステータスバーで、データベース名をクリックします。

Hub コンソールに、ターゲットデータベースの選択を求めるプロンプトが表示されます。
2. 接続する MDM Hub マスターデータベースまたはオペレーショナル参照ストアを選択します。

3. **【接続】** をクリックします。

別のユーザーとしてのログイン

Hub コンソールには別のユーザーとしてログインすることもできます。

1. 次のいずれかのオプションを選択します。
 - ステータスバーでユーザー名をクリックします。
 - **【オプション】** > **【再ログイン】** の順にクリックします。
2. 使用したいユーザーアカウントのユーザー名とパスワードを指定します。
3. **【OK】** をクリックします。

ユーザーのパスワードの変更

Hub コンソールに現在ログインしているユーザーのパスワードを変更することができます。

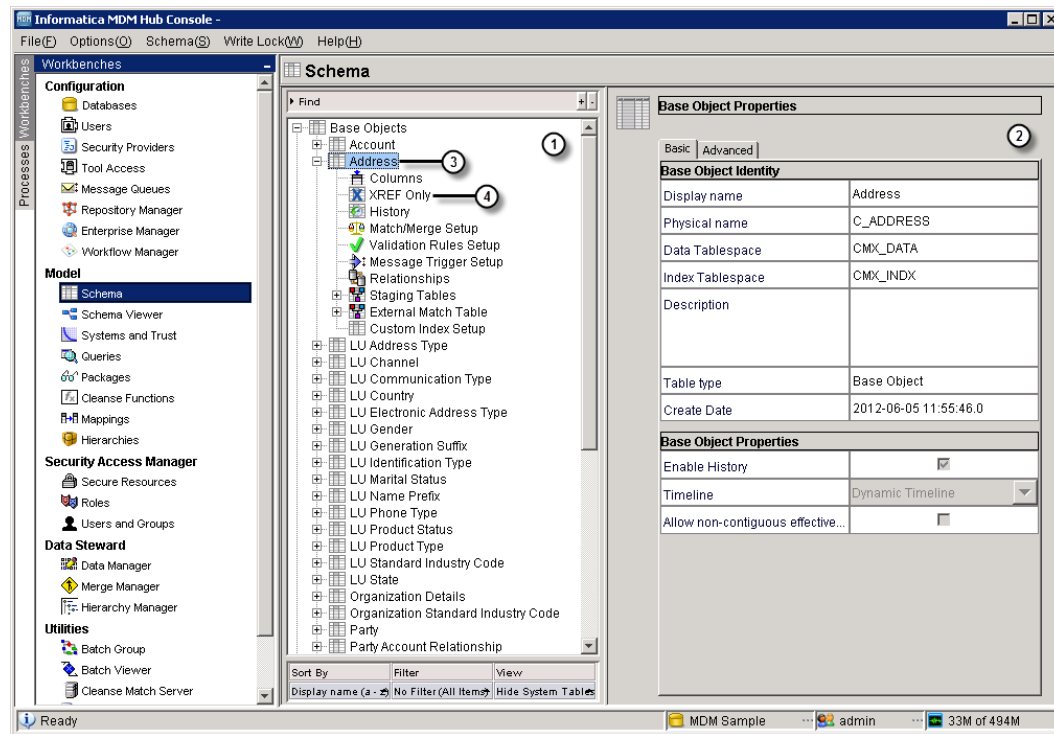
1. **【オプション】** > **【パスワードの変更】** の順にクリックします。
2. 使用するパスワードを指定します。
3. **【OK】** をクリックします。

ナビゲーションペインでのナビゲーションツリーの使用

Hub コンソールのナビゲーションツリーでは、オブジェクトの階層集合を表示および管理できます。

ナビゲーションツリーは、Hub コンソールの**ナビゲーションペイン**にあります。各名前付きオブジェクトは、ナビゲーションツリーでノードとして表されます。他のノードを含むノードは、**親ノード**と呼ばれます。親ノードに属するノードは**子ノード**と呼ばれます。

以下の図は、Hub コンソールインタフェースを示しています。



1. ナビゲーションペイン
2. プロパティペイン
3. 親ノード
4. 子ノード

子ノードの表示および非表示

子ノードを親ノードの下に表示する手順

- 親ノードの隣のプラス (+) 記号をクリックします。

親ノードの下の子ノードを非表示にする手順

- 親ノードの隣のマイナス (-) 記号をクリックします。

表示名によるソート

表示名は、ナビゲーションツリーに表示されるオブジェクトの名前です。ツリーオプション領域で【ソート基準】をクリックして適切なオプションを選択すると、ナビゲーションツリーに表示されるオブジェクトの順序を変更できます。

次のソートオプションから選択します。

- 【表示名 (a-z)】は、ツリー内のオブジェクトを表示名のアルファベット順にソートします。
- 【表示名 (z-a)】は、ツリー内のオブジェクトを表示名のアルファベットの逆順にソートします。

フィルタオプション

ナビゲーションツリーに表示される項目をフィルタリングできます。そのためには、ナビゲーションペイン下部の【フィルタ】領域をクリックし、適切なフィルタオプションを選択します。

フィルタオプションを以下から選択します。

- フィルタなし（全項目）。以前に定義されたすべてのフィルタを削除します。
- 1 項目。ナビゲーションツリーの上に項目を選択するためのドロップダウンリストを表示します。
例えばスキーママネージャでは、[テーブルタイプ] または [テーブル] を選択できます。
[テーブルタイプ] を選択した場合は、下矢印をクリックしてテーブルタイプのリストを表示し、そこからフィルタに使用するものを選択します。
- 複数項目。1 つ以上の項目を選択できます。

項目のフィルタリング

[1 項目] フィルタオプションまたは **[複数項目]** フィルタオプションを選択すると、フィルタリングする項目を選択できます。

例えばスキーママネージャでは、テーブルタイプまたはテーブル名を基に複数のテーブルを選択できます。**[複数項目]** を選択すると、Hub コンソールでナビゲーションツリーの上に **[項目フィルタの定義]** ボタンが表示されます。

1. **[項目フィルタの定義]** ボタンをクリックします。
2. フィルタに含める項目を選択し、**[OK]** をクリックします。

項目の表示の変更

Hub コンソールの特定のツールでは、ナビゲーションツリーの下に **[表示]** 領域または **[表示方法]** 領域が表示されます。

- スキーママネージャでは、ナビゲーションツリーの下に **[表示]** 領域をクリックして該当するコマンドを選択することによって、公開されている Informatica MDM Hub の項目の表示と非表示を切り替えることができます。
例えば、すべてのシステムテーブルを表示することができます。
- マッピングツールでは、マッピング、ステージングテーブル、またはランディングテーブルごとに項目を表示できます。
- パッケージツールでは、パッケージまたはテーブルごとに項目を表示できます。
- ユーザーとグループツールでは、サブグループやサブユーザーを表示できます。
- バッチビューアでは、テーブル、日付、またはプロシージャタイプごとにジョブをグループ化できます。

項目の検索

フィルタがない場合や、**[複数項目]** フィルタが選択されている場合、**[ナビゲーション]** ペインには **[検索]** ボックスが表示されます。**[検索]** ボックスで、完全な名前や名前の一部を基準にして、項目を見つけます。例えば、スキーママネージャで、テーブル名に「Lookup」が含まれるすべてのテーブルを見つけることができます。検出プロセスでは、大文字と小文字が区別されます。

1. **[検索]** ボックスをクリックします。
[検索] ウィンドウが表示されます。
2. 検索する名前、または名前の最初の数文字を入力します。
3. **[F3 - 検索]** ボタンをクリックします。
ツリー内で最初の一致項目が強調表示されます。
4. 次の一致項目に移動するには、**[F3 - 検索]** ボタンを再度クリックします。
5. **[検索]** ウィンドウを閉じるには、**[検索]** ボックスをクリックします。

ナビゲーションツリー内のオブジェクトに対するコマンドの実行

ナビゲーションツリーでオブジェクトにコマンドを実行するには、次のいずれかの操作を実行します。

- オブジェクト名を右クリックすると、オブジェクトに対して実行可能なコマンドがポップアップメニューに表示されます。
- ナビゲーションツリーでオブジェクトを選択して、ウィンドウ上部の Hub コンソールメニューからコマンドを選択します。





例えばスキーママネージャで、あるタイプのオブジェクトをナビゲーションツリー上で右クリックすると、そのオブジェクトに対して実行できるコマンドがポップアップメニューに表示されます。

コマンドボタンを使用したオブジェクトの追加、編集、および削除

Hub コンソールでは、コマンドボタンを使用してオブジェクトを作成、編集、および削除することができます。

コマンドボタン

Hub コンソールウィンドウでオブジェクトを作成、変更、または削除するアクセス権がある場合、書き込みロックを取得したときに、以下の一部またはすべてのコマンドボタンが【プロパティ】ペインに表示されます。コマンドボタンはこれ以外にもあります。

ボタン	名前	説明
	追加	新しいオブジェクトを追加します。
	編集	選択した項目のプロパティを【プロパティ】ペインで編集します。プロパティが編集可能であることを示します。
	削除	選択した項目を削除します。
	保存	変更を保存します。

注: コマンドボタンの処理に関する説明を確認するには、ボタンの上にマウスを合わせてツールチップを表示します。

オブジェクトの追加

Hub コンソールではオブジェクトを追加できます。

1. 書き込みロックを取得します。
2. 【追加】ボタンをクリックします。
Hub コンソールに、追加するオブジェクトのタイプに応じて、オブジェクトを追加するためのウィンドウが表示されます。
3. オブジェクトのプロパティを指定します。
4. 【OK】をクリックします。

オブジェクトのプロパティの編集

Hub コンソールではオブジェクトのプロパティを編集できます。

1. 書き込みロックを取得し、編集するオブジェクトを選択します。
2. 編集するプロパティごとに、**【編集】** ボタンをクリックし、新しい値を指定します。
3. **【保存】** ボタンをクリックして変更を保存します。

オブジェクトの削除

Hub コンソールではオブジェクトを削除できます。

1. 書き込みロックを取得し、削除するオブジェクトを選択します。
2. **【削除】** ボタンをクリックします。
オブジェクトが他のオブジェクトに依存しているかまたは関連付けられている場合は、**【Impact Analyzer】** ダイアログボックスが表示されます。 オブジェクトが他のオブジェクトに依存しておらず関連付けもされていない場合は、**【Informatica MDM Hub コンソール】** ダイアログボックスが表示されます。
3. 適切なオプションを選択します。

MDM Hub コンソールインタフェースのカスタマイズ

Hub コンソールインタフェースはカスタマイズすることができます。

1. **【オプション】** > **【オプション】** の順にクリックします。
【オプション】 ダイアログボックスが表示されます。
2. 以下のタブで必要なオプションを指定します。
 - **【全般】** タブ:ウィザードの「ようこそ」画面を表示するかどうか、およびウィンドウのサイズと位置を保存するかどうかを指定します。
 - **【クイック起動】** タブ:メニューの下のクイック起動バーにアイコンとして表示するツールを指定します。

バージョンの詳細の表示

Multidomain MDM のインストールされているバージョンについて、バージョンの詳細を表示できます。

1. Hub コンソールで、**【ヘルプ】** > **【バージョン情報】** の順に選択します。
Informatica Multidomain MDM ダイアログボックスが表示されます。
2. **【インストールの詳細】** をクリックします。
【インストールの詳細】 ダイアログボックスが表示されます。
3. **【閉じる】** をクリックします。
4. **【閉じる】** をクリックします。








Informatica MDM Hub ワークベンチおよびツール

ここでは、Informatica MDM Hub のワークベンチおよびツールの概要を示します。






設定ワークベンチのツール




設定ワークベンチには、Hub の設定に役立つツールが含まれています。

以下の表に、設定ワークベンチのツールおよびそれらに関連付けられたアイコンを一覧表示します。




アイコン	ツール名	説明
	データベース	オペレーショナル参照ストアを登録および管理します。
	ユーザー	ユーザーを定義し、ユーザーがアクセスできるデータベースを指定します。グローバルパスワードポリシーおよび個々のパスワードポリシーを管理します。MDM Hub では、ユーザーの外部認証（LDAP など）がサポートされています。
	セキュリティプロバイダ	MDM Hub にアクセスするユーザーに対してセキュリティサービス（認証、承認、およびユーザープロファイルサービス）を提供するサードパーティのセキュリティプロバイダを設定します。
	ツールアクセス	ユーザーがアクセスできる Hub コンソールツールおよびプロセスを定義します。デフォルトでは、新しいユーザーアカウントは、アクセス権が明示的に割り当てられるまでどのツールにもアクセスできません。
	メッセージキュー	MDM Hub に対するインバウンドおよびアウトバウンドのメッセージキューインタフェースを定義します。
	リポジトリマネージャ	オペレーショナル参照ストアのメタデータを検証、リポジトリ間で変更を昇格、オブジェクトをリポジトリにインポート、およびリポジトリをエクスポートします。詳細については、『 <i>Multidomain MDM Repository Manager ガイド</i> 』を参照してください。
	エンタープライズマネージャ	Hub サーバー、プロセスサーバー、MDM Hub マスターデータベース、およびオペレーショナル参照ストアについての設定の詳細とバージョン情報を表示します。

モデルワークベンチのツール




アイコン	ツール名	説明
	スキーマ	ベースオブジェクト、リレーション、履歴とセキュリティの要件、ステージングテーブルとランディングテーブル、検証ルール、マッチング基準、およびその他のデータモデル属性を定義します。
	スキーマビューア	現在のスキーマを表示し、そのスキーマ内を移動します。
	システムと信頼	Informatica MDM Hub で統合するためのデータを提供できるソースシステムの名前を示します。ベースオブジェクトのカラムごとに各ソースシステムに関連付けられた信頼設定を定義します。
	クエリ	パッケージで使用されるクエリグループおよびクエリを定義します。
	パッケージ	パッケージ（テーブルビュー）を定義します。

アイコン	ツール名	説明
	クレンジング関数	データに対して実行するクレンジング関数を定義します。
	マッピング	クレンジング関数の出力をステージングテーブルのターゲットカラムにマップします。
	階層	階層マネージャでデータリレーションを表示および操作するために必要な構造を設定します。

セキュリティアクセスマネージャワークベンチのツール







アイコン	ツール名	説明
	セキュアリソース	MDM Hub のセキュアリソースを管理します。MDM Hub リソースごとにステータス（非公開、セキュア）を設定し、リソースグループを定義してセキュアリソースを整理します。
	ロール	リソースとリソースグループに対するロールと特権の割り当てを定義します。ユーザーとユーザーグループにロールを割り当てます。
	ユーザーとグループ	単一の Hub ストア内のユーザーとユーザーグループを管理します。

データスチュワードワークベンチのツール

アイコン	ツール名	説明
	データマネージャ	統合されたデータの内容の管理、相互参照の表示、データの編集、履歴の表示、および統合されたレコードのマージ解除を行います。 注: データマネージャツールは、暗号化されたカラム内のデータの編集または更新には使用できません。
	Merge Manager: マージマネージャ	手動でマージするためにキューに追加された一致レコードを確認およびマージします。
	階層マネージャ	Hub ストアの階層リレーションを定義および管理します。

データスチュワードワークベンチ内のツールについての詳細は、「*Multidomain MDM のデータスチュワードガイド*」を参照してください。

ユーティリティワークベンチのツール

アイコン	ツール名	説明
	バッチグループ	1つのコマンドで実行できる個々のバッチジョブ（ステージジョブ、ロードジョブ、マッチジョブなど）の集合であるバッチグループを設定および実行します。
	バッチビューア	バッチジョブを実行してデータをクレンジング、ロード、一致、または自動マージし、ジョブログを表示します。
	プロセスサーバー	プロセスサーバー情報（名前、ポート、サーバータイプ、サーバーがオンラインかオフラインかなど）を表示します。
	監査マネージャ	アプリケーション要求およびメッセージキューイベントの監査とデバッグを設定します。
	SIF マネージャ	オペレーショナル参照ストア固有のサービス統合フレームワーク（SIF）要求 API を生成します。SIF マネージャでは、オペレーショナル参照ストアのパッケージ、リモートパッケージ、マッピング、およびクレンジング関数の SIF 要求 API をサポートするコードが生成およびデプロイされます。生成されたオペレーショナル参照ストア固有の API は、Web サービスとして、および Siperian API JAR（siperian-api.jar）を介して使用できます。
	ユーザーオブジェクトレジストリ	オペレーショナル参照ストアに関する登録済みユーザーイグジット、カスタムの Java クレンジング関数、およびカスタムの GUI 関数を表示します。

第 3 章

インターナショナルデータのサポートの設定

この章では、以下の項目について説明します。

- [インターナショナルデータのサポートの設定の概要, 45 ページ](#)
- [Unicode データベースの設定 \(Oracle のみ\), 45 ページ](#)
- [US 以外のポピュレーションの一致設定の設定, 46 ページ](#)
- [Windows レジストリでの ANSI コードページの設定, 48 ページ](#)
- [Unicode 用のクレンジング設定, 48 ページ](#)
- [UTF-8 を使用する場合は UNIX 向けのロケールの推奨事項, 48 ページ](#)
- [破損したデータのトラブルシューティング, 49 ページ](#)
- [Oracle 環境での言語の設定, 49 ページ](#)

インターナショナルデータのサポートの設定の概要

複数の国のデータがある場合は、MDM Hub の実装で文字セットを設定できます。使用するデータベースで、選択した文字セットがサポートされている必要があります。

複数の国の、異なる文字セットのデータを含む Oracle 環境がある場合は、Unicode Transfer Format (UTF-8) エンコードを使用する必要があります。また、NLS_LANG を設定して、クライアントの Oracle ソフトウェアのロケール動作を指定する必要があります。

Unicode データベースの設定 (Oracle のみ)

MDM Hub 実装で Oracle データベースを使用する場合は、使用する文字セットを設定する必要があります。実装環境で、異なる文字セットを使用する複数の国のデータなど、混在ロケール情報を使用する場合は、MDM Hub とデータベースを、Unicode Transfer Format (UTF-8) エンコーディングを使用するように設定する必

要があります。ただし、データベースに1つのロケールのデータが含まれる場合、通常 UTF-8 データベースは不要です。

1. UTF-8 データベースを作成し、以下の設定を選択します。

- **データベース文字セット:** AL32UTF8

- **各国語文字セット:** AL16UTF16

注: Oracle では、Oracle 10g のデータベース文字セットとして AL32UTF8 を推奨しています。以前の Oracle リリースの詳細については、Oracle のマニュアルを参照してください。

2. サーバーとクライアントの両方で、データベース文字セットに一致するように **NLS_LANG** を設定します。例えば、米国内の場合は、NLS_LANG を次の値に設定します。

AMERICAN_AMERICA.AL32UTF8

ただし、日本語の実装環境では、NLS_LANG を次の値に設定します。

JAPANESE_JAPAN.AL32UTF8

3. クライアントで地域のフォントを設定します。

例えば、中国のデータの場合は、中国語のフォントをインストールします。

4. マルチバイトの文字セットを使用する場合は、Unicode 値をサポートするため、C_REPOS_DB_RELEASE テーブルで次の設定を変更します。

column_length_in_bytes_ind = 0

US 以外のポピュレーションの一致設定の設定

MDM Hub 実装で US 以外のポピュレーションを使用する場合は、ポピュレーションセットを設定し、一致処理のエンコーディングを有効にします。ポピュレーションセットでは、名前、アドレス、および特定のポピュレーションに関するその他の識別情報がカプセル化されます。ポピュレーションセットの詳細については、[「ポピュレーションセット」 \(ページ 288\)](#)を参照してください。

MDM Hub には、demo.ysp ファイルと<population>.ysp ファイルが含まれます。demo.ysp ファイルには、デモ専用のデモポピュレーションのみが格納されており、実際の一致ルールで使用することはできません。MDM Hub 実装では、購入したライセンスの対象<population>.ysp ポピュレーションファイルを使用します。ポピュレーションファイルをお持ちでない場合は、Informatica グローバルカスタマサポートに問い合わせ、お使いの実装に適したポピュレーションファイルを入手してください。

次の考慮事項に基づいてポピュレーションセットを決定します。

- データが1つの国のデータのみで構成されており、その国用のポピュレーションセットが Informatica から提供されている場合は、そのポピュレーションを使用します。
- データのほとんどが1つの国のデータで構成され、1つまたは複数の別の国のデータがごく少量だけ混在している場合は、大部分を占めるデータのポピュレーションを使用します。
- データに異なる国の混成データが大量に含まれている場合は、それらのデータセットを一致させることが有効かどうか検討します。有効な場合は、international ポピュレーションを使用します。

一致ポピュレーションの有効化の詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

一致処理に使用するエンコードの設定

照合時に UTF-8 文字を処理できるようにするには、プロセスサーバーの設定を編集します。

1. テキストエディタを使用して次のファイルを開きます: <MDM Hub Installation Directory>\hub\cleanse\resources\cmxcleanse.properties
2. 次の設定を手動で追加します。
cmx.server.match.server_encoding = 1

1つのベースオブジェクト内での複数のポピュレーションの使用

MDM Hub の 1 つのベースオブジェクト内で複数のポピュレーションを使用できます。

これは、ベースオブジェクトのデータが異なるポピュレーションから取得される場合に便利です。例えば、レコードの 70%が米国から、30%が中国から来るとします。レコードごとに異なるポピュレーションを使用できます。

ベースオブジェクト内で複数のポピュレーションを使用するには、次の手順に従います。

1. Informatica グローバルカスタマサポートに連絡して、実装に該当する<population>.ysp ファイルと、ポピュレーションを有効にするための手順を入手します。
2. 使用する各ポピュレーションを C_REPOS_SSA_POPULATION メタデータテーブルで有効にします。
3. 該当するポピュレーションファイルを次の場所にコピーします。

UNIX の場合:<MDM Hub installation directory>/hub/cleanse/

Windows の場合:<MDM Hub installation directory>\cleanse\resources\match

4. アプリケーションサーバーを再起動します。
5. スキーママネージャで、各レコードに使用するポピュレーションを格納する SIP_POP という名前の VARCHAR カラムをベースオブジェクトに追加します。

注: VARCHAR カラムの幅には、使用しているポピュレーション名のうち最長の名前が収まる必要があります。ほとんどの実装では、30 あれば十分です。

6. 一致カラムを、名前が SIP_POP の完全一致カラムとして設定します。
7. デフォルトではないポピュレーションを使用するベースオブジェクトの各レコードについて、使用するポピュレーションの名前を SIP_POP カラムに入力します。
以下のいずれかの方法で SIP_POP カラムの値を指定できます。

- ランディングテーブルに UTF-8 データを追加する。
- ステージプロセス時に値を計算するクレンジング関数を使用する。
- 外部アプリケーションから SIF 要求を呼び出す。
- データマネージャツールを使用して、カラムの値を手動で編集する。

注: SIP_POP カラムのデータでは大文字小文字は区別されませんが、NULL 値や空の文字列などの無効な値があった場合はデフォルトのポピュレーションが使用されます。

8. ベースオブジェクトに対して一致トークンの生成プロセスを実行し、一致キーテーブルを更新します。
9. ベースオブジェクトに対して一致プロセスを実行します。

注: 一致プロセスでは、同じポピュレーションを共有するレコードのみが比較されます。例えば、中国のレコードは中国のレコードと、アメリカのレコードはアメリカのレコードと比較されます。

Windows レジストリでの ANSI コードページの設定

Windows で、Windows のレジストリエディタを使用して ANSI コードページを設定します。

1. コマンドプロンプトから、「regedit」と入力し、**[OK]** をクリックします。
2. 次のレジストリエントリに移動します。
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\ACP`
3. ANSI コードページを変更するには、Windows のコントロールパネルでロケールと言語の設定を行います。手順は、Windows のバージョンによって異なります。詳細な手順については、Microsoft Windows のマニュアルを参照してください。

Unicode 用のクレンジング設定

Informatica Address Verification のクレンジングライブラリを使用する場合は、適切なデータベースと Informatica Address Verification のロック解除コードがあることを確認します。実装に必要なすべての国の Informatica Address Verification データベースを入手する必要があります。詳細については、Informatica グローバルカスタマサポートまでお問い合わせください。

Trillium を使用する場合は、正しいテンプレートを使用してプロジェクトを作成してください。サポートされている国を確認するには、Trillium のインストールマニュアルを参照してください。国固有のプロジェクトを Trillium から直接取得できます。

UTF-8 を使用する場合は UNIX 向けのロケールの推奨事項

UNIX システムの多くは、互換性のない文字エンコードを使用して各国語の文字をバイナリデータとして表しています。そのため、たとえば、韓国語のシステムのテキストは中国のシステムでは表示できません。ただし、どの言語にも UTF-8 エンコードを使用するように UNIX システムを設定できます。UTF-8 テキストエンコードは複数の言語をサポートしており、ある言語が別の言語と干渉しあうことはありません。

UTF-8 を使用するようにシステムロケールを設定するには、次の手順を実行します。

1. 次のコマンドを実行します。
`locale -a`
2. サフィックス utf8 を使用して、自分の言語のロケールが見つかるかどうかを確認します。
`localedef -f UTF-8 -i en_US en_US.utf8`
3. UTF-8 を許可するロケールがある場合は、そのロケールを使用するように指定します。
`Export LC_ALL="en_US.utf8"
export LANG="en_US.utf8"
export LANGUAGE="en_US.utf8"`

破損したデータのトラブルシューティング

SQL*Loader を使用して国際文字セットで表されたデータをロードしていて、ロードしたデータが破損している場合、cmxcleanse.properties ファイルのプロパティを設定することによってこの問題を修正できます。

1. 次のディレクトリに移動します。
`<MDM installation directory>/hub/cleanse/resources`
2. エディタで cmxcleanse.properties ファイルを開きます。
3. ファイルの末尾に次のプロパティを入力し、ロードするデータに一致する国際文字セットの Unicode 識別子を指定します。
`cmx.server.stage.sqlldr.charset=AL32UTF8`
4. ファイルを保存します。
5. ステージプロセスを実行します。
ステージプロセスにより、指定した文字セットを使用した SQL*Loader の制御ファイルが生成されます。
6. データを再ロードするときに、この制御ファイルを使用します。

Oracle 環境での言語の設定

クライアントの Oracle ソフトウェアのロケールの動作を指定するには、NLS_LANG 設定を設定し、クライアントの言語、地域、および文字セットを指定する必要があります。NLS_LANG 設定の設定方法は、オペレーティングシステムに応じて異なります。

Windows

NLS_LANG 設定は、Windows レジストリエディタを使用して設定するか、環境変数として設定できます。

UNIX

LANG 環境変数を設定し、必要なロケールをインストールする必要があります。

NLS_LANG の構文

NLS_LANG 設定では、次の形式を使用します。

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

次の表に、これらのパラメータについて説明します。

パラメータ	説明
LANGUAGE	日付と月の名前の他に、Oracle メッセージで使用する言語を指定します。
TERRITORY	週数と日数の計算に使用する地域および規則の他に、通貨と数値の形式を指定します。
CHARACTERSET	クライアントアプリケーションが使用する文字セットを管理します。このパラメータは Windows コードページと一致します。または Unicode アプリケーション用に UTF-8 に設定できます。

注: NLS_LANG 設定で定義された文字セットでは、クライアントの文字セットは変更されません。代わりに Oracle データベースが、データを定義された文字セットに変換します。NLS_LANG 設定は、サーバーから文字セットを継承することはありません。

Windows レジストリでの NLS_LANG の設定

Windows システムでは、Oracle ホームごとに NLS_LANG レジストリサブキーを設定していることを確認します。

このサブキーは、Windows のレジストリエディタを使用して変更できます。

1. コマンドプロンプトから、「regedit」と入力し、**[OK]** をクリックします。
2. 次のレジストリエントリに移動します。

Oracle 10g の場合:

HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_<Oracle_Home_name>

3. **[NLS_LANG]** サブキーを編集します。

環境変数としての NLS_LANG の設定 (Windows)

Informatica では推奨しませんが、NLS_LANG は、システムのプロパティでシステムまたはユーザー環境変数として設定できます。すべての Oracle Home で設定された値が使用されます。

システムまたはユーザー環境変数を確認するには、次のステップを実行します。

1. **[マイ コンピューター]** を右クリックし、**[プロパティ]** を選択します。
2. **[詳細設定]** タブで、**[環境変数]** をクリックします。
[ユーザー環境変数] リストに、現在ログインしている Windows ユーザー用の設定が表示されます。
[システム環境変数] リストに、すべてのユーザー用のシステム全体の変数が表示されます。
3. 必要に応じて設定を変更します。

これらの環境変数は Windows レジストリで指定されているパラメータより優先されるので、特別な理由がない限り、ここでは Oracle パラメータを設定しないようにしてください。特に、ORACLE_HOME パラメータが、Windows ではなく UNIX に設定されることに注意してください。

LANG 環境変数とロケールの設定 (UNIX)

MDM Hub 全体で UTF-8 データの処理を統一するには、アプリケーションサーバーをホストする UNIX サーバーに対して正しいロケールおよび LANG 環境変数を設定します。

データベースとサーバーを含む MDM Hub のすべてのシステムで、次の環境変数を設定します。

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

Oracle 環境では、次の環境変数を設定します。

- export NLS_LANG=AMERICAN_AMERICA.AL32UTF8

たとえば、米国のデフォルトの LANG 環境変数は、export LANG=en_US です。

したがって、UTF-8 を使用する場合は、次のコマンドを使用して LANG 環境変数を設定します。

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

1 台のマシンに複数のアプリケーションがインストールされていて、正しいロケールもすべてインストールされている場合は、アプリケーションを起動するプロファイルに対して正しい環境変数を設定できます。同じユーザープロファイルが複数のアプリケーションを起動する場合は、アプリケーションの起動スクリプトにローカルに環境変数を設定できます。これにより、環境変数はアプリケーションプロセスのコンテキスト内でのみ、ローカルに適用されます。

通常、すべての LANG 環境変数で同じ設定を使用しますが、異なる設定を使用する場合もあります。例えば、インタフェース言語は英語で、ソートが必要なデータはフランス語である場合、LC_MESSAGES を en_US に設定し、LC_COLLATE を fr_FR に設定します。異なる LANG 設定を使用する必要がない場合は、LC_ALL または LANG を設定します。

アプリケーションでは次のルールを使用して、使用するロケールを決定します。

- LC_ALL 環境変数が定義されていて NULL でない場合、アプリケーションでは LC_ALL の値が使用されます。
- LC_COLLATE などの適切なコンポーネント固有の環境変数が設定されていて NULL でない場合、アプリケーションではこの環境変数の値が使用されます。
- LANG 環境変数が定義されていて NULL でない場合、アプリケーションでは LANG の値が使用されます。
- LANG 環境変数が設定されていないか NULL の場合、アプリケーションでは実装に依存するデフォルトのロケールが使用されます。

注: シナリオによって異なるロケールを使用する必要がある場合は、LC_ALL を設定しないでください。

パート II: ハブコンソールツールの設定

この部には、以下の章があります。

- [Hub コンソールのツールへのアクセスの設定, 53 ページ](#)
- [Hub コンソールツールでのカスタムボタンの実装, 56 ページ](#)

第 4 章

Hub コンソールのツールへのアクセスの設定

この章では、以下の項目について説明します。

- [Hub コンソールのツールへのアクセスの設定に関する概要, 53 ページ](#)
- [ユーザー設定, 53 ページ](#)
- [ツールおよびプロセスに対するユーザーアクセス, 54 ページ](#)

Hub コンソールのツールへのアクセスの設定に関する概要

MDM Hub ユーザーが Hub コンソールツールにアクセスする方法を制御することができます。例えば、データスチュワードには、データマネージャツールとマージマネージャツールのみに対するアクセス権限がある場合があります。

Hub コンソールのツールへのアクセスを設定するには、設定ワークベンチのツールアクセスツールを使用します。ツールアクセスツールを使用するには、マスターデータベースに接続する必要があります。

ツールアクセスツールは、管理者として設定されていない MDM Hub ユーザーにのみ適用されます。ユーザー設定の詳細については、*Multidomain MDM のセキュリティガイド*を参照してください。

ユーザー設定

MDM Hub 内のユーザーを作成、編集、削除できます。

設定ワークベンチのユーザーツールは、MDM Hub ユーザーのユーザーアカウントを設定したり、パスワードを変更したり外部認証を有効化するために使用します。最新のパスワードおよびパスワードを変更したユーザーに関する最新情報が保持されます。パスワード履歴を使用することはできません。

Hub ストアで問題が発生する可能性があるため、他の方法を使用して MDM Hub に直接ユーザー情報をインポートしないことをお勧めします。

また、ユーザーツールを使用して、外部アプリケーションユーザーのユーザーアカウントを設定することもできます。外部アプリケーションユーザーは、サードパーティの信頼されるアプリケーションを介して MDM Hub データに間接的にアクセスする、MDM Hub ユーザーです。

ユーザー設定の詳細については、*Multidomain MDM* のセキュリティガイドを参照してください。

ツールおよびプロセスに対するユーザーアクセス

Hub コンソールのツールへのアクセスを設定するには、設定ワークベンチのツールアクセスツールを使用します。

ツールアクセスツールの起動

ツールアクセスツールを起動する手順

1. Hub コンソールでマスターデータベースに接続します（接続していない場合）。
2. 設定ワークベンチを展開し、**【ツールアクセス】** をクリックします。
ツールアクセスツールが表示されます。

ツールおよびプロセスに対するアクセス権のユーザーへの付与

Hub コンソールのツールおよびプロセスへのアクセス権を特定の MDM Hub ユーザーに付与する手順

1. 書き込みロックを取得します。
2. ツールアクセスツールで、ユーザーリストをスクロールし、設定するユーザーを選択します。
3. 次のいずれかを実行します。
 - **【使用可能なプロセス】** リストで、アクセス権を付与するプロセスを選択します。
 - **【使用可能なワークベンチ】** リストで、アクセス権を付与するツールを含むワークベンチを選択します。
4. **【ツールの追加】** または **【プロセスの追加】** をクリックします。

ツールアクセスツールの **【使用可能なツールとプロセス】** リストに、選択したツールまたはプロセスが追加されます。プロセスにアクセス権を付与すると、MDM Hub によりそのプロセスで使用されるすべてのツールにアクセス権が付与されます。ツールにアクセス権を付与すると、MDM Hub によりそのツールを使用するすべてのプロセスにアクセス権が付与されます。

ユーザーはアクセス権を持つすべてのオペレーショナル参照ストアについて、これらのプロセスとツールにアクセスできます。ユーザーに対し、あるオペレーショナル参照ストアと別のオペレーショナル参照ストアについてそれぞれ別のツールへのアクセス権を付与することはできません。

注: ワークベンチの一部のツールにアクセス権を付与しない場合は、**【使用可能なツールとプロセス】** リストで関連付けられているワークベンチを展開し、選択したツールのアクセス権を取り消します。

ツールおよびプロセスに対するアクセス権のユーザーからの取り消し

Hub コンソールのツールおよびプロセスへのアクセス権を特定の MDM Hub ユーザーで取り消す手順

1. 書き込みロックを取得します。
2. ツールアクセスツールで、ユーザーリストをスクロールし、設定するユーザーを選択します。
3. **【使用可能なツールとプロセス】** リストをスクロールして、アクセス権を取り消すプロセス、ワークベンチ、またはツールを選択します。
ツールを選択するには、関連付けられたワークベンチを展開します。

4. **【ツールの削除】** または **【プロセスの削除】** をクリックします。

アクセス権限の取り消しを確認するメッセージが表示されます。

5. **【はい】** をクリックします。

選択した項目が、**【使用可能なツールとプロセス】** リストから削除されます。プロセスへのアクセス権限を取り消すと、そのプロセスが使用するあらゆるツールへのアクセス権限が MDM Hub によって取り消されます。ツールへのアクセス権限を取り消すと、そのツールを使用するあらゆるプロセスへのアクセス権限が MDM Hub によって取り消されます。

第 5 章

Hub コンソールツールでのカスタムボタンの実装

この章では、以下の項目について説明します。

- [概要, 56 ページ](#)
- [Hub コンソールのカスタムボタンについて, 56 ページ](#)
- [カスタムボタンの追加, 57 ページ](#)
- [カスタムボタンの外観の制御, 60 ページ](#)
- [カスタムボタンのデプロイ, 60 ページ](#)

概要

この章では、Informatica MDM Hub 実装で Hub コンソールのツールにカスタムボタンを追加し、外部サービスをオンデマンドで呼び出せるようにする方法について説明します。

Hub コンソールのカスタムボタンについて

Informatica MDM Hub の実装環境では、Hub コンソールのユーザーに対し、Informatica MDM Hub 実装を拡張するためのカスタムボタンを提供することができます。

カスタムボタンにより、専門のデータサービスにオンデマンドで、リアルタイムなアクセスをユーザーに提供することができます。カスタムボタンは、マージマネージャおよび階層マネージャに追加できます。

ユーザーはカスタムボタンを使用して、特定の外部サービス（データの取得や結果の計算など）の呼び出しや特殊な操作（ワークフローの起動など）の実行などのタスクを行うことができます。カスタムボタンは、さまざまなサービスプロバイダから提供されるデータサービスにアクセスするように設計できます。これには、エンタープライズアプリケーション（CRM や ERP などのアプリケーション）、外部のサービスプロバイダ（外国為替計算、金融市場指数の発行元、政府機関など）、Informatica MDM Hub 本体など、ほかにもさまざまなサービスが含まれます（詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照）。

例えば、特殊なクレンジング関数を呼び出すカスタムボタンを追加することができます。これは、ベンダから Web サービスとして提供される機能で、マージマネージャの画面で現在選択されている顧客レコードのデータをクレンジングできます。ユーザーがこのボタンをクリックすると、ベースとなるコードにより、選択したレコードから関連するデータがキャプチャされ、Web サービスで必要な形式の要求（例えば認証情報など）が作成された後、その要求が処理のために Web サービスに送信されます。結果が返されると、独立した Swing ダ

イアログ（作成した場合およびクライアントのカスタム関数として実装した場合）に、Informatica MDM Hub の顧客の rowid_object とともに情報が表示されます。

カスタムボタンはデフォルトではインストールされず、すべての Informatica MDM Hub 実装に必ずしも必要なものではありません。Java インタフェースを実装する必要があるカスタムボタンごとに、実装を JAR ファイルにパッケージし、コマンドラインユーティリティを実行してデプロイします。Hub コンソールのカスタムボタンの外観を制御するには、テキストまたはアイコンのグラフィック（JPG、PNG、GIF などの Swing と互換性があるグラフィック形式）を指定します。

ユーザーがカスタムボタンをクリックしたときの動作

ユーザーが Hub コンソールで顧客レコードを選択してカスタムボタンをクリックすると、Hub コンソールによって要求が呼び出され、コンテンツとコンテキストが Java 外部（カスタム）サービスに渡されます。データのタイプの例には、ベースオブジェクトのレコードキーおよびその他のデータ、パッケージ情報などがあります。実行は非同期なので、ユーザーは要求の処理中に Hub コンソールでの作業を続行できます。

カスタムコードでは、必要に応じてサービス応答を処理できます。結果をログに記録したり、データを別の Swing ダイアログ（カスタムコーディングされており、カスタム関数がクライアント側の関数である場合）に表示したり、ユーザーが結果をコピーしてデータ入力フィールドに貼り付けることができるようにしたり、PUT 文をリアルタイムで実行してデータを正しいビジネスオブジェクトに戻したりします。

Hub コンソールにカスタムボタンを表示する手順

この節では、実装したカスタムボタンを Hub コンソールのマージマネージャツールおよび階層マネージャツールに表示する方法を説明します。

マージマネージャのカスタムボタン

カスタムボタンは、マージマネージャの上部パネルの右側（マージマネージャの標準ボタンと同じ場所）に表示されます。

階層マネージャのカスタムボタン

カスタムボタンは、階層マネージャ画面の上部パネルの最上部に表示されます。これは階層マネージャの他のボタンと同じ場所です。

カスタムボタンの追加

Informatica MDM Hub 実装の Hub コンソールにカスタムボタンを追加するには、以下の作業を実行します。

1. 要求メッセージと応答メッセージの形式やパラメータなど、呼び出す外部サービスの詳細を確認します。
2. カスタムボタンで実行するビジネスロジックを記述してパッケージ化します。
3. Hub コンソールのアプリケーションツールに表示されるようにパッケージをデプロイします。

ユーザーは、Hub コンソールに表示された外部サービスのボタンをクリックして、そのサービスを呼び出すことができます。

カスタム関数の作成

外部サービスの呼び出しを作成するには、ユーザーが Hub コンソールでカスタムボタンをクリックしたときにアプリケーションロジックを実行するカスタム関数を記述します。アプリケーションロジックでは、次の Java インタフェースを実装します。

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

このインタフェースの詳細については、使用している Informatica MDM Hub ディストリビューションに付属の Javadoc を参照してください。

サーバーベースおよびクライアントベースのカスタム関数

アプリケーションロジックは以下のいずれかの環境で実行します。

環境	説明
クライアント	UI ベースのカスタム関数—応答情報を表示する個別のダイアログなど、エレメントをユーザーインタフェースに表示する場合に推奨されます。
サーバー	サーバーベースのカスタムボタン—ネットワークまたはパフォーマンス上の理由により、サーバーから外部サービスを呼び出すのが適切な場合に推奨されます。

カスタム関数の例

この節では、`com.siperian.mrm.customfunctions.api.CustomFunction` インタフェースを実装する 2 つのカスタム関数例の Java コードを示します。このコードは、（標準エラー発生時に）情報をサーバーログまたは Hub コンソールログに出力します。

クライアントベースのカスタム関数の例

以下のサンプルコードで使用されるクライアント関数クラスの名前は、`com.siperian.mrm.customfunctions.test.TestFunction` です。

```
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; }
}
```

サーバーベースの関数の例

以下のコードで 사용되는サーバー関数クラスの名前は、
com.siperian.mrm.customfunctions.test.TestFunctionClient です。

```
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;
    }
}
```

カスタムボタンの外観の制御

Hub コンソールにおけるカスタムボタンの外観を制御するには、`com.siperian.mrm.customfunctions.api.CustomFunction` インタフェースの以下のいずれかのメソッドを実装します。

メソッド	説明
<code>getActionText</code>	ボタンラベルのテキストを指定します。外観については、カスタムボタン用のデフォルトの外観が使用されます。
<code>getGuilcon</code>	Swing と互換性があるグラフィック形式（JPG、PNG、GIF など）のアイコンのグラフィックを指定します。このイメージファイルは、このカスタム関数用の JAR ファイルとバンドルすることができます。

Hub コンソールでは、カスタムボタンは名前のアルファベット順で表示されます。

カスタムボタンのデプロイ

Hub コンソールにカスタムボタンを表示するには、コマンドラインから `DeployCustomFunction` ユーティリティを使用してそれらのボタンを明示的に追加する必要があります。

カスタムボタンをデプロイする手順

1. コマンドプロンプトを開きます。
2. `DeployCustomFunction` ユーティリティを実行します。これにより、ユーザーが作成した JAR ファイルがロードされて登録されます。

注: `DeployCustomFunction` を実行するには、2 つの JAR ファイル（`siperian-server.jar` と JDBC ドライバ（この場合は `ojdbc14.jar`））がこれらのファイルを指すディレクトリパスとともに `CLASSPATH` に含まれている必要があります。

コマンドプロンプトで以下のコマンドを指定します。

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

Informatica MDM Hub の実装で構成されている設定に基づいてプロンプトに応答します。以下に例を示します。

```
Database Type:oracle
Host:localhost
Port(1521):
Service:orcl
Username:ds_ui1
Password:!!cmx!!
(L)ist, (A)dd, (U)pdate, (C)hange Type, (S)et Properties, (D)elete or (Q)uit:l
No custom actions
(L)ist, (A)dd, (U)pdate Jar, (C)hange Type, (S)et Properties, (D)elete or (Q)uit:q
```

3. それぞれのプロンプトで、Informatica MDM Hub の実装で構成されている設定に基づいて以下の情報を指定します。
 - データベースホスト
 - ポート
 - サービス

- ログインユーザー名（スキーマ名）
 - ログインパスワード
4. プロンプトが表示されたら、データベース接続情報を指定します（データベースホスト、ポート、サービス、ログインユーザー名、およびパスワード）。
 5. DeployCustomFunction ツールにより、以下のオプションを含んだメニューが表示されます。

ラベル	説明
(L)ist (リスト)	現在定義されているカスタムボタンのリストを表示します。
(A)dd (追加)	新しいカスタムボタンを追加します。DeployCustomFunction ツールから以下の情報を指定するように求められます。 <ul style="list-style-type: none"> - カスタムボタンの JAR ファイル - com.siperian.mrm.customfunctions.api.CustomFunction インタフェースを実装するカスタム関数クラスの名前 - カスタムボタンのタイプ: m—マージマネージャ、d—データマネージャ、h—階層マネージャ（1 文字または 2 文字を指定できます）
(U)pdate (更新)	既存のカスタムボタンの JAR ファイルを更新します。DeployCustomFunction ツールから以下の情報を指定するように求められます。 <ul style="list-style-type: none"> - 更新するカスタムボタンの行 ID - カスタムボタンの JAR ファイル - com.siperian.mrm.customfunctions.api.CustomFunction インタフェースを実装するカスタム関数クラスの名前 - カスタムボタンのタイプ: m—マージマネージャ、h—階層マネージャ（1 文字または 2 文字を指定できます）
(C)hange Type (タイプの変更)	既存のカスタムボタンのタイプを変更します。DeployCustomFunction ツールから以下の情報を指定するように求められます。 <ul style="list-style-type: none"> - 更新するカスタムボタンの行 ID - カスタムボタンのタイプ: m—マージマネージャおよび h—階層マネージャまたはそのいずれか（1 文字または 2 文字を指定できます）
(S)et Properties (プロパティの設定)	プロパティファイルを指定します。このファイルでは、カスタム関数が実行時に必要とする名前/値のペアが定義されます（名前=値）。DeployCustomFunction ツールから、使用するプロパティファイルを指定するように求められます。
(D)elete (削除)	既存のカスタムボタンを削除します。DeployCustomFunction ツールから、削除するカスタムボタンを行 ID を指定するように求められます。
(Q)uit (終了)	DeployCustomFunction ツールを終了します。

6. アクションの選択が終了したら、**[(Q)uit (終了)]** を選択します。
7. ブラウザウィンドウを更新すると、追加したカスタムボタンが表示されます。
8. カスタムボタンをテストして、正しく機能することを確認します。

パート III: データモデルの構築

この部には、以下の章があります。

- [Hub Store について](#), 63 ページ
- [オペレーショナル参照ストアとデータソースの設定](#), 65 ページ
- [スキーマの構築](#), 79 ページ
- [クエリとパッケージ](#), 128 ページ
- [タイムライン](#), 145 ページ
- [状態管理および BPM ワークフローツール](#), 173 ページ
- [データ暗号化](#), 187 ページ
- [階層](#), 194 ページ
- [階層マネージャのチュートリアル](#), 229 ページ

第 6 章

Hub Store について

この章では、以下の項目について説明します。

- [概要, 63 ページ](#)
- [Hub Store のデータベース, 63 ページ](#)
- [Hub Store 内の各データベースの関係, 64 ページ](#)
- [Hub ストアデータベースの作成, 64 ページ](#)
- [バージョン要件, 64 ページ](#)

概要

Hub ストアは、Informatica MDM Hub のビジネスデータが格納および統合される場所です。

Hub ストアには、Informatica MDM Hub 実装の一部であるすべてのデータベースに関する共通の情報が含まれます。

Hub Store のデータベース

Hub ストアはデータベースの集まりで、以下の要素が含まれます。

要素	説明
マスターデータベース	Informatica MDM Hub 環境構成設定（ユーザーアカウント、セキュリティ設定、オペレーショナル参照ストアレジストリ、メッセージキュー設定など）が含まれています。1 つの Informatica MDM Hub 環境で利用できるマスターデータベースは 1 つだけです。マスターデータベースのデフォルト名は、CMX_SYSTEM です。 Hub コンソールでは、設定ワークベンチの各ツール（データベース、ユーザー、セキュリティプロバイダ、ツールアクセス、およびメッセージキュー）によって、マスターデータベース内の構成設定が管理されます。
オペレーショナル参照ストア (オペレーショナル参照ストア)	最善データ (BVT) の定義時に Informatica MDM Hub によって使用される処理ルールと補助ロジックに加えて、マスターデータ、コンテンツメタデータ、マスターデータの処理ルール、マスターデータオブジェクトセットの管理ルールを格納するデータベース。 Informatica MDM Hub 設定には、1 つ以上の ORS データベースを指定できます。オペレーショナル参照ストアのデフォルト名は CMX_ORS です。

Hub ストアデータベースのユーザーはマスターデータベース内でグローバルに作成され、その後に特定のオペレーショナル参照ストアに割り当てられます。マスターデータベースには、サイトレベルの情報（ユーザーアカウントがロックアウトされるまでのログインの試行回数など）も格納されます。

Hub Store 内の各データベースの関係

Informatica MDM Hub の実装には、1 つのマスターデータベースおよび 0 個以上の ORS が含まれています。

ORS が存在しない場合、Hub コンソールでは設定ワークベンチツールのみが利用可能です。Informatica MDM Hub の実装では、複数のオペレーショナル参照ストアを作成して、例えば開発用とプロダクション用にオペレーショナル参照ストアを分けたり、地域別または組織の部署別にオペレーショナル参照ストアを分けたりすることができます。

複数のオペレーショナル参照ストアを、1 つのマスターデータベースからアクセスして管理することができます。マスターデータベースには、各オペレーショナル参照ストアの接続設定およびプロパティが格納されます。

注: オペレーショナル参照ストアは、1 つのマスターデータベースにのみ登録できます。複数のマスターデータベースで同じオペレーショナル参照ストアを共有することはできません。1 つのオペレーショナル参照ストアを複数のマスターデータベースに関連付けることはできません。

Hub ストアデータベースの作成

データベースは、Informatica MDM Hub をインストールするときに最初に作成され、構成されます。

- マスターデータベースおよび 1 つの ORS を作成するには、`setup.sql` スクリプトを実行します。
- 個々の ORS を作成するには、`setup_ors.sql` スクリプトを実行します。

詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

バージョン要件

異なるバージョンの Informatica MDM Hub を同じ環境内で同時に動作させることはできません。

Informatica MDM Hub ソフトウェアや Hub Store のデータベースなど、インストールに含まれるすべてのコンポーネントが同じバージョンである必要があります。

サイトで複数のバージョンの Informatica MDM Hub を使用する場合は、各バージョンを個別の環境にインストールする必要があります。異なるバージョンのデータベースを使用しようとすると、データベースを現在のバージョンにアップグレードするように求めるメッセージが表示されます。

第 7 章

オペレーショナル参照ストアとデータソースの設定

この章では、以下の項目について説明します。

- [オペレーショナルリファレンスストアとデータソースの設定の概要, 65 ページ](#)
- [作業を開始する前に, 65 ページ](#)
- [データベースツールの概要, 66 ページ](#)
- [データベースツールの起動, 66 ページ](#)
- [オペレーショナルリファレンスストアの設定, 66 ページ](#)
- [データソースの設定, 77 ページ](#)

オペレーショナルリファレンスストアとデータソースの設定の概要

Hub コンソールのデータベースツールを使用して、Hub ストアのオペレーショナルリファレンスストアとデータソースを設定することができます。オペレーショナルリファレンスストアを作成したら、データベースツールで登録して定義する必要があります。また、データベースツールを使用してデータソースを作成または削除することもできます。

作業を開始する前に

開始する前に、MDM Hub をインストールし、MDM Hub マスタデータベースを作成して、オペレーショナルリファレンスストアを少なくとも 1 つ作成する必要があります。MDM Hub マスタデータベースとオペレーショナルリファレンスストアを作成するには、『*Multidomain MDM のインストールガイド*』の手順を参照してください。

データベースツールの概要

Hub ストアを作成したら、Hub コンソールのデータベースツールを使用して、オペレーショナルリファレンスストアを登録して定義します。

データベースツールを使用してオペレーショナルリファレンスストアを登録し、MDM Hub が接続できるようにします。オペレーショナルリファレンスストアを登録すると、データベース接続プロパティが MDM Hub マスタデータベースに格納されます。

データベースツールを使用して、オペレーショナルリファレンスストアのデータソースを作成します。オペレーショナルリファレンスストアのデータソースには、オペレーショナルリファレンスストアのプロパティのセットが含まれています。これには、データベースサーバーの場所、データベースの名前、サーバーとの通信に使用するネットワークプロトコル、データベースユーザーの ID とパスワードなどのプロパティが含まれます。

注: データベースツールでは、オペレーショナルリファレンスストアがデータベースとして参照されます。

データベースツールの起動

Hub コンソールでデータベースツールを起動します。

1. Hub コンソールで、MDM Hub マスタデータベースに接続します。
2. 設定ワークベンチを展開し、**[データベース]** をクリックします。

Hub コンソールにデータベースツールが表示され、登録したオペレーショナルリファレンスストアが表示されます。

データベースツールには以下のデータベース情報が表示されます。

データベース情報	説明
データベース数	Hub ストアで定義されたオペレーショナルリファレンスストアの数。
データベースリスト	登録されている MDM Hub オペレーショナルリファレンスストアのデータベースのリスト。
データベースのプロパティ	選択したオペレーショナルリファレンスストアのデータベースのプロパティ。

オペレーショナルリファレンスストアの設定

Hub コンソールのデータベースツールを使用して、Hub ストアでオペレーショナルリファレンスストアを設定することができます。

オペレーショナルリファレンスストアの設定についてサポートが必要な場合は、データベース管理者に問い合わせてください。オペレーショナルリファレンスストアの詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

Microsoft SQL Server 用のオペレーショナルリファレンスストアの接続プロパティ

オペレーショナル参照ストアを Microsoft SQL Server に登録する場合は、次の接続プロパティを設定します。

データベース表示名

Hub コンソールに表示する必要のあるオペレーショナル参照ストアの名前。

マシン識別子

Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。

データベースホスト名

Microsoft SQL Server データベースをホストするサーバーの IP アドレスまたは名前。

ポート

Microsoft SQL Server データベースのポート。デフォルトは 1433。

データベース名

オペレーショナル参照ストアの名前。

ユーザー名

オペレーショナル参照ストアのユーザー名。オペレーショナル参照ストアの作成時に指定するユーザー名は、デフォルトでこの名前になります。このユーザーは、Hub ストア内のオペレーショナル参照ストアデータベースオブジェクトをすべて所有します。

注: Microsoft SQL Server のユーザー名を指定する必要はありません。

パスワード

オペレーショナル参照ストアのユーザー名に関連付けられているパスワード。

DDM 接続 URL

オプション。Dynamic Data Masking サーバーの URL。Dynamic Data Masking を使用しない場合は、空のままにする。

また、次に示すその他のプロパティを **【サマリ】** ページで設定できます。

接続 URL

接続 URL。接続ウィザードでは、デフォルトで接続 URL が生成されます。

登録後にデータソースを作成する

登録後にアプリケーションサーバーのデータソースを作成する場合は選択します。

オペレーショナルリファレンスストアの Oracle 用接続プロパティ

オペレーショナル参照ストアを Oracle に登録する場合は、次の接続プロパティを設定します。

データベース表示名

Hub コンソールに表示する必要のあるオペレーショナル参照ストアの名前。

マシン識別子

Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。

データベースホスト名

Microsoft SQL Server データベースをホストするサーバーの IP アドレスまたは名前。

SID

サーバー上で実行される Oracle データベースのインスタンスを参照する Oracle システム識別子。このフィールドは、SID 接続タイプを選択した場合に表示されます。

サービス

Oracle データベースへの接続に使用する Oracle サービスの名前。このフィールドは、[サービス] 接続タイプを選択した場合に表示されます。

ポート

Oracle データベースサーバー上で実行される Oracle リスナの TCP ポート。デフォルトは 1521。

Oracle TNS 名

ネットワーク上で認識されているデータベースの名前（アプリケーションサーバーの TNSNAMES.ORA ファイルで定義）。

例: mydatabase.mycompany.com

Oracle TNS 名は、Oracle データベースのインストール時に設定します。Oracle TNS 名の詳細については、Oracle のマニュアルを参照してください。

スキーマ名

オペレーショナル参照ストアの名前。

注: [スキーマ名] と [ユーザー名] はどちらも、オペレーショナル参照ストアの作成時に指定したオペレーショナル参照ストアの名前です。この情報が必要な場合はデータベース管理者に問い合わせます。

パスワード

オペレーショナル参照ストアのユーザー名に関連付けられているパスワード。

Oracle の場合、このパスワードに大文字小文字の区別はありません。

デフォルトでは、これがオペレーショナル参照ストアの作成時に指定するパスワードになる。

DDM 接続 URL

オプション。Dynamic Data Masking サーバーの URL。Dynamic Data Masking を使用しない場合は、空のままにする。

また、次に示すその他のプロパティを [サマリ] ページで設定できます。

接続 URL

接続 URL。接続ウィザードでは、デフォルトで接続 URL が生成されます。次の例では、接続 URL のフォーマットを示します。

サービス接続タイプ:

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

SID 接続タイプ:

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

サービス接続タイプのカスタム URL を指定できます。例:

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

登録後にデータソースを作成する

登録後にアプリケーションサーバーのデータソースを作成する場合は選択します。

注: このオプションを選択しない場合、手動でデータソースを設定する必要があります。

オペレーショナルリファレンスストアの IBM DB2 用接続プロパティ

オペレーショナル参照ストアを IBM DB2 に登録する場合は、次の接続プロパティを設定します。

データベース表示名

Hub コンソールに表示する必要のあるオペレーショナル参照ストアの名前。

マシン識別子

Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。

データベースサーバー名

IBM DB2 データベースをホストするサーバーの IP アドレスまたは名前。

データベース名

作成するデータベースの名前

データベースホスト名

IBM DB2 データベースをホストするサーバーの IP アドレスまたは名前。

ポート

IBM DB2 データベースサーバーの TCP ポート。デフォルトは 5000。

スキーマ名

オペレーショナル参照ストアの名前。

注: スキーマ名とユーザー名は、どちらもオペレーショナル参照ストアを作成するときに指定したオペレーショナル参照ストアの名前になる。この情報が必要な場合はデータベース管理者に問い合わせます。

ユーザー名

オペレーショナル参照ストアのユーザー名。デフォルトでは、これがオペレーショナル参照ストアを作成するために使用するスクリプトで指定するユーザー名になる。このユーザーは、Hub ストア内のオペレーショナル参照ストアデータベースオブジェクトをすべて所有する。

パスワード

オペレーショナル参照ストアのユーザー名に関連付けられているパスワード。

IBM DB2 の場合、パスワードでは大文字と小文字が区別される。

デフォルトでは、これがオペレーショナル参照ストアの作成時に指定するパスワードになる。

DDM 接続 URL

オプション。Dynamic Data Masking サーバーの URL。Dynamic Data Masking を使用しない場合は、空のままにする。

また、次に示すその他のプロパティを【サマリ】ページで設定できます。

接続 URL

接続 URL。接続ウィザードでは、デフォルトで接続 URL が生成されます。次の例は、接続 URL のフォーマットを示しています。

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

登録後にデータソースを作成する

登録後にアプリケーションサーバーのデータソースを作成する場合は選択します。

注: このオプションを選択しない場合、手動でデータソースを設定する必要があります。

オペレーショナル参照ストアの登録

オペレーショナル参照ストアを介して Hub コンソールを登録できます。オペレーショナル参照ストアに複数の MDM Hub マスタデータベースを登録することはできません。

注: ORS パスワードの最大文字数は 25 文字です。この制限を超えるパスワードを設定すると、ORS を登録できません。

1. Hub コンソールを開始します。

【データベースの変更】ダイアログボックスが表示されます。

2. MDM Hub マスタデータベースを選択して、**【接続】** をクリックします。

3. 設定ワークベンチにある**データベース**ツールを起動します。

4. 書き込みロックを取得します。

5. **【データベースの登録】** ボタンをクリックします。

Informatica MDM Hub 接続ウィザードが表示され、データベースタイプの選択が求められます。

6. データベースのタイプを選択して **【次へ】** をクリックします。

7. Oracle データベースへの接続を作成する場合は、接続方法を選択して、**【次へ】** をクリックします。

- **【サービス】** を選択して、サービス名を使用して Oracle に接続します。

- **【SID】** を選択して、Oracle システム ID を使用して Oracle に接続します。

注: サービス名と SID 名の詳細については、Oracle のマニュアルを参照してください。

【接続プロパティ】 ページが表示されます。

8. データベースの接続プロパティを設定します。

9. **【サマリ】** ページで変更を確認して、その他の接続プロパティを指定します。

10. **【完了】** をクリックします。

【データベースの登録】 ダイアログボックスが表示されます。

11. **【OK】** をクリックします。

MDM Hub により、オペレーショナル参照ストアが登録されます。

12. 登録したオペレーショナル参照ストアを選択し、**【データベース接続のテスト】** ボタンをクリックしてデータベース設定をテストします。

WebSphere を使用している場合は、データベース接続をテストする前に WebSphere を再起動します。

【データベースのテスト】 ダイアログボックスに、データベース接続テストの結果が表示されます。

13. **【OK】** をクリックします。

オペレーショナル参照ストアが登録され、データベースへの接続がテストされます。

注: 他の場所で使用されているオペレーショナル参照ストアを登録するときに、オペレーショナル参照ストアにプロセスサーバーが登録されている場合、他のサーバーは登録できないことがあります。プロセスサーバーのいずれかを再登録する必要があります。これにより、c_repos_db_release のデータが更新されます。

オペレーショナルリファレンスストア登録プロパティの編集

特定のオペレーショナル参照ストア登録プロパティを編集できます。編集できないプロパティを編集するには、オペレーショナル参照ストアを登録解除して、新しいプロパティを使用して再登録します。

1. データベースツールを起動します。
2. 書き込みロックを取得します。
3. 設定するオペレーショナル参照ストアを選択します。

4. **【データベース接続プロパティの編集】** ボタンをクリックします。
選択したオペレーショナル参照ストアの **【データベースの登録】** ダイアログボックスが表示されます。
5. データベース登録設定を編集します。
 - Oracle の以下のデータベース登録設定を編集します。

プロパティ	説明
データベース表示名	Hub コンソールに表示する必要のあるオペレーショナル参照ストアの名前。
マシン識別子	Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。
Oracle TNS 名	ネットワーク上でデータベースが認識される名前。 TNS 名は TNSNAMES.ORA ファイルで定義されます。
パスワード	オペレーショナル参照ストアの作成時に指定したユーザー名に関連付けられているパスワード。
DDM 接続 URL	オプション。Dynamic Data Masking サーバーの URL。 Dynamic Data Masking を使用しない場合は、空のままにする。

- Microsoft SQL Server の次のデータベース登録設定を編集します。

プロパティ	説明
データベース表示名	Hub コンソールに表示する必要のあるオペレーショナル参照ストアの名前。
マシン識別子	Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。
パスワード	オペレーショナル参照ストアの作成時に指定したユーザー名に関連付けられているパスワード。
DDM 接続 URL	オプション。Dynamic Data Masking サーバーの URL。 Dynamic Data Masking を使用しない場合は、空のままにする。

6. 変更した設定を使用してアプリケーションサーバーのデータソースを更新するには、**【登録後にデータソースを更新】** オプションを有効にして、**【OK】** をクリックします。
データソースパラメータ、および対応する接続プールパラメータをデフォルト値にリセットすることを要求するプロンプトが Hub コンソールに表示されます。
7. デフォルトのデータソースおよび接続プールパラメータにリセットするには、**【はい】** をクリックします。
データソースおよび接続プールパラメータを保持するには、**【いいえ】** をクリックします。
【データベース登録の更新】 ダイアログボックスが表示されます。
8. **【OK】** をクリックします。
データベースツールで変更が保存されます。
9. 更新されたデータベース接続の設定をテストします。

オペレーショナルリファレンスストアのプロパティの編集

登録されたオペレーショナルリファレンスストアのプロパティを変更することができます。

1. データベースツールを起動します。
2. 書き込みロックを取得します。
3. 設定するオペレーショナルリファレンスストアを選択します。

選択したオペレーショナルリファレンスストアのデータベースのプロパティがデータベースツールに表示されます。

以下の表は、データベースのプロパティの説明です。

プロパティ	説明
データベース タイプ	Oracle や Microsoft SQL Server などのデータベースのタイプ。
データベース ID	<p>オペレーショナルリファレンスストアの ID。MDM Hub は、SIF 要求でデータベース ID を使用します。データベース ID のルックアップでは、大文字と小文字が区別されます。</p> <p>Oracle のデータベース ID の形式は以下のとおりです。</p> <ul style="list-style-type: none">- Oracle 接続タイプが SID である場合、形式は hostname-sid-databasename になります。- Oracle 接続タイプがサービスである場合、形式は hostname-sid-databasename になります。 <p>Microsoft SQL Server のデータベース ID の形式は以下のとおりです。</p> <p>hostname-databasename</p> <p>オペレーショナルリファレンスストアを登録すると、ホスト、サーバー、およびデータベースの名前が正規化されます。</p> <p>MDM Hub は、スキーマやテーブルなどのデータベースオブジェクトの標準に従って、ホスト名を小文字に、データベース名を大文字に変換します。</p>
JNDI データ ソース名	<p>選択したオペレーショナルリファレンスストアのデータソースの JNDI 名。アプリケーションサーバーの JDBC 接続に対して MDM Hub で設定した JNDI 名です。</p> <p>Oracle の JNDI データソース名の形式は以下のとおりです。</p> <ul style="list-style-type: none">- Oracle 接続タイプが SID である場合、形式は jdbc/siperian-hostname-sid-databasename-ds になります。- Oracle 接続タイプがサービスである場合、形式は jdbc/siperian-servicename-databasename-ds になります。 <p>Microsoft SQL Server の JNDI データソースの形式は以下のとおりです。</p> <p>jdbc/siperian-hostname-databasename-ds</p>
マシン識別子	Hub ストアインスタンスからのレコードを一意に識別するためにキーに割り当てられるプレフィックス。
GETLIST 制限 (レコード)	searchQuery、searchMatch、getLookupValues などの SIF 検索要求を通じて返されるレコードの最大数。デフォルトは 200、最大値は 5,999 です。

プロパティ	説明
プロダクションモード	<p>オペレーショナルリファレンスストアがプロダクションモードであるかノーマルモードであるかを指定します。</p> <p>無効にすると、権限のあるユーザーがオペレーショナルリファレンスストアのメタデータを Hub コンソールで編集できます。デフォルトでは無効になっています。</p> <p>有効にすると、ユーザーはオペレーショナルリファレンスストアのメタデータを編集することができません。ユーザーがプロダクションモードのオペレーショナルリファレンスストアで書き込みロックを取得しようとする、ロックを取得できないことを説明するメッセージが Hub コンソールに表示されます。</p> <p>注: オペレーショナルリファレンスストアのプロダクションモードを有効または無効にできるのは、MDM Hub 管理者ユーザーだけです。</p>
遷移モード	<p>オペレーショナルリファレンスストアが遷移モードで実行されているかどうかを指定します。遷移モードは、オペレーショナルリファレンスストアのプロダクションモードを有効にした場合に使用できます。</p> <p>有効にすると、ユーザーはリポジトリマネージャの昇格アクションを実行できます。</p> <p>無効にすると、ユーザーはリポジトリマネージャの昇格アクションを実行することができません。</p>
バッチと API の相互運用性	<p>MDM Hub で、SIF API 書き込み呼び出しとバッチ操作を同時処理するために、オペレーショナルリファレンスストアの行レベルのロックを使用できるかどうかを指定します。</p> <p>有効にすると、SIF API 書き込み呼び出しで行レベルのロックが使用可能になり、並行処理とパフォーマンスが向上します。</p> <p>無効にすると、行レベルのロックを使用できなくなります。</p>
ZDT 対応	<p>オペレーショナルリファレンスストアがゼロダウンタイム（ZDT）モードで実行されるかどうかを指定します。</p> <p>有効にすると、オペレーショナルリファレンスストアは ZDT モードで実行されます。</p> <p>無効にすると、オペレーショナルリファレンスストアは ZDT モードで実行されません。</p>

4. プロパティを変更するには、プロパティの横の【編集】ボタンをクリックし、プロパティを編集します。
5. 【保存】ボタンをクリックして変更を保存します。

オペレーショナルリファレンスストアのプロダクションモードを有効にすると、データベースツールで、リスト内のオペレーショナルリファレンスストアの横にロックアイコンが表示されます。

オペレーショナルリファレンスストアの接続

オペレーショナルリファレンスストアとの接続を確保する必要があります。

オペレーショナルリファレンスストアとの接続をテストする場合には、Test Database コマンドによって以下の接続プロパティをテストします。

- データベース接続パラメータ
- データソースの存在
- データソースの接続
- オペレーショナルリファレンスストアのバージョンの有効性

オペレーショナルリファレンスストアの接続テスト

オペレーショナルリファレンスストアとの接続をテストできます。

1. データベースツールを起動します。
2. テストするオペレーショナルリファレンスストアを選択します。
3. **【データベース接続のテスト】** ボタンをクリックします。

【データベースのテスト】 ダイアログボックスが表示されます。

注: WebSphere では、Hub コンソールを使用したテスト接続に失敗した場合は、WebSphere コンソールからの接続を確認します。JNDI 名では大文字小文字の区別があるため、この名前は Hub コンソールで生成された名前とマッチする必要があります。

4. **【OK】** をクリックします。

パスワードの変更

MDM Hub マスターデータベース、またはオペレーショナル参照ストアのパスワードは、MDM Hub をインストールした後で変更できます。

Oracle 環境で MDM Hub マスターデータベースのパスワードを変更する

Oracle 環境で MDM Hub マスターデータベースのパスワードを変更するには、データベースサーバーでパスワードを変更してから、アプリケーションサーバーでパスワードを更新します。

1. データベースサーバーで、CMX_SYSTEM データベースのパスワードを変更します。
2. アプリケーションサーバーの管理コンソールにログインします。
3. データソース接続情報を編集して、ステップ 1 で作成した CMX_SYSTEM のパスワードを指定します。変更を保存します。

IBM DB2 環境で MDM Hub マスターデータベースのパスワードを変更する

IBM DB2 環境で MDM Hub マスターデータベースのパスワードを変更するには、オペレーティングシステムでパスワードを変更してから、アプリケーションサーバーでパスワードを更新します。

1. オペレーティングシステム環境で、CMX_SYSTEM のパスワードを変更します。
2. アプリケーションサーバーの管理コンソールにログインします。
3. データソース接続情報を編集して、ステップ 1 で作成した CMX_SYSTEM のパスワードを指定します。変更を保存します。

オペレーショナルリファレンスストアのパスワードの変更

オペレーショナルリファレンスストアのパスワードを変更することができます。

1. IBM DB2 の環境にいる場合、Microsoft SQL Server の環境でオペレーティングシステムの認定を有効にしている場合、オペレーションシステムのパスワードを変更します。このパスワードはオペレーショナルリファレンスストアのパスワードと同じにする必要があります。
2. データベースサーバーで、オペレーショナルリファレンスストアスキーマのパスワードを変更します。
3. Hub コンソールを起動し、ターゲットデータベースとして MDM Hub マスターデータベースを選択します。
4. **データベースツール** を起動します。
5. 書き込みロックを取得します。

6. 設定するオペレーショナルリファレンスストアを選択します。
7. **【データベースのプロパティ】** パネルで、選択したオペレーショナルリファレンスストアの JNDI データソース名をメモします。
8. アプリケーションサーバーの管理コンソールにログインします。オペレーショナルリファレンスストアのデータソース接続情報を編集し、メモしておいた JNDI データソース名の新しいパスワードを指定して、変更を保存します。
9. データベースツールで、データベースへの接続をテストします。

パスワードの暗号化

スキーマパスワードを変更するには、アプリケーションサーバーで定義されているデータソース内でそのパスワードを変更する必要があります。

このパスワードは、アプリケーションサーバーで保護されているため、暗号化されていません。アプリケーションサーバーでデータソースを更新するとともに、パスワードを暗号化して各種のテーブルに格納する必要があります。

スキーマのパスワードの暗号化

データベーススキーマパスワードを暗号化して、保護することができます。

- ▶ データベーススキーマパスワードを暗号化するには、コマンドプロンプトで以下のコマンドを実行します。

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

結果は端末のウィンドウにエコーされます。

Plaintext Password: *password*

Encrypted Password: *encrypted password*

スキーマのパスワードの更新

MDM Hub マスタデータベースのパスワードまたはオペレーショナルリファレンスストアのパスワードを更新することができます。

1. マスタデータベースのパスワードまたはオペレーショナルリファレンスストアのパスワードを更新するには、cmx_system ユーザーとして接続して次の文を実行します。

Oracle および IBM Db2 の場合

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

Microsoft SQL Server の場合

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

2. アプリケーションサーバを再起動します。

オペレーショナル参照のプロダクションモード

MDM Hub 管理者は Hub コンソールを使用して、オペレーショナル参照ストアのプロダクションモードを有効にすることができます。オペレーショナル参照ストアがプロダクションモードの場合、オペレーショナル参照ストアの設計はロックされています。

オペレーショナル参照ストアがプロダクションモードの場合は、管理権限のないユーザーが書き込みロックを取得することはできません。オペレーショナル参照ストアでスキーマ定義を変更することはできません。プロダクションモードのオペレーショナル参照ストアでロックを取得しようとすると、オペレーショナル参照ストアがプロダクションモードであるためにロックを取得できないことを説明するメッセージが Hub コンソールに表示されます。

オペレーショナル参照のプロダクションモードの有効化または無効化

Hub コンソールを使用して、オペレーショナル参照ストアのプロダクションモードを有効にすることができます。

オペレーショナル参照ストアのプロダクションモードを有効にするには、データベースツールを実行するための管理権限があり、マスターデータベースでロックを取得する必要があります。

1. MDM Hub 実装に対する管理者権限で Hub コンソールにログインします。
2. **データベースツール**を起動します。
3. オペレーショナル参照ストアに対する排他ロックをクリアします。

オペレーショナル参照ストアに排他ロックが設定されている場合は、オペレーショナル参照ストアのプロダクションモードを有効または無効にすることはできません。

4. 書き込みロックを取得します。
5. 設定するオペレーショナル参照ストアを選択します。
データベースツールに、選択したオペレーショナル参照ストアのデータベースのプロパティが表示されます。
6. **【プロダクションモード】** オプションを有効または無効にします。
7. **【保存】** をクリックします。

オペレーショナルリファレンスストアの登録解除

データベースツールを使用して、オペレーショナルリファレンスストア（ORS）を登録解除できます。ORS を登録解除すると、MDM Hub では、ORS との接続情報が MDM Hub マスタデータベースから削除されます。また、ORS を登録解除すると、アプリケーションサーバー環境からデータソース定義が削除されます。

1. MDM Hub コンソールで、**【書き込みロック】** > **【ロックの取得】** の順にクリックします。
2. **【設定】** ワークベンチで、**【データベース】** ツールを選択します。
【データベース情報】 ページが表示されます。
3. データベースのリストから、登録を解除する ORS を選択します。
4. **【データベースの登録解除】** をクリックします。WebLogic を使用している場合は、アプリケーションサーバーのユーザー名とパスワードを入力します。
データベースツールに、ORS の登録解除を確認するメッセージが表示されます。
5. **【はい】** をクリックします。

IBM DB2 でオペレーショナル参照ストアを削除する

オペレーショナル参照ストアを使用しない場合は、データベースからドロップして削除できます。

1. MDM Hub コンソールを使用して、オペレーショナル参照ストアを登録解除します。
2. IBM DB2 コマンドウィンドウを開きます。
3. 次のコマンドを入力します。

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


コマンドで指定されたオペレーショナル参照ストアが削除されました。

データソースの設定

それぞれのオペレーショナルリファレンスストアについて、アプリケーションサーバー環境でデータソースを定義する必要があります。MDM Hub では、データソースでオペレーショナルリファレンスストアのプロパティを指定します。これには、データベースサーバーの場所、データベースの名前、データベースユーザー ID とパスワードなどがあります。

MDM Hub のデータソースは、アプリケーションサーバー環境で定義された JDBC リソースを指します。JDBC データソースの詳細については、アプリケーションサーバーのマニュアルを参照してください。

WebLogic でのデータソースの管理

WebLogic でデータソースの追加、削除、または更新を試みると、MDM Hub から WebLogic の管理者ユーザーの名前とパスワードを指定するように求められます。

データベースツールで複数の操作を実行している場合は、最後に入力したユーザー名がダイアログボックスに表示されますが、パスワードは毎回入力するように要求されます。

データソースの作成

異なるアプリケーションサーバーを使用してオペレーショナルリファレンスストアを作成する場合、またはオペレーショナルリファレンスストアの登録時にデータソースを作成しなかった場合には、データソースを明示的に作成する必要があります。

1. **データベースツール**を起動します。
2. 書き込みロックを取得します。
3. データベースリストでオペレーショナルリファレンスストアを右クリックし、**[データソースの作成]** をクリックします。
4. WebLogic を使用している場合は、表示された入力画面で WebLogic のユーザー名とパスワードを入力します。

デフォルトのパスワード `ChangeMe` が `cmxserver.properties` ファイルで `cmx.server.database.authentication.method=windowsauthentication` と設定されている場合、**[パスワード]** フィールドには `*****` という文字が表示されます。アプリケーションは、Windows 認証方式を使用してデータソースに接続します。

1. データベースツールでデータソースが作成され、進捗メッセージが表示されます。
5. **[OK]** をクリックします。

データソースの削除

オペレーショナルリファレンスストアを登録してデータソースを設定している場合には、データベースツールを使用して、アプリケーションサーバーからデータソース定義を手動で削除できます。

ただしデータソース定義を削除しても、オペレーショナルリファレンスストアは Hub コンソールに表示されたままになります。オペレーショナルリファレンスストアを Hub コンソールから完全に削除するには、登録を解除する必要があります。

1. **データベースツール**を起動します。
2. 書き込みロックを取得します。
3. データベースリストでオペレーショナルリファレンスストアを右クリックし、**[データソースの削除]** をクリックします。

4. WebLogic を実行している場合は、表示された入力画面で WebLogic のユーザー名とパスワードを入力します。
データベースツールによってデータソースが削除され、進捗メッセージが表示されます。
5. **【OK】** をクリックします。

第 8 章

スキーマの構築

この章では、以下の項目について説明します。

- [概要, 79 ページ](#)
- [作業を開始する前に, 79 ページ](#)
- [スキーマについて, 79 ページ](#)
- [スキーママネージャの起動, 98 ページ](#)
- [ベースオブジェクトの設定, 98 ページ](#)
- [テーブルのカラムの設定, 112 ページ](#)
- [ベースオブジェクト間の外部キーリレーションの設定, 120 ページ](#)
- [スキーマの表示, 123 ページ](#)

概要

この章では、Informatica MDM Hub でスキーマを設計および構築する方法について説明します。

作業を開始する前に

作業を開始する前に、Informatica MDM Hub をインストールし、『*Multidomain MDM のインストールガイド*』の手順に従って、Hub ストア（オペレーショナル参照ストアを含む）を作成する必要があります。

スキーマについて

スキーマは、MDM Hub 実装で使用するデータモデルです。

MDM Hub には、必須のスキーマはありません。スキーマは MDM Hub 内に存在し、MDM Hub にデータを提供するソースシステムからは独立しています。

注: MDM Hub 実装用スキーマの設計プロセスは、このドキュメントの対象範囲外になります。業界標準のデータモデリング手法を使って、組織の要件を満たすデータモデルを開発済みであることを前提とします。

Informatica のスキーマは、リポジトリ駆動型の柔軟なモデルであり、どのような階層の事業部門のデータ構造にも対応します。Hub ストアは、MDM Hub 機能のベースとなるデータベースです。Hub ストアはどのインストールにも必ずあり、1 つの MDM Hub マスターデータベースと 1 つ以上のオペレーショナル参照ストアデータベースが含まれています。オペレーショナル参照ストアデータベースは、システム構成に応じて 1 つのインストールに複数含めることが可能です。例えば、開発用のオペレーショナル参照ストア、テスト用のオペレーショナル参照ストア、およびプロダクション用のオペレーショナル参照ストアをそれぞれ含めることができます。

スキーマの実装を開始する前に、基盤となる MDM Hub スキーマとそのコンポーネントの基本的な構造について理解しておく必要があります。

注: 統合スキーマを定義および管理するには、Hub コンソールツールを使用する必要があります。データベースに直接変更を加えることはできません。例えば、テーブルやカラムを定義する場合はスキーママネージャを使用する必要があります。

オペレーショナル参照ストアのテーブルのタイプ

オペレーショナル参照ストアにはユーザーが設定するテーブルとシステムサポートテーブルが含まれています。

設定可能なテーブル

設定可能なテーブルを使用してビジネス参照データをモデル化できます。これらのテーブルは明示的に作成および設定する必要があります。

次の表に、設定可能な MDM Hub テーブルのタイプを示します。

テーブルのタイプ	説明
ベースオブジェクト	主要なビジネスエンティティ（顧客、製品、従業員など）やルックアップテーブル（国、都道府県など）のデータの格納に使用されます。ベースオブジェクトでは、複数のソースシステムのデータを統合し、信頼設定を使用して各ベースオブジェクトセルの最も信頼できる値を特定することができます。ベースオブジェクト間では 1 対多のリレーションを定義できます。ベースオブジェクトは、ユーザー自身で作成し、設定する必要があります。
ランディングテーブル	ソースシステムからバッチロードを受け取るために使用されます。ランディングテーブルは、ユーザー自身で作成し、設定する必要があります。
ステージングテーブル	データをベースオブジェクトにロードするために使用されます。ランディングテーブルとステージングテーブル間のマッピングを定義することにより、データをランディングテーブルからステージングテーブルに移動するときにそのデータのクレンジングと標準化をどのように行うかを指定します。ステージングテーブルは、ユーザー自身で作成し、設定する必要があります。

インフラストラクチャテーブル

MDM Hub インフラストラクチャテーブルは、Hub ストア内でのデータフローを管理し、サポートします。MDM Hub インフラストラクチャテーブルは、ベースオブジェクトを設定するたびに MDM Hub によって作成および設定され、保持されます。

次の表に、MDM Hub インフラストラクチャテーブルのタイプを示します。

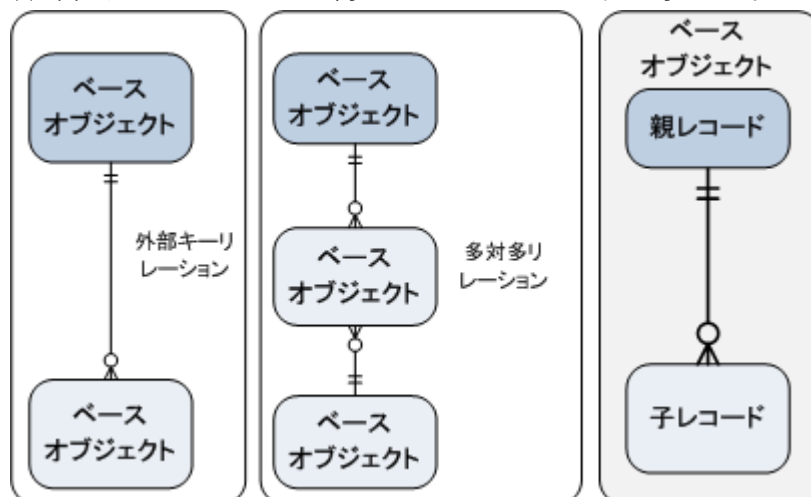
テーブルのタイプ	説明
相互参照テーブル	<p>ベースオブジェクトの各レコードのソースを追跡するために使用されます。</p> <p>相互参照テーブルの名前は次のパターンに従って付けられます。</p> <p>C_<base object name>_XREF</p> <p>ここで、<base object name>はベースオブジェクトのルート名です。例えば、ベースオブジェクト名が PARTY の場合、相互参照テーブル名は C_PARTY_XREF です。ベースオブジェクトを作成すると、ソースシステムからのデータに関する情報を格納するために、MDM Hub によって相互参照テーブルが作成されます。</p>
履歴テーブル	<p>ベースオブジェクトに対して履歴を有効にする場合に使用されます。</p> <p>履歴テーブルの名前は次のパターンに従って付けられます。</p> <ul style="list-style-type: none"> - C_<base object name>_HIST。ベースオブジェクト履歴テーブル。 - C_<base object name>_HXRF。相互参照履歴テーブル。 <p>ここで、<base object name>はベースオブジェクトのルート名です。例えば、PARTY というベースオブジェクトには、C_PARTY_HIST 履歴テーブルと C_PARTY_HXRF 相互参照履歴テーブルが作成されます。</p> <p>MDM Hub では、さまざまなタイプの履歴テーブルが作成および保持されます。これらの履歴テーブルでは、マージおよびマージ解除の履歴、事前にクレンジングされたデータの履歴、ベースオブジェクトの履歴、相互参照の履歴などの詳しい変更追跡オプションを使用できます。</p>
一致キーテーブル	<p>すべてのベースオブジェクトレコードに対して MDM Hub が生成する一致キーが含まれます。一致キーテーブルの名前は次のパターンに従って付けられます。</p> <p>C_<base object name>_STRP</p> <p>ここで、<base object name>はベースオブジェクトのルート名です。例えば、ベースオブジェクト名が PARTY の場合、一致キーテーブル名は C_PARTY_STRP です。</p>
一致テーブル	<p>ベースオブジェクトに対して実行された一致プロセスの結果であるベースオブジェクトの一致レコードのペアが含まれます。一致テーブルの名前は次のパターンに従って付けられます。</p> <p>C_<base object name>_MTCH</p> <p>ここで、<base object name>はベースオブジェクトのルート名です。例えば、ベースオブジェクト名が PARTY の場合、一致テーブル名は C_PARTY_MTCH です。</p>

テーブルのタイプ	説明
外部一致テーブル	<p>外部一致ジョブのデータが含まれます。次のタイプの外部一致テーブルを使用できます。</p> <ul style="list-style-type: none"> - EMI テーブル。外部一致入力テーブル。ベースオブジェクトのレコードと一致するレコードが含まれます。 - EMO テーブル。外部一致出力テーブル。外部一致ジョブの出力データが含まれます。EMO テーブルの各行は、一致したレコードのペアを表します。レコードペアの一方は EMI テーブルから取得されたもので、もう一方はベースオブジェクトから取得されたものです。 <p>外部一致テーブルの名前は次のパターンに従って付けられます。</p> <ul style="list-style-type: none"> - C_<base object name>_EMI - C_<base object name>_EMO <p>ここで、<base object name>はベースオブジェクトのルート名です。例えば、ベースオブジェクト名が PARTY の場合、一致テーブル名は C_PARTY_EMI と C_PARTY_EMO です。</p>
一時テーブル	<p>MDM Hub では、バッチジョブの実行中にデータを処理するために必要な一時テーブルが作成されます。MDM Hub によってデータの処理が完了した後、一時テーブルは不要になるため、バッチプロセスによって削除されます。バッチプロセス、アプリケーションサーバー、またはデータベースサーバーでエラーが発生した場合は、一時テーブルが作成されない可能性があります。</p> <p>一時テーブルにはプレフィックス T\$_が付いています。バッチプロセスが完了した後、バッチプロセスによって作成された一時テーブルがリポジトリに含まれる場合は、一時テーブルを手動で削除できます。一時テーブルの中にはプレフィックス T\$_が付かないものがあります。例えば、クレンジング一時テーブルのプレフィックスは_CL であり、差分一時テーブルのプレフィックスは_DLT です。</p>

サポートされるデータ間のリレーション

MDM Hub では、同じベースオブジェクト内のレコード間の階層リレーションに加えて、テーブル間の 1 対多および多対多のリレーションもサポートされます。MDM Hub では、レコード間のリレーションをいくつかの方法で定義できます。

次の図に、ベースオブジェクト間のさまざまなリレーションを示します。



次の表に、データ間のリレーションタイプを示します。

リレーションのタイプ	説明
ベースオブジェクト間の外部キー関係	一方のベースオブジェクト（子テーブル）には、もう一方のベースオブジェクト（親テーブル）のプライマリキーカラムの値に一致する値を持つ外部キーカラムが含まれています。
同じベースオブジェクト内のレコード	ベースオブジェクト内のレコードは階層的に関連し合っています。ベースオブジェクト内では多対多のリレーションを定義できます。

Hub コンソールでリレーションを設定した後、そのリレーションを使用してレコード間の一致パスを定義し、一致カラムルールを設定できます。

スキーマオブジェクトを定義する際の要件

このセクションでは、スキーマオブジェクトの設定要件について説明します。

スキーマ変更は Hub コンソールでのみ実行する

MDM Hub は、すべてのモデル変更が Hub コンソールツールを使用して実行されていて、データベースに対して直接変更が加えられない場合にのみ、スキーマの一貫性を維持します。MDM Hub には、スキーマを維持するために必要なすべてのツールが揃っています。

スキーマを変更する前に検討する

スキーマの変更はリスクを伴うため、管理された方法でデータを扱う必要があります。変更を実施する前に計画を作成して、影響を分析する必要があります。また、変更前にデータベースのバックアップも作成する必要があります。

スキーマを変更するための書き込みロックが必要

スキーマに変更を加えるには、書き込みロックを設定する必要があります。

データベースオブジェクト名の命名規則

データベースオブジェクト名は 24 文字以下にする必要があります。

データベースオブジェクト名の予約済み文字列

MDM Hub は、ベースオブジェクトの名前にプレフィックスとサフィックスを追加してメタデータオブジェクトを作成します。混乱を避け、データ損失を防ぐために、データベースオブジェクト名には、単独の名前またはカラム名の一部（プレフィックスとサフィックス）どちらについても、次の文字列は使用しないでください。

_BVTB	_OPL	_TMGB	BVTX_	TCMO_	TMF_
_BVTC	_ORAW	_TMIN	BVTXC_	TCRN_	TMMA_
_BVTV	_STRP	_TML0	BVTXV_	TCRO_	TMR_
BO1	_STRPT	_TMMA	CLC_	TCSN_	TPBR_
BO2	_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	_TEMO	_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_	-

予約カラム名

ユーザー定義カラム名のどの部分にも、予約語や予約カラム名を使用することはできません。

例えば、LAST_UPDATE_DATE は予約カラム名であるため、ユーザー定義カラムには使用できません。また、S_LAST_UPDATE_DATE のように、LAST_UPDATE_DATE をユーザー定義カラム名の一部として使用することもできません。

予約カラム名を使用すると、警告メッセージが表示されます。以下に例を示します。

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

次のカラム名の一部または全体をベースオブジェクトのカラム名に使用することはできません。

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	-

次のカラム名の一部または全体をランディングテーブル のカラム名に使用することはできません。

DEP_PKEY_SRC_OBJECT	ROWID_JOB
ERROR_DESCRIPTION	SRC_ROWID
PKEY_SRC_OBJECT	VERSION_SEQ
ROWID	ONLINE_IND

次の単語をカラム名として使用することはできません。

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

Oracle 環境における予約語

次の単語を Oracle 環境におけるカラム名として使用することはできません。

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	VARCHAR2
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

IBM DB2 環境における予約語

次の単語を IBM DB2 環境におけるカラム名として使用することはできません。

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	SQLException
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	FREETEXTABLE	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	

Microsoft SQL Server 環境における予約語

次の単語を Microsoft SQL Server 環境におけるカラム名として使用することはできません。

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	SQLEXCEPTION
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

予約特殊文字

MDM Hub では、すべての特殊文字を使用できますが、一部の特殊文字は、Informatica Data Director で問題が発生する原因となります。そのため、MDM Hub の属性の表示名には次の特殊文字は使用しないでください。

- ! (感嘆符)
- ' (単一引用符)
- # (番号記号)
- & (アンパサンド)
- < (小なり記号)

ただし、特殊文字&と<を SIF CleansePut および Put 要求で使用するために、これらを次の有効な String 値に置き換えることができます。

- &を&に置き換える
- <を<に置き換える

技術的な理由によるカラムの追加

純粹に技術的な理由から、ベースオブジェクトへのカラムの追加が必要な場合があります。例えば、セグメント一致にはセグメントカラムの追加が必要です。

混乱を避けるため、技術的な理由でベースオブジェクトに追加したカラムは、その他のビジネス上の理由で追加したカラムと区別することをお勧めします。技術的な理由で追加したカラムを除外するには、カラム名に CSTM_などの特定の識別子をプレフィックスとして追加します。

スキーママネージャの起動

スキーマ、ステー징テーブル、およびランディングテーブルを定義するには、Hub コンソールでスキーママネージャを使用します。

スキーママネージャは、一致およびマージ、検証、メッセージキューに対するルールの定義でも使用します。

スキーママネージャを開始する手順

1. Hub コンソールでモデルワークベンチを展開して、**[スキーマ]** をクリックします。
2. スキーママネージャが表示されます。

スキーママネージャは2つのペインで構成されています。

ペイン	説明
ナビゲーションペイン	主要なスキーマオブジェクトを（ツリービューで）表示します（ベースオブジェクトおよびランディングテーブル）。ツリーのオブジェクトを展開すると、そのオブジェクトに対して使用可能なプロパティのグループが表示されます。
プロパティペイン	左ペインで選択されているオブジェクトのプロパティが表示されます。スキーマツリー内のノードをクリックすると、対応する（表示および編集可能な）プロパティページが右側のペインに表示されます。

オペレーショナル参照ストアでテーブルを定義するにはスキーママネージャを使用する必要があります。

ベースオブジェクトの設定

MDM Hub では、顧客、アカウント、製品などの中心的なビジネスエンティティがベースオブジェクトというテーブルで表されます。ベースオブジェクトは、個々のエンティティに関するデータのコレクションを含む Hub ストアのテーブルです。

エンティティごとに1つのマスタレコードがあります。マスタレコードがベストバージョン オブ トゥールズです。それ以外に複数バージョンの信頼データが格納されたベースオブジェクトのレコードが含まれる場合があります。これらのレコードはマスタレコードに統合する必要があります。統合とは、重複するレコードを統合され

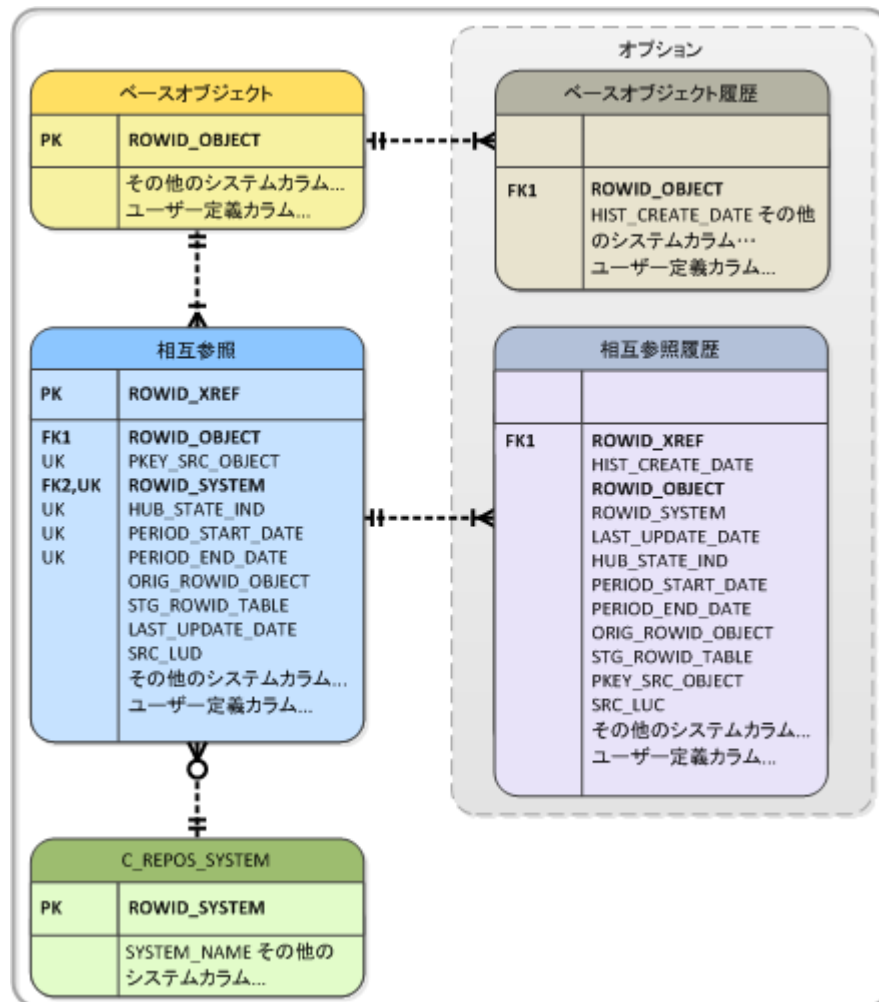
たレコードにマージするプロセスで、統合されたレコードにはすべてのソースレコードのうち最も信頼できるセルの値が格納されます。

Hub コンソールのスキーママネージャを使用して、ベースオブジェクトを定義します。ベースオブジェクトをデータベースで直接設定することはできません。

スキーマに変更を加えると、メタデータの検証によって警告が生成され、MDM Hub がエントリを C_REPOS_MET_VALID_RESULT テーブルに追加します。

ベースオブジェクトと Hub ストア内のその他のテーブルの間のリレーション

以下の図に、ベースオブジェクトと Hub Store 内のその他のテーブルの間のリレーションを示します。



ベースオブジェクトを定義するプロセスの概要

ベースオブジェクトを定義するにはスキーママネージャを使用します。

1. スキーママネージャを使用して、ベースオブジェクトテーブルを作成します。
スキーママネージャによってシステムカラムが自動的に追加されます。
2. ビジネスデータを含むユーザー定義カラムを追加します。

注: カラム名は 26 文字以下にする必要があります。

3. カラムプロパティを設定する際、別個のソースシステムが同じセルに対して異なる値を提供する場合、最も信頼できる値を判定するためにどのカラムを信頼するかを指定します。
4. ベースオブジェクトに対して、ソースシステムごとに 1 つのステージングテーブルを作成します。ステージングテーブルごとに、含めるベースオブジェクトカラムを選択します。
5. データを格納するランディングテーブルをソースシステムから作成します。
6. ランディングテーブルをステージングテーブルにマップします。

データクレンジングが必要なカラムがある場合、マッピングでクレンジング関数を指定します。各ステージングテーブルは、(クレンジング関数を指定せずに) データを 1 つのランディングテーブルから取得する必要がありますが、同一のランディングテーブルは複数のステージングテーブルにデータを提供できます。ランディングテーブルのプライマリキーカラムを、ステージングテーブルの PKEY_SRC_OBJECT カラムにマップします。

7. 各ランディングテーブルに、ETL ツールまたはその他のプロセスを使用してデータを入力します。

ベースオブジェクトのカラム

ベースオブジェクトにはシステムカラムとユーザー定義カラムがあります。システムカラムはスキーママネージャが作成と保持を行うカラムです。ユーザー定義カラムはユーザーが追加するカラムです。

以下の表に、ベースオブジェクトのシステムカラムを示します。

物理名	MDM Hub データタイプ (サイズ)	説明
ROWID_OBJECT	CHAR (14)	プライマリキー。Informatica MDM Hub は、レコードがベースオブジェクトに挿入されるときに一意の値を割り当てます。
CREATOR	VARCHAR (50)	レコードを作成したユーザーまたはプロセス。
CREATE_DATE	TIMESTAMP	レコードが作成された日付。
UPDATED_BY	VARCHAR (50)	レコードに対して最後に更新を行ったユーザーまたはプロセス。
LAST_UPDATE_DATE	TIMESTAMP	レコードのいずれかのセルに対して更新が行われた直近の日付。
CONSOLIDATION_IND	INT	このレコードの統合状態を示す整数値。有効な値は以下のとおりです。 <ul style="list-style-type: none">- 1 = 一意。レコードが最善データであることを表します。- 2 = 統合準備完了。- 3 = 一致準備完了。このレコードは現在実行中のマッチプロセスの一致候補です。- 4 = 一致に利用可能。レコードは新しいため、マッチプロセスで処理する必要があります。- 9 = 保留。データスチュワードによって保留になっています。
DELETED_IND	INT	将来の使用のために予約済み。
DELETED_BY	VARCHAR (50)	将来の使用のために予約済み。
DELETED_DATE	TIMESTAMP	将来の使用のために予約済み。

物理名	MDM Hub データタイプ (サイズ)	説明
LAST_ROWID_SYSTEM	CHAR (14)	ベースオブジェクトレコード内の最後に処理されたセルに更新を行ったシステムの識別子。 LAST_ROWID_SYSTEM は、C_REPOS_SYSTEM テーブルの ROWID_SYSTEM カラムを参照する外部キーです。
DIRTY_IND	INT	DIRTY_IND システムカラムは廃止されました。代わりに、MDM Hub では未処理テーブルと呼ばれるシステムテーブルを使用して、トークン化が必要なレコードを判断します。
INTERACTION_ID	INT	状態が有効なベースオブジェクト向け。INTERACTION_ID は、保留中の相互参照レコードが元の相互参照レコードと異なるプロセスで更新されないようにする処理識別子です。
HUB_STATE_IND	INT	状態が有効なベースオブジェクト向け。このレコードの状態を示す整数値。有効な値は以下のとおりです。 - 0=保留中 - 1=アクティブ - -1=削除済み デフォルトは 1 です。
CM_DIRTY_IND	INT	ゼロダウンタイムプロセスを使用してアップグレードするときに、データが変更されているかどうかを示します。

注: sourceKey や ROWID_OBJECT といったシステムカラムの値には、~や'のような特殊文字を含めないでください。

相互参照テーブル

この節では、Hub ストア内の相互参照テーブルについて説明します。

相互参照テーブルについて

各ベースオブジェクトには、1 つの相互参照 (XREF) テーブルが関連付けられます。XREF テーブルはリネージュを追跡し、タイムラインが有効な場合はさまざまな有効期間に対するレコードバージョンも追跡します。

Informatica MDM Hub では、ベースオブジェクトを作成すると自動的に相互参照テーブルが作成されます。また、その相互参照テーブルを使用して、すべてのソースシステムの識別子が適切な ROWID_OBJECT 値に変換されます。

相互参照テーブルのレコード

相互参照テーブルの各行は、ソースシステムの個別のレコードを表します。1 つのカラムのデータが複数のソースから提供される場合 (例えば、電話番号が CRM システムと ERP システムの両方から提供される場合など)、相互参照テーブルには、それぞれのソースシステムのレコードが別々に格納されます。タイムラインが有効なベースオブジェクトの場合、相互参照テーブルには、ベースオブジェクトレコードのバージョンごとにレコードが別々に格納されます。各ベースオブジェクトレコードには、1 つ以上の相互参照レコードが関連付けられます。

相互参照レコードには、以下の情報が格納されます。

- レコードの提供元のソースシステムの識別子。
- ソースシステム内のそのレコードのプライマリキー値。

- そのシステムから提供された最新のセルの値。
- レコードの元の ROWID_OBJECT の値。
- レコードの元の GBID の値（該当する場合）。
- レコードの有効期間の開始日と終了日（該当する場合）。

ロードプロセスと相互参照テーブル

相互参照テーブルのデータはロードプロセスで取り込まれます。ロードの挿入では、相互参照テーブルに新しいレコードが追加されます。ロードの更新では、影響を受ける相互参照レコードに変更が書き込まれます。

データスチュワードのツールと相互参照テーブル

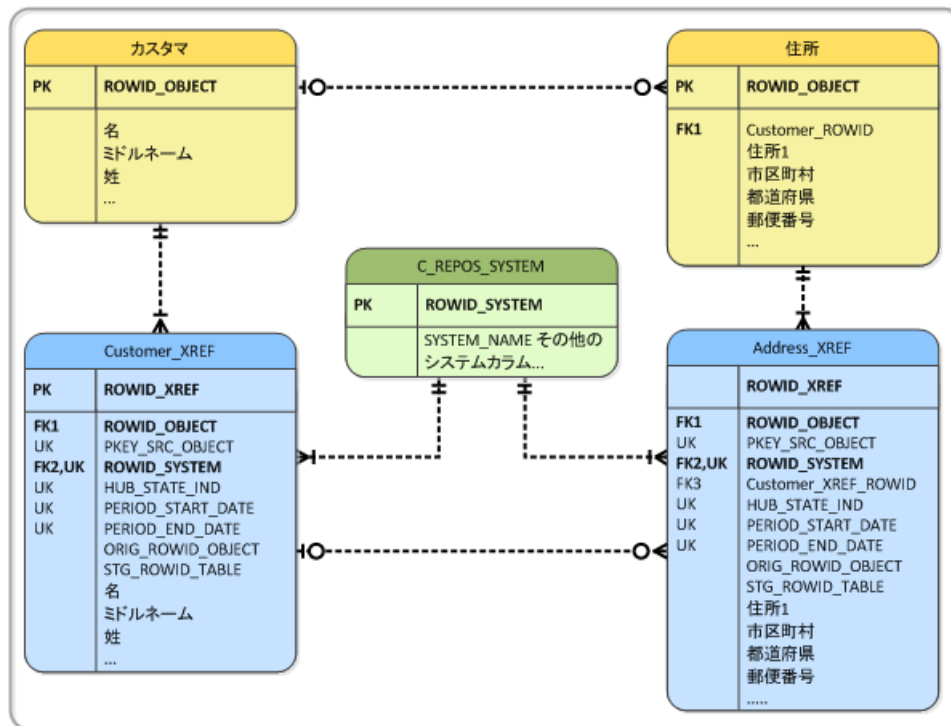
相互参照テーブルレコードは、マージマネージャに表示され、データマネージャを使用して変更できます。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

ベースオブジェクトと相互参照テーブル間のリレーション

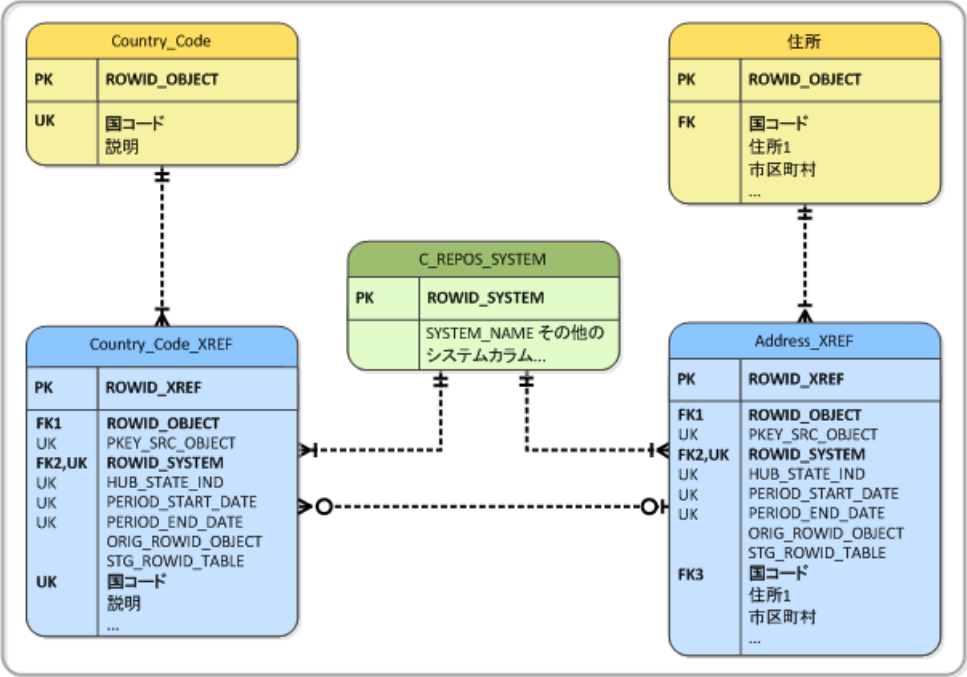
ベースオブジェクトと相互参照テーブルは、次の2つの方法でリンクできます。

- ROWID でリンクする。
- 一意のキーでリンクする。

以下の図は、ROWID を基にしたベースオブジェクト、相互参照テーブル、および C_REPOS_SYSTEM 間のリレーションの例を示しています。



以下の図は、一意のキーを基にしたベースオブジェクト、相互参照テーブル、および C_REPOS_SYSTEM 間のリレーションの例を示しています。



相互参照テーブルカラム

相互参照テーブルには、以下のシステムカラムが含まれます。

注: 相互参照テーブルには、PKEY_SRC_OBJECT カラムと ROWID_SYSTEM カラムの組み合わせを表す一意のキーがあります。

物理名	MDM Hub データ型 (サイズ)	説明
ROWID_XREF	NUMBER (38)	相互参照テーブル内のこのレコードを一意に識別するプライマリキー。
PKEY_SRC_OBJECT	VARCHAR2 (255)	<p>ソースシステムのプライマリキー値。複数のフィールドまたは複数のカラムを含む複数のキーを1つのキー値に結合する必要があります。キーを連結するには、MDM Hub の内部クレンジングプロセス、または ETL ツールやその他のデータロードユーティリティなどの外部クレンジングプロセスを使用します。</p> <p>編集の結果生成される管理システムの相互参照レコードの場合、PKEY_SRC_OBJECT は SYS0:<ROWID_OBJECT>となります (ROWID_OBJECT は編集されたレコードの行 ID です)。</p> <p>PKEY_SRC_OBJECT が SYS0:<ROWID_OBJECT>であり、かつ PUT_UPDATE_MERGE_IND が 1 の場合、MDM Hub はアンマージ中に相互参照レコードを削除します。</p>

物理名	MDM Hub データ型 (サイズ)	説明
ROWID_SYSTEM	CHAR (14)	C_REPOS_SYSTEM に対する外部キー。 C_REPOS_SYSTEM は、ORS にデータを設定できる各ソースシステムの MDM Hub 識別子および説明を格納する MDM Hub リポジトリテーブルです。
ROWID_OBJECT	CHAR (14)	ベースオブジェクトに対する外部キー。MDM Hub は関連するベースオブジェクトレコードに ROWID_OBJECT を割り当てます。
ORIG_ROWID_OBJECT	CHAR (14)	相互参照レコードの元の ROWID_OBJECT。
<i>GBID_Column_Name_GOV</i>	VARCHAR または INT	元のグローバル識別子の値。
STG_ROWID_TABLE	CHAR (14)	MDM Hub が相互参照レコードをロードしたときにルックアップ設定が使用されたステージングテーブルの名前。
<i>S_Foreign_Key_Column_Name</i>	VARCHAR(255)	外部キーカラムのルックアップに使用された値が格納されているシャドウカラム。
SRC_LUD	TIMESTAMP	ソースの最終更新日。ソースシステムが相互参照レコードに更新を送信すると、MDM Hub は SRC_LUD を更新します。
CREATOR	VARCHAR2 (50)	相互参照レコードを作成したユーザーまたはプロセス。
CREATE_DATE	TIMESTAMP	相互参照レコードが作成された日付。
UPDATED_BY	VARCHAR2 (50)	相互参照レコードを最後に更新したユーザーまたはプロセス。
LAST_UPDATE_DATE	TIMESTAMP	相互参照レコード内のいずれかのセルが最後に更新された日付。MDM Hub は、ロードおよび統合プロセス中、必要に応じて LAST_UPDATE_DATE を更新します。
DELETED_IND	NUMBER (38)	将来の使用のために予約済み。
DELETED_BY	VARCHAR2 (50)	将来の使用のために予約済み。
DELETED_DATE	TIMESTAMP	将来の使用のために予約済み。
PERIOD_START_DATE	TIMESTAMP	レコードの有効期間の開始日。タイムラインが有効になったベースオブジェクトの相互参照レコードには PERIOD_START_DATE の値が必要です。
PERIOD_END_DATE	TIMESTAMP	レコードの有効期間の終了日。タイムラインが有効になったベースオブジェクトの相互参照レコードには PERIOD_END_DATE の値が必要です。
PUT_UPDATE_MERGE_IND	NUMBER (38)	この値が 1 の場合、PUT API 要求に ROWID_OBJECT の値を指定することによって、PUT API 呼び出しがそのレコードを更新したことを示します。

物理名	MDM Hub データ型 (サイズ)	説明
INTERACTION_ID	NUMBER (38)	状態が有効なベースオブジェクト向け。保留中の相互参照レコードが元の相互参照レコードと異なるプロセスで更新されないようにするために使用される処理識別子。
HUB_STATE_IND	NUMBER (38)	状態が有効なベースオブジェクト向け。レコードの状態を示す整数値。有効な値は以下のとおりです。 - 0 = 保留中 - 1 = アクティブ - -1 = 削除済み デフォルト値は 1 です。
PROMOTE_IND	NUMBER (38)	状態が有効なベースオブジェクト向け。昇格ステータスを示す整数値。昇格プロセスは PROMOTE_IND を使用して、レコードをアクティブ状態に昇格するかどうかを判断します。有効な値は以下のとおりです。 - 0 = レコードを昇格しない。 - 1 = このレコードをアクティブに昇格する。 昇格プロセスがレコードの昇格に成功しなかった場合、MDM Hub は PROMOTE_IND を 0 に変更しません。

履歴テーブル

履歴テーブルは Hub ストアに存在します。

ベースオブジェクトに対して履歴が有効になっている場合は、Informatica MDM Hub でベースオブジェクトおよび相互参照テーブルの履歴テーブルが保持されます。Informatica MDM Hub には、詳細な変更追跡オプションが用意されています。例えば、マージとアンマージ/マージ解除の履歴、事前クレンジング済みデータの履歴、ベースオブジェクトの履歴、履歴テーブルの相互参照履歴などがあります。

自動マージジョブを実行すると、Informatica MDM Hub は各マージ操作の履歴テーブルにレコードを作成します。同様に、Informatica MDM Hub が子ベースオブジェクトの外部キーを更新するたびに、対応する履歴テーブルにレコードが挿入されます。

ベースオブジェクト履歴テーブル

履歴が有効なベースオブジェクトには、ベースオブジェクトでのデータ変更に関する履歴情報を保存する 1 つの履歴テーブル (C_baseObjectName_HIST) があります。ベースオブジェクトでレコードの追加や更新が行われると、そのイベントをキャプチャする新しいレコードがベースオブジェクト履歴テーブルに挿入されます。

相互参照履歴テーブル

履歴が有効なベースオブジェクトには、相互参照テーブルでのデータ変更に関する履歴情報を保存する 1 つの相互参照履歴テーブル (C_baseObjectName_HXRF) があります。相互参照テーブルでレコードが変更されると、そのイベントをキャプチャする新しいレコードが相互参照履歴テーブルに挿入されます。

ベースオブジェクトのプロパティ

ここでは、ベースオブジェクトの基本プロパティと詳細プロパティについて説明します。

ベースオブジェクトの基本プロパティ

ここでは、ベースオブジェクトの基本プロパティについて説明します。

項目タイプ

追加する必要があるテーブルのタイプ。[ベースオブジェクト] を選択します。

表示名

Hub コンソールに表示する必要があるベースオブジェクトの名前。わかりやすい名前を入力します。表示名は 67 文字未満にします。

物理名

データベース内のテーブルの実際の名前。Informatica MDM Hub では、テーブルの物理名の候補として、入力した表示名に基づく名前が示されます。予約済みの名前である接尾辞は使用しないように注意してください。

データテーブルスペース

データテーブルスペースの名前。読み取り専用。

インデックステーブルスペース

インデックステーブルスペースの名前。読み取り専用。

説明

このベースオブジェクトの簡単な説明。

履歴を有効にする

このベースオブジェクトに対して履歴を有効にするかどうかを指定します。有効にした場合、Informatica MDM Hub はこのベースオブジェクトについて、挿入、更新、または削除されたレコードのログが記録されます。履歴テーブルのこの情報は、監査のために使用できます。

タイムライン

このベースオブジェクトに対してタイムラインを有効にするかどうかを指定します。デフォルトは[タイムラインなし]です。[動的タイムライン]を選択すると、タイムラインが有効になります。タイムラインが有効な場合、MDM Hub はエンティティやリレーションを含むベースオブジェクトレコードのバージョンを管理します。

連続しない有効期間を許可する

レコードに対して連続しない有効期間を使用します。レコードに対して連続しない有効期間を指定できます。レコードの連続した有効期間の使用を無効にします。デフォルトでは無効になっています。

ベースオブジェクトの詳細プロパティ

ここでは、ベースオブジェクトの詳細プロパティについて説明します。

完全トークン化率

変更されたレコードの割合が完全トークン化率を超えると、完全な再トークン化が実行されます。トークン化されるレコードの数がこのしきい値を超えない場合、Informatica MDM Hub により再トークン化が必要なレコードが一致キーテーブルから削除され、それらのレコードのトークンが計算されて、一致キーテーブルに再び挿入されます。デフォルトは 60 です。

注: 削除プロセスには時間がかかることがあります。ただし、プロセスサーバーが高速で、プロセスサーバーとデータベースサーバー間のネットワーク接続も高速である場合は、デフォルトよりもずっと低いしきい値 (10% など) を試すことも可能です。これにより、パフォーマンスが向上するかどうかを確認することができます。

制約を無効にする

初期ロード/更新の実行時（つまりリアルタイムの同時アクセスがない場合）、パフォーマンスを向上させるために、ベースオブジェクトに対する参照整合性の制約を無効にすることができます。デフォルト値は 1 で、制約は無効になっています。

重複一致しきい値

このパラメータは、初期データロードに対する重複データの致ジョブでのみ使用されます。デフォルトは 0 です。この機能を有効にするには、2 以上の値を設定する必要があります。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

ロードバッチサイズ

ロードプロセスでは、ベースオブジェクトのレコードの挿入や更新がバッチに分けられ処理されます。ロードバッチサイズでは、各バッチサイクルでロードするレコード数を指定します（デフォルトは 1000000）。

最大一致経過時間（分）

一致ルールを実行するときのタイムアウトを分単位で指定します。この制限時間を過ぎると、一致プロセスは（一致ルールが手動またはバッチジョブで実行されるたびに）終了します。一致プロセスがバッチジョブの一部として実行される場合は、次の一致に移ります。単独の一致プロセスの場合は、そのプロセスが停止します。デフォルトは 20 です。この値を大きくするのは、一致ルールやデータが非常に複雑な場合だけにしてください。通常、20 分（デフォルト）以内にルールの処理は完了します。

並列度

ベースオブジェクトテーブルとその関連テーブルに対して設定される並列処理の度合いを指定します。すべてのバッチプロセスに対して効果があるわけではありませんが、使用するとパフォーマンスが向上する場合があります。ただし、データベースサーバマシンの CPU の数や使用可能なメモリの量に基づく制約があります。デフォルト値は 1 です。

注: Microsoft SQL Server 環境では並列度を設定できません。

親がマージされたときにキューに再追加する

デフォルト値は NONE です。[親がマージされたときにキューに再追加する] プロパティには、次のいずれかの値を設定できます。

- **NONE。**
子ベースオブジェクトで値が NONE に設定されている場合、親がマージされても子レコードの統合インジケータは変更されません。
- **UNCONSOLIDATED ONLY。**
子ベースオブジェクトで値が UNCONSOLIDATED ONLY に設定されている場合、親がマージされたときに、子レコードの統合インジケータは 4 に変更されます。ただし統合インジケータが 1 に設定されているレコードは例外です。統合インジケータが 1 に設定されている子レコードをキューに再追加するには、[親がマージされたときにキューに再追加する] 設定を手動で 2 に設定する必要があります。
- **ALL。**
子ベースオブジェクトで値が ALL に設定されている場合、親がマージされたときに、統合インジケータが 1 に設定されているレコードも含めてすべての子レコードの統合インジケータは 4 に変更されます。これにより、再一致できるようになります。

ロード時の照合トークンの生成

ロードプロシージャが完了した後で実行するトークン生成プロシージャを有効にします。一致トークンの生成は、依存する子が存在しないベースオブジェクトのロードプロセスの後で有効にすることができます。親をトークン化するには、その前に依存する子をロードする必要があります。ロードプロセスを実行する時間が限られている場合は、ロードプロセスの後で一致トークンの生成を無効にします。デフォルトでは無効になっています。

一致フラグ監査テーブル

一致フラグ監査テーブルを作成するかどうかを指定します。

- 有効にした場合、監査テーブル (*BusinessObjectName_FMHA*) が作成され、マージマネージャで自動マージ用のキューに手動マッチレコードを追加したユーザーのユーザー ID が取り込まれます。
- 無効にした場合、Updated_By カラムに、自動マージバッチジョブを実行したユーザーのユーザー ID が設定されます。

API のロック待機間隔 (秒)

SIF 要求が行レベルのロックを取得する際の最大待機時間 (秒) を指定します。ORS に対して行レベルのロックが有効になっている場合にのみ適用されます。

バッチのロック待機間隔 (秒)

バッチジョブが行レベルのロックを取得する際の最大待機時間 (秒) を指定します。ORS に対して行レベルのロックが有効になっている場合にのみ適用されます。

状態管理を有効にする

ベースオブジェクト内のレコードのシステムの状態を Informatica MDM Hub で管理するかどうかを指定します。デフォルトでは、状態管理は無効になっています。このベースオブジェクトの状態管理を有効にして承認ワークフローをサポートするには、このチェックボックスを選択 (チェック) します。このプロパティが有効なベースオブジェクトのことを、このドキュメントでは **状態が有効なベースオブジェクト** と呼びます。

注: ベースオブジェクトにカスタムクエリがあり、そのベースオブジェクトの状態管理を無効にした場合は、hub_state_ind がカスタムクエリに含まれていなくても、常に警告ポップアップウィンドウが表示されます。

相互参照の昇格の履歴を有効にする

状態が有効なベースオブジェクトについて、状態が保留 (0) からアクティブ (1) に遷移した相互参照レコードの昇格の履歴を Informatica MDM Hub で保持するかどうかを指定します。デフォルトでは、このオプションは無効になっています。

ベースオブジェクトのスタイル

ベースオブジェクトスタイルはマージスタイルです。マージスタイルのベースオブジェクトは、MDM Hub の一致およびマージ機能で使用できます。これはデフォルトで選択されます。

ルックアップインジケータ

MDM Hub Informatica Data Director での値の取得方法を指定します。

- 有効にした場合、Informatica Data Director でルックアップ値のドロップダウンリストが表示されます。
- 無効にした場合、Informatica Data Director で、データテーブルの値を選択するための検索ウィザードが表示されます。

ベースオブジェクトの作成

スキーマにベースオブジェクトを作成します。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーママネージャの左ペインで右クリックし、ポップアップメニューから **【項目の追加】** を選択します。
[テーブルの追加] ダイアログが表示されます。
4. ベースオブジェクトの基本プロパティを指定します。

5. **[OK]** をクリックします。

新しいベーステーブルとそのサポートテーブルがオペレーショナル参照ストア（ORS）内に作成され、スキーマのツリーに新しいベースオブジェクトテーブルが追加されます。

ベースオブジェクトのプロパティの編集

既存のベースオブジェクトのプロパティを編集できます。ベースオブジェクトのプロパティを編集したら、オペレーショナル参照ストアを検証します。

注: オペレーショナル参照ストアを検証しないと、アプリケーションがビジネスエンティティサービスを使用してデータを取得したときに、サービスの呼び出しに失敗してエラーが表示される可能性があります。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマツリーで、変更するベースオブジェクトを選択します。
スキーママネージャで、[ベースオブジェクトのプロパティ] ページの [基本] タブが表示されます。
4. 次の手順に従って、ベースオブジェクトの基本プロパティを編集します。
 - **[編集]** ボタンをクリックして、[表示名] フィールドで新しい値を指定します。
 - **[編集]** ボタンをクリックして、[説明] フィールドで新しい値を指定します。
 - **[履歴を有効にする]** チェックボックスをオンにして、挿入、更新、または削除されたレコードのログを Informatica MDM Hub で記録するようにします。監査には履歴テーブルが使用されます。
 - **[タイムライン]** ドロップダウンリストから、次のいずれかを選択します。
 - ベースオブジェクトでタイムラインを無効にするには、**[タイムラインなし]** を選択します。
 - ベースオブジェクトでタイムラインを有効にするには、**[動的タイムライン]** を選択します。
 - **[連続しない有効期間を許可する]** チェックボックスをオンにして、エンティティやリレーションなど、ベースオブジェクトレコードで連続しない有効期間を使用できるようにします。このプロパティは、タイムライン対応ベースオブジェクトに対してのみ設定できます。
5. ベースオブジェクトのその他のプロパティを変更するには、**[詳細設定]** タブをクリックします。
6. このベースオブジェクトの詳細プロパティを指定します。
7. 左ペインで、ベースオブジェクト名の下に **[一致/マージ設定]** をクリックします。
8. 一致/マージオブジェクトのプロパティを指定します。少なくとも以下のプロパティの設定を検討してください。
 - 手動統合の最大一致数
 - 一致ジョブバッチサイクルあたりの行数プロパティを編集するには、**[編集]** ボタンをクリックし、新しい値を入力します。
9. **[保存]** ボタンをクリックして変更を保存します。

スキーマに変更を加えると、メタデータの検証によって警告が生成され、MDM Hub がエントリを C_REPOS_MET_VALID_RESULT テーブルに追加します。

ベースオブジェクトのカスタムインデックス

カスタムインデックスを作成すると、バッチジョブや SIF の API のパフォーマンスを改善できます。

Informatica MDM Hub はプライマリキーと一意のカラムに対してシステムインデックスを作成します。また、その他のカラムに対してカスタムインデックスを作成して、パフォーマンスを向上させることもできます。カスタムインデックスの値は一意でない可能性があります。

例えば、外部アプリケーションが SIF の API の SearchQuery 要求を呼び出し、ベースオブジェクトを姓で検索するとします。この場合、姓のカラムにカスタムインデックスを作成すると、検索のパフォーマンスが向上します。

バッチジョブがシステムインデックスと登録済みのカスタムインデックスを削除して再作成することで、パフォーマンスが向上します。SIF の API の RegisterCustomIndex を使用してカスタムインデックスを登録します。

ベースオブジェクトにグローバルビジネス識別子のインデックスがある場合、カスタムインデックスは作成できません。ベースオブジェクトのいずれかのカラムがグローバルビジネス識別子である場合、ベースオブジェクトにはグローバルビジネス識別子のインデックスが存在します。

Microsoft SQL Server の制約により、Microsoft SQL Server 上の MDM Hub では、16 を超えるカラムにカスタムインデックスを作成することはできません。

[カスタムインデックスのセットアップ] ノードへの移動

[カスタムインデックスのセットアップ] ノードに移動する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマツリーで [ベースオブジェクト] ノードを展開し、操作するベースオブジェクトのノードを展開します。
4. **[カスタムインデックスのセットアップ]** ノードをクリックします。
[カスタムインデックスのセットアップ] ページが表示されます。

カスタムインデックスの作成

ベースオブジェクトのカスタムインデックスを作成するには、ベースオブジェクトのカスタムインデックスのセットアップを設定します。ベースオブジェクトについては、複数のカスタムインデックスを作成できます。

1. **モデルワークベンチ**で、**[スキーマ]** を選択します。
2. スキーママネージャで、目的のベースオブジェクトの **[カスタムインデックスのセットアップ]** ノードに移動します。
3. **[追加]** ボタンをクリックします。
スキーママネージャでは、`NI_C_<base_object_name>_inc` という名前のカスタムインデックスが作成されます。`inc` は増分数です。
4. **[インデックス内のカラム]** ペインのベースオブジェクトカラムのリストで、カスタムインデックスに含めるカラムを選択します。**[保存]** ボタンをクリックします。

カスタムインデックスの編集

カスタムインデックスを変更するには、既存のカスタムインデックスを削除し、必要なカラムを使用して新しいカスタムインデックスを追加する必要があります。

カスタムインデックスの削除

カスタムインデックスを削除する手順

1. スキーママネージャで、目的のベースオブジェクトの **[カスタムインデックスのセットアップ]** ノードに移動します。
2. **[インデックス]** リストで、削除するカスタムインデックスを選択します。

3. **【削除】** ボタンをクリックします。
削除を確認するように求められます。
4. **【はい】** をクリックします。

MDM Hub の外でのカスタムインデックスの作成

MDM Hub の外にカスタムインデックスを作成するには、お使いのデータベースのデータベースユーティリティを使用します。

MDM Hub の外にインデックスを作成することで、特殊な操作をサポートすることができます。例えば、関数ベースのインデックス（インデックス式の Upper>Last_Name)など）を作成することができます。MDM Hub の外にカスタムインデックスを作成する場合、インデックスは登録できません。MDM Hub は未登録のインデックスを保持できません。インデックスを保持しない場合、バッチジョブのパフォーマンスは低下することがあります。

ベースオブジェクトの影響分析の表示

スキーママネージャを使用すると、ベースオブジェクトに関連付けられているすべてのテーブル、パッケージ、およびクエリを表示できます。

通常は、他の関連付けられたオブジェクトを誤って削除しないようにするために、ベースオブジェクトを削除する前にこの操作を実行します。

注: カラムがベースクエリから削除されると、それに依存するクエリとパッケージが完全に削除されます。

ベースオブジェクトの影響分析を表示する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマツリーで、表示するベースオブジェクトを選択します。
4. 右クリックし、**【影響分析】** を選択します。
スキーママネージャに **【テーブル影響分析】** ダイアログボックスが表示されます。
5. **【閉じる】** をクリックします。

ベースオブジェクトの削除

ベースオブジェクトを削除する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマツリーで、削除するベースオブジェクトを選択します。
4. 右クリックし、**【削除】** を選択します。
削除を確認するように求められます。
5. **【はい】** を選択します。
スキーママネージャから、ベースオブジェクトを削除する前に影響分析を表示するかどうかを尋ねられます。
6. 影響分析を表示せずにベースオブジェクトを削除する場合は、**【いいえ】** を選択します。
スキーママネージャによって、削除したベースオブジェクトがスキーマツリーから削除されます。

テーブルのカラムの設定

テーブル（ベースオブジェクトまたはランディングテーブル）を作成したら、スキーママネージャを使用してそのテーブルのカラムを定義します。テーブルのカラムは、スキーママネージャを使用して定義する必要があります。データベースで直接設定することはできません。

注: スキーママネージャには、相互参照テーブルや履歴テーブルのカラムも表示されますが、これらについては編集することはできません。

カラムについて

ここでは、テーブルのカラムに関する全般的な情報を示します。

ORS テーブルのカラムのタイプ

Hub ストアのテーブルには、2 種類のカラムがあります。

カラム	説明
システムカラム	Informatica MDM Hub が自動的に作成して保持するカラム。システムカラムはメタデータを含みます。
ユーザー定義カラム	テーブル内のシステムカラムではない任意のカラム。ユーザー定義カラムはスキーママネージャで追加され、このカラムには一般的にビジネスデータが含まれます。

注: システムカラムには Informatica MDM Hub メタデータが含まれます。Informatica MDM Hub メタデータは変更しないでください。変更すると、Informatica MDM Hub で予期しない動作が発生し、データが失われる可能性があります。

カラムのデータ型

MDM Hub には、カラムのデータ型セットがあります。MDM Hub のデータ型は、Oracle、IBM DB2、および Microsoft SQL Server データ型に直接マッピングされます。データベースのデータ型の詳細については、データベースの製品マニュアルを参照してください。

注: カラムのデータ型を変更するには、Hub コンソールを使用します。データベースのデータ型は変更しないでください。データベースでデータ型を変更すると、メタデータの検証プロセスで問題が発生することがあります。

次の表に、MDM Hub のデータ型からデータベースのデータ型へのマッピング方法を示します。

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

MDM Hub	Oracle	IBM DB2	Microsoft SQL Server
TIMESTAMP ²	TIMESTAMP	TIMESTAMP	DATETIME2
NUMBER	NUMBER	DECIMAL	NUMERIC
INT	INTEGER	DECIMAL (31,0)	BIGINT

1. グローバル識別子 (GBID) カラムでは、Microsoft SQL Server での行の幅制限によって発生する問題を避けるため、CHAR または NCHAR の代わりに VARCHAR を使用します。

2. 日付のシステムカラムの場合のデータ型は TIMESTAMP です。日付のユーザー定義カラムの場合、TIMESTAMP データ型は使用できません。代わりに DATE を使用します。

カラムのプロパティ

Informatica MDM Hub カラムのプロパティを設定することができます。

注: MDM Hub では、空の文字列は、その空の文字列に關与するデータベースタイプに關係なく、NULL 値と同等です。

Informatica MDM Hub のカラムには、次のプロパティがあります。

表示名

Hub コンソールが表示するカラムの名前。

物理名

ベースオブジェクト内のカラムの名前。Hub コンソール では、カラムの物理名の候補として、表示名に基づく名前が示されます。

物理的なカラム名に予約カラム名は指定できません。また、ドル記号 (\$) を含めることもできません。

NULL 可能

これが有効な場合、カラムには NULL 値を含めることができます。NULL 可能プロパティを有効にしない場合は、デフォルトの値を指定する必要があります。

NULL 可能プロパティがカラムで無効になっていて、Data Director でまたは SIF PUT 操作中にレコードが更新された場合、相互参照レコードは、更新されたフィールドの値のみを使用して作成されます。NULL にできないフィールドを含むその他の相互参照レコードのフィールドは、すべて NULL 値になります。レコードの最善データ (BVT) の計算中に、NULL にできないフィールドが NULL 値になる場合があります、そうするとエラーが発生します。

BVT の計算で NULL にできないフィールドが確実に考慮に入れられるようにするには、cmxserver.properties ファイルの cmx.server.put.autopopulate.missing.user.columns.bo.list プロパティを設定します。NULL 可能プロパティが無効なカラムを持つベースオブジェクトの名前のカンマ区切りリストに、プロパティ値を設定します。プロパティを設定すると、MDM Hub は、関連するベースオブジェクトの値を使用して、相互参照レコードの NULL 値を更新します。これにより、BVT 計算の間に、NULL にできないフィールドに NULL 値が設定されることがなくなります。

データ型

カラムのデータ型。文字データ型については、長さのプロパティを指定できます。特定の数値データ型については、精度や位取りのプロパティを指定できます。

長さ

文字データ型については、使用できる文字数を指定します。

精度

数値データ型については、数値に使用できる桁数（すべての小数点以下の桁数を含む）を指定します。

スケール

数値データ型については、小数点以下に使用できる桁数を指定します。

デフォルトあり

これが有効な場合、デフォルトの値を指定できます。

デフォルト

カラムのデフォルト値。カラムが NULL 可能ではなく、また値がない場合、Informatica MDM Hub はデフォルトの値を使用します。

一意プロパティを有効にした場合や、NULL 可能プロパティを無効にした場合は、カラムのデフォルト値を指定する必要があります。

信頼

有効な場合、Informatica MDM Hub は信頼スコアの最も高いソースシステム値を使用します。

無効な場合、Informatica MDM Hub は最後に更新された値を使用します。

一意

有効な場合、Informatica MDM Hub はカラムに一意制約を適用します。一意制約が有効な場合、一意なカラムには重複した値が存在しなくなります。ほとんどの組織では、ルックアップ値にソースシステムのプライマリキーを使用しています。一意プロパティを有効にすると、Informatica MDM Hub はカラム内に重複した値を持つレコードを拒否します。一意プロパティを有効にする場合は、そのカラムにデフォルト値を設定する必要があります。

統合ベースオブジェクトの一意プロパティが有効な場合、ベースオブジェクトに挿入を行うと失敗します。これは、Informatica MDM Hub が別のシステムから同じキーをロードしようとするからです。このカラムのキーがすべてのソースシステムで一意であることを確認します。

検証

有効な場合、Informatica MDM Hub はロードプロセス中に検証ルールを適用し、有効ではないセルの値に対する信頼スコアをダウングレードします。検証プロパティを有効にする場合は、検証ルールを設定する必要があります。

NULL 値を適用する

このプロパティは、このカラムで NULL 値が最も信頼できる値の場合に使用するデフォルト値に設定します。プロセスで、ステージングテーブルのカラムの **【NULL の更新を許可する】** プロパティの設定を判別できない場合、このプロパティが使用されます。

- True。有効にすると、このカラムの最も信頼できる値が NULL 値の場合、プロセスではベースオブジェクトレコードに NULL 値を書き込むことができます。
- False。デフォルト。無効にすると、別のソースシステムがこのカラムの非 NULL 値に寄与する限り、プロセスではベースオブジェクトレコードに NULL 値を書き込むことができません。

【NULL 値を適用する】 プロパティは、**【NULL の更新を許可する】** プロパティと同様に機能します。

【NULL の更新を許可する】 プロパティの詳細については、[「ステージングテーブルのカラムのプロパティ」](#)（ページ 366）を参照してください。

GBID

有効な場合、Informatica MDM Hub は、ベースオブジェクトのグローバルビジネス識別子（GBID）としてこのカラムを使用します。GBID カラムは、API アクセスやバッチロードに対していくつでも設定できます。

Oracle では、GBID が有効な場合、カラムを INT データ型として、または最長 255 文字の CHAR、NCHAR、VARCHAR、および NVARCHAR2 データ型として設定する必要があります。

IBM DB2 では、GBID が有効な場合、カラムを INT データ型として、または最長 255 文字の VARCHAR、および NVARCHAR データ型として設定する必要があります。

Microsoft SQL Server では、GBID が有効な場合、カラムを INT データ型として、または最長 255 文字の VARCHAR、および NVARCHAR2 データ型として設定する必要があります。

任意のベースオブジェクトカラムに GBID を有効にすると、Informatica MDM Hub はそのベースオブジェクトのためのインデックスを作成します。ベースオブジェクトにはインデックスがあるため、Informatica MDM Hub ではベースオブジェクトにカスタムインデックスを作成できません。

PUT 可能

システムカラムに対して有効な場合、バッチジョブと SIF の API 要求ではシステムカラムにデータを挿入したり更新することができます。次のシステムカラムに対しては PUT 可能プロパティを有効にできません。

- ROWID_OBJECT
- CONSOLIDATION_IND
- HUB_STATE_IND
- LAST_ROWID_SYSTEM
- CM_DIRTY_IND

PUT API リクエストまたは Cleanse Put API リクエストがシステムカラムを更新しようとしたときに PUT 可能プロパティを有効にしていない場合、API リクエストは失敗します。

ユーザー定義のカラムと INTERACTION_ID および LAST_UPDATE_DATE システムカラムは常に PUT 可能です。

グローバル識別子 (GBID) カラム

グローバル識別子 (GBID) カラムには、ビジネスニーズに基づいてレコードを一意的かつグローバルに識別する共通識別子 (キー値) が含まれています。以下に例を挙げます。

- ERP システム (SAP や Siebel 顧客番号など) や CRM システムなど、MDM Hub 外部のアプリケーションによって定義された識別子。
- 外部の組織で定義されている識別子 (AMA 番号や DEA 番号などの業界固有のコードなど) あるいは、政府が発行した識別子 (社会保障番号、税金 ID 番号、運転免許証番号など)。

注: GBID カラムとして設定するには、カラムのデータタイプが整数型、CHAR、VARCHAR、NCHAR、または NVARCHAR であることが必要です。整数以外のカラムは、長さが正確に 255 文字でなければなりません。

スキーママネージャで、ベースオブジェクト内に複数の GBID カラムを定義することができます。例えば、従業員テーブルに社会保障番号と運転免許証番号のカラムを定義したり、ベンダテーブルに税金 ID 番号を定義したりできます。

マスター識別子 (MID) は、他の製品 (CIF、レガシーハブ、MDM Hub、取引先のハブなど) で使用されている参照システムまたはレコードシステムによって生成される共通の識別子です。MDM Hub での MID は、各種ソースシステムからの個々のレコードを一意的に識別する ROWID_OBJECT です。

GBID は ROWID_OBJECT の代わりにはなりません。GBID は、MDM Hub 実装と外部システムとの統合に役立つ別の方法を提供します。これにより、(『*Multidomain MDM サービスの統合フレームワークガイド*』で説明されている SIF リクエストを使用して) 自分で選択した一意の識別子を使用してデータをクエリしたりデータにアクセスしたりできます。さらに、定義済みの識別子を使用して GBID カラムを設定することにより、カスタムの識別子を定義する必要がなくなります。

GBID は、データのトレーサビリティに役立ちます。トレーサビリティとは、データを追跡してそのリネージュ（統合されたレコードに関与したシステムおよびそのシステムのレコード）を判別できるようにすることです。ベースオブジェクトで GBID カラムを定義すると、スキーママネージャは相互参照テーブルに `<GBID_column_name>` と `<GBID_column_name>_GOV` の 2 つのカラムを作成して、現在および元の GBID 値を追跡します。

例えば、2 件の顧客（それぞれ異なる税金 ID 番号を使用）を 1 つの会社にマージし、一方の税金 ID 番号を生かしてもう一方の番号を廃止するとします。税金 ID 番号カラムを GBID として定義した場合は、MDM Hub で現在の税金 ID 番号と過去の税金 ID 番号の両方を追跡できるため、過去の値を使用して（SIF 要求を通じて）データにアクセスできます。

注: MDM Hub では、GBID カラムのデータ検証やエラー検出を実行しません。ソースシステムに重複する GBID 値がある場合、これらの重複値が MDM Hub に渡されます。

ステージングテーブルのカラム

ステージングテーブルのカラムは、カラムエディタを使用して定義することはできません。ステージングテーブルのカラムは特殊であり、ステージングテーブルのターゲットオブジェクト内の一部またはすべてのカラムに基づいて作成されます。ステージングテーブルからデータを取り込めるターゲットテーブルのカラムは、**[ステージングテーブルの追加/編集]** ウィンドウで選択できます。これらのカラムを選択すると、Informatica MDM Hub により、ステージングテーブルの各カラムがターゲットテーブル内の対応するカラムと同じデータ型で作成されます。

ベースオブジェクトのカラム数の上限

自動統合用に設定されている一致ルールがある場合、ベースオブジェクトには 200 を超えるユーザー定義カラムを設定できません。



カラムエディタへの移動








ベースオブジェクトおよびランディングテーブルにカラムを設定する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. カラムを追加するオブジェクトのスキーマツリーを展開します。
4. **[カラム]** を選択します。
プロパティペインにカラム定義が表示されます。
[システムカラムの表示] チェックボックスが選択されている場合は、カラムエディタのシステムカラムの横に「ロック」アイコンが表示されます。

カラムエディタのコマンドボタン

カラムエディタのプロパティペインには、以下のコマンドボタンが表示されます。

ボタン	名前	説明
	カラムの追加	新しいカラムを追加します。
	カラムの削除	既存のカラムを削除します。

ボタン	名前	説明
	カラムの説明の編集	選択したカラムの説明を編集できます。
	上に移動	選択したカラムの表示順序を上に移動します。
	下に移動	選択したカラムの表示順序を下に移動します。
	スキーマのインポート	別のテーブルからカラム定義をインポートして、新しいカラムを追加できます。
	テーブルカラムビューの展開	テーブルカラムビューを展開します。
	テーブルカラムビューのリストア	テーブルカラムビューをリストアします。
	保存	カラム定義に変更を保存します。

システムカラムの表示または非表示

[システムカラムの表示] チェックボックスで、システムカラムの表示と非表示を切り替えることができます。

テーブルカラムビューの展開

プロパティペインを展開して、1つのペインにすべてのカラムプロパティを表示することができます。デフォルトでは、カラム定義は収縮されたビューに表示されます。

展開されたテーブルカラムビューを表示する手順

- **【テーブルカラムビューの展開】** ボタンをクリックします。
展開されたテーブルカラムビューが表示されます。

デフォルトのテーブルカラムビューを表示する手順

- **【テーブルカラムビューのリストア】** ボタンをクリックします。
デフォルトのテーブルカラムビューが表示されます。

カラムの追加

カラムを追加する手順

1. 設定するテーブルのカラムエディタに移動します。
2. 書き込みロックを取得します。
3. **【追加】** ボタンをクリックします。
空の行が表示されます。
4. 各カラムのプロパティを指定します。
5. **【保存】** ボタンをクリックして、追加したカラムを保存します。

別のテーブルからのカラム定義のインポート

別のテーブルからカラム定義をインポートする手順

- 1. 設定するテーブルのカラムエディタに移動します。
- 2. 書き込みロックを取得します。
- 3. **【スキーマのインポート】** ボタンをクリックします。
[スキーマのインポート] ダイアログが表示されます。
- 4. インポートするスキーマの接続プロパティを指定します。
ここで指定する接続情報の詳細については、データベース管理者にお問い合わせください。
- 5. **【次へ】** をクリックします。

注: 入力するデータベースは、現在作業中の Informatica ORS と同じである必要はありません。また、Informatica ORS でなくてもかまいません。

唯一の制限事項として、現在作業中のデータベースとは異なるタイプのリレーショナルデータベースからインポートすることはできません。例えば、使用するデータベースが Oracle データベースの場合は、別の Oracle データベースからのみカラムをインポートできます。

スキーママネージャに、インポート可能なテーブルのリストが表示されます。

- 6. インポートするテーブルを選択します。
- 7. **【次へ】** をクリックします。
選択したテーブルのカラムのリストが表示されます。
- 8. インポートするカラムを選択します。
- 9. **【完了】** をクリックします。
- 10. **【保存】** ボタンをクリックして、追加したカラムを保存します。

カラムのプロパティの編集

カラムが追加および保存された後は、特定のカラムのプロパティを変更することができます。

注: テーブルを定義して変更を保存した後は、CHAR、VARCHAR、NCHAR、および NVARCHAR2 フィールドの長さを減らしたり、NUMBER フィールドの位取りや精度を変更することはできません。

テーブルにデータが入力された後にスキーマ変更を行う場合、カラムに対するそのような変更は計画的かつ制御された方法で管理し、変更の前に適切なデータベースのバックアップが実行されるようにしてください。

- 1. 設定するテーブルのカラムエディタに移動します。
- 2. 書き込みロックを取得します。
- 3. 各カラムに対して次のプロパティを変更できます。

プロパティ	説明
表示名	Hub コンソールに表示されるカラムの名前。
長さ	CHAR、VARCHAR、NCHAR、または NVARCHAR2 フィールドの長さを増やすことのみ可能です。

プロパティ	説明
デフォルト	<p>NULL が許容されないカラムに値が指定されていない場合に使用される値です。</p> <p>注: デフォルトを有効にしても、デフォルトが有効になる前にロードされたレコードは影響を受けません。既存の NULL 値は、NULL のままです。データをリロードして（ステージングテーブルからが推奨されます）、カラムの値が NULL ではないことを確認します。</p>
信頼	<p>カラムが信頼されているカラムであるかどうかを指定します。</p> <p>信頼を有効にした場合、メタデータを同期する必要があります。すでにデータが入っているテーブルのカラムに対して信頼を有効にすると、信頼の設定が変更されたこと、およびこれ以降にテーブルへのロードを行う場合は事前にバッチビューアツールで信頼同期バッチジョブを実行する必要があることが警告されます。</p> <p>Informatica MDM Hub によって、バッチビューアツールで同期ジョブを使用できることが自動的に確認されます。</p> <p>同期プロセスは、追加のロードジョブを実行する前に実行する必要があります。そうしないと、カラムへの入力に使用される信頼値が正しい値でなくなります。</p> <p>信頼を無効にすると、基礎となるメタデータテーブルの一部でカラムが削除され、データが損失します。</p> <p>誤って信頼を無効にして変更を保存した場合は、エラーを訂正するために、信頼を再び有効にし、即座に同期ジョブを実行してメタデータを再作成する必要があります。</p>
一意	<p>カラムの値が一意かどうかを指定します。カラムに一意な値が入っている場合は、一意プロパティを有効にします。あるカラムに対して一意プロパティを有効にする場合は、そのカラムに重複した値が存在しないことを確認します。重複した値が入っているカラムに対して一意プロパティを有効にすることはできません。特に結合されている可能性のあるベースオブジェクトでは一意プロパティを使用しないでください。</p>
検証	<p>カラムに検証が必要かどうかを指定します。検証を無効にすると、関連のカラムに対するメタデータが失われます。</p>
PUT 可能	<p>PUT 可能プロパティを、SIF 要求を使用して、およびハブコンソールで実行するバッチジョブを使用してデータを PUT（挿入または更新）するシステムカラムで有効にします。ROWID_OBJECT、CONSOLIDATION_IND、HUB_STATE_IND、LAST_ROWID_SYSTEM、および CM_DIRTY_IND を除くすべてのシステムカラムに適用されます。</p>

4. **【保存】** ボタンをクリックして変更を保存します。

カラムの表示順の変更

カラムを上下に移動して表示順を変更することができます。表示順を変更しても、データベース内の物理テーブルには影響しません。

カラムの表示順を変更する手順

1. 設定するテーブルのカラムエディタに移動します。
2. 書き込みロックを取得します。
3. 移動するカラムを選択します。
4. 次のいずれかを実行します。
 - 選択したカラムの表示順を上に移すには、**【上に移動】** ボタンをクリックします。
 - 選択したカラムの表示順を下に移すには、**【下に移動】** ボタンをクリックします。
5. **【保存】** ボタンをクリックして変更を保存します。

カラムの削除

カラムの削除は、細心の注意を払って行う必要があります。カラムを削除すると、カラムにロードされたデータはすべて失われます。影響を受ける基本テーブルの数によっては、カラムの削除に時間がかかる可能性があります。

ベースオブジェクトおよびランディングテーブルからカラムを削除する手順

1. Hub コンソールで、スキーマツールを起動します。
2. 設定するテーブルのカラムエディタに移動します。
3. 書き込みロックを取得します。
4. プロパティペインのカラム定義リストから、削除するカラムを選択します。
5. **【カラムの削除】** ボタンをクリックします。
カラムが分析され、影響アナライザが表示されます。
6. 影響アナライザで **【OK】** をクリックして、カラムを削除します。
スキーマツールにより、カラムが削除されます。
7. **【保存】** ボタンをクリックして変更を保存します。

ベースオブジェクト間の外部キーリレーションの設定

ここでは、Informatica MDM Hub 実装のベースオブジェクト間の外部キー関係を設定する方法について説明します。

外部キー関係の概要については、[「外部キーリレーションを定義するプロセスの概要」](#) (ページ 121) を参照してください。

外部キー関係について

Informatica MDM Hub では、外部キーリレーションは、マッチングカラムによって 2 つのベースオブジェクト間の関連付けを確立します。外部キーリレーションでは、一方のベースオブジェクト（子）には外部キーカラムが含まれ、もう一方のベースオブジェクト（親）のプライマリキーカラムの値にマッチする値がこのカラムに格納されます。子ベースオブジェクトの外部キーが親ベースオブジェクトの ROWID_OBJECT を指している場合、関連する子の相互参照テーブルの外部キーは、関連する親の相互参照テーブルの ROWID_XREF を指しています。

オペレーショナル参照ストアテーブルの外部キー関係の種類

Hub ストアテーブルの外部キーリレーションには次の 2 種類があります。

タイプ	説明
システム外部キー関係	スキーマの参照整合性を確保するために、Informatica MDM Hub で自動的に定義されて適用されます。
ユーザー定義の外部キーリレーション	この後に説明する手順に従って手動で定義したカスタムの外部キー関係。

外部キーリレーションを定義するプロセスの概要

外部キーリレーションを作成する手順

1. 親テーブルを作成します。
2. 子テーブルを作成します。
3. これらの間に外部キーリレーションを定義します。

子テーブルに親テーブルからの生成キーが含まれている場合は、ロードプロセスで適切なプライマリーキー値が親テーブルから子テーブルにコピーされます。

外部キー関係の追加

2つのベースオブジェクト間の外部キー関係を追加することができます。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、ベースオブジェクト（リレーションの子側になるベースオブジェクト）を展開します。
4. **[リレーション]** を右クリックします。
[リレーション] ページの [プロパティ] タブが表示されます。
5. **[追加]** ボタンをクリックします。
[リレーションの追加] ダイアログが表示されます。
6. 以下をそれぞれ選択して、新しいリレーションを定義します。
 - [関連元] ツリーのカラム
 - [関連先] ツリーのカラム
7. この外部キーリレーションのインデックスを作成する場合は、**[関連するインデックスを作成する]** チェックボックスをチェック（選択）します。インデックスが存在するオペレーショナル参照ストア内にメタデータが定義されます。
8. **[OK]** をクリックします。
9. **[図]** タブをクリックして、外部キー関係の図を確認します。
10. **[保存]** ボタンをクリックして変更を保存します。

注: リレーションを作成した後に、戻って別のリレーションを作成する場合、このカラムは使用中であるため表示されません。リレーションを削除すると、カラムが表示されるようになります。

外部キー関係編集

外部キーリレーションでは、ルックアップ表示名のみを変更できます。その他のプロパティを変更するには、リレーションを削除して再び追加し、必要なプロパティを指定する必要があります。

2つのベースオブジェクト間の外部キー関係のルックアップ表示名を編集する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、ベースオブジェクトを展開し、**[リレーション]** を右クリックします。
[リレーション] ページの [プロパティ] タブが表示されます。
4. **[プロパティ]** タブで、プロパティを表示する外部キー関係をクリックします。
リレーションの詳細が表示されます。

5. [ルックアップ表示名] の横の【編集】 ボタンをクリックし、新しい値を指定します。
6. 必要に応じて、**【関連するインデックスがある】** チェックボックスを選択してこの外部キーリレーションにインデックスを追加するか、またはチェックボックスをクリアして既存のインデックスを削除します。
7. **【保存】** ボタンをクリックして変更を保存します。

リレーションの詳細

[リレーションの詳細] 画面でテーブルと外部キー間のリレーションの詳細を表示できます。

説明

外部キーとオブジェクト間のリレーションの説明。

制約テーブル

外部キー制約が定義されたテーブルの名前。

制約カラム

外部キー制約が定義されたカラムの名前。

親テーブル

外部キーの親テーブルの名前。

親カラム

親テーブル内にある外部キーの親カラムの名前。

リレーションの適用

- 制限。関連する子レコードが利用可能な場合、MDM Hub は、親テーブルからのレコードの削除を制限します。親テーブルのレコードを削除するには、最初に子テーブルから関連レコードを削除します。
- カスケードの削除。MDM Hub は、親レコードが C_REPOS_TABLE から削除されると、該当するすべてのメタデータを削除します。親テーブルからレコードを削除すると、MDM Hub によって親テーブルからレコードが削除され、子テーブルから関連レコードが削除されます。

タイプ

- 適用済み。MDM Hub によって、適用済みリレーションが作成されます。適用済みリレーションは、データベース内のデータベース制約です。例えば、ベースオブジェクトとその相互参照テーブル間のリレーションが適用されます。適用済みリレーションの問題は、データベースレイヤから報告されます。
- 仮想。ユーザーが仮想リレーションを作成します。MDM Hub では、仮想リレーションの制約を作成しません。仮想リレーションの場合、MDM Hub によって外部キーリレーションのメタデータが内部に格納されます。仮想リレーションの問題は、アプリケーションサーバーを実行したときにだけ報告されます。

関連するインデックスがある

外部キーリレーションにインデックスを追加する場合は、**【関連するインデックスがある】** チェックボックスを選択します。外部キーリレーションから既存のインデックスを削除する場合は、**【関連するインデックスがある】** チェックボックスをクリアします。

ルックアップ表示名

Hub コンソールに表示されるルックアップテーブルの名前。表示名は、**【編集】** ボタンをクリックして変更できます。

外部キーリレーションのルックアップの設定

外部キーリレーションを作成したら、カラムのルックアップを設定できます。ルックアップを設定すると、Informatica MDM Hub のロードプロセスで、親テーブルからデータの値が取得されるようになります。例えば、Address ステージングテーブルに CONSUMER_CODE_FK カラムが含まれている場合、Informatica MDM Hub で Consumer ベースオブジェクトの ROWID_OBJECT カラムに対するルックアップを実行し、Consumer テーブル内の関連付けられた親レコードの ROWID_OBJECT 値を取得することができます。

外部キー関係の削除

追加されたユーザー定義の外部キー関係は、削除することができます。スキーマの参照整合性を保護するために Informatica MDM Hub で自動的に定義されて適用される、システム外部キーリレーションは削除できません。

2つのベースオブジェクト間の外部キーリレーションを削除する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、ベースオブジェクトを展開し、**[リレーション]** を右クリックします。
4. **[プロパティ]** タブで、削除する外部キーリレーションをクリックします。
5. **[削除]** ボタンをクリックします。
削除を確認するように求められます。
6. **[はい]** をクリックします。
スキーママネージャによって、外部キーリレーションが削除されます。
7. **[保存]** ボタンをクリックして変更を保存します。

スキーマの表示

Hub コンソールのスキーマビューアツールを使用して、オペレーショナル参照ストアのスキーマを視覚化できます。スキーマビューアは複雑なスキーマを可視化する際に特に便利です。

スキーマビューアの起動

注: スキーマビューアはリポジトリマネージャ内から起動することもできます (*Multidomain MDM Repository Manager ガイド*を参照)。ただし、起動後のスキーマビューアの使用手順は、どこから起動しても同じです。

スキーマビューアツールを開始する手順

- Hub コンソールでモデルワークベンチを展開して、**[スキーマビューア]** をクリックします。
スキーマビューアが開始され、進行状況ダイアログが表示されてデータモデルがロードされます。
データモデルのロード後、Hub コンソールにスキーマビューアツールが表示されます。








スキーマビューアのペイン

スキーマビューアは2つのペインで構成されています。

ペイン	説明
図ペイン	スキーマの詳細図が表示されます。
概要ペイン	スキーマの概要が表示されます。灰色のボックスに、スキーマ図全体の内で現在、図ペインに表示されている部分がハイライト表示されます。灰色のボックスをドラッグすると、スキーマの対応する部分に表示が移動します。

スキーマビューアのコマンドボタン

スキーマビューアの図ペインには、以下のコマンドボタンが表示されます。

ボタン	名前	説明
	ズームイン	スキーマ図の特定の部分を拡大して表示します。
	ズームアウト	ズームアウトして、より多くの領域を表示します。
	全体を表示	ズームアウトしてスキーマ図全体を表示します。
	レイアウト	階層ビューと直交ビューを切り替えます。
	オプション	カラム名の表示と非表示を切り替え、階層ビューの方向を制御します。
	保存	スキーマ図を保存します。
	印刷	スキーマ図を印刷します。

スキーマ図のズームインとズームアウト

スキーマ図のズームインとズームアウトを実行できます。

ズームイン

スキーマ図の一部にズームインする手順

- **【ズームイン】** ボタンをクリックします。
スキーマビューアで画面の一部が拡大されます。

注: 概要ペインのグレーで強調表示されたボックスは、図ペインに表示されているスキーマの一部を示すために小さくなります。

ズームアウト

スキーマ図をズームアウトする手順

- **【ズームアウト】** ボタンをクリックします。
スキーマビューアがスキーマ図からズームアウトします。
注: 概要ペインのグレーのボックスは、より広範囲の表示領域を示すために大きくなります。

全体のズーム

スキーマ図全体をズームする（スキーマ図全体が図ペインに表示されるようにする）手順

- **【全体を表示】** ボタンをクリックします。
スキーマビューアがズームアウトして、スキーマ図全体が表示されます。

スキーマ図のビューの切り替え

スキーマビューアには以下の 2 つのビューが表示されます。

- 階層ビュー（デフォルト）
- 直交ビュー

ビューの切り替え

階層と直交ビューを切り替える手順

- **【レイアウト】** ボタンをクリックします。
別のビューが表示されます。

関連するデザインオブジェクトやバッチジョブへの移動

スキーマビューアでオブジェクトを右クリックするとコンテキストメニューが表示されます。

コンテキストメニューには以下のコマンドが表示されます。

コマンド	説明
＜ベースオブジェクト＞に移動	スキーママネージャが起動し、ベースオブジェクトノードが展開された状態でベースオブジェクトが表示されます。
ステージングテーブルに移動	スキーママネージャが起動し、関連するベースオブジェクトの下に、選択したステージングテーブルが表示されます。
マッピングに移動	マッピングツールが起動し、選択したマッピングのプロパティが表示されます。
ジョブに移動	バッチビューアが起動し、選択したバッチジョブのプロパティが表示されます。
バッチグループに移動	バッチグループツールを起動します。

スキーマビューアのオプションの設定

スキーマビューアのオプションを設定する手順

1. **【オプション】** ボタンをクリックします。
【オプション】ダイアログが表示されます。

2. 必要なオプションを指定します。

ペイン	説明
カラム名を表示する	エンティティのボックスにカラム名を表示するかどうかを制御します。 エンティティのボックスにカラム名を表示する場合は、このオプションをチェック（選択）します。 エンティティのボックスにカラム名を表示せずに、エンティティ名だけを表示する場合は、このオプションをチェック解除（選択解除）します。
方向	スキーマ階層の方向を制御します。次のうち 1 つの値になります。 <ul style="list-style-type: none">- 上から下（デフォルト）: 最上位のノードを一番上にして、階層が上から下の順になります。- 下から上: 最上位のノードを一番下にして、階層が下から上の順になります。- 左から右: 最上位のレベルを左にして、階層が左から右の順になります。- 右から左: 最上位のレベルを右にして、階層が右から左の順になります。

3. **【OK】** をクリックします。

スキーマ図の JPG イメージの保存

JPG イメージとしてスキーマ図を保存する手順

1. **【保存】** ボタンをクリックします。
[保存] ダイアログが表示されます。
2. JPG ファイルを保存するファイルシステムの場所に移動します。
3. JPG ファイルのわかりやすい名前を指定します。
4. **【保存】** をクリックします。
スキーマビューアによってファイルが保存されます。

スキーマ図の印刷

スキーマ図を印刷する手順

1. **【印刷】** ボタンをクリックします。
[印刷] ダイアログが表示されます。
2. 印刷オプションを選択します。

ペイン	説明
印刷領域	印刷する範囲。 [すべて印刷] は、スキーマ図全体を印刷します。 [可視部分を印刷] は、図ペインに現在表示されている部分のみを印刷します。
ページ設定	用紙、方向、マージンなどのページ出力オプション。
プリンタ設定	プリンタオプションは、環境内の使用可能なプリンタによって異なります。

3. **【印刷】** をクリックします。
スキーマ図がプリンタに送信されます。

第 9 章

クエリとパッケージ

この章では、以下の項目について説明します。

- [クエリとパッケージの概要, 128](#) ページ
- [クエリグループ, 131](#) ページ
- [汎用クエリ, 132](#) ページ
- [カスタムクエリ, 139](#) ページ
- [パッケージ, 141](#) ページ

クエリとパッケージの概要

MDM Hub のクエリは、Hub ストアからデータを取得する要求です。要求は、SQL SELECT 文の形式で送信されます。クエリを実行すると、MDM Hub はクエリ SQL 文を Hub ストアを含むデータベースに送信し、データベースはクエリの結果を返します。パッケージはクエリ結果のパブリックビューです。

汎用クエリ

汎用クエリでは、クエリウィザードと基本単位を使用してクエリを定義します。SQL の知識は必要ありません。クエリツールは、選択した基本単位から SQL SELECT 文を生成します。生成された SELECT 文は、サポート対象のすべてのデータベースと連携して機能します。

カスタムクエリ

カスタムクエリでは、独自の SQL SELECT 文を定義します。データベース固有の SQL 構文と文法を使用する場合は、カスタムクエリを作成します。

クエリグループ

クエリグループは、クエリ用のユーザー定義のコンテナです。クエリグループを使用すると、クエリを整理して簡単に検索と実行を行えます。

ヒント: クエリを作成する前にクエリグループを作成すると、クエリの作成時にグループを選択できます。

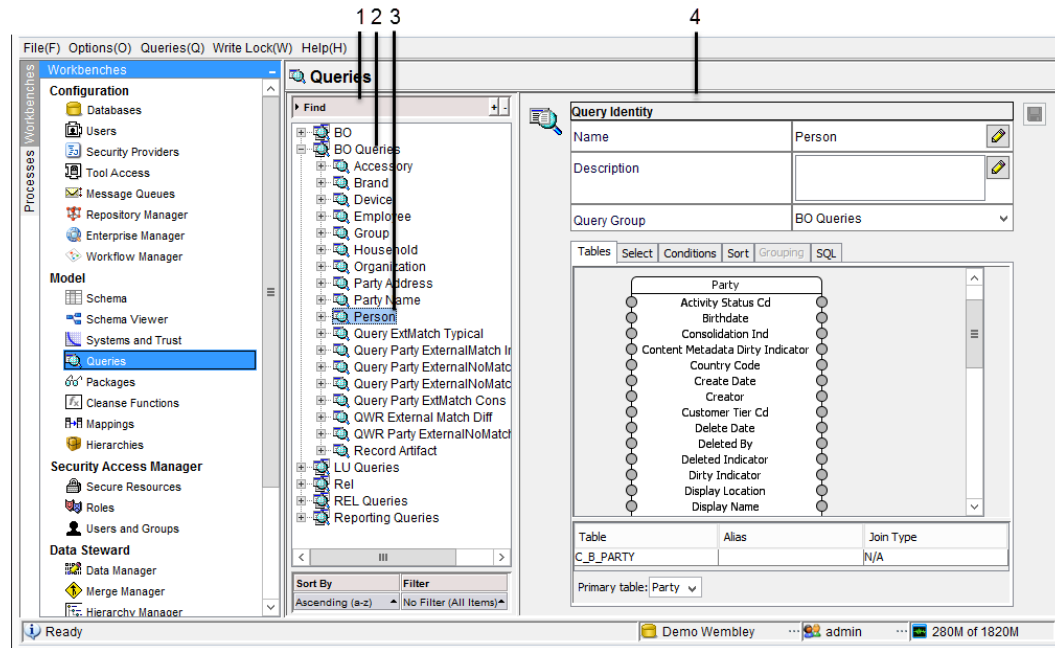
パッケージ

パッケージは、クエリ結果のパブリックビューです。データスチュワードは、データマネージャとマージマネージャでパッケージを使用します。パッケージは外部アプリケーションでも使用できます。

クエリツール

汎用クエリ、カスタムクエリ、およびクエリグループを追加、編集、削除するときは、クエリツールを使用します。クエリ結果を表示したり、クエリに応じたパッケージを表示したりすることもできます。

次の図は、クエリが選択されているクエリツールを示しています。

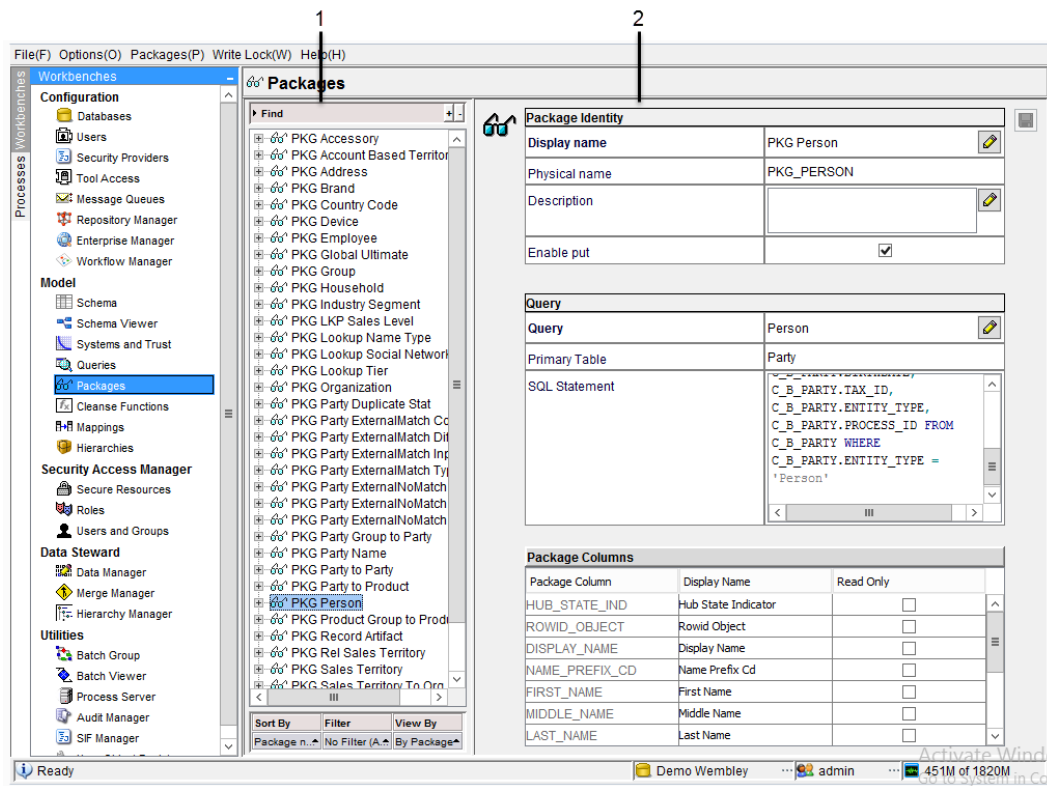


1. ナビゲーションペイン。ユーザー定義のクエリグループおよびクエリが表示されます。
2. クエリグループ。
3. クエリ。
4. プロパティペイン。選択したクエリグループまたはクエリのプロパティが表示されます。

パッケージツール

パッケージツールを使用して、パッケージの追加、編集、削除を行います。

次の図は、パッケージが選択されているパッケージツールを示しています。



1. ナビゲーションペイン。ユーザー定義のパッケージが表示されます。
2. プロパティペイン。選択されたパッケージのプロパティが表示されます。

クエリおよびパッケージのメンテナンス

クエリとパッケージの作成後にデータベーススキーマを変更すると、MDM Hub によりクエリとパッケージが更新されます。更新方法は、スキーマの変更の種類に応じて異なります。場合によっては、一部のクエリとパッケージを手動で編集する必要があります。

次の表に、スキーマ変更の種類、MDM Hub によって行われるアクションの説明、および実行する必要があるアクションについての注意点を示します。

スキーマ変更	汎用クエリとパッケージ	カスタムクエリとパッケージ
改訂されたカラム名	改訂されたカラム名を使用するように汎用クエリとパッケージを更新します。	改訂されたカラム名を使用するようにカスタムクエリとパッケージを更新します。
削除されたカラム	汎用クエリとパッケージから削除されたカラムを削除します。	カスタムクエリは更新されません。削除されたカラムを使用するカスタムクエリとパッケージは有効ではなくなります。 注: クエリとパッケージは、削除されたカラムを削除するように編集する必要があります。
削除されたベースオブジェクト	削除されたベースオブジェクトを使用する汎用クエリとパッケージを削除します。	カスタムクエリは更新されません。削除されたベースオブジェクトを使用するカスタムクエリとパッケージは有効ではなくなります。 注: それらのクエリと依存パッケージを削除する必要があります。

クエリグループ

クエリグループは、クエリ用のユーザー定義のコンテナです。クエリグループを使用すると、クエリを整理して簡単に使用、検索できます。

例えば、次のようなクエリのグループを作成できます。

- 更新パッケージでの使用に適した汎用クエリ。
- 複数のベースオブジェクトテーブルから選択するクエリ。
- ルックアップテーブルから選択するクエリ。

クエリグループの追加

クエリグループを使用してクエリを整理します。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで右クリックし、**【クエリグループの新規作成】** をクリックします。
4. **【クエリグループの追加】** ウィンドウで、クエリグループの名前を入力し、必要に応じて説明を入力します。
5. **【OK】** をクリックします。
ナビゲーションペインにクエリグループがアルファベット順に表示されます。

クエリグループの編集

クエリグループの名前と説明を編集できます。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、クエリグループを選択します。
プロパティペインにプロパティが表示されます。
4. プロパティを編集するには、**【編集】** アイコンをクリックし、テキストを編集して、**【編集を許可】** アイコンをクリックします。
5. **【保存】** アイコンをクリックします。

クエリグループの削除

クエリグループにクエリが含まれている場合は、グループを削除できるようにするため、まずクエリを移動するか削除する必要があります。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、ターゲットのクエリグループを展開します。
4. クエリグループにクエリが含まれている場合は、グループ内のクエリを移動または削除します。
ヒント: クエリを移動するには、クエリを編集し、別のクエリグループを選択します。
5. 空のクエリグループを右クリックし、**【クエリグループの削除】** をクリックします。

汎用クエリ

クエリツールの基本単位を使用して、汎用クエリを作成します。汎用クエリの作成に SQL の知識は必要ありません。

基本単位を選択することで、テーブル名、カラム名、および条件セットなど、データを取得するために使用する条件を指定します。クエリには、結果のソートやグルーピングに関する指示を含めることもできます。MDM Hub により、基本単位から SQL 文が生成されます。生成された SQL 文は、すべてのデータベースで使用できます。

汎用クエリの追加

基本単位を使用して、サポートされるすべてのデータベースで認識できるクエリを作成する場合は、汎用クエリを追加します。

ヒント: SQL 文をコード化する場合は、代わりにカスタムクエリを追加します。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. 必要に応じて、ナビゲーションペインで、クエリを追加するクエリグループを選択します。
4. ナビゲーションペインで右クリックし、**【新しいクエリ】** をクリックします。

新しいクエリウィザードが開きます。

5. **【ようこそ】** 画面が表示されたら、**【次へ】** をクリックします。
6. 汎用クエリのプロパティを指定して、**【次へ】** をクリックします。

プロパティ	説明
クエリ名	クエリの分かりやすい名前を入力します。
説明	必要に応じて、クエリの説明を入力します。
クエリグループ	必要に応じて、別のクエリグループを選択します。
プライマリテーブルの選択	データの取得元となるテーブルを選択します。

7. カラムのサブセットを取得する場合は、カラムを選択します。
 - a. **【クエリカラムを選択】** 画面で、含めるカラムを選択し、他のすべてのカラムのチェックボックスの選択を解除します。
 - b. PUT 対応パッケージでクエリを使用する場合は、**【行 ID オブジェクト】** カラムを選択します。

注: 【行 ID オブジェクト】 は、PUT 対応パッケージの必須カラムです。

8. **【完了】** をクリックします。

クエリは選択したクエリグループ内に表示されます。
9. クエリの結果を表示するには、ナビゲーションペインでクエリを展開し、**【表示】** をクリックします。

クエリツールにクエリの結果が表示されます。

クエリをさらに絞り込むには、クエリの基本単位を使用してクエリを編集します。

汎用クエリの絞り込み

汎用クエリを作成したら、基本単位を使用してクエリを絞り込むことができます。クエリを編集して、基本単位を開きます。各基本単位は、プロパティペインのタブにあります。

次の表に、タブの一覧、基本単位の説明、同等の SQL 構文を示します。

タブ名	基本単位の説明	SQL 構文
テーブル	このクエリに関連付けられているテーブル。	FROM 句
選択	このクエリに関連付けられているカラム。カラムに関数と定数を追加できます。	SELECT <list of columns>
条件	このクエリに関連付けられている条件。個々のレコードの選択条件を決定します。	WHERE 句
ソート	このクエリの結果のソート順。	ORDER BY 句
グルーピング	このクエリの結果のグループ化。	GROUP BY 句
SQL	選択した基本単位から生成された SQL 文を表示します。	すべての句を含む SELECT 文

汎用クエリの編集

クエリ条件を絞り込むには、クエリを編集します。

1. モデルワークベンチで、**[クエリ]** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、クエリを選択します。
プロパティペインにクエリのプロパティが表示されます。
4. 名前または説明を変更するには、**[編集]** アイコンをクリックし、テキストを編集して、**[編集を許可]** アイコンをクリックします。
5. クエリグループを変更するには、**[クエリグループ]** リストからグループを選択します。
6. クエリ条件を絞り込むには、タブを選択し、基本単位を定義します。

関連項目：

- [「追加テーブルの選択」 \(ページ 134\)](#)
- [「カラムの選択」 \(ページ 134\)](#)
- [「関数の定義」 \(ページ 135\)](#)
- [「定数の定義」 \(ページ 135\)](#)
- [「比較条件の定義」 \(ページ 136\)](#)
- [「結果のソート順の定義」 \(ページ 137\)](#)
- [「結果のグルーピングの定義」 \(ページ 137\)](#)

追加テーブルの選択

汎用クエリを作成したときに、クエリ対象のプライマリテーブルを選択しました。クエリを編集するときに、追加のテーブルを追加できます。またテーブル間で外部キーのリレーションを定義することもできます。

注: 更新パッケージでクエリを使用する場合は、追加のテーブルを選択しないでください。更新パッケージの目的は、プライマリテーブルのデータを更新することです。

1. プロパティペインで **【テーブル】** タブをクリックします。
2. **【追加】** アイコンをクリックします。
ヒント: **【追加】** アイコンを利用できない場合、クエリは更新パッケージに関連付けられています。更新パッケージは1つのテーブルのみを参照できます。
3. **【追加するテーブルを選択】** ダイアログボックスで、テーブルを選択して **【OK】** をクリックします。
テーブルがビュー領域とリストに表示されます。リストでは、**【結合タイプ】** が **【クロス】** になっています。
4. 必要に応じて、テーブル間に外部キーのリレーションを作成します。
 - a. ビュー領域で、関連付けるカラムを見つけます。カラムは結合に対応している必要があります。
 - b. 1つのカラムの横の円から別のカラムの横の円まで、コネクタ線をドラッグします。
リストで、**【結合タイプ】** が **【内部】** に変更されます。
 - c. 必要に応じて、リストからタイプを選択して結合タイプを変更します。
注意: 1つのクエリに複数のリレーションが含まれている場合は、リレーションの1つのみが外部結合になります。
5. 必要に応じて、他のテーブルを追加します。
テーブルまたはリレーションの追加でエラーが発生する場合は、テーブルまたはリレーションを削除できます。

オプション	説明
リレーションの削除	ビュー領域でコネクタ線を右クリックして、 【削除】 をクリックします。
テーブルの削除	ビュー領域でテーブルを選択して、 【削除】 アイコンをクリックします。プライマリテーブルは削除できません。

6. **【保存】** アイコンをクリックします。

カラムの選択

各テーブルのカラムのサブセットにクエリを制限できます。削除するカラムを追加した場合は、そのカラムをクエリから削除できます。

1. プロパティペインで、**【選択】** タブをクリックします。
2. **【追加】** アイコンをクリックします。
【テーブルカラムの追加】 ダイアログボックスが開きます。
3. テーブルカラムのリストを展開します。
4. クエリに追加するカラムを選択します。
5. **【OK】** をクリックします。
選択したカラムがテーブルに表示されます。

6. 必要に応じて、カラムの順序を変更したり、削除したりすることができます。

オプション	説明
順序変更	カラムを選択して、 【上に移動】 または 【下に移動】 アイコンをクリックします。
削除	カラムを選択して、 【削除】 アイコンをクリックします。

7. **【保存】** アイコンをクリックします。

関数の定義

クエリのカラムに集計関数を追加することができます。例えば、カラムから取得したデータについて、COUNT、MIN、または MAX を使用できます。また、クエリの関数を編集および削除することもできます。

次のコードサンプルは、SQL 文の関数を示しています。

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

- プロパティペインで、**【選択】** タブをクリックします。
- 【関数の追加】** アイコンをクリックします。
【関数の追加】 ダイアログボックスが開きます。
- カラムを選択します。
- 関数を選択します。
- 【OK】** をクリックします。
関数がテーブルに表示されます。
- 必要に応じて、関数を編集、順序変更、または削除することができます。

オプション	説明
編集	関数を選択して、 【編集】 アイコンをクリックします。
順序変更	関数を選択して、 【上に移動】 または 【下に移動】 アイコンをクリックします。
削除	関数を選択して 【削除】 アイコンをクリックします。

7. **【保存】** アイコンをクリックします。

定数の定義

クエリで取得したカラムデータに適用する定数を追加できます。また、クエリの定数を編集および削除することもできます。

- プロパティペインで、**【選択】** タブをクリックします。
- 【定数の追加】** アイコンをクリックします。
【定数の追加】 ダイアログボックスが開きます。
- データ型を選択します。
- 【値】** フィールドが表示された場合は、データ型に適合する値を入力します。
- 【OK】** をクリックします。
定数がテーブルに表示されます。

6. 必要に応じて、定数を編集、順序変更、または削除することができます。

オプション	説明
編集	定数を選択して、 編集 アイコンをクリックします。
順序変更	定数を選択して、 上に移動 または 下に移動 アイコンをクリックします。
削除	定数を選択して、 削除 アイコンをクリックします。

7. **保存** アイコンをクリックします。

比較条件の定義

クエリで比較条件を定義できます。比較条件には、1つのカラム、1つの SQL 比較演算子、および別のカラムまたは定数が含まれます。クエリを実行すると、そのクエリでカラムの値と条件が比較され、条件を満たすレコードが返されます。

注: カラム内のデータが暗号化されている場合は、文字列やワイルドカード文字を使用する条件を作成できません。

- プロパティペインで、**条件** タブをクリックします。
- 追加** アイコンをクリックします。
比較の追加 ダイアログボックスが開きます。
- [カラム] フィールドで、比較を定義するカラムを選択します。
- [演算子] フィールドで、SQL 比較演算子を選択します。
- 別のカラムや定数などの比較のターゲットを選択します。
 - カラム** を選択する場合は、[カラムの編集] リストからカラムを選択します。
 - 定数** を選択する場合、デフォルト値は NULL です。値を変更するには、**編集** アイコンをクリックし、データ型を選択し、[値] フィールドが表示される場合は値を入力します。

例えば、次の図は、LAST_NAME カラムの値を文字列%JO%と比較する比較条件を示しています。

クエリを実行すると、「Johnson」、「Vallejo」、および「Major」のような値のレコードが返されます。

6. **[OK]** をクリックします。

条件がテーブルに表示されます。

7. 必要に応じて、条件を編集、順序変更、または削除することができます。

オプション	説明
編集	条件を選択して、 編集 アイコンをクリックします。
順序変更	条件を選択して、 上に移動 または 下に移動 アイコンをクリックします。
削除	条件を選択して、 削除 アイコンをクリックします。

8. **保存** アイコンをクリックします。

結果のソート順の定義

データベースでクエリの結果をソートする順序を指定できます。ソートを実行するカラムを選択します。

- プロパティペインで、**ソート** タブをクリックします。
- 追加** アイコンをクリックします。
テーブルカラムの追加 ダイアログボックスが開きます。
- テーブルを見つけ、カラムのリストを展開します。
- ソートに含めるカラムを選択します。
- OK** をクリックします。

選択したカラムが **ソート** テーブルに表示されます。デフォルトでは、カラムが昇順にソートされます。

- 降順にカラムをソートする場合は、**昇順** カラムでチェックボックスの選択を解除します。
- 必要に応じて、カラムの順序を変更したり、削除したりすることができます。

オプション	説明
順序変更	カラムを選択して、 上に移動 または 下に移動 アイコンをクリックします。
削除	カラムを選択して、 削除 アイコンをクリックします。

8. **保存** アイコンをクリックします。

結果のグルーピングの定義

データベースでクエリの結果をグループ化する方法を指定できます。グルーピングに含めるカラムを選択します。

- プロパティペインで、**グルーピング** タブをクリックします。
- 追加** アイコンをクリックします。
テーブルカラムの追加 ダイアログボックスが開きます。
- テーブルカラムのリストを展開します。
- グループに含めるカラムを選択します。
- OK** をクリックします。

選択したカラムがテーブルに表示されます。

6. 必要に応じて、カラムの順序を変更したり、削除したりすることができます。

オプション	説明
順序変更	カラムを選択して、 【上に移動】 または 【下に移動】 アイコンをクリックします。
削除	カラムを選択して、 【削除】 アイコンをクリックします。

7. **【保存】** アイコンをクリックします。

クエリの SQL の表示

汎用クエリの場合、**クエリツール**は、指定した基本単位から SQL 文を生成します。クエリを更新するたびに、SQL 文が生成されます。

- 生成された SQL 文を表示するには、プロパティペインで **【SQL】** タブをクリックします。

クエリ結果の表示

クエリの結果は、クエリツール内でプレビューできます。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. ナビゲーションペインで、クエリが含まれるクエリグループを展開します。
3. クエリを展開します。
4. **【表示】** をクリックします。

クエリツールにクエリの結果が表示されます。

クエリの影響の表示

クエリごとに、その依存関係を表示できます。例えば、クエリを使用するパッケージを表示できます。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. ナビゲーションペインで、クエリが含まれるクエリグループを展開します。
3. クエリを右クリックして、**【影響分析】** をクリックします。
【影響分析】 ダイアログボックスが開きます。
4. クエリを使用するシステムオブジェクトを確認します。
5. **【閉じる】** をクリックします。

クエリの削除

クエリは、使用しなくなったら削除します。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、クエリが含まれるクエリグループを展開します。
4. クエリを右クリックして、**【影響分析】** をクリックします。
【影響分析】 ダイアログボックスが開きます。

5. [パッケージ] セクションにパッケージが含まれる場合は、パッケージ名を書き留め、[閉じる] をクリックします。
【影響分析】 ダイアログボックスが閉じられます。
6. クエリがパッケージにリンクされていた場合は、そのパッケージを削除します。
 - a. **モデルワークベンチ**で、[パッケージ] をクリックします。
 - b. パッケージ名を右クリックし、[パッケージの削除] をクリックします。操作を繰り返して、すべての依存パッケージを削除します。
 - c. **モデルワークベンチ**で、[クエリ] をクリックします。
7. ナビゲーションペインで、クエリを右クリックし、[クエリの削除] をクリックします。

カスタムクエリ

カスタムクエリは、オプションを選択して構築する代わりに、SQL 文を直接入力するクエリです。カスタムクエリは表示パッケージで使用できます。任意のオブジェクトに対してカスタムクエリを作成できます。

IBM DB2 環境でプロキシユーザーを使用する場合は、プロキシユーザーがクエリ内のオブジェクトにアクセスできることを確認してください。リポジトリマネージャにクエリ内のすべてのオブジェクトのメタデータがない場合、リポジトリマネージャを移行すると、クエリに対する特権の警告が発生する可能性があります。警告を修正するには、プロキシユーザーにオブジェクトへのアクセス権を手動で付与します。

カスタムクエリ用 SQL 構文

Hub コンソールは、カスタムクエリで使用できる SQL 構文にいくつかの制約を適用します。これらの制約以外は、データベースがサポートする SQL 構文と文法を使用してください。

次の表は、SQL 構文の制約を示しています。

SQL	制限
文	カスタムクエリは <code>SELECT</code> 文である必要があります。他の SQL 文はサポートされません。
カラム名	カラム名には、英数字とアンダースコアを使用できます。スペースや他の特殊文字はサポートされません。
エイリアス	エイリアス名には、英数字と特殊文字を使用できます。スペースはサポートされません。
定数カラム	定数を一重引用符で囲んで追加するには、エイリアスを使用する必要があります。 例えば、次のクエリではエイリアス <code>const_alias</code> が使用されています。 <code>SELECT ID, 'CONST_COL' AS const_alias FROM c_party</code>
集計関数	特殊文字を含む集計関数を追加するには、エイリアスを使用する必要があります。 例えば、次のクエリではエイリアス <code>new_rowid</code> が使用されています。 <code>SELECT rowid_object*0 AS new_rowid FROM c_party</code>

SQL の検証

カスタムクエリを保存すると、MDM Hub は、SQL 文のクライアント側検証を実行した後に SQL 文をデータベースに送信し、データベースでさらに検証が行われます。

クライアント側 SQL 検証

MDM Hub は、SQL 文が MDM Hub によって要求される SQL 構文を満たしていることを検証します。例えば、クライアント側検証プロセスは、文が SELECT キーワードで始まり、スペースまたは特殊文字が含まれていないことをチェックします。検証プロセスでエラーが検出された場合、エラーメッセージが表示されます。

データベース側 SQL 検証

SQL 文を受け取ったデータベースは、SQL 文の構文と文法がデータベースの要件に準拠しているかどうかを検証します。SQL 文でエラーが生成された場合、データベースはこのエラーを MDM Hub に返し、Hub コンソールにデータベースエラーが表示されます。

カスタムクエリの追加

カスタムクエリでは、データベース固有の SQL 構文を使用できます。SQL 文が、データベースと MDM Hub コンソールの両方の構文要件に準拠していることを確認してください。

1. **モデルワークベンチ**で、**【クエリ】** をクリックします。
2. 書き込みロックを取得します。
3. クエリグループを定義した場合は、クエリを追加するグループを選択します。
4. グループを右クリックし、**【新しいカスタムクエリ】** をクリックします。
カスタムクエリの新規作成ウィザードが表示されます。
5. 最初の画面が表示されたら、**【次へ】** をクリックします。
6. カスタムクエリの以下のプロパティを設定します。

プロパティ	説明
クエリ名	このクエリの分かりやすい名前を入力します。
説明	必要に応じて、このクエリの説明を入力します。
クエリグループ	必要に応じて、選択したクエリグループを変更します。

7. **【完了】** をクリックします。
選択したクエリグループの下ナビゲーションペインにクエリが表示されます。
8. **【SQL】** フィールドの横にある **【編集】** アイコンをクリックします。
9. データベース環境の構文規則に従って SQL クエリを入力します。
10. **【保存】** アイコンをクリックします。
Hub コンソールでクエリが保存され、データベースに送信されます。データベースが SQL 文のエラーを検出すると、クエリツールによってデータベースエラーメッセージが表示されます。エラーを修正し、変更を保存します。
11. クエリの結果を表示するには、ナビゲーションペインでクエリを展開し、**【表示】** をクリックします。
クエリツールにクエリの結果が表示されます。

カスタムクエリの編集

カスタムクエリを編集できます。

1. **モデル**ワークベンチで、**[クエリ]** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、カスタムクエリを選択します。
プロパティペインにプロパティと SQL 文が表示されます。
4. プロパティを編集するには、**[編集]** アイコンをクリックし、テキストを編集して、**[編集を許可]** アイコンをクリックします。
5. **[保存]** アイコンをクリックします。
クエリツールでクエリ設定が検証され、エラーが検出された場合はプロンプトが表示されます。

関連項目：

- [「クエリの削除」 \(ページ 138\)](#)
- [「クエリの影響の表示」 \(ページ 138\)](#)
- [「クエリ結果の表示」 \(ページ 138\)](#)

パッケージ

パッケージは、1 つ以上のクエリのパブリックビューです。パッケージは、Hub コンソールのデータスチュワードツール、およびサービスを使用してマスタデータにアクセスする外部アプリケーションで使用します。

2 種類のパッケージ（表示パッケージと更新パッケージ）を作成できます。表示パッケージでは、マスタデータの読み取り専用ビューを定義します。更新パッケージでは、認証ユーザーがマスタデータを変更できるビューを定義します。

パッケージの使用を認証ユーザーに限定する場合は、パッケージをセキュアリソースにします。すべてのセキュアリソースはロールツールから管理します。詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

表示パッケージ

データスチュワードツールまたは外部アプリケーションのクエリ結果をユーザーが表示できるようにする場合は、表示パッケージを作成します。ユーザーは、クエリ結果を表示できますが、データに変更を加えることはできません。

権限のあるロールごとに、そのロールで使用できる表示パッケージに対する読み取り特権を設定します。他の特権は有効にしないでください。セキュアリソースと特権の詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

更新パッケージ

更新パッケージは、PUT 対応パッケージ、PUT 可能パッケージ、またはマージパッケージとも呼ばれます。更新パッケージを作成するとき、パッケージ内の PUT API を有効にします。データスチュワードは更新パッケージを使用して、マスタデータの追加、変更、マージを行うことができます。

認証されたユーザーが次のアクションを実行できるようにするには、更新パッケージを作成します。

- データマネージャまたは外部アプリケーションから、レコードのデータを更新します。
- データマネージャまたは外部アプリケーションから、レコードを挿入します。
- マージマネージャまたは外部アプリケーションから、レコードをマージします。

認証されたロールごとに、ロールが使用できる更新パッケージの特権を設定します。作成、更新、またはマージ特権のうち 1 つ以上の特権と読み取り特権を付与します。セキュリティソースと特権の詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

更新パッケージの要件

更新パッケージには、パッケージレベルおよびクエリレベルでいくつかの要件があります。

パッケージレベルの要件

更新パッケージは、次の要件を満たしている必要があります。

- 更新パッケージ用の汎用クエリを選択します。更新パッケージでは、カスタムクエリおよび他のパッケージはサポートされていません。
- **[put を有効にする]** オプションを選択します。
- 汎用クエリ内で定義されたテーブルおよびリレーションのみにアクセスします。更新パッケージには、他のテーブルへの結合を含めることはできません。

クエリレベルの要件

汎用クエリは、次の要件を満たしている必要があります。

- プライマリテーブルのみを選択します。汎用クエリには、追加テーブルを含めることはできません。
- 個々のカラムを選択するときは、ROWID_OBJECT カラムを追加する必要があります。
- 汎用テーブルには、システムテーブル、定数カラム、集計関数、またはグループを含めることはできません。

パッケージの追加

表示パッケージまたは更新パッケージのどちらかを作成できます。

ヒント: 更新パッケージを作成する場合は、作成する前に、更新パッケージの要件を満たす汎用クエリを作成します。

1. モデルワークベンチで、**[パッケージ]** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで右クリックして、**[新しいパッケージ]** をクリックします。

新しいパッケージウィザードが開きます。

4. **[ようこそ]** 画面が表示されたら、**[次へ]** をクリックします。

5. 以下のプロパティを設定します。

プロパティ	説明
表示名	パッケージの分かりやすい名前を入力します。この名前はナビゲーションペインに表示されます。
物理名	必要に応じて、提案された物理名を変更します。このウィザードでは、指定した表示名に基づいて物理名の候補が示されます。
説明	必要に応じて、クエリの説明を入力します。
クエリグループ	必要に応じて、別のクエリグループを選択します。
PUT を有効にする	更新パッケージを作成するには、このオプションを選択します。表示パッケージを作成するには、このオプションを選択解除します。
セキュアリソース	パッケージの使用者を制限するには、このオプションを選択します。ロールツールを使用して、セキュアなパッケージにユーザーロールを割り当てます。

6. **[Next (次へ)]** をクリックします。

新しいパッケージウィザードに **[クエリの選択]** ダイアログボックスが表示されます。

7. パッケージで使用するクエリを選択します。

留意: 更新パッケージの場合は、汎用クエリを選択する必要があります。表示パッケージの場合は、汎用クエリまたはカスタムクエリを選択できます。

- 既存のクエリを使用するには、リストからクエリを選択します。
- クエリを作成するには、**[新しいクエリ]** をクリックしてクエリを作成します。
- クエリグループを作成するには、**[クエリグループの新規作成]** をクリックしてクエリグループを作成します。

8. **[完了]** をクリックします。

9. パッケージの結果をプレビューするには、ナビゲーションペインでパッケージを展開して、**[表示]** をクリックします。

パッケージツールにパッケージのプレビューが表示されます。

ヒント: カスタムクエリを選択していて、パッケージの生成に失敗する場合は、そのカスタムクエリが SQL 構文の制約に従っていることを確認してください。

パッケージの編集

パッケージのプロパティと基になるクエリを変更できます。

1. **モデルワークベンチ**で、**[パッケージ]** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、パッケージを選択します。
プロパティペインにプロパティが表示されます。
4. テキストプロパティを編集するには、**[編集]** アイコンをクリックし、テキストを編集して、**[編集を許可]** アイコンをクリックします。
5. **[保存]** アイコンをクリックします。

6. 必要に応じて、このパッケージのクエリを編集できます。
 - a. ナビゲーションペインでパッケージを展開します。
 - b. **【クエリ】** をクリックします。
 - c. クエリを編集します。
7. パッケージの結果をプレビューするには、ナビゲーションペインでパッケージを展開して、**【表示】** をクリックします。
パッケージツールにパッケージのプレビューが表示されます。

クエリ変更後のパッケージの更新

クエリを変更した場合は、クエリを使用するすべてのパッケージを更新します。

注: 更新後、パッケージとクエリとの非同期状態が続く場合、クエリに一致するカラムを選択またはクリアします。

1. **モデルワークベンチ**で、**【パッケージ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインでパッケージを展開します。
4. **【更新】** をクリックします。

パッケージの削除

パッケージが不要になったら、そのパッケージを削除して基本のクエリから依存関係を削除します。

1. **モデルワークベンチ**で、**【パッケージ】** をクリックします。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、パッケージを右クリックし、**【パッケージの削除】** をクリックします。

結合クエリの指定

データマネージャまたはマージマネージャで、データスチュワードがベースオブジェクト情報と他のテーブルの情報を表示できるようにするパッケージを作成できます。

1. ベースオブジェクトに対してクエリを実行する更新パッケージを作成します。
2. 更新パッケージを他のテーブルと結合するクエリを作成します。
3. 作成したクエリに基づいて表示パッケージを作成します。

第 10 章

タイムライン

この章では、以下の項目について説明します。

- [概要, 145 ページ](#)
- [ガイドライン, 146 ページ](#)
- [例, 146 ページ](#)
- [レコードバージョン, 147 ページ](#)
- [タイムラインの粒度, 149 ページ](#)
- [履歴と状態管理, 150 ページ](#)
- [タイムライン適用ルール, 150 ページ](#)
- [ベースオブジェクトのタイムラインの設定, 159 ページ](#)
- [バッチジョブでの複数のレコードバージョンのロード, 160 ページ](#)
- [レコードバージョンの有効期間の編集, 161 ページ](#)
- [レコードバージョンの追加, 164 ページ](#)
- [レコード内のデータの更新, 165 ページ](#)
- [リレーションの更新, 165 ページ](#)
- [リレーションの終了, 168 ページ](#)
- [リレーション期間の削除, 169 ページ](#)
- [すべてのリレーション期間の削除, 170 ページ](#)
- [タイムライン抽出の使用, 170 ページ](#)

概要

ビジネスエンティティとそのリレーションのデータ変更イベントは、タイムライン管理を通して管理できます。データ変更イベントとは、ある期間内で有効なデータに対する変更のことです。データ変更イベントが発生すると、MDM Hub は既存のデータを上書きするのではなく、エンティティのバージョンを作成します。

ビジネスエンティティとそのリレーションのデータ変更イベントやバージョンは、その有効期間に基づいて定義できます。データの変更は、時間の経過とともに発生し、ほかのデータとのリレーションには関係ありません。データに変更が発生すると、有効期間が新しくなったり、既存または今後の有効期間が更新されたりします。タイムライン管理は、データの有効期間への変更を追跡するために使用します。

タイムラインを有効にすることで、ビジネスエンティティ（顧客の住所、電話番号、それらのリレーションなど）のデータイベントを管理できます。ベースオブジェクトレコードの有効期間を管理するために、MDM Hub では、タイムラインを有効にしたベースオブジェクトに関連付けられた相互参照テーブルを使用します。

Hub コンソールでベースオブジェクトプロパティを設定するときには、ベースオブジェクトのタイムラインを有効にします。タイムラインを有効にすると、顧客住所などのデータの過去、現在、未来の変更が MDM Hub によって追跡されます。

階層管理リレーションベースオブジェクトでは、タイムラインはデフォルトで有効になっています。

タイムラインを有効にしたベースオブジェクトには、指定した有効期間に関連するデータのバージョンが含まれます。ベースオブジェクトが現在の値を反映しない可能性があります。任意の時点におけるベースオブジェクトのデータのバージョンは、関与する相互参照テーブルに依存します。これらのテーブルは、時間の経過とともに変化する可能性があります。SIF の API 呼び出しを使用すると、現在の有効日のレコードを取得できます。また、任意の有効日を渡して、指定した有効日のデータを取得することもできます。

追加された変更がベースオブジェクトの現在の有効期間に影響する場合、MDM Hub では現在の信頼設定を使用して BVT を再計算します。次に、MDM Hub でベースオブジェクトレコードを更新します。

注: Hub サーバーでは常に、現在の信頼設定を使用して過去、現在、または将来のデータの最善データ (BVT) を計算します。

ベースオブジェクトのタイムラインを有効にした後は、無効にできません。読み込まれたベースオブジェクトのタイムラインを有効にした場合、Hub サーバーによって有効期間の開始日と終了日が NULL に設定されます。

ガイドライン

ベースオブジェクトのタイムラインを設定する際、MDM Hub はベースオブジェクトとそのリレーションのデータ変更イベントを管理します。

リレーションベースオブジェクトのタイムラインはデフォルトで有効化されていますが、エンティティベースオブジェクトについては有効化されていません。リレーションベースオブジェクトのタイムラインは無効にできません。また、タイムラインを有効にした後、ベースオブジェクトのタイムラインを無効にすることはできません。

ベースオブジェクトのタイムラインを有効にすると、履歴および状態管理もデフォルトで有効になります。タイムラインが有効なベースオブジェクトの履歴または状態管理を無効にすることはできません。

MDM Hub では、タイムラインが有効なベースオブジェクトのベストバージョンオブトゥールズ (BVT) が自動的に更新されません。指定した有効期間のレコードの BVT を表示できます。

例

組織にタイムラインを有効にする顧客ベースオブジェクトがあるとします。CUSTOMER ベースオブジェクトには、2011 年 1 月 31 日から 2013 年 10 月 20 日までの有効期間でロサンゼルスに住んだことがあり、現在はサンフランシスコに住んでいてその有効期間は 2013 年 10 月 21 日から始まり、2015 年 11 月 25 日からはラスベガスに住む予定の John Smith 氏のレコードが含まれています。MDM Hub では、John Smith 氏の住所データのようなデータの過去、現在、未来の変更を追跡します。

過去のデータを含むレコード

顧客ベースオブジェクトには、過去に住んでいたロサンゼルスに住む John Smith 氏のレコードが格納されます。相互参照テーブルには、ロサンゼルスに住んでいた期間を含む John Smith 氏のレコードが格納されます。

John Smith 氏の相互参照レコードは、氏が過去にロサンゼルスに住んでいたことを示しています。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	ロサンゼルス	2011 年 1 月 31 日	2013 年 10 月 20 日

John Smith 氏のベースオブジェクトレコードは、指定した有効日（2012 年 2 月 12 日など）の最善データ（BVT）を示します。

行 ID オブジェクト	顧客 ID	名	姓	市区町村
2	25	John	Smith	ロサンゼルス

過去および現在のデータを含むレコード

顧客ベースオブジェクトには、指定した日の BVT である住所を含む John Smith 氏のレコードが格納されます。相互参照テーブルには、John Smith 氏のレコードが格納されます。相互参照テーブルは、氏がロサンゼルスに住んでいた期間と、サンフランシスコに住んでいた期間を含むレコードを示します。

John Smith 氏の相互参照レコードは、氏が過去にロサンゼルスに住んでいたことを示しています。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	ロサンゼルス	2011 年 1 月 31 日	2013 年 10 月 20 日
2		2	25	John	Smith	サンフランシスコ	2013 年 10 月 21 日	2015 年 11 月 24 日

John Smith 氏のベースオブジェクトレコードには、指定した期間の BVT が示されています（例えば、2012 年 2 月 12 日）。

行 ID オブジェクト	顧客 ID	名	姓	市区町村
2	25	John	Smith	ロサンゼルス

過去、現在および未来のデータを含むレコード

顧客ベースオブジェクトには、指定した日の BVT である住所を含む John Smith 氏のレコードが格納されます。相互参照テーブルには、ロサンゼルスに住んでいた期間を含む John Smith 氏のレコードが格納されます。

過去ロサンゼルスに住み、現在はサンフランシスコに住んでいる、2015 年 11 月 25 日からラスベガスに住む予定の、John Smith 氏の相互参照レコード。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	ロサンゼルス	2011 年 1 月 31 日	2013 年 10 月 20 日
2		2	25	John	Smith	サンフランシスコ	2013 年 10 月 21 日	2015 年 11 月 24 日
3		2	25	John	Smith	ラスベガス	2015 年 11 月 25 日	null

John Smith 氏のベースオブジェクトレコードには、指定した期間の BVT が示されています（例えば、2012 年 2 月 12 日）。

行 ID オブジェクト	顧客 ID	名	姓	市区町村
2	25	John	Smith	ロサンゼルス

レコードバージョン

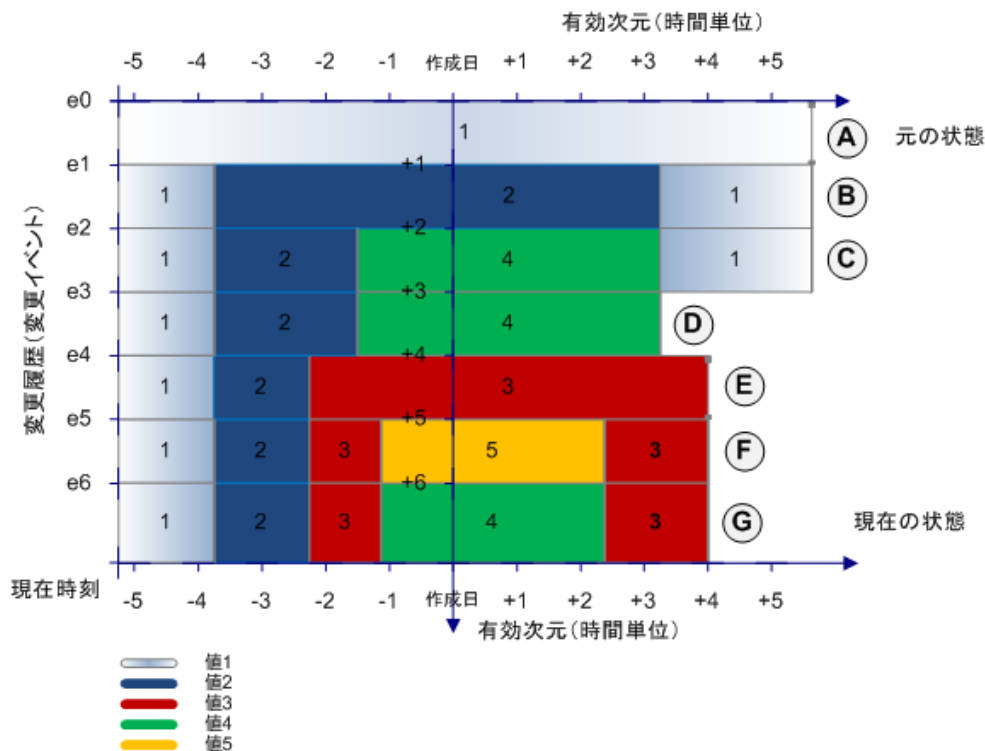
タイムラインがベースオブジェクトに対して有効になっている場合、データを二次元的に確認することができます。この二次元表示は、レコードの有効期間と履歴に基づいています。有効期間とは、レコードが有効である期間のことです。レコードの有効期間は、レコードの開始日と終了日を指定することによって定義できます。

履歴とは、レコードが存続している間の過去のデータイベントのことです。レコードの履歴を表示するには、レコードが有効であった過去の日付を指定します。

レコードバージョンの例

組織で、タイムラインを有効にするレコードがベースオブジェクト内に存在しているとします。このレコードのタイムラインには、e0、e1、e2、e3、e4、e5、e6 などの複数のデータ変更イベントが含まれます。データ変更イベント（新規または更新されたレコード値）は、指定された有効期間中有効な新しい相互参照レコードのバージョンになります。

次の図は、レコードの有効期間全体、存続期間、および変更履歴を示しています。



このレコードには、その存続期間中に次の変更が加えられます。

- A. 値 1 である元のレコードは有効期間が指定されていないため、常に有効です。
- B. MDM Hub で、特定の有効期間でのレコードの新しいバージョンを新しい値 2 で追加します。
- C. MDM Hub で、新しい有効期間での別の新しいバージョンのレコードを新しい値 4 で追加します。
- D. MDM Hub で、特定の有効期間でのレコードのバージョンを値 1 で削除します。
- E. MDM Hub で、特定の有効期間でのレコードのバージョンを新しい値 3 で追加します。MDM Hub で、有効期間がこの新しいバージョンの有効期間と重なっていて、値 4 であるレコードのバージョンを削除します。MDM Hub で、レコードバージョンの有効期間を値 2 で更新します。
- F. MDM Hub で、新しい有効期間でのレコードの新バージョンを新しい値 5 で追加します。MDM Hub で、レコードに値 3 のレコードバージョンを 2 つ作成し、2 つの新しい有効期間を設定します。
- G. レコードバージョンの値が 5 から 4 に変更されますが、有効期間は変更されません。

タイムラインの粒度

タイムラインの粒度は、レコードバージョンの有効期間の定義に使用される、時間の計測単位です。例えば、年単位、月単位、秒単位などで有効期間を選択できます。

年、月、日、時、分、または秒単位でタイムラインの粒度を設定することも、MDM Hub 実装でデータの有効期間を指定することもできます。オペレーショナル参照ストアを作成、または更新するときに必要なタイムラインの粒度を設定することができます。

重要: 設定したタイムラインの粒度は変更できません。

いずれかのタイムラインの粒度で有効期間を指定すると、システムでは有効期間に渡ってデータベース時間のロケールを使用します。1つのタイムラインの計測単位に有効なバージョンを作成するには、開始日と終了日を同じにする必要があります。

次の表に、設定可能なタイムラインの粒度オプションを示します。

タイムラインの粒度	説明
年	タイムラインの粒度が年の場合、2010 のような年のフォーマット (yyyy) で有効期間を指定できます。レコードの有効期間の開始日は年の最初の日、有効期間の終了日はその年の最後の日になります。例えば、有効期間の開始日が 2013 で終了日が 2014 の場合、レコードは 01/01/2013～31/12/2014 の間有効になります。
月	タイムラインの粒度が月の場合、01/2013 のような月のフォーマット (mm/yyyy) で有効期間を指定できます。レコードの有効期間の開始日は月の最初の日です。レコードの有効期間の終了日は月の最後の日になります。例えば、有効期間の開始日が 02/2013 で、終了日が 04/2013 の場合、レコードは、01/02/2013～30/04/2013 の間有効になります。
日	タイムラインの粒度が日の場合、13/01/2013 のような日付フォーマット (dd/mm/yyyy) で有効期間を指定できます。レコードの有効期間の開始日は、1 日の開始時 (12:00) に始まります。レコードの有効期間の終了日は 1 日の終了時 (23:59) に終わります。例えば、有効期間の開始日が 13/01/2013、有効期間の終了日が 15/04/2013 の場合、レコードは 13/01/2013 の 12:00～15/04/2013 の 23:59 の間有効になります。
時間	タイムラインの粒度が時間の場合、有効期間には、年、月、日、および時が含まれます。タイムラインの形式は dd/mm/yyyy hh です (例: 13/01/2013 15)。レコードの有効期間の開始日は 1 日の時刻の最初になります。レコードの有効期間の終了日は指定した時刻の終わりになります。例えば、有効期間の開始日が 13/01/2013 15、終了日が 15/04/2013 10 の場合、レコードは 13/01/2013 の 15:00～15/04/2013 の 10:59 の間有効になります。
分	タイムラインの粒度が分の場合、有効期間には、年、月、日、時、および分が含まれます。タイムラインの形式は dd/mm/yyyy hh:mm です (例: 13/01/2013 15:30)。レコードの有効期間の開始日は分の最初になります。レコードの有効期間の終了日は指定した分の終わりになります。例えば、有効期間の開始日が 13/01/2013 15:30、終了日が 15/04/2013 10:45 の場合、レコードは 13/01/2013 の 15:30:00～15/04/2013 の 10:45:59 の間有効になります。
秒	タイムラインの粒度が秒の場合、有効期間には、年、月、日、時、分、秒が含まれます。タイムラインの形式は dd/mm/yyyy hh:mm:ss です (例: 13/01/2013 15:30:45)。レコードの有効期間の開始日は秒の最初になります。有効期間の終了日は指定した秒の終わりになります。例えば、有効期間の開始日が 13/01/2013 15:30:55、終了日が 15/04/2013 10:45:15 の場合、レコードは 13/01/2013 の 15:30:55:00～15/04/2013 の 10:45:15:00 の間有効になります。

履歴と状態管理

履歴と状態管理は、デフォルトではタイムラインが有効になっているベースオブジェクトと、関連付けられているリレーションベースオブジェクトに対して有効になります。

ベースオブジェクトに対してタイムラインが有効になっている場合、そのベースオブジェクトの履歴または状態の管理を無効にすることはできません。MDM Hub は、相互参照テーブルに関連付けられた履歴テーブルのタイムラインに関連する変更履歴を保持します。MDM Hub は、レコードの状態（アクティブ、保留、削除済みなど）を、関連付けられた相互参照テーブルで管理します。

更新するレコードに承認が必要な場合、データ変更イベントが発生し、MDM Hub はレコードを保留として保存します。更新によるデータ変更イベントは、相互参照バージョンに影響します。MDM Hub は状態管理の相互作用 ID を使用して、相互参照バージョンへの変更を防ぎます。関連付けられている相互参照バージョンに対する影響は、同じソースシステムからの期間が重複していることが原因で発生することがあります。保留中の相互参照を昇格すると、相互参照レコードの状態はアクティブに変更されます。レコードを論理削除すると、状態は削除済みに変更されます。

タイムライン適用ルール

タイムライン情報の定義と保守を行う際には、ビジネスエンティティとビジネスリレーションのタイムラインを管理するため、事前定義されたタイムライン適用ルールが MDM Hub によって適用されます。MDM Hub は、期間の開始日と終了日に一連のルールを適用して、ロードおよび入力操作中の有効期間を管理します。

任意の時点で、MDM Hub では有効となるレコードのバージョンは 1 つのみとみなします。これは、有効期間の開始日と有効期間の終了日に基づいて判断されます。バッチプロセス、サービス統合フレームワーク、または Informatica Data Director を使用してデータを変更する際は、MDM Hub で現在の有効なデータが維持されます。また、1 つのベースオブジェクトレコードに多くのシステムが関係している場合、関与している有効なレコードに基づいて、MDM Hub でレコードのバージョンを更新するルールを適用します。

ユーザー出口を使用して、タイムラインと有効日を管理するカスタムルールを定義して適用することも使用できます。

有効期間の計算

ベースオブジェクトレコードの有効期間全体は、MDM Hub によって計算されます。

データ変更イベントを追跡するベースオブジェクトでロードまたは入力操作を使用する場合、ソースレコードごとに期間の開始日と終了日を指定できます。ベースオブジェクトレコードには、複数のソースシステムまたはプライマリキーソースからのコントリビュータまたはその両方がある場合があります。ベースオブジェクトレコードの全有効期間を計算するために、MDM Hub ではすべてのソースシステムのレコードの有効期間を集計します。

次の図は、2 つのソースシステム（PKey 1 を持つ System 1、PKey 2 を持つ System 2）からのコントリビュータのあるベースオブジェクトを示しています。

システム1/ PKeyソース1	値1 値2
システム2/ PKeyソース2	値4 値5
全体	1 1/4 2/4 2/5 5

上の図では、重複する異なる有効期間に、System 1 に値 1 と 2、System 2 に値 4 と 5 があります。レコードの全有効期間を計算するために、MDM Hub では両方のソースシステムのレコードの有効期間を集計します。レコードの有効期間を集計した結果は、各有効期間の最善データ（BVT）です。図に示されている全有効期間 1、1/4、2/4、2/5、5 は、特定の有効期間の BVT です。

注: 有効期間の管理ルールを変更することはできません。

ルール 1. 有効期間を指定せずにレコードを追加する

相互参照レコードに期間の開始日と終了日が指定されておらず、既存の相互参照レコードがない場合、レコードは常に有効です。

次の図は、既存の相互参照レコードがない場合の、期間の開始日と終了日が指定されずに挿入された相互参照レコードを示しています。

以前	レコードなし
変更	初期値
以後	初期値

ルール 2. 有効期間を指定してレコードを追加する

相互参照レコードに期間の開始日と終了日が指定されていて、既存の相互参照レコードがない場合、MDM Hub では指定した有効日を含むレコードを挿入します。

次の図は、既存の相互参照レコードがない場合の、期間の開始日と終了日を指定して挿入された相互参照レコードを示しています。

以前	レコードなし
変更	初期値
以後	初期値

ルール 3。有効期間のレコードバージョンを追加する

既存の有効期間のレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 3 が適用されます。

- 期間の開始日と終了日が指定され、有効期間が同じ相互参照レコードが存在します。
- 挿入される相互参照レコードと既存の相互参照レコードが、同じソースシステムに属しています。

ルール 3 により、次の MDM Hub の動作が保証されます。

- 相互参照レコードが更新されます。
- 有効期間は変更されません。

次の図は、同じ有効期間の相互参照レコードが存在する場合の、期間の開始日と終了日を指定して挿入された相互参照レコードを示しています。

以前	初期値
変更	その他の値
以後	その他の値

ルール 4。ある有効期間と重なり、開始日が拡張されたレコードバージョンを追加する

ある有効期間と重なり、開始日が拡張されたレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 4 が適用されます。

- 指定された有効期間と既存の相互参照レコードの有効期間が交差する。
- 変更された期間の開始日が既存の相互参照レコードの期間の開始日より後である。

ルール 4 により、以下の MDM Hub の動作が保証されます。

- 既存の相互参照バージョンの期間の終了日が、選択されているタイムライン単位で「変更された期間開始日-1」で更新され、最初の期間の開始日は変更されません。
- 新規バージョンの相互参照レコードが、指定された有効期間で挿入されます。

次の図は、既存の相互参照レコードの有効期間と交差し、期間の開始日が遅い有効期間を含む相互参照レコードを示しています。

以前	初期値
変更	その他の値
以後	初期値 その他の値

ルール 5。ある有効期間と重なり、終了日が早まったレコードバージョンを追加する

ある有効期間と重なり、終了日が早まったレコードバージョンを追加することができます。

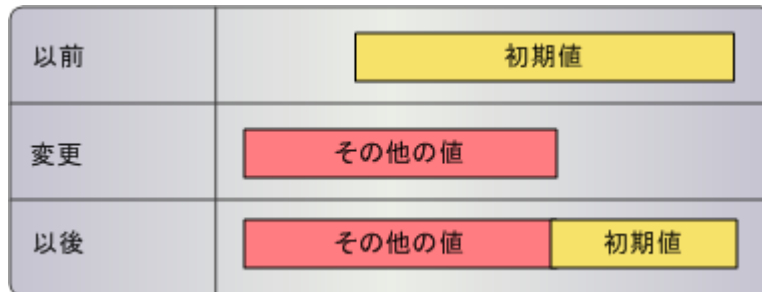
次の条件が存在する場合は、ルール 5 が適用されます。

- 指定された有効期間と既存の相互参照レコードの有効期間が交差する。
- 変更された終了日が相互参照レコードの既存の終了日より前である。

ルール 5 により、以下の MDM Hub の動作が保証されます。

- 選択されているタイムライン計測単位について、1 増えた変更後の終了日で、既存の相互参照バージョンの期間開始日が更新されます。
- 既存の相互参照バージョンの元の期間の終了日は変更されません。
- 新規バージョンの相互参照レコードが、指定された有効期間で追加されます。

次の図は、既存の相互参照レコードの有効期間と交差し、期間の終了日が早い有効期間を含む相互参照レコードを示しています。



ルール 6。1 つの有効期間内に含まれるレコードバージョンを追加する

1 つの有効期間内に含まれるレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 6 が適用されます。

- 新規相互参照バージョンの有効期間が既存の相互参照レコードの有効期間に含まれている。
- 変更された期間の開始日が既存の相互参照レコードの期間の開始日より後である。
- 変更された期間の終了日が既存の相互参照レコードの期間の終了日より前である。

ルール 6 により、以下の MDM Hub の動作が保証されます。

- 選択されているタイムライン計測単位について、1 減った変更後の日付で、既存の相互参照バージョンの期間終了日が更新されます。
- 既存の相互参照バージョンの期間の開始日は変更されません。
- 新規バージョンの相互参照レコードが、指定された有効期間で挿入されます。
- 選択されているタイムラインの計測単位について、1 増えた変更後の終了日に設定された期間開始日で、2 番目の相互参照レコードバージョンが挿入されます。
- 2 番目の相互参照レコードバージョンの終了日が、既存の相互参照レコードの期間の終了日に設定されます。

次の図は、有効期間が既存の相互参照レコードの有効期間に含まれる相互参照レコードを示しています。

以前	初期値
変更	その他の値
以後	初期値 その他の値 初期値

ルール 7。1 つの有効期間を含むレコードバージョンを追加する

レコードバージョンの既存の有効期間を含むレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 7 が適用されます。

- 既存の相互参照バージョンの有効期間が同じ日またはそれ以降に開始する。
- 既存の相互参照バージョンの有効期間が、新規の相互参照バージョンと同じ日またはそれ以前に終了する。

ルール 7 により、以下の MDM Hub の動作が保証されます。

- 有効期間が新規の相互参照バージョンと同じ日またはそれ以降に開始し、同じ日またはそれ以前に終了する既存の相互参照バージョンは、削除されます。
- 新規の相互参照バージョンは指定された有効期間で挿入されます。
- その他のすべての既存の相互参照バージョンの場合は、ルール 4 および 5 が適用されます。

次の図は、有効期間が新規の相互参照レコードの有効期間以降に開始し、それ以前に終了する既存の相互参照レコードを示しています。

以前	初期値1 初期値2 初期値3
変更	その他の値
以後	初期値1 その他の値 初期値3

ルール 8。有効期間が連続していないレコードバージョンを追加する

有効期間が連続していないレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 8 が適用されます。

- 指定された有効期間が既存の相互参照レコードの有効期間と連続していない。
- 変更された期間の終了日が既存の相互参照データの期間の開始日より前である、または変更された期間の開始日が既存の相互参照データの期間の終了日の後である。

ルール 8 により、以下の MDM Hub の動作が保証されます。

- 既存のバージョンの相互参照レコードは変更されません。
- ベースオブジェクトで連続しない有効期間が許可される場合には、指定する有効期間で新バージョンの相互参照レコードが挿入されます。

- ベースオブジェクトで連続した有効期間のみが許可される場合は、新バージョンの相互参照レコードは挿入されず、エラーが表示されます。

次の図に、有効期間が連続していない相互参照レコードを示します。

前	初期値
変更	その他の値
後	初期値 その他の値

ルール 9。有効期間内に保留状態でレコードバージョンを追加する

既存の有効期間内に含まれる保留状態のレコードバージョンを追加できます。

既存の相互参照レコードの有効期間が保留中の新規相互参照バージョンの有効期間にまたがる場合は、ルール 9 が適用されます。

ルール 9 により、以下の MDM Hub の動作が保証されます。

- 既存のバージョンの相互参照レコードは変更されませんが、昇格時に使用される相互作用 ID でロックされます。
- 新規相互参照バージョンが指定された有効期間および保留状態で挿入されます。
- 保留中の相互参照バージョンが昇格される場合は、ルール 1 から 8 が適用されます。

次の図は、既存の相互参照レコードの有効期間が保留中の新規相互参照バージョンの有効期間にまたがっていることを示しています。

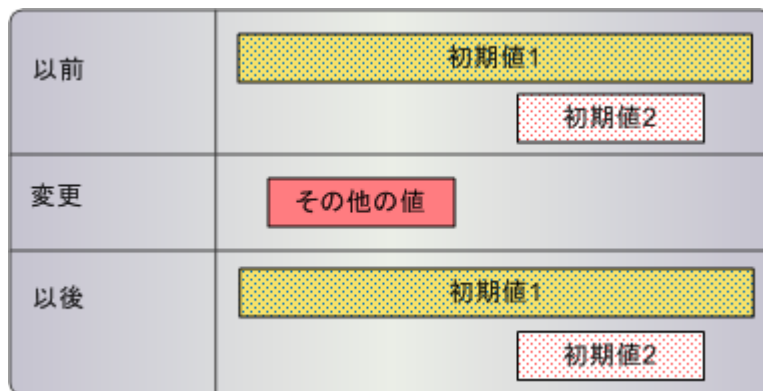
以前	初期値
変更	その他の値
以後	初期値 その他の値

ルール 10。レコードバージョンがロックされているときにレコードバージョンを追加する

既存のレコードバージョンが相互作用 ID によってロックされているときに、レコードバージョンを追加できません。

ルール 1 から 9 に基づく変更が、相互作用 ID でロックされている既存の相互参照レコードに影響を与える場合、MDM Hub では変更が制限されます。

次の図は、相互作用 ID でロックされている相互参照レコードが存在するときに新規相互参照レコードが挿入される場合、変更が発生しないことを示しています。



ルール 11。バージョンが保留状態のときにレコードバージョンを追加する

既存のバージョンが保留状態のときには、アクティブ状態または保留状態でレコードバージョンを追加できません。

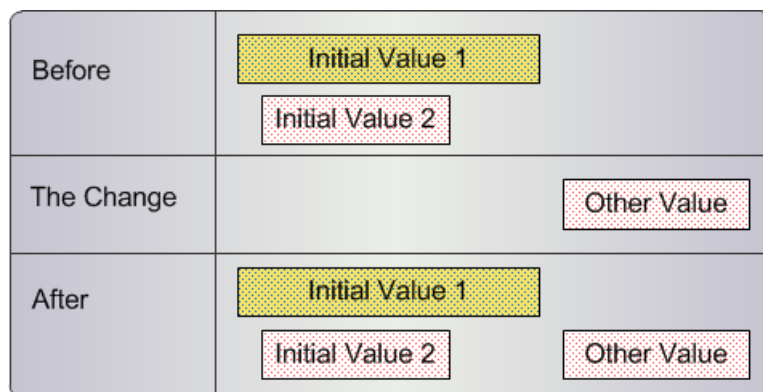
次の条件が存在する場合は、ルール 11 が適用されます。

- 既存の相互参照レコードが保留状態である。
- 相互参照レコードがアクティブまたは保留状態で挿入され、相互参照レコードの有効期間はロックされたもののレコードとも交差していない。

ルール 11 により、以下の MDM Hub の動作が保証されます。

- 既存の相互参照バージョンは変更されません。
- 相互参照レコードの新しいバージョンは、指定された有効期間で、状態を変更せずに挿入されます。

次の図は、既存の相互参照レコードが保留状態で、新規相互参照レコードが保留状態で挿入されることを示しています。



ルール 12。連続したベースオブジェクトでレコードバージョンを削除または更新する

連続したベースオブジェクトのレコードバージョンの連続性を中断するような、レコードバージョンの削除または更新はできません。

ベースオブジェクトのプロパティ設定で連続しない有効期間が許可されない場合、ルール 12 によって MDM Hub が次のように動作します。

- 連続性を中断するような、有効期間の削除はできません。
- 連続性を中断する相互参照レコードの有効期間への変更は保存できません。

次の図は、相互参照レコードのバージョンが削除される前後の、有効期間が連続している相互参照レコードを示しています。

Before	Initial Value 1	Initial Value 2	Initial Value 3
The Change	Initial Value 1		Initial Value 3
After	Initial Value 1	Initial Value 2	Initial Value 3

ルール 13。データを更新する

既存レコードバージョン内のデータを更新できます。

次の条件が存在する場合は、ルール 13 が適用されます。

- 既存の相互参照レコードの有効期間が変更されない。
- 既存の相互参照データが変更される。
- タイムラインアクションが 1 に設定されている。

ルール 13 により、以下の MDM Hub の動作が保証されます。

- 既存のバージョンの相互参照レコードは変更されません。
- 既存バージョンの相互参照レコード内のデータが更新されます。

次の図に、既存レコードバージョンのデータ更新を示します。

Before	Initial Value
The Change	Updated Value
After	Updated Value

ルール 14。有効期間を更新する

レコードバージョンの有効期間を更新できます。

次の条件が存在する場合は、ルール 14 が適用されます。

- 既存の相互参照レコードの有効期間は、有効期間を延長または短縮するように更新される。
- 既存の相互参照レコード内のデータは変更されない。
- タイムラインアクションが 2 に設定されている。

ルール 14 により、以下の MDM Hub の動作が保証されます。

- 既存バージョンの相互参照レコードが更新されます。
- 既存バージョンの相互参照レコードの有効期間はユーザーによって指定され、データは変更されません。
- ベースオブジェクトに連続しない有効期間を設定できる場合、レコードバージョンの有効期間を延長すると、直前および直後のレコードバージョンの有効期間は短縮されます。レコードバージョンの重複を回避するには、MDM Hub では隣接するレコードバージョンの有効日が短縮されます。
- ベースオブジェクトに連続しない有効期間を設定できる場合、レコードバージョンの有効期間を短縮すると、MDM Hub ではレコードバージョン間にギャップが発生します。
- ベースオブジェクトの有効期間が連続している必要がある場合、MDM Hub では隣接するレコードの有効期間が延長または短縮されて連続性が保持されます。

次の図に、ベースオブジェクトで連続性が有効になっている場合の有効期間の更新を示します。

変更前	<div>初期値1</div> <div>初期値2</div>
変更	<div>更新値1</div>
変更後	<div>更新値1</div> <div>更新値2</div>

ルール 15。有効期間を追加する

新しい有効期間でレコードバージョンを追加できます。

次の条件が存在する場合は、ルール 15 が適用されます。

- 新しい有効期間のレコードバージョンが相互参照テーブルに追加される。
- 指定した有効期間によって有効期間の間にギャップが生じる。
- ベースオブジェクトプロパティ設定で連続しない有効期間が許可されていない。
- タイムラインアクションが 4 に設定されている。

ルール 15 により、以下の MDM Hub の動作が保証されます。

- 新しい有効期間の相互参照レコードがロードされます。
- 有効期間の連続性が中断したときにギャップを埋めるには、既存レコードバージョンの有効期間を延長してレコードバージョン間のギャップを埋めます。
- 有効期間の連続性が中断したときにギャップを埋めないと、新しいレコードバージョンが挿入されず、エラーが発生します。バッチロード中に、レコードがリジェクトテーブルに移動されます。

次の図に、関連付けられたベースオブジェクトで連続性が有効に設定されているときの相互参照テーブルへのレコードの追加を示します。

変更前	初期値
変更	他の値
変更後	初期値 他の値

ベースオブジェクトのタイムラインの設定

ベースオブジェクトのレコードのタイムラインを設定するには、MDM Hub のタイムラインと Hub サーバプロパティファイルを有効にします。

1. Hub コンソールのベースオブジェクトのタイムラインを有効にします。
2. MDM Hub プロパティファイルを設定します。

手順 1。Hub コンソールでタイムラインを有効にする

タイムラインが必要な各ベースオブジェクトに対してタイムラインを有効にします。

1. モデルワークベンチを開き、[スキーマ] をクリックします。
2. 書き込みロックを取得します。
3. スキーマツリーで、ベースオブジェクトを選択します。
スキーママネージャで、[ベースオブジェクトのプロパティ] ページの [基本] タブが表示されます。
4. [タイムライン] リストで [動的タイムライン] を選択します。
5. 有効期間を設定します。
 - 連続した有効期間がレコードに必要な場合は、[連続しない有効期間を許可] チェックボックスを無効にします。
 - 連続しない有効期間がレコードに必要な場合は、[連続しない有効期間を許可] チェックボックスを有効にします。

6. [保存] をクリックします。

MDM Hub により、ベースオブジェクトのタイムラインが有効になります。

手順 2。プロパティファイルを設定する

ベースオブジェクトのタイムラインを有効にする場合は、ベストバージョンオブトゥールズ (BVT) の計算中に使用するレコードの最大数を設定します。BVT 計算の有効日を指定すると、SIF の API である

SearchQuery、SearchHMQQuery、GetOneHop、および GetEntityGraph はユーザーが設定するレコードの最大数を計算中に使用します。

1. 次のディレクトリにある cmxserver.properties ファイルを開きます。
UNIX の場合: <infadm installation directory>/hub/server/resources
Windows の場合: <infadm installation directory>\hub\server\resources
2. 以下のプロパティを追加します。
searchQuery.buildBvtTemp.MaxRowCount
sif.search.result.querytemptableTimeToLive.seconds
searchQuery.buildBvtTemp.MaxRowCount のデフォルトは 10000 です。
sif.search.result.querytemptableTimeToLive.seconds のデフォルトは 30 です。
3. ファイルを保存し、閉じます。

バッチジョブでの複数のレコードバージョンのロード

複数のレコードバージョンをステージングテーブルからベースオブジェクトに関連付けられた相互参照テーブルに、1つのバッチジョブでロードできます。レコードの有効期間の連続性に関するベースオブジェクトプロパティを有効にしてください。

複数のレコードバージョンをロードするときに、レコードの有効期間にギャップが発生するようなレコードバージョンは MDM Hub によって拒否されます。レコードの有効期間の連続性に影響を与えることなく複数のレコードバージョンをロードするには、連続性を保持する順番でレコードバージョンをロードするように MDM Hub を設定します。

複数のレコードバージョンをロードするロードバッチの設定

1つのロードバッチジョブで複数のレコードバージョンをロードするように MDM Hub を設定します。

1. ベースオブジェクトレコードに連続する有効期間を設定します。
 - a. モデルワークベンチを開き、[スキーマ] をクリックします。
 - b. 書き込みロックを取得します。
 - c. スキーマツリーで、ベースオブジェクトを選択します。
スキーママネージャで、[ベースオブジェクトのプロパティ] ページの [基本] タブが表示されます。
 - d. [連続しない有効期間を許可] チェックボックスを無効にします。
2. ベースオブジェクトにロードするレコードバージョンの順序を設定します。
 - a. 次のディレクトリにある cmxserver.properties ファイルを開きます。
UNIX の場合: <infadm installation directory>/hub/server/resources
Windows の場合: <infadm installation directory>\hub\server\resources
 - b. 次のプロパティを追加します。
cmx.server.batch.load.smart_resequencing = true
 - c. ファイルを保存し、閉じます。

複数のレコードバージョンのバッチロード例

データ変更イベントの追跡対象である CUSTOMER というベースオブジェクトが存在します。この CUSTOMER ベースオブジェクトには、2014 年 3 月 21 時点でニューヨークに住んでおり、2015 年 1 月 15 日までニューヨークに住むことを予定している John Smith のレコードが含まれています。過去および将来における他の有効期間のレコードバージョンをロードする必要があるとします。

ステージングテーブルから次の有効期間のレコードを、CUSTOMER ベースオブジェクトに関連付けられた相互参照テーブルにロードする必要があるとします。

- 2011 年 1 月 31 日～2013 年 10 月 20 日
- 2013 年 10 月 21 日～2014 年 3 月 20 日
- 2015 年 1 月 16 日～2015 年 11 月 24 日
- 2015 年 11 月 25 日～NULL

単一のロードバッチジョブで複数バージョンのレコードをロードするには、ベースオブジェクト内のレコードの有効期間が連続するように設定します。さらに、既存のレコードバージョンとの連続性が保持される順番でレコードバージョンをロードできるように MDM Hub を設定します。

ロードバッチジョブを行う前は、相互参照には John Smith がニューヨークに住んでいることを示すレコードが 1 つ含まれています。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	New York	March 21, 2014	January 15, 2015

ロードバッチジョブの後、相互参照テーブルには John Smith について複数の連続したレコードバージョンが含まれた状態になります。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
3		2	25	John	Smith	ロサンゼルス	2011 年 1 月 31 日	2013 年 10 月 20 日
2		2	25	John	Smith	サンフランシスコ	2013 年 10 月 21 日	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	2015 年 11 月 24 日
5		2	25	John	Smith	ラスベガス	2015 年 11 月 25 日	null

このレコードは、John Smith が過去にロサンゼルスとサンフランシスコに住んでいたが現在はニューヨークに住んでおり、これらの複数バージョンのレコードがロードされた後はオースチンとラスベガスに住む可能性があることを示しています。

レコードバージョンの有効期間の編集

レコードバージョンの有効期間は編集可能です。レコードバージョンに間違った値を保存した場合は、有効期間の値を編集します。レコードバージョンの開始日と終了日を変更することで、レコードの有効期間を変更できます。

指定した日付よりも遅く開始し、早く終了するようにレコードバージョンの有効期間を編集した場合、有効期間は短縮されます。指定した日付よりも早く開始し、遅く終了するようにレコードバージョンの有効期間を編集した場合、有効期間は延長されます。

関連付けられているベースオブジェクトでレコードの連続性が有効になっている場合、レコードバージョンの有効期間を編集できます。MDM Hub は、ユーザーが編集するレコードバージョンの隣接するレコードバージョンの有効期間を延長または短縮して、連続するように調整します。ベースオブジェクトに連続しない有効期間が存在することがある場合、有効期間を延長すると、隣接するレコードバージョンは影響を受けます。

ロードバッチジョブ中にレコードバージョンの有効期間を編集するには、次の設定を実行します。

- **TIMELINE_ACTION**。TIMELINE_ACTION ステージングテーブルカラムの値を 2 に設定します。プロパティが 2 に設定されている場合、MDM Hub はレコードバージョンの有効期間を編集できます。TIMELINE_ACTION カラムは、対応するランディングテーブルカラムからマッピングされます。
- **TIMELINE_FILL_ON_GAP**。レコードバージョンの有効期間を編集する際、レコードバージョンの有効日が連続するように調整します。このプロパティは、オペレーショナルリファレンスストアの C_REPOS_TABLE、または Hub コンソールのステージングテーブルプロパティで設定します。true に設定すると、ベースオブジェクトに新しいレコードバージョンを追加可能な場合に、MDM Hub によって各レコードバージョンの有効期間が連続するよう調整されます。false に設定すると、各レコードバージョンの有効期間にギャップが発生するようなレコードバージョンの追加は MDM Hub によって拒否されます。デフォルトは false です。

注: 編集内容は、状態管理オーバーライドシステムのデータを介して、またはレコードが生成されたソースシステムを介して有効期間に反映させることができます。

レコードバージョンの有効期間の延長

終了日と開始日を編集して、レコードバージョンの有効期間を延長できます。ベースオブジェクトのレコードバージョンが連続している場合、MDM Hub は編集するレコードバージョンの隣接するレコードバージョンの有効期間を短縮します。

終了日を延長すると、有効期間を延長できます。終了日を延長した後、編集するレコードバージョンが隣接するレコードバージョンと重複することがあります。編集するレコードバージョンの後にあるレコードバージョンと重複します。MDM Hub は、隣接するレコードバージョンの開始日を拡張します。レコードバージョンが重複せずに連続性を確保するには、隣接するレコードバージョンの有効期間を短縮します。

開始日を早い日付に移動すると、有効期間を延長できます。開始日を移動した後、レコードバージョンが隣接するレコードバージョンと重複することがあります。編集するレコードバージョンの前にあるレコードバージョンと重複します。MDM Hub は、隣接するレコードバージョンの終了日を早い日付に移動します。レコードバージョンが重複せずに連続性を確保するには、隣接するレコードバージョンの有効期間を短縮します。

ベースオブジェクト内のレコードの連続性がある場合、レコードバージョンの重複がない限り、隣接するレコードバージョンは変更されません。

有効期間延長の例

データ変更イベントの追跡対象である CUSTOMER というベースオブジェクトが存在します。CUSTOMER ベースオブジェクトに関連付けられている相互参照テーブルには、ニューヨークに住んでいる John Smith の、2014 年 3 月 21 日から 2014 年 11 月 30 日まで有効なレコードが含まれています。相互参照テーブルには、2014 年 12 月 1 日から有効な別のバージョンのレコードも含まれています。2014 年 3 月 21 日から 2014 年 11 月 30 日まで有効なレコードバージョンを編集して、2015 年 12 月 28 日まで有効にしたいと考えています。

最初のレコードバージョンを編集する前は、John Smith の 2 番目のレコードバージョンは 2014 年 12 月 1 日から有効です。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名 姓	市区町村	期間の開始日	期間の終了日
1		2	25	John Smith	New York	March 21, 2014	November 30, 2014
2		2	25	John Smith	ロサンゼルス	December 1, 2014	null

2014 年 11 月 30 日ではなく 2015 年 2 月 28 日に終了するようにレコードバージョンの有効期間を編集すると、有効期間は延長されます。レコードバージョンの重複を回避するために、隣接するレコードバージョンの開始日は 2014 年 12 月 1 日から 2015 年 3 月 1 日に変更されます。

最初のレコードバージョンの開始日を拡張した後、MDM Hub は 2 番目のレコードバージョンを、最初のバージョンが終了した後に開始するように更新します。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	New York	March 21, 2014	February 28, 2015
2		2	25	John	Smith	ロサンゼルス	March 1, 2015	null

レコードバージョンの有効期間の短縮

開始日または終了日を編集すると、レコードバージョンの有効期間を短縮することができます。ベースオブジェクトのレコードバージョンが連続する場合、MDM Hub は、編集するレコードバージョンに隣接するレコードバージョンの有効期間を延長します。

開始日を拡張すると、有効期間が短縮されます。編集するレコードバージョン以前に隣接するレコードバージョンがある場合、2 つのレコードバージョンの間にギャップが生じます。MDM Hub は、ギャップが生じたレコードバージョンの終了日を延長して連続するように調整します。

終了日を前の日付に変更すると、有効期間を短縮できます。編集するレコードバージョン以降に隣接するレコードバージョンがある場合、2 つのレコードバージョンの間にギャップが生じます。連続するようにするために、MDM Hub は隣接するレコードバージョンの開始日を前の日付に変更します。

ベースオブジェクトのレコードの連続性を有効にしない場合は、隣接するレコードバージョンは変更されません。

有効期間短縮の例

データ変更イベントの追跡対象である CUSTOMER というベースオブジェクトが存在します。この CUSTOMER ベースオブジェクトには、2014 年 3 月 21 日時点でニューヨークに住んでいる John Smith のレコードが含まれています。有効期間が連続するように調整する必要があります。2014 年 12 月 1 日から有効なレコードバージョンをロードします。最後に、2014 年 12 月 1 日から有効なレコードバージョンを 2015 年 3 月 1 日から有効になるように編集します。

John Smith 氏の相互参照レコードは、彼が 2014 年 3 月 21 日からニューヨークに住んでいることを示しています。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	New York	March 21, 2014	null

2014 年 12 月 1 日から有効なレコードバージョンをロードすると、既存のレコードバージョンは 2014 年 11 月 30 日で終了します。既存のレコードバージョンの有効期間は、重複を回避し、連続するように短縮されます。

2014 年 12 月 1 日から有効なレコードバージョンをロードした後は、相互参照テーブルの John Smith の既存のレコードバージョンの有効期間の終了日は 2014 年 11 月 30 日と表示されます。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	New York	March 21, 2014	November 30, 2014
2		2	25	John	Smith	ロサンゼルス	December 1, 2014	null

2014 年 12 月 1 日から有効なレコードバージョンを 2015 年 3 月 1 日から有効になるように編集すると、有効期間は短縮されます。また、2 つのレコードバージョンの有効期間の間にギャップが生じます。MDM Hub は、編集するレコードバージョンに隣接するレコードバージョンの終了日を延長します。終了日が 2014 年 11 月 30 日から 2015 年 2 月 28 日に変更され、ギャップが埋まります。

レコードバージョンの有効日付を 2014 年 12 月 1 日から 2015 年 3 月 1 日に変更した場合。

行 ID	XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1		2	25	John	Smith	New York	March 21, 2014	February 28, 2015

行 ID XREF	行 ID オブジェクト	顧客 ID	名 姓	市区町村	期間の開始日	期間の終了日
2	2	25	John Smith	ロサンゼルス	March 1, 2015	null

レコードバージョンの追加

新しい期間で有効なレコードバージョンを追加できます。

MDM Hub では、ベースオブジェクト内のレコードの連続性を有効にすると、各レコードバージョンの有効日が連続するように調整されます。追加するレコードバージョンに隣接するレコードバージョンの有効期間が MDM Hub によって延長され、連続性が保持されます。

レコードバージョンの連続性を確保するには、ステージングテーブルのカラムを設定して新しいレコードバージョンと C_REPOS_TABLE カラムを追加します。

ロードバッチジョブ中にレコードバージョンを追加するには、次の設定項目を設定します。

- **TIMELINE_ACTION**。ステージングテーブルの **TIMELINE_ACTION** カラムの値を 4 に設定します。プロパティを 4 に設定すると、MDM Hub で新しいレコードバージョンを追加できます。TIMELINE_ACTION カラムは、対応するランディングテーブルカラムからマッピングされます。
- **TIMELINE_FILL_ON_GAP**。新しいレコードバージョンを追加した時に、各レコードバージョンの有効期間が連続するよう調整されます。このプロパティは、オペレーショナルリファレンスストアの C_REPOS_TABLE、または Hub コンソールのステージングテーブルプロパティで設定します。true に設定すると、ベースオブジェクトに新しいレコードバージョンを追加可能な場合に、MDM Hub によって各レコードバージョンの有効期間が連続するよう調整されます。false に設定すると、各レコードバージョンの有効期間にギャップが発生するようなレコードバージョンの追加は MDM Hub によって拒否されます。デフォルトは false です。

レコードバージョンの追加例

データ変更イベントの追跡対象である CUSTOMER というベースオブジェクトが存在します。この CUSTOMER ベースオブジェクトには、2014 年 3 月 21 日時点でニューヨークに住んでいる John Smith のレコードが含まれています。有効期間が連続するように調整する必要があります。2014 年 12 月 1 日から有効なレコードバージョンをロードします。

新しいレコードバージョンをロードする前は、John Smith のレコードには彼が 2014 年 3 月 21 日からニューヨークに住んでいると示されます。

行 ID XREF	行 ID オブジェクト	顧客 ID	名 姓	市区町村	期間の開始日	期間の終了日
1	2	25	John Smith	New York	March 21, 2014	null

2014 年 12 月 1 日から有効なレコードバージョンをロードすると、隣接するレコードバージョンの有効期間が短縮されます。MDM Hub では、レコードバージョンが重複しないよう既存レコードの有効期間の終了日を調整して、連続するようにします。

2014 年 12 月 1 日から有効なレコードバージョンをロードした後は、相互参照テーブルの John Smith の既存のレコードバージョンの有効期間の終了日は 2014 年 11 月 30 日と表示されます。

行 ID XREF	行 ID オブジェクト	顧客 ID	名 姓	市区町村	期間の開始日	期間の終了日
1	2	25	John Smith	New York	March 21, 2014	November 30, 2014
2	2	25	John Smith	ロサンゼルス	December 1, 2014	null

レコード内のデータの更新

あるレコード内のデータを更新した場合、MDM Hub は、そのレコード内のデータを変更します。新しいレコードバージョンは作成されません。

レコードデータ更新の例

データ変更イベントの追跡対象である CUSTOMER というベースオブジェクトが存在します。この CUSTOMER ベースオブジェクトには、2014 年 3 月 21 から 2015 年 1 月 15 日までニューヨークに住んでいた John Smith 氏の記録が入っています。氏が住んでいる町をニューアークに更新する必要があります。

データを更新する前は、John Smith 氏の相互参照レコードでは氏がニューヨークに住んでいると示されます。

行 ID XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1	2	25	John	Smith	New York	March 21, 2014	January 15, 2015

データを更新した後は、John Smith 氏の相互参照レコードでは氏がニューアークに住んでいると示されます。

行 ID XREF	行 ID オブジェクト	顧客 ID	名	姓	市区町村	期間の開始日	期間の終了日
1	2	25	John	Smith	Newark	March 21, 2014	January 15, 2015

MDM Hub により、正しい都市名ニューアークで既存のバージョンが更新されます。John Smith 氏のレコードの有効期間は変更されません。

リレーションの更新

リレーションベースオブジェクトレコード内のリレーション情報を更新すると、MDM Hub は、レコードバージョンを挿入または更新します。

リレーションベースオブジェクトレコードのカスタムカラム内のリレーション情報を更新すると、MDM Hub は、そのリレーションレコードに関連付けられている相互参照レコードを更新します。

リレーションベースオブジェクトレコードのシステムカラム内のリレーション情報を更新すると、MDM Hub は、関連する相互参照テーブルに新しいレコードバージョンを挿入します。システムカラムは、リレーションタイプ、階層タイプ、期間の開始日、期間の終了日などのカラムです。また、対応する相互参照レコードがないソースシステムからリレーションを更新した場合、MDM Hub は、相互参照テーブルに新しいレコードバージョンを追加します。

リレーションレコードのカスタムカラムの更新の例

組織に PARTY ベースオブジェクトがあり、Mary Adam のレコードと ABC 社のレコードが含まれているとします。Mary Adam と ABC 社間のリレーションは、関連する PARTY REL リレーションベースオブジェクトで定義されています。カスタムカラムである Rel Desc カラム内の値を「個人」から「組織」に更新します。

注: IDD ソースシステムを使用するように Informatica Data Director を設定しているため、更新は階層ビューで実行されます。

SFDC ソースシステムから更新が行われる場合

SFDC ソースシステムレコードから Rel Desc カラムの更新が行われた場合、MDM Hub は、次のように RL PARTY CROSS-REFERENCE テーブルを更新します。

- 既存の相互参照レコードは変更しません。
- 開始日と終了日は変更せずに、新しい相互参照レコードを挿入します。
- 新しい相互参照レコード内のカスタムカラムと行 ID システムカラムの値を、ソースシステムの値で更新します。

RL PARTY CROSS-REFERENCE テーブルには、Rel Desc カスタムカラムの値を変更する必要があるレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Rel Desc	期間の開始日	期間の終了日
SFDC	414722	2402007	2402147	個人	Null	Null

Rel Desc カラムの値を「個人」から「組織」に変更すると、RL PARTY CROSS-REFERENCE テーブルには既存レコードと新しいレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Rel Desc	期間の開始日	期間の終了日
SFDC	414722	2402007	2402147	個人	Null	Null
IDD	414722	2402007	2402147	組織	Null	Null

IDD ソースシステムから更新が行われる場合

IDD ソースシステムのレコードから Rel Desc カラムの更新が行われた場合、MDM Hub は、次のように RL PARTY CROSS-REFERENCE テーブルを更新します。

- 既存の相互参照レコード内のカスタムカラムの値を更新します。
- 相互参照レコードの開始日と終了日を変更しません。

RL PARTY CROSS-REFERENCE テーブルには、Rel Desc カスタムカラムの値を変更する必要があるレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Rel Desc	期間の開始日	期間の終了日
IDD	414722	2402007	2402147	個人	Null	Null

Rel Desc カラムの値を「個人」から「組織」に変更すると、RL PARTY CROSS-REFERENCE テーブルには更新したレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Rel Desc	期間の開始日	期間の終了日
IDD	414722	2402007	2402147	組織	Null	Null

リレーションレコードのシステムカラムの更新例

組織に PARTY ベースオブジェクトがあり、Mary Adam のレコードと ABC 社のレコードが含まれているとします。Mary Adam と ABC 社間のリレーションは、関連する PARTY REL リレーションベースオブジェクトで定義されています。期間の開始日と終了日を更新します。

注: IDD ソースシステムを使用するように Informatica Data Director を設定しているため、更新は階層ビューで実行されます。

期間の開始日を「null」から「2016 月 1 月 1 日」に、期間の終了日を「null」から「2017 月 1 月 1 日」に更新します。

SFDC ソースシステムから更新が行われる場合

SFDC ソースシステムレコードから期間の開始日と終了日が更新されると、MDM Hub は、次のように RL PARTY CROSS-REFERENCE テーブルを更新します。

- 既存の相互参照レコードは変更しません。
- 開始日を「2016 月 1 月 1 日」に、終了日を「2017 月 1 月 1 日」に設定した新しい相互参照レコードを挿入します。

RL PARTY CROSS-REFERENCE テーブルには、期間の開始日と終了日を変更する必要があるアクティブなレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Hub 状態インジケータ	Rel Desc	期間の開始日	期間の終了日
SFDC	414722	2402007	2402147	アクティブ	個人	Null	Null

期間の開始日と終了日を変更した後、RL PARTY CROSS-REFERENCE テーブルには、既存レコードと新しいレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Hub 状態インジケータ	Rel Desc	期間の開始日	期間の終了日
SFDC	414722	2402007	2402147	アクティブ	個人	Null	Null
IDD	414722	2402007	2402147	アクティブ	個人	2016 月 1 月 1 日	2017 月 1 月 1 日

IDD ソースシステムから更新が行われる場合

IDD ソースシステムレコードから期間の開始日と終了日を更新されると、MDM Hub は、次のように RL PARTY CROSS-REFERENCE テーブルを更新します。

- 既存の相互参照レコードは変更しません。
- 終了日を「2015 年 12 月 31 日」に設定した新しい相互参照レコードを挿入します。
- 開始日を「2016 月 1 月 1 日」に、終了日を「2017 月 1 月 1 日」に設定した新しい相互参照レコードを挿入します。
- 開始日を「2017 月 1 月 2 日」に、終了日を「null」に設定した新しい相互参照レコードを挿入します。

RL PARTY CROSS-REFERENCE テーブルには、期間の開始日と終了日を変更する必要があるアクティブなレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Hub 状態インジケータ	Rel Desc	期間の開始日	期間の終了日
IDD	414722	2402007	2402147	アクティブ	個人	Null	Null

期間の開始日と終了日を変更した後、RL PARTY CROSS-REFERENCE テーブルには、既存レコードと 3 つの新しいレコードが表示されます。

行 ID システム	行 ID オブジェクト	グループ ID1	グループ ID2	Hub 状態インジケータ	Rel Desc	期間の開始日	期間の終了日
IDD	414722	2402007	2402147	アクティブ	個人	Null	Null
IDD	414722	2402007	2402147	アクティブ	個人	Null	2015 年 12 月 31 日
IDD	414722	2402007	2402147	アクティブ	個人	2016 月 1 月 1 日	2017 月 1 月 1 日
IDD	414722	2402007	2402147	アクティブ	個人	2017 年 1 月 2 日	Null

リレーションの終了

2つのレコード間のリレーションを終了できます。リレーションを終了すると、MDM Hub によってリレーションベースオブジェクトに関連付けられている相互参照テーブルに新しいレコードバージョンが挿入されます。また、MDM Hub が、レコードの削除されたインジケータを [-999999999] に設定し、Hub 状態インジケータを [削除済み] に更新します。新しいレコードバージョンの期間開始日は、リレーションを終了した日に設定されます。

レコードが状態管理オーバーライドシステムに属する場合、新しいレコードバージョンの期間開始日は、リレーションを終了する日に 1 タイムライン単位を加えた日に設定されます。例えば、期間終了日が 2014 年 1 月 31 日で、タイムラインの粒度が 1 日である場合、状態管理オーバーライドシステムのレコードバージョンの期間開始日は 2014 年 2 月 1 日になります。

リレーションの終了の例

組織に PARTY ベースオブジェクトがあり、Mary Adam のレコードが含まれているとします。PARTY GROUP ベースオブジェクトには、Mary が属する Adam 世帯の詳細が含まれています。PARTY と PARTY GROUP ベースオブジェクト間のリレーションは、関連付けられた PARTY GROUP REL リレーションベースオブジェクトで定義されます。2015 年 5 月 17 日に、Mary から Adam 世帯の一員ではなくなったと連絡がありました。Mary のレコードを更新して、Adam 世帯とのリレーションを終了する必要があります。

Mary の Adam 世帯とのリレーションの終了日を設定します。Mary のリレーションレコードに終了日を設定すると、MDM Hub によって削除済み状態のレコードが RL PARTY GROUP CROSS-REFERENCE リレーションベースオブジェクトに関連付けられている相互参照テーブルに追加されます。

PARTY GROUP REL リレーションベースオブジェクトに関連付けられている RL PARTY GROUP CROSS-REFERENCE 相互参照テーブルで、次の変更が行われます。

- Mary と Adam 世帯間のリレーションにレコードが挿入されます。
- このレコードは、2015 年 5 月 18 日から有効です。
- 削除されたインジケータが [-999999999] に設定されます。
- Hub 状態インジケータが [削除済み] に設定されます。
- 元のレコードの終了日が 2015 年 5 月 17 日に更新されます。

Mary と Adam 世帯間のリレーションを終了する前、相互参照テーブルには、2006 年 10 月 6 日から 2999 年 12 月 31 日までが有効になっているレコードが示されています。

行 ID	XREF 行 ID	オブジェクトグループ ID	削除されたインジケータ	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63	22	147		アクティブ	28	2006 年 10 月 6 日	2999 年 12 月 31 日

Mary と Adam 世帯間のリレーションを終了した後、相互参照テーブルには、2015 年 5 月 18 日以降 Mary が Adam 世帯の一員ではなくなったことが示されています。

行 ID	XREF 行 ID	オブジェクトグループ ID	削除されたインジケータ	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63	22	147		アクティブ	28	2006 年 10 月 6 日	2015 年 5 月 17 日
621	22	147	-999999999	削除済み	28	2015 年 5 月 18 日	

リレーション期間の削除

特定の期間有効なリレーションレコードバージョンを削除できます。リレーションレコードバージョンを削除すると、MDM Hub が、リレーションベースオブジェクトに関連付けられている相互参照テーブルに新しいレコードバージョンを挿入します。このレコードバージョンの有効期間は、削除するリレーションレコードバージョンの有効期間と同じです。また、MDM Hub が、このレコードバージョンの削除されたインジケータを [-999999999] に設定し、Hub 状態インジケータを [削除済み] に設定します。

リレーションの削除の例

組織に PARTY ベースオブジェクトがあり、Mary Adam のレコードが含まれているとします。PARTY GROUP ベースオブジェクトには、Mary が属する Adam 世帯の詳細が含まれています。PARTY と PARTY GROUP ベースオブジェクト間のリレーションは、関連付けられた PARTY GROUP REL リレーションベースオブジェクトで定義されます。2006 年 10 月 6 日から有効だった Mary と Adam 世帯間のリレーションを削除する必要があります。

Mary のリレーションレコードの不正なバージョンを削除することにします。Mary のリレーションレコードを削除すると、MDM Hub が、PARTY GROUP REL リレーションベースオブジェクトに関連付けられている相互参照テーブルに、削除済み状態のレコードを追加します。

PARTY GROUP REL リレーションベースオブジェクトに関連付けられている RL PARTY GROUP CROSS-REFERENCE 相互参照テーブルで、次の変更が行われます。

- Mary と Adam 世帯間のリレーションのレコードが挿入されますが、有効期間に変更はありません。
- 新しいレコードの削除されたインジケータは [-999999999] です。
- 新しいレコードの Hub 状態インジケータは [削除済み] です。
- 元のレコードは変更されません。

Mary と Adam 世帯間のリレーションを削除する前、相互参照テーブルにはこのリレーションがアクティブであることが示されています。

行 ID	XREF	行 ID	オブジェクト	グループ ID	削除されたインジケータ	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63		22		147		アクティブ	28	2006 年 10 月 6 日	2999 年 12 月 31 日

Mary と Adam 世帯間のリレーションを削除した後、相互参照テーブルには、このリレーションが削除済みの新しいレコードが表示されます。

行 ID	XREF	行 ID	オブジェクト	グループ ID	削除されたインジケータ	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63		22		147		アクティブ	28	2006 年 10 月 6 日	2999 年 12 月 31 日
621		22		147	-999999999	削除済み	28	2006 年 10 月 6 日	2999 年 12 月 31 日

リレーションベースオブジェクトに、Mary と Adam 世帯間のリレーションが削除されたことが示されます。

行 ID	オブジェクト	グループ ID	削除されたインジケータ	Hub 状態インジケータ	関係者グループ ID
22		147	-999999999	-1	28

すべてのリレーション期間の削除

リレーションレコードのすべてのリレーション期間が正しくない場合、リレーション期間をすべて削除できます。すべてのリレーション期間を削除すると、MDM Hub が、PARTY GROUP REL リレーションベースオブジェクトに関連付けられている相互参照テーブルで、レコードバージョンの Hub 状態を「削除済み」に変更します。また、MDM Hub が、PARTY GROUP REL リレーションベースオブジェクトのレコードの Hub 状態インジケータおよび削除されたインジケータを「削除済み」に設定します。

すべてのリレーション期間の削除の例

組織に PARTY ベースオブジェクトがあり、Mary Adam のレコードが含まれているとします。PARTY GROUP ベースオブジェクトには、Mary が属する Adam 世帯の詳細が含まれています。PARTY と PARTY GROUP ベースオブジェクト間のリレーションは、関連付けられた PARTY GROUP REL リレーションベースオブジェクトで定義されます。Mary と Adam 世帯間のリレーションを削除する必要があります。

Mary と Adam 世帯間のリレーションは正しくないため、Mary のリレーションレコードを削除することにします。有効期間の異なるすべてのリレーションレコードバージョンを削除する必要があります。すべての有効期間の Mary のリレーションレコードを削除すると、MDM Hub が、PARTY GROUP REL リレーションベースオブジェクトに関連付けられている相互参照テーブルで、レコードバージョンの Hub 状態を「削除済み」に変更します。

リレーション期間をすべて削除すると、PARTY GROUP REL リレーションベースオブジェクトに関連付けられている RL PARTY GROUP CROSS-REFERENCE 相互参照テーブルで、元のレコードバージョンの Hub 状態インジケータがすべて「削除済み」に設定されます。

リレーション期間をすべて削除する前に、相互参照テーブルに、Mary と Adam 世帯間の有効なリレーションがすべてアクティブであることが示されます。

行 ID	XREF 行 ID	オブジェクト	グループ ID	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63	22		147	アクティブ	28	2006 年 10 月 6 日	2015 年 12 月 31 日
66	22		147	アクティブ	28	2016 年 1 月 1 日	2999 年 12 月 31 日

リレーション期間をすべて削除した後、相互参照テーブルに、Mary と Adam 世帯間のすべてのリレーションが削除されたことが示されます。

行 ID	XREF 行 ID	オブジェクト	グループ ID	Hub 状態インジケータ	関係者グループ ID	期間の開始日	期間の終了日
63	22		147	削除済み	28	2006 年 10 月 6 日	2015 年 12 月 31 日
66	22		147	削除済み	28	2016 年 1 月 1 日	2999 年 12 月 31 日

リレーションベースオブジェクトに、Mary と Adam 世帯間のリレーションが削除されたことが示されます。

行 ID	オブジェクト	グループ ID	Hub 状態インジケータ	関係者グループ ID
22		147	-1	28

タイムライン抽出の使用

次の方法で、タイムライン抽出を実行できます。

- MDM Hub コンソールのバッチビューアツールで、BVT バージョンの抽出バッチジョブを実行する
- executeBatchExtractBVTVersions SIF の API を実行する

BVT バージョンの抽出バッチジョブを実行すると、抽出されたレコードが次のテーブルに追加されます。

- E\$_BO_NAME

BVT バージョンの抽出バッチジョブを実行する

影響を受けるベースオブジェクトに対して、MDM Hub コンソールのバッチビューアツールで、BVT バージョンの抽出バッチジョブを実行します。

注: バッチビューアツールで BVT バージョンの抽出バッチジョブを実行するときは、期日を指定できません。BVT バージョンの抽出バッチジョブでは、期日として現在のアプリケーションサーバーの時間が使用されます。期日を指定するには、executeBatchExtractBVTVersions SIF の API を実行します。

バッチビューアツールの詳細については、[「バッチビューアツールを使用したバッチジョブの実行」 \(ページ 544\)](#)を参照してください。

executeBatchExtractBVTVersions SIF の API を実行する

要求

executeBatchExtractBVTVersions 要求には、次のパラメータが含まれます。

orsId

オペレーショナル参照ストアの名前

tableName

ベースオブジェクトの名前

limitDate

BVT バージョンの抽出バッチジョブは、最終更新日が期日より前の相互参照レコードで機能します。

応答

executeBatchExtractBVTVersions 応答は、次のパラメータを返します。

Message

要求のステータスに関するメッセージが含まれます。

RetCode

戻りコードが含まれます。

EJB 要求の例

次の EJB 要求では、BVT バージョンの抽出バッチジョブが実行されます。

```
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 要求の例

次の SOAP 要求では、CMX_ORS10A オペレーショナル参照ストアの C_TIMELINE ベースオブジェクトに対して、BVT バージョンの抽出バッチジョブが実行されます。

```
<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:password>
    </urn:executeBatchExtractBVTVersions>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <urn:encrypted>false</urn:encrypted>
    </urn:password>
    <urn:orsId>localhost-orcl-CMX_ORS10A</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>

```

タイムライン抽出のプロパティの設定

タイムライン抽出機能を使用する場合は、cmxserver.properties ファイルでオプションのプロパティを設定できます。

- ▶ 次のオプションのプロパティを cmxserver.properties ファイルに追加します。

プロパティ	説明
cmx.server.batch.extractbvtversions.removePreviousExtractRecords	ディスク領域が過度に使用されないようにするため、最新のスナップショットを保持するかどうかを指定します。ディスク領域が過度に使用されないようにするには、true に設定します。デフォルトは false です。
cmx.server.batch.extractbvtversions.dropOutOfSyncBOExtractTable	ベースオブジェクトのカラムの変更によって既存の抽出テーブルが削除されないようにするかどうかを指定します。抽出テーブルが削除されないようにするには、false に設定します。デフォルトは true です。
cmx.server.batch.extractbvtversions.maxJobRejectExtractToKeep	拒否されたレコードの履歴を保持するジョブの数を指定します。デフォルトは 10 です。

第 11 章

状態管理および BPM ワークフローツール

この章では、以下の項目について説明します。

- [状態管理と BPM ワークフローツールの概要, 173 ページ](#)
- [MDM Hub での状態管理, 174 ページ](#)
- [BPM ワークフローツール, 178 ページ](#)
- [ActiveVOS 用に MDM Hub を設定する, 178 ページ](#)
- [保留中のレコードに対する一致の有効化, 183 ページ](#)
- [状態遷移のメッセージトリガ, 184 ページ](#)
- [レコードの昇格, 184 ページ](#)

状態管理と BPM ワークフローツールの概要

更新済みエンティティデータが変更承認ワークフローを通過してからその更新済みレコードがベストバージョンオブトゥールズ (BVT) レコードを提供することを確保できます。例えば、ビジネスプロセスでは、顧客データに対する更新がマスターデータになる前に、シニアマネージャによるその確認と承認が必要となる場合があります。

変更承認ワークフローをサポートするために、MDM Hub と Data Director (IDD) が ActiveVOS^(R)サーバーと統合されます。定義済み MDM ワークフロー、タスクタイプ、およびロールにより、コンポーネントが相互に同期するようになります。

コンポーネントは、以下の方法で変更承認ワークフローをサポートします。

- MDM Hub は、状態管理が有効になっているベースオブジェクトテーブルのレコードの状態を管理します。未承認レコードは保留状態となり、承認済みレコードはアクティブ状態となります。
- IDD アプリケーションで、権限を持つビジネスユーザーがエンティティデータを変更し、承認のために更新を送信します。
- ActiveVOS サーバーは、MDM ワークフローにおいてアクティビティを実行し、ビジネスマネージャとデータスチュワードのタスクを作成します。

例

ビジネスプロセスでは、顧客データがマスターデータとなる前に、シニアマネージャが更新を確認して承認する必要があります。IDD アプリケーションおよび MDM 環境は、ActiveVOS サーバーをデフォルトワークフローエンジンとして使用するように構成されています。

定義済みの一段階承認ワークフローは、1 人のシニアマネージャによる更新の確認と承認を必要とするビジネスプロセスを定義します。次の手順に、データスチュワードが一段階承認ワークフローを開始すると何が起きるかの概要を示します。

1. IDD アプリケーションで、データスチュワードは顧客エンティティを更新し、承認のために更新を送信します。
2. MDM オペレーショナル参照ストアで、レコードが保留状態になります。
3. ActiveVOS サーバーが、一段階承認ワークフロープロセスのアクティビティを実行開始します。
4. ActiveVOS サーバーが最終確認のユーザーアクティビティに到達すると、サーバーはシニアマネージャのためにタスクを作成します。
5. IDD アプリケーションで、すべてのシニアマネージャが各自のタスクインボックスでタスクを受信します。タスクは、更新された顧客エンティティにリンクされています。
6. シニアマネージャが更新を承認すると、以下のイベントが発生します。
 - IDD で、タスクが完了します。
 - ActiveVOS サーバーは、最終確認のユーザーアクティビティを完了としてマークし、一段階承認ワークフロープロセスにおける次のアクティビティを実行します。
 - MDM オペレーショナル参照ストアで、承認済みレコードがアクティブ状態になります。
 - MDM Hub で、承認されたレコードは BVT レコードにデータを提供します。

MDM Hub での状態管理

状態管理は、MDM Hub のレコードに関連付けられた状態の割り当てと変更を行うプロセスです。MDM Hub では、ベースオブジェクトテーブル上の状態管理を有効にすることができます。状態が有効なベースオブジェクトテーブルとその相互参照テーブルのすべてのレコードにはすべて、デフォルトの状態が割り当てられています。

レコードの状態は次の処理によって変更されます。

- ビジネスユーザーが BPM ワークフローツールでタスクを完了した場合。承認タスクの完了により、関連レコードの状態が変化することがあります。
- データスチュワードが、Hub コンソールでツールを使用して、一連のレコードを昇格、削除、またはリストアした場合。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。
- アプリケーションユーザーが SiperianClient API を使用して、一連のレコードを昇格、削除、またはリストアした場合。詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

レコードの状態

事前定義された MDM Hub レコードの状態は、アクティブ、保留、および削除済みです。

注: デフォルトでは、アクティブ状態の承認済みレコードだけがベストバージョン オブ トゥールズ(BVT)レコードとして有効です。アクティブレコードおよび保留レコード間で一致を許可するように操作を変更できます。

次の表では、レコードの状態について説明します。

レコードの状態	説明
アクティブ	<p>デフォルト。アクティブ状態のレコードが確認、承認されます。レコードが BMP 承認ワークフローで処理される場合、アクティブなレコードはその処理を通して承認されます。</p> <p>アクティブなレコードは次のような条件に一致します。</p> <ul style="list-style-type: none"> - すべての MDM Hub プロセスに、アクティブなレコードがデフォルトで含まれている。 - 少なくとも相互参照レコードの 1 つがアクティブ状態であれば、ベースオブジェクトレコードはアクティブである。 - アクティブな相互参照レコードのみが統合ベースオブジェクトに渡される。 - テーブルの HUB_STATE_IND システムカラム内のアクティブ状態のレコードには 1 と表示されている。
保留中	<p>保留状態のレコードとは、確認を待つ状態のレコードです。レコードが BPM 承認ワークフローで処理される場合、保留中のレコードに関連する確認タスクは未完了の状態になります。ワークフローに加えた承認保留中のレコードを編集することはできません。</p> <p>アクティブなレコードには次の品質が適用されます。</p> <ul style="list-style-type: none"> - 保留状態のレコードは、MDM Hub によって一般的な使用が承認されていない。 - 保留レコードは MDM Hub プロセスからデフォルトで除外されている。保留レコードに対してほとんどの操作を実行できますが、保留レコードを要求する必要があります。 - 保留中の相互参照レコードのみがある場合、MDM Hub は、保留レコードの信頼性によって、ベースオブジェクトの BVT を判別する。 - 保留状態のレコードを削除すると、そのレコードはテーブルから削除され、削除済みの状態にならず、復元できない。 - テーブルの HUB_STATE_IND システムカラム内の保留状態のレコードには 0 と表示されている。
削除済み	<p>削除済み状態のレコードは、MDM Hub データの一部として含まれなくなります。</p> <p>アクティブなレコードには次の品質が適用されます。</p> <ul style="list-style-type: none"> - 削除済みレコードは MDM Hub プロセスからデフォルトで除外されている。削除済みレコードを含めるには、これらのレコードを要求する必要があります。 - 削除済み状態のレコードは復元できない。 - テーブルの HUB_STATE_IND システムカラム内の削除済み状態のレコードには -1 と表示されている。

Hub 状態インジケータ

状態が有効なベースオブジェクトテーブルと相互参照テーブルには、テーブル内のレコードの状態を表すための値を指標するシステムカラム、HUB_STATE_IND があります。

次の表では、HUB_STATE_IND の可能な値を示して、レコードの状態と対応付けます。

HUB_STATE_IND の値	レコードの状態
1	アクティブ
0	保留

HUB_STATE_IND の値	レコードの状態
-1	削除済み
-9	物理削除後、相互参照履歴が有効になっている場合に使用。相互参照が削除された場合、相互参照履歴テーブル（HXRF）に HUB_STATE_IND が-9 のレコードが書き込まれます。同様に、ベースオブジェクトレコードが物理的に削除された場合、削除されたベースオブジェクトの履歴テーブル（HIST）に、HUB_STATE_IND が-9 のレコードが追加されます。

状態遷移

状態遷移ルールは、レコードがある状態から別の状態に変化可能かどうか、またいつ変化するかを決定します。

次の表では、一般的な遷移について説明し、ベースオブジェクトレコードと相互参照レコードの相違について記します。

状態が変わる前	変わった後の状態	注意事項
アクティブ	削除済み	この遷移は、論理削除と言います。
保留	アクティブ	この遷移は、昇格と言います。
削除済み	アクティブ	この遷移は、復元と言います。 相互参照レコードは復元可能です。ベースオブジェクトレコードは、相互参照レコードが復元された場合にのみ復元できます。
削除済み	保留	削除済みレコードを保留レコードに変更することはできません。この遷移は間接的にしか発生しません。アクティブなベースオブジェクトレコードを削除して、保留中の相互参照レコードをその削除済みレコードに追加すると、ベースオブジェクトレコードの状態はアクティブから削除済みへ、次に保留へと変わります。

注: 次の状態遷移は無効です。

- アクティブから保留。アクティブなレコードを保留レコードに変えることはできません。
- 保留から削除済み。保留レコードは削除することはできますが、削除済みの状態に遷移しません。代わりに、そのレコードは MDM Hub によってデータベースから削除されます。これを物理削除と言います。物理削除されたレコードは復元できません。

保留中のレコードの保護

同じプロセスの一部ではない更新から、保留レコードを保護することができます。この設定は相互作用 ID によって制御されます。

ベースオブジェクトテーブルまたは相互参照テーブルに相互作用 ID が含まれている場合、Hub コンソールでツールを使用して、ベースオブジェクトテーブルまたは相互参照テーブルのレコードの状態を保留からアクティブに変更することはできません。相互参照レコードに相互作用 ID が含まれている場合、保留中の相互参照レコードは、元の相互参照レコードと同じプロセスの一部ではない更新から相互作用 ID によって保護されます。代わりに、状態管理 SIF API 要求のいずれか 1 つを使用します。

注: 相互作用 ID は、API を使用して指定できます。ただし、バッチ処理を実行する場合は指定できません。例えば、相互作用 ID によって保護されるレコードは、ロードバッチプロセスによって更新できません。

次の表では、既存のレコードと新しいレコードの一致プロセス中に、相互作用 ID フィールドがどのように作用するかを例示します。この例では、interaction_ID フィールドにバージョン A、バージョン B、および Null の値が含まれています。

新しいレコードの相互作用 ID	既存のレコードの相互作用 ID		
	バージョン A	バージョン B	Null
バージョン A	OK	エラー	OK
バージョン B	エラー	OK	OK
Null	エラー	エラー	OK

データのロードのルール

ロードバッチプロセスは、任意の状態のレコードをロードします。状態は、ステージングテーブルの入力カラムとして指定されます。入力状態は、マッピングでランディングテーブルカラムとして指定するか、派生させることができます。マッピングに入力状態が指定されていない場合、状態はアクティブと見なされます（ロードの挿入の場合）。レコードをロードバッチジョブから更新し、入力状態が NULL の場合、更新するレコードの既存の状態は変更されず元のままになります。

次の表に、最初のカラムの入力相互参照レコードの状態、先頭行の既存の相互参照レコードの状態、およびセル内の実行アクションを示します。

外部参照の状態	既存: アクティブ	既存: 保留中	既存: 削除済み	既存: 外部参照なし (rowid によるロード)	既存: ベースオブジェクトレコードなし
入力: アクティブ	更新	更新+昇格	更新+リストア	挿入	挿入
入力: 保留中	更新を保留	更新を保留	更新+リストアを保留	挿入を保留	挿入を保留
入力: 削除済み	論理削除	物理削除	論理削除	非推奨	非推奨
入力: 未定義	アクティブとして扱う	保留として扱う	削除として扱う	アクティブとして扱う	アクティブとして扱う

注: 物理削除後に履歴が有効になった場合、HUB_STATE_IND が-9 のレコードは、相互参照が削除されるときに相互参照履歴テーブル（HXRF）に書き込まれます。同様に、ベースオブジェクトレコードが物理的に削除された場合、HUB_STATE_IND が-9 のレコードは削除されたベースオブジェクトの履歴テーブル（HIST）に追加されます。

レコードの状態およびベースオブジェクトレコード値の存続性

マージ、配置、ロードプロセスの後、ベースオブジェクトレコードには、レコードの状態が異なる複数の相互参照レコードが含まれる可能性があります。この場合、ベースオブジェクトレコードの値には、アクティブな相互参照レコードの値のみが反映されます。

一般に、異なる状態の複数のレコードを解決する場合、MDM Hub プロセスによって次のルールが適用されます。

- アクティブなレコードの値が、保留または削除済みレコードの値に優先する。
- 保留レコードの値が、削除済みレコードの値に優先する。

注: 一致およびマージ操作の場合、マージ対象のレコードのソースとターゲットによって、レコードの状態ではなく、存続行 ID が決定される。

BPM ワークフローツール

ビジネスプロセス管理 (BPM) ツールを使用すると、ビジネスプロセスを自動化できます。

デフォルトの事前定義済みのワークフローエンジンは、Multidomain MDM に付属の ActiveVOS^(R)サーバーのライセンス供与されたバージョンです。インストールプロセスによって、このバージョンの ActiveVOS サーバーが MDM Hub と Data Director に統合され、事前定義済みの MDM ワークフロー、タスクタイプ、およびルールがデプロイされます。

Informatica ActiveVOS ワークフローエンジンは、次のアダプタをサポートしています。

- ビジネスサービスを通じてビジネスエンティティで操作するタスクのアダプタ。アダプタ名は **[BE ActiveVOS]** です。
- SIF API を通じてサブジェクト領域で操作するタスクのアダプタ。アダプタ名は **[Informatica ActiveVOS]** です。

また、BPM ツールのスタンドアロンインスタンスの統合も選択できます。

Informatica ActiveVOS

Informatica ActiveVOS のスタンドアロンインスタンスを環境内で実行すると、インスタンスを MDM Hub と Data Director に手動で統合できます。事前定義済みの MDM ワークフローをデプロイしたり、カスタムワークフローを作成したりすることができます。詳細については、『*Multidomain MDM Data Director - ActiveVOS の統合ガイド*』を参照してください。

サードパーティ BPM ツール

サードパーティインスタンスを環境内で実行すると、インスタンスを MDM Hub と Data Director に手動で統合できます。事前定義済みの MDM ワークフローをデプロイしたり、カスタムワークフローを作成したりすることができます。詳細については、『*Multidomain MDM Business Process Manager Adapter SDK の実装ガイド*』を参照してください。

重要: ビジネスエンティティベースの ActiveVOS ワークフローアダプタへの移行をお勧めします。Siperian ワークフローアダプタは非推奨です。非推奨扱いのアダプタも継続してサポートしますが、古くなるため、将来のリリースではサポートが廃止されます。MDM Hub はプライマリワークフローエンジンとセカンダリワークフローエンジンをサポートしています。Siperian ワークフローアダプタからビジネスエンティティベースの ActiveVOS ワークフローアダプタに移行できます。

ActiveVOS 用に MDM Hub を設定する

組み込みの ActiveVOS サーバーを所要するように MDM Hub を設定するには、次の手順を実行する必要があります。

1. オペレーショナル参照ストア内のベースオブジェクトテーブルでの状態管理を有効にします。

2. オペレーショナル参照ストアを ActiveVOS ワークフローエンジンにマッピングします。
3. 事前定義されたワークフローのユーザーロールをビジネスマネージャに割り当てます。

状態管理の有効化

Data Director アプリケーションを使用して更新できるレコードを含む各ベースオブジェクトテーブルで、状態管理を有効にする必要があります。

1. モデルワークベンチで、**[スキーマ]** をクリックします。
2. 書き込みロックを取得します。
3. スキーマツールで、アプリケーションを使用して更新できるレコードを含むベースオブジェクトテーブルを選択します。
4. **[詳細]** タブをクリックします。
5. **[状態管理を有効にする]** チェックボックスを選択します。
6. ベースオブジェクトに属する相互参照レコードの状態がいつ PENDING から ACTIVE に変わったかを追跡し続けるには、**[相互参照昇格の履歴]** チェックボックスを選択します。

ワークフローエンジンの追加

ワークフローエンジンを MDM Hub に追加する場合は、ワークフローエンジンとワークフローアダプタを関連付けます。

ワークフローエンジンを追加すると、これがプライマリワークフローエンジンになり、既存のプライマリワークフローエンジンはセカンダリワークフローエンジンになります。既存のセカンダリワークフローエンジンがある場合、これはオペレーショナル参照ストアから削除され、そのタスクがタスクインボックスから削除されます。

1. 設定ワークベンチで、**[Workflow Manager]** をクリックします。
2. 書き込みロックを取得します。
3. **[ワークフローエンジン]** タブを選択し、**[追加]** ボタンをクリックします。
4. **[ワークフローの追加]** ダイアログボックスで、ワークフローエンジンのプロパティを入力します。

次の表に、ワークフローエンジンのプロパティを示します。

フィールド	説明
ワークフローエンジン	ワークフローエンジンの表示名
アダプタ名	ビジネスエンティティに基づいて ActiveVOS ワークフローアダプタに [BE ActiveVOS] を選択します。
ホスト	Informatica ActiveVOS インスタンスのホスト名。
ポート	Informatica ActiveVOS インスタンスのポート名。
ユーザー名	信頼されたユーザーのユーザー名。

フィールド	説明
パスワード	信頼されたユーザーのパスワード。
プロトコル	MDM Hub と ActiveVOS 間の通信プロトコル。プロトコルには http または https が使用できます。

5. **[OK]** をクリックします。

プライマリワークフローアダプタとセカンダリワークフローアダプタの設定

BE ActiveVOS ワークフローアダプタに移行するには、BE ActiveVOS ワークフローエンジンをプライマリワークフローエンジンとして選択します。ビジネスエンティティのワークフローアダプタの名前は、BE ActiveVOS です。既存のタスクをセカンダリワークフローエンジンで処理することはできますが、タスクを作成することはできません。プライマリとセカンダリに同じワークフローエンジンを選択しないでください。

ActiveVOS を搭載した Multidomain MDM を今まで使用したことがない場合、プライマリワークフローアダプタとして BE ActiveVOS アダプタを選択します。既存タスクを処理するのにセカンダリワークフローアダプタを選択する必要はありません。

注: Data Director アプリケーションがサブジェクト領域を使用する場合は、プライマリワークフローエンジンとして Informatica ActiveVOS アダプタの使用を継続します。

ワークフローエンジンを追加すると、これがプライマリワークフローエンジンになり、既存のプライマリワークフローエンジンはセカンダリワークフローエンジンになります。既存のセカンダリワークフローエンジンがある場合、これはオペレーショナル参照ストアから削除され、そのタスクがタスクインボックスから削除されます。

1. **[設定]** ワークベンチで、**[Workflow Manager]** をクリックします。
2. 書き込みロックを取得します。
3. **[ワークフローエンジン]** タブを選択し、次の BE ActiveVOS ワークフローアダプタ情報が正確であることを確認します。
 - ActiveVOS サーバーホスト
 - ActiveVOS サーバーポート
 - 信頼できるユーザーのユーザー名
 - 信頼できるユーザーのパスワード
 - MDM Hub と ActiveVOS サーバーとの間の通信のプロトコル
4. **[オペレーショナルリファレンスストアのワークフローマッピング]** タブを選択します。
タブのテーブルでは、Hub ストア内のすべてのオペレーショナル参照ストアデータベースが含まれています。
5. **[プライマリワークフローエンジン]** カラムで、BE ActiveVOS ワークフローアダプタのワークフローエンジンを選択します。
6. **[セカンダリワークフローエンジン]** カラムで、ワークフローエンジンを選択します。

ユーザーロールの設定

MDM が ActiveVOS サーバーでユーザーを確実に認証できるようにするには、MDM Hub、Data Director、および ActiveVOS サーバーで同一のユーザーロールを設定する必要があります。これらのユーザーロールを

MDM Hub 内のユーザーに割り当てることで、すべてのコンポーネントへのユーザーアクセスがセキュリティアクセスマネージャによって管理されます。

複数のロールをユーザーに割り当てることができます。ロール特権は累積なので、複数のロールを持つユーザーは各ロールに関連する組み合わせされた特権を受け取ります。例えば、ユーザーにデータスチュワードロール、abAdmin ロール、abadmin ロールを割り当てることができます。ユーザーは組み合わせされたワークフローと、各ロールに関連するタスク管理の特権を受け取ります。

次の表に、必要なユーザーロールを示し、ワークフローアクションを識別し、ロールに関連するタスク管理アクションを示します。

ユーザーロール	ワークフローアクション	タスク管理アクション
データスチュワード	更新、マージ、マージ解除、通知	<p>ユーザーは自分のユーザーロールで使用可能なタスクでの次のアクションを実行できます。</p> <ul style="list-style-type: none"> - クレーム - 解放 - 編集 - 許容、拒否、または引き受け解除などのタスクアクション <p>注: 使用可能なタスクアクションは、ロールおよび ActiveVOS ワークフローによって異なります。</p>
マネージャ	確認不承認	<p>ユーザーは自分のユーザーロールで使用可能なタスクでの次のアクションを実行できます。</p> <ul style="list-style-type: none"> - クレーム - 解放 - 編集 - 許容、拒否、または引き受け解除などのタスクアクション <p>注: 使用可能なタスクアクションは、ロールおよび ActiveVOS ワークフローによって異なります。</p>
SrManager	最終確認	<p>ユーザーは自分のユーザーロールで使用可能なタスクでの次のアクションを実行できます。</p> <ul style="list-style-type: none"> - クレーム - 解放 - 編集 - 許容、拒否、または引き受け解除などのタスクアクション <p>注: 使用可能なタスクアクションは、ロールおよび ActiveVOS ワークフローによって異なります。</p>

ユーザーロール	ワークフローアクション	タスク管理アクション
abAdmin	-	<p>ユーザーはすべてのタスクで次のアクションを実行できます。</p> <ul style="list-style-type: none"> - 割り当て - 解放 - 編集 <p>注: ユーザーはデフォルト承認のタスクを管理し、ActiveVOS ワークフローのマージを解除できます。</p>
abadmin	-	<p>ユーザーはすべてのタスクで次のアクションを実行できます。</p> <ul style="list-style-type: none"> - 割り当て - 解放 - 編集 <p>注: ユーザーはデフォルトマージ ActiveVOS ワークフローのタスクを管理できます。</p>

注: ビジネスプロセス管理で ActiveVOS を使用する場合は、ユーザーロールにスペースを含めることはできません。

abAdmin および abadmin ロールのタスク管理特権を有効にするには、プロビジョニングツールでタスク管理者ロールを設定する必要があります。詳細については、『*Multidomain MDM のプロビジョニングツールガイド*』を参照してください。

ワークフローユーザーロールの割り当て

定義済みの各ワークフローユーザーロールを、ビジネスマネージャまたはデータスチュワードに割り当てます。

事前に、ワークフローに参加する必要のあるすべてのビジネスユーザーの MDM Hub ユーザーアカウントがあることを確認しておきます。必要に応じて、ビジネスユーザーの新しいユーザーアカウントを追加します。新しいユーザーを追加する場合は、同じユーザーをアプリケーションサーバーのコンテナに追加します。

1. Hub コンソールで、Informatica Data Director アプリケーションがアクセスするオペレーショナル参照ストアに接続します。
2. 書き込みロックを取得します。
3. セキュリティアクセスマネージャワークベンチを開き、**[ユーザーとグループ]** をクリックします。
[ユーザーとグループ] ツールが開きます。
4. **[ロールへのユーザー/グループの割り当て]** タブをクリックします。
5. ワークフローのユーザーロールを選択し、**[編集]** をクリックします。
[ロールへのユーザーの割り当て] ダイアログボックスが開きます。
6. このロールを必要とするユーザーまたはユーザーグループを選択し、**[OK]** をクリックします。
7. 別のワークフローロールをユーザーに割り当てル場合は、手順 5 と 6 を繰り返します。

ActiveVOS Web アプリケーションへのアクセスの有効化

Informatica ActiveVOS には、ActiveVOS コンソールと ActiveVOS Central の 2 つの Web アプリケーションが含まれています。アプリケーションサーバーでユーザーとワークフローのロールを定義することで、これらの Web アプリケーションへのユーザーアクセスが可能になります。

MDM Hub で定義されていたものと同じユーザーロールとユーザー資格情報を定義します。ロールとユーザーの作成方法については、アプリケーションサーバーのマニュアルを参照してください。

タスク管理者ロールの設定

プロビジョニングツールを使用して、タスク管理者ロールを設定し、タスク管理者ロールにマッピングする MDM Hub ロールを指定できます。

開始する前に、デフォルトの ActiveVOS ワークフローを使用している場合は、MDM Hub で abAdmin と abadminMDM Hub ロールを作成する必要があります。

重要: ユーザーがすべてのワークフローのすべてのタスクでタスク管理アクションを実行できるようにするには、ユーザーに abAdmin および abadmin の両方の MDM Hub ロールを割り当てる必要があります。

1. プロビジョニングツールにログインします。
2. **【データベース】** リストから、設定に関連付けるデータベースを選択します。
3. **【ビジネスエンティティ】** > **【タスク】** をクリックします。
4. **【タスク】** パネルで **【タスク管理者ロール】** を選択し、**【作成】** をクリックします。
5. **【プロパティ】** パネルで、**【名前】** フィールドに「TaskAdministrator」と入力します。
6. **【タスク管理者ロールの有効化】** チェックボックスを選択します。
7. **【MDM Hub ロール】** リストから、タスク管理者ロールにマッピングする MDM Hub ロールを選択します。

デフォルトの ActiveVOS ワークフローを使用している場合は、**【abAdmin】** を選択します。カスタム ActiveVOS ワークフローを使用している場合は、ActiveVOS でビジネス管理に使用する MDM Hub ロールを選択します。

8. **【適用】** をクリックします。

変更は保存されますが、MDM Hub にはパブリッシュされません。

9. 変更内容を MDM Hub にパブリッシュします。

- a. **【パブリッシュ】** をクリックします。

確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。

- b. 変更内容を確認するか、確認せずにパブリッシュします。

- 確認せずにパブリッシュする場合は、**【パブリッシュ】** をクリックします。
- 確認後にパブリッシュするには、**【変更の確認】** をクリックし、画面に表示される指示に従います。

タスク管理者ロールを設定しました。マッピングされた MDM Hub ロールを割り当てられたユーザーは、Data Director のすべてのタスクに対してタスク管理アクションを実行できます。

保留中のレコードに対する一致の有効化

デフォルトでは、一致プロセスには、アクティブなレコードが含まれ、保留レコードは除外されます。保留レコードを含めることができます。

1. モデルワークベンチを開き、**【スキーマ】** をクリックします。
2. スキーマツールで、状態が有効なベースオブジェクトを選択します。
3. ベースオブジェクトのツリーを展開して、**【一致/マージ設定】** を選択します。
4. **【一致/マージ設定の詳細】** パネルの **【プロパティ】** タブで、**【保留中のレコードに対して一致を有効にする】** チェックボックスを選択します。

状態遷移のメッセージトリガ

MDM Hub は、メッセージキューのメッセージを使用して、外部アプリケーションと通信します。ベースオブジェクトレコードと相互参照レコードの状態が変化すると報告を行うメッセージトリガを有効にすることができます。

ルールの定義に該当する処理が発生すると、メッセージキューにメッセージが配置されます。メッセージトリガによって、メッセージが配置されるキューが指定されます。

次のメッセージトリガイベントをレコードの状態の変化に利用できます。

イベント	アクション
新しい保留データの追加	新しい保留レコードが作成されます。
既存の保留データの更新	保留中のベースオブジェクトレコードが更新されます。
相互参照レコードが変化した場合にのみ保留を更新	保留中の相互参照レコードが更新されます。このイベントにはレコードの昇格が含まれます。
ベースオブジェクトデータの削除	ベースオブジェクトレコードが論理削除されます。
相互参照データの削除	相互参照レコードが論理削除されます。
保留中のベースオブジェクトデータの削除	ベースオブジェクトレコードが物理削除されます。
保留中の相互参照データの削除	相互参照レコードが物理削除されます。

状態遷移のメッセージトリガの有効化

レコードの状態が変化したときに外部アプリケーションに通知するメッセージトリガを有効にすることができます。

MDM Hub は、メッセージキューを使用して、外部アプリケーションと通信します。そうしない場合は、アプリケーションサーバーでメッセージキューを設定する必要があります。メッセージキューは公開プロセスの一部です。

1. **モデルワークベンチ**を開いて、**[スキーマ]** をクリックします。
2. メッセージキューツールで、メッセージキューの **【保留の更新時にトリガを実行】** チェックボックスを選択します。

レコードの昇格

レコードの昇格とは、一連のレコードのシステム状態を保留状態からアクティブ状態に変更する処理です。レコードの昇格は、データスチュワードツールまたは昇格バッチプロセスのいずれかを使用して手動で設定できます。

昇格バッチプロセスでは、昇格対象のベースオブジェクトに関連付けられている子も昇格させます。子レコードと孫レコードを昇格させるには、昇格バッチを2度実行する必要があります。最初に、親ベースオブジェク

トに対して昇格バッチジョブを実行します。続いてもう一度、孫ベースオブジェクトに対して昇格バッチジョブを実行します。

注: MDM Hub でレコードを削除するときには、同様に行います。子レコードを削除するには、ベースオブジェクトを削除します。続いて 2 度目の削除ジョブを実行し、ベースオブジェクトの孫レコードを削除します。

データスチュワードツールでのレコードの昇格

保留状態のベースオブジェクトまたは相互参照レコードは、データスチュワードワークベンチ内のツールを使用して直ちにアクティブ状態に昇格させることができます。また、データマネージャまたはマージマネージャを使用し、これらのレコードに後で昇格させるためのフラグを設定することもできます。Hub コンソールを使用してこれらのタスクを実行する方法については、*Multidomain MDM のデータスチュワードガイド*を参照してください。

ベースオブジェクトレコードまたは相互参照レコードに後で昇格させるためのフラグを設定する

データマネージャを使用して、昇格するレコードにフラグを付けることができます。フラグ付きレコードを昇格するには、昇格バッチジョブを実行します。

1. データスチュワードワークベンチを開き、**[データマネージャ]** をクリックします。
2. データマネージャツールで、ベースオブジェクトレコードまたは相互参照レコードをクリックします。
3. 関連するパネルで **[昇格フラグを設定]** をクリックします。

注: パッケージに対して HUB_STATE_IND フィールドが読み取り専用設定されている場合は、Hub コンソールのデータマネージャツールおよびマージマネージャツールで関連するレコードに対して **[レコード状態を設定]** ボタンが無効になります。ただし、**[昇格フラグを設定]** ボタンは、レコードの HUB_STATE_IND カラムを直接変更しないので、アクティブなままとなります。

マージマネージャを使用して、一致レコードに昇格のフラグを付ける

マージマネージャを使用して、昇格する相互参照レコードにフラグを付けることができます。フラグ付きレコードを昇格するには、昇格バッチジョブを実行します。

1. データスチュワードワークベンチを開き、**[データマネージャ]** をクリックします。
2. 一致レコードをクリックします。
3. **[一致レコード]** パネルで **[昇格フラグを設定]** をクリックします。

バッチビューアを使用した昇格バッチジョブの設定

昇格フラグの付いたレコードを昇格するバッチジョブを設定できます。

1. 昇格する保留レコードにフラグを付けます。
2. ユーティリティワークベンチを開き、**[バッチビューア]** をクリックします。
3. バッチビューアの **[ベースオブジェクト]** ノードで、**[昇格]** バッチジョブをクリックします。
4. **[フラグ付きレコード abc の昇格]** を選択します。
「abc」は、あらかじめ昇格用にフラグ設定した関連レコードを表します。
5. 昇格フラグが付いたレコードを昇格するには、**[バッチの実行]** をクリックします。

バッチグループツールを使用した昇格バッチジョブの設定

フラグ付きレコードを昇格するには、バッチグループツールを使用して昇格バッチジョブを追加します。

1. 昇格する保留レコードにフラグを付けます。
2. ユーティリティワークベンチを開き、**[バッチグループ]** をクリックします。
3. 書き込みロックを取得します。
4. バッチグループツリーで **[バッチグループ]** ノードを右クリックして、**[バッチグループの追加]** を選択します。
5. バッチグループツリーでいずれかのレベルを右クリックして、レベルをバッチグループに追加するオプションを選択します。

[バッチグループに追加するジョブの選択] ダイアログボックスが開きます。

6. 追加するジョブのベースオブジェクトを展開します。
7. **[< XREF テーブル>のフラグ付きレコードの昇格]** を選択します。
8. **[OK]** をクリックします。

選択したジョブがバッチグループに追加されます。

9. **[保存]** をクリックします。

設定が完了しました。バッチグループジョブを実行できます。

第 12 章

データ暗号化

この章では、以下の項目について説明します。

- [データ暗号化の概要, 187 ページ](#)
- [データ暗号化アーキテクチャ, 187 ページ](#)
- [データ暗号化の制限, 188 ページ](#)
- [データ暗号化ユーティリティ, 188 ページ](#)
- [データ暗号化の設定, 189 ページ](#)
- [サービス統合フレームワーク API 要求および応答, 191 ページ](#)
- [サンプルのデータ暗号化プロパティファイル, 192 ページ](#)

データ暗号化の概要

MDM Hub 実装を利用し、機密データを格納したり、ネットワークを介して機密データを転送したりするには、データ暗号化が有効になるように MDM Hub を設定します。

データ暗号化が行われるように MDM Hub を設定した後は、暗号化された形式で機密データをデータベースに格納できます。MDM Hub によってデータがデータベースから Hub サーバー、Process サーバー、およびサブジェクト領域を含む Data Director に転送されるときには機密データが暗号化された形式で転送されます。データ暗号化を使用すると、ネットワーク上で送信するあらゆる機密データを安全に保つことができます。

注: 暗号化できるのは、VARCHAR データ型のデータのみです。

データ暗号化アーキテクチャ

機密データは、暗号化された形式でデータベースに格納することも、暗号化された形式で Hub サーバー、Process サーバー、Data Director などに転送することもできます。データ暗号化を使用すると、MDM Hub 実装内の機密データを安全に保つことができます。

暗号化されたデータは、ユーザーに表示される前に Data Director などのサービス統合フレームワーク (SIF) クライアントで復号化されます。データベース内の暗号化されたデータは、クレンジングプロセスの前にも復号化が行われます。

MDM Hub は、SiperianClient オブジェクトを介して各 SIF API 呼び出しを行います。各呼び出しは要求と応答から構成されています。送信する要求は暗号化する必要があり、受信する応答は復号化する必要があります。

MDM Hub で、ネットワークにおける以下のデータ交換を暗号化できます。

- Hub サーバーとデータベースの間
- クレンジング操作時における Hub サーバーと Process サーバーの間
- Data Director と Hub サーバーの間

データ暗号化の制限

ベースオブジェクトのカラムのデータ暗号化を設定する際には、暗号化された形式でデータが格納されます。

データ暗号化を設定する前に、次の制限について考慮します。

- データマネージャまたはマージマネージャのような Hub コンソールツールを使用して、データを暗号化されたカラムに追加または編集する場合、データは暗号化されません。
- データの暗号化はビジネスエンティティを含む Data Director ではサポートされません。データ暗号化はサブジェクト領域を含む Data Director でのみサポートされます。
- データ暗号化は REST API ではサポートされません。データ暗号化はサービス統合フレームワーク (SIF) API でのみサポートされます。
- SIF API が EjbSiperianClient を使用しない場合、要求内のデータは暗号化されず、応答は復号化されません。要求を送信する前に機密データを暗号化し、応答を受信した後に暗号化データを復号化する必要があります。
- SOAP 呼び出しはデータを暗号化しません。SOAP 呼び出しを使用してデータを暗号化されたカラムに挿入する場合、データを事前に暗号化する必要があります。
- SearchMatch および SearchQuery などの SOAP 呼び出しは、暗号化されたカラムに存在するデータを検索できません。
- 暗号化されたカラムから Get などの SOAP 呼び出しでデータを取得する場合、そのデータは暗号化された形式で返されます。
- 一致フィールドが複数のカラムの連結である場合、暗号化されたデータにはスペースを含めることができません。これらの各カラムは、カラムにデータが含まれない場合でも暗号化される必要があります。
- ワイルドカード文字およびリテラルを使用する検索クエリは検索結果を取得しません。

データ暗号化ユーティリティ

MDM Hub にデータ暗号化を設定するには、リソースキットに含まれるデータ暗号化ユーティリティを使用できます。

リソースキットには、以下のデータ暗号化サンプルとデータ暗号化ユーティリティが含まれています。

データ暗号化ライブラリ

MDM Hub で、データ暗号化ライブラリをデータ暗号化 JAR ファイルにバンドルする必要があります。データ暗号化ライブラリには、API と共通クラスが含まれます。特に、データ暗号化を設定するときに実装する必要がある DataEncryptor インタフェースなどが含まれています。DataEncryptor インタフェースは、MDM Hub でのデータ暗号化のために実装する必要がある encrypt メソッドと decrypt メソッドを定義します。

データ暗号化プロパティファイルのサンプル

サンプルのデータ暗号化プロパティファイルには、データ暗号化実装に必要なパラメータが入っています。プロパティファイルの名前は、dataencryption.properties です。このサンプルのプロパティファイルをカスタマイズしてデータ暗号化の実装のオプションを指定できます。

サンプルの DataEncryption インタフェースの実装

サンプルの DataEncryption インタフェースの実装では、InformaticaDataEncryptor クラスを使用して DataEncryptor インタフェースが実装されます。カスタム暗号化アルゴリズムを作成する場合は、このサンプルの実装を参照することができます。データ暗号化プロパティファイル内の mainClass プロパティは、インタフェースの実装で使用するクラス名を参照する必要があります。

Ant ビルドスクリプト

データ暗号化 JAR ファイルを作成する Ant ビルドスクリプト、build.xml。

データ暗号化の設定

データ暗号化を使用するには、データ暗号化の MDM Hub を設定する必要があります。

1. DataEncryptor インタフェースを実装します。
2. データ暗号化プロパティファイルを設定します。
3. Hub サーバーのデータ暗号化を設定します。
4. Process サーバーのデータ暗号化を設定します。

手順 1. DataEncryptor インタフェースの実装

DataEncryptor インタフェースを実装する必要があります。DataEncryptor インタフェースは、encrypt メソッドと decrypt メソッドを定義します。encrypt メソッドと decrypt メソッドの実装は、スレッドセーフである必要があります。

1. Java の統合開発環境で Java プロジェクトを作成します。
2. その Java プロジェクトに、以下の MDM Hub JAR ファイルを追加します。
 - siperian-api.jar
 - siperian-common.jarjar ファイルは次のディレクトリにあります。
UNIX の場合: `<infadm_install_dir>/resourcekit/samples/DataEncryption/lib`
Windows の場合: `<infadm_install_dir>\resourcekit\samples\DataEncryption\lib`
3. encrypt メソッドと decrypt メソッドを含む、データ暗号化の Java クラスを作成します。
4. このデータ暗号化 Java クラスをコンパイルします。
5. クラスファイルをカスタムデータ暗号化 JAR ファイルとしてパッケージ化するため、次のコマンドを実行します。

```
ant build
```
6. ビルドが完了した後、生成されたファイルをクリーンアップするため、次のコマンドを実行します。

```
ant clean
```
7. 作成したカスタムデータ暗号化 JAR ファイルに、DataEncryptor インタフェースを実装します。

手順 2. データ暗号化プロパティファイルの設定

データ暗号化の実装に必要なパラメータが入っているサンプルのデータ暗号化プロパティファイルが Informatica から提供されています。このサンプルのプロパティファイルをカスタマイズしてデータ暗号化の実装のオプションを指定できます。

1. 次のディレクトリにある dataencryption.properties ファイルを検索します。
UNIX の場合: `<ResourceKit_install_dir>/DataEncryption/resources`
Windows の場合: `<ResourceKit_install_dir>\DataEncryption\resources`
2. dataencryption.properties ファイルのバックアップコピーを作成します。
3. テキストエディタを使用してこのファイルを開き、データ暗号化パラメータを編集します。
4. DataEncryptor インタフェース実装への参照を指定します。
5. 機密データ、機密データに関連するテーブル（履歴テーブルや相互参照テーブルなど）、およびパッケージが表示されるベースオブジェクトのカラムの暗号化と復号化を行うように EJBServerClient を設定します。

以下の構文を使用して、ベースオブジェクトのカラム名、関連するテーブル（履歴テーブルや相互参照テーブルなど）、およびパッケージを指定します。

```
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. あいまい一致およびあいまい検索操作を使用する場合、使用する一致フィールドを指定します。

次の構文を使用して一致フィールド名を指定します。

```
MATCHFIELD.<ORS_ID>.BASE_OBJECT.<BASE_OBJECT_NAME>.<MATCH_FIELD_NAME>
```

注: 指定するベースオブジェクトが、一致パスコンポーネントのルートである、照合しているベースオブジェクトであることを確認します。一致フィールドが一致パスコンポーネントのルートではないベースオブジェクトのものである場合でも、一致パスコンポーネントのルートであるベースオブジェクトを指定します。

一致フィールドが複数のカラムの連結である場合、カラムにデータが含まれない場合でも、これらのカラムをそれぞれ暗号化する必要があります。

7. データの暗号化と復号化を行う必要がある、クレンジング関数の入力ポートと出力ポートを設定します。

以下の構文を使用して、クレンジング関数の入力ポートと出力ポートを指定します。

```
CLEANSE.<PORT_NAME>=true
```

8. 同じ名前（dataencryption.properties）でプロパティファイルを保存します。

手順 3. Hub サーバーのデータ暗号化の設定

データ暗号化を使用するように Hub サーバーを設定するには、カスタムデータ暗号化 JAR ファイルを使用するように Hub サーバーを設定します。

1. 次のディレクトリにある cmxserver.properties ファイルを検索します。
UNIX の場合: `<infadm_install_directory>/hub/server/resources`
Windows の場合: `<infadm_install_directory>\hub\server\resources`
2. テキストエディタを使用してこのファイルを開き、encryption.plugin.jar プロパティのデータ暗号化 JAR のパスを指定します。
`encryption.plugin.jar=<Path to the data encryption JAR>`
3. cmxserver.properties プロパティファイルを保存します。

手順 4. プロセスサーバーのデータ暗号化の設定

データ暗号化を使用するようにプロセスサーバーを設定するには、カスタムデータ暗号化 JAR ファイルを使用するようにプロセスサーバーを設定します。

1. 次のディレクトリにある cmxcleanse.properties ファイルを検索します。
UNIX の場合: `<infadm_install_directory>/hub/cleanse/resources`
Windows の場合: `<infadm_install_directory>\hub\cleanse\resources`
2. テキストエディタを使用してこのファイルを開き、encryption.plugin.jar プロパティのデータ暗号化 JAR のパスを指定します。
`encryption.plugin.jar=<Path to the data encryption JAR>`
3. cmxcleanse.properties プロパティファイルを保存します。

サービス統合フレームワーク API 要求および応答

MDM Hub は、データ暗号化 JAR ファイルに含まれるデータ暗号化 Java クラスを使用して、サービス統合フレームワーク (SIF) API を呼び出します。

以下の表に、データ暗号化に関連する、SIF API の要求と応答を示します。

要求/応答	説明
PutRequest	キーで識別される単一レコードをベースオブジェクトに挿入または更新します。 入力レコードを暗号化します。
MultiMergeRequest	同じオブジェクトを表す複数のベースオブジェクトレコードをマージします。マージされたレコードに対してフィールドレベルのオーバーライドを指定できます。 オーバーライドする側のフィールドを暗号化します。
GetResponse	既知のキーを使用してパッケージから単一レコードを取得します。 返されるレコードを暗号化します。
SearchHmQueryRequest	階層マネージャのエンティティとリレーションを検索します。 入力フィルタパラメータを暗号化します。パラメータではなく SiperianObjectUidField を使用して暗号化を有効にします。
SearchQueryRequest	パッケージから、指定した基準を満たすレコードのセットを取得します。 入力フィルタパラメータを暗号化します。パラメータではなく SiperianObjectUidField を使用して暗号化を有効にします。
SearchMatchRequest	照合カラムとルール定義に基づいてパッケージ内でレコードを検索します。 入力フィルタパラメータと、照合を行うレコードを暗号化します。パラメータではなく SiperianObjectUidField を使用して暗号化を有効にします。

要求/応答	説明
SearchHmQueryResponse	エンティティと、1つ以上のエンティティに関連付けられているリレーションを返します。 返されたレコードを暗号化します。
SearchMatchResponse	レコードとマッチトークンのリストを返します。 返されたレコードを暗号化します。
SearchQueryResponse	パッケージ内のレコードのセットを返します。 返されたレコードを暗号化します。
AddRelationshipRequest	2つのエンティティ間にリレーションを追加します。 レコードとして指定されたカスタムフィールドを暗号化します。
UpdateRelationshipRequest	リレーションの特性を変更する階層マネージャの要求。 レコードとして指定されたカスタムフィールドを暗号化します。
GetOneHopResponse	指定された階層設定内の指定されたエンティティのグループに直接関連するエンティティについての情報を返します。 返されたレコードを暗号化します。
GetEntityGraphResponse	指定されたエンティティのセットに関連するエンティティとリレーションのグラフを返します。 返されたレコードを暗号化します。
GetXrefForEffectiveDateResponse	指定された有効日の複数の相互参照レコードを返します。 返されたレコードを暗号化します。

サンプルのデータ暗号化プロパティファイル

リソースキットには、サンプルのデータ暗号化プロパティファイル `dataencryption.properties` が含まれます。

サンプルのデータ暗号化プロパティファイルは、次のディレクトリにあります。

UNIX の場合: `<infadm_install_dir>/hub/resourcekit/samples/DataEncryption/resources`

Windows の場合: `<infadm_install_dir>\hub\resourcekit\samples\DataEncryption\resources`

以下の例は、サンプルのディレクトリ暗号化プロパティファイルの内容を示しています。

```
# 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

```

第 13 章

階層

この章では、以下の項目について説明します。

- [階層の概要, 194 ページ](#)
- [階層の例, 195 ページ](#)
- [階層の設定について, 195 ページ](#)
- [作業を開始する前に, 196 ページ](#)
- [設定手順の概要, 196 ページ](#)
- [階層マネージャ用のデータの準備, 196 ページ](#)
- [階層マネージャ用にデータを準備する方法の例, 197 ページ](#)
- [HM リポジトリベースオブジェクトの作成, 201 ページ](#)
- [デフォルトのエンティティアイコンのアップロード, 202 ページ](#)
- [エンティティアイコンの設定, 202 ページ](#)
- [エンティティ, 203 ページ](#)
- [エンティティベースオブジェクト, 203 ページ](#)
- [エンティティタイプ, 207 ページ](#)
- [エンティティの表示オプション, 211 ページ](#)
- [エンティティベースオブジェクトをベースオブジェクトに戻す, 211 ページ](#)
- [階層タイプ, 212 ページ](#)
- [リレーションオブジェクト, 213 ページ](#)
- [リレーションベースオブジェクト, 213 ページ](#)
- [外部キーリレーションベースオブジェクト, 217 ページ](#)
- [リレーションタイプ, 217 ページ](#)
- [パッケージ, 221 ページ](#)
- [プロファイルの概要, 225 ページ](#)

階層の概要

MDM Hub コンソールで階層を設定することができます。階層が個々のレコード間の関係を示す一方で、データモデルはベースオブジェクト間の関係を示します。

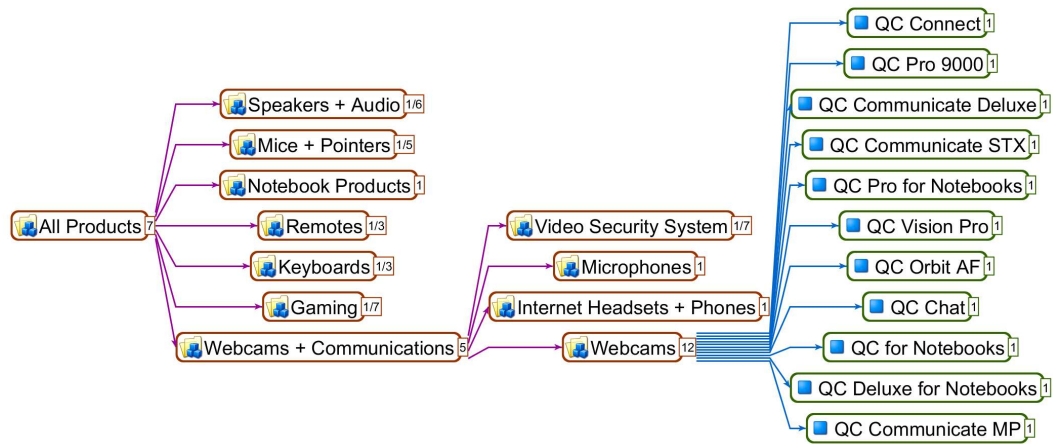
たとえば、組織内の各部門の所属する従業員を追跡するための階層を作成できます。また、各種製品グループ内の製品の階層を作成することもできます。

階層を設定するには、最初にエンティティベースオブジェクトを作成する必要があります。以下、同様。
階層を設定するには、階層マネージャのライセンスが必要です。

階層の例

リソースキット内のサンプルのオペレーショナルリファレンスストアには、事前設定された階層を持つスキーマが含まれています。この章では、製品階層の一部を使用して階層設定の手順を示します。

次の図に、Hub コンソールの階層マネージャツールに表示された、例で使用する階層の一部を示します。



階層の設定について

MDM Hub 管理者は、階層ツールを使用して、階層マネージャでデータのリレーションを表示および操作するために必要な構造を設定します。

階層ツールを使用して、MDM Hub 実装で使用する階層マネージャのコンポーネント（エンティティタイプ、階層、リレーションタイプ、パッケージ、およびプロファイル）を定義します。

階層マネージャのコンポーネントの定義が完了したら、パッケージまたはクエリのマネージャツールを使用してクエリ条件を更新できます。

注: 階層マネージャで使用するパッケージを設定して、プロファイルを検証する必要があります。

階層とそのコンポーネントの概念を理解するには、次の章で紹介する概念に関する知識が必要です。

- [第 8 章, 「スキーマの構築」 \(ページ 79\)](#)
- [第 9 章, 「クエリとパッケージ」 \(ページ 128\)](#)
- [第 24 章, 「統合プロセスの設定」 \(ページ 492\)](#)

作業を開始する前に

階層マネージャ（HM）の設定を開始する前に、完了しておく必要があるタスクがいくつかあります。

以下のタスクを完了させます。

- 空のオペレーショナル参照ストアまたは有効な ORS で開始し、データベースを CMX_SYSTEM に登録する。
- 階層マネージャのライセンスがあることを確認する。詳細については、Informatica MDM Hub 管理者に問い合わせてください。
- データ分析を実行する。

設定手順の概要

階層マネージャを設定するには、以下の手順を実行します。

1. Hub コンソールを開始します。
2. 階層ツールを起動します。
リポジトリベースオブジェクト（RBO）テーブルを作成していない場合、Hub コンソールに作成手順が表示されます。
3. エンティティオブジェクトとタイプを作成します。
4. 階層を作成します。
5. リレーションオブジェクトとタイプを作成します。
6. パッケージを作成します。
7. プロファイルを設定します。
8. プロファイルを検証します。

注: 階層マネージャの右クリックメニューで表示されるオプションは、階層ツールのメニューでも選択できます。

階層マネージャ用のデータの準備

HM を最大限に活用するため、情報を分析して以下の処理が実行済みであることを確認します。

- データソースが検証済みであり信頼できることを確認します。
- Informatica MDM Hub および HM で機能する有効なスキーマを作成します。
- エンティティ間に階層リレーション、外部キーリレーション、ワンホップ/マルチホップリレーション（直接および間接リレーション）を作成します。
すべての子エンティティには有効な親エンティティが関連付けられている必要があります。
HM に入力するデータは「孤立している」子エンティティを持つことはできません。
すべての階層が検証済みである必要があります。
1 ホップリレーションとマルチホップリレーションの詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。
- HM タイプを派生させます。

- 複数のソースシステムから重複エンティティを統合します。
例えば、あるエンティティグループ（ソース A）と別のエンティティグループ（ソース B）が同じであるのに、この 2 つのエンティティグループの名前が異なる場合があります。エンティティが同一と判断されたら、2 つのグループは統合可能です。
- エンティティを論理カテゴリにグループ化します（例えば、医師の名前を「医師」カテゴリにグループ化）。
- データが、参照整合性、無効なデータ、およびデータの変動性に関するルールに準拠していることを確認します。
これらのデータベースの概念の詳細については、データベースの参照資料を参照してください。

階層マネージャ用にデータを準備する方法の例

この節では、データを Informatica MDM Hub に入力する前、および階層マネージャに表示する前にデータを操作する方法の例を示します。

通常、企業のデータはここに示す例よりもはるかに大規模です。

シナリオ

John は会社のデータを操作して、最も効率的な方法でデータを階層マネージャに表示して使用できるようにする担当者です。

例を簡素化するために、コンピュータ部品を販売する企業の製品とそのタイプに関するデータのサブセットを対象にします。

この企業は 3 種類の製品を販売しています。マウス、トラックボール、およびキーボードです。これらの製品にはそれぞれ、複数のベンダーおよびさまざまなレベルの製品が含まれています（ゲーム用キーボードや TrackMan トラックボールなど）。

方法論

この節ではデータの簡素化の方法について説明します。

手順 1: データを階層にまとめる

この手順では、HM 設定に変換するデータを階層に整理します。

John は製品および製品グループ階層を分析することから始めます。製品を製品グループ別に、製品グループを親の製品グループ別に整理します。大量のデータおよびそのデータに含まれるリレーションは可視化するのが難しいため、John はカテゴリをリストして、それらの間に関係性があるかどうかを確認しました。

（マーケティング部門からのデータを含む）以下のテーブルは、John がどのようにデータを整理したかの一例です。

ProdGroup		ProdGroup		ProdGroup		製品	
ProdNumber	説明	ProdNumber	説明	ProdNumber	説明	ProdNumber	説明
ALL	全製品	100	マウス +ポイン タ	120	マウス	120-0001	レーザーマウス

ProdGroup		ProdGroup		ProdGroup		製品	
						120-0002	ナノコードレスレーザーマウス
						120-0003	コードレス光学式マウス
						120-0004	ナノコードレスレーザーマウス II
						120-0005	ノートブック用レーザーマウス
						120-0006	レボリューション
						120-0007	リチャージャブルコードレスマウス
						120-0008	コードレス光学式マウス II
		200	キーボード	210	キーボード	-	-
				220	キーボードとマウスのコンボ	-	-

注: ほとんどのデータセットには、これよりも多くの項目が存在します。

このテーブルは Products BO に格納されることになるデータを示しています。これは、HM で変換（または作成）される BO です。このテーブルには、Mice や Laser Mouse などのエンティティがあります。リレーションはグループ化によって示されており、Mice と Laser Mouse の間にはリレーションがあります。見出しの値はエンティティタイプです。Mice は Product Group で、Laser Mouse は Product です。これは、Product テーブルのフィールドに格納されるタイプです。

このようにデータを整理することによって、John はデータに含まれるエンティティおよびエンティティタイプの数や、それらのエンティティにどのようなリレーションがあるかを理解できました。

主要なカテゴリは ProdGroup で、これには製品グループ（Mice + Pointers など）、カテゴリ Product、および製品自体（Trackman Wheel など）が含まれます。これらの項目間のリレーションは、リレーションオブジェクトにカプセル化できます。これを John は Product Rel と呼ぶことにしました。Product Rel に関する情報に、John はリレーションの説明を含めました。製品グループは製品と製品グループ両方の親です。

手順 2: リレーションベースオブジェクトテーブルを作成する

データを分析し、John は以下の結論に達しました。

- 製品 (BO) はエンティティオブジェクトに変換する必要がある。
- 製品グループと製品はエンティティタイプである。
- 製品リレーションは作成する必要があるリレーションオブジェクトである。
- 以下のリレーションタイプを作成する必要がある (テーブルには表示されていないものもある)。
 - 製品は製品の親 (非表示)
 - 製品グループは製品の親 (例: Laser Mouse に対する Mice)。
 - 製品グループは製品グループの親 (例: Mice + Pointers は Mice の親)。

John は階層ツールを使用してアクセスを開始します。ツールにアクセスすると、リレーションベースオブジェクトテーブル (RBO テーブル) が作成されます。RBO テーブルは基本的に、特定のカラムを含むベースオブジェクトに必要なシステムベースオブジェクトです。手順 1 のテーブルにあるデータのような HM 構成データを格納します。

ベースオブジェクトの作成手順については、「[ベースオブジェクトの作成](#)」 (ページ 108) を参照してください。この節では、スキーマツールでベースオブジェクトの例を作成する場合に行う選択について説明します。

前の手順で特定したエンティティオブジェクトとリレーションオブジェクトごとにベースオブジェクトを作成して設定します。この例では、製品のベースオブジェクトを作成して HM エンティティオブジェクトに変換します。製品リレーション BO は変換せずに HM に直接作成します (より簡単な方法です)。新しいベースオブジェクトはそれぞれ、スキーマパネルの [ベースオブジェクト] カテゴリに表示されます。この手順を繰り返してすべてのベースオブジェクトを作成します。

次の節では、ベースオブジェクトが HM での使用に対して最適化されるようにベースオブジェクトを設定します。

手順 3: ベースオブジェクトを設定する

前の節で、2 つのベースオブジェクト (Product と Product Rel) を作成しました。この節ではそれらの設定方法について説明します。











ベースオブジェクトの設定では、カラムの数や種類、ステージングテーブルのコンテンツ、相互参照テーブルの名前 (ある場合) など、オブジェクトのプロパティに関する基準の入力を行います。履歴機能の有効化、検証ルールとメッセージトリガの設定、カスタムインデックスの作成、および外部一致テーブルの設定 (ある場合) も行うことができます。

これらのオプションを選択するかどうか、およびどのように設定するかは、選択するデータモデルおよびベースオブジェクトによって決まります。

この例では、John は次の節の説明どおりにベースオブジェクトを設定します。

注: ここでは、ベースオブジェクトの作成に関して、すべてのコンポーネントについて説明しているわけではなく、HM で使用するデータに関する重要なコンポーネントについてのみ説明しています。ここで説明されていないコンポーネントの詳細については、[第 8 章, 「スキーマの構築」](#) (ページ 79) を参照してください。










以下の図はベースオブジェクトのカラムです。

	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

このテーブルは、HM エンティティオブジェクトへの変換後の Product BO を示しています。このリストでは、[Product Type] フィールドのみが HM フィールドです。

各ベースオブジェクトにはシステムカラムとユーザー定義カラムがあります。システムカラムは自動的に作成され、必須カラムを含みます（[行 ID オブジェクト]）。これは各ベースオブジェクトテーブルのプライマリキーであり、Hub が生成した一意の値を含みます。この値はクラスコードの HM ルックアップであるため、NULL にできません。HM は、データベースに外部キー制約を作成するため、ROWID_OBJECT 値が必須であり、NULL にできません。

ユーザー定義カラムに対して、John は、製品番号、製品の種類、および製品の説明などの製品に関する情報を効果的に含む論理名を選択します。これらの同じカラムおよびカラム値が、以下の図のようにステー징テーブルに表示されます。

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 は、ステー징テーブルからのすべてのユーザー定義カラムが、ベースオブジェクトに上の図のようにカラムとして追加されていることを確認しました。[ルックアップカラム] に、HM が追加したルックアップ値が表示されます。

ステー징テーブルのいくつかのカラムは（[Status Cd]、[Product Type]、および [Product Type Cd]）、ルックアップテーブルへの参照を持ちます。これらの参照は、ステー징テーブルの作成時に設定します。ステー징テーブルで値をハードコードせずに、親テーブルの値をサーバーでルックアップする場合、ルックアップを使用できます。

ほとんどのルックアップは HM と関連がなく、データモデルの一部です。ただし [Rbo Bo Class] は HM によって追加された例外です。HM は、[Product Type] カラムにルックアップを追加します。

注: エンティティをエンティティベースオブジェクト（HM で使用するために設定されたエンティティ）に変換する際、ルックアップテーブルで [Status Cd]、[Product Type]、および [Product Type Cd] の値を確認する必要があります。

注: HM エンティティオブジェクトには開始日および終了日は不要です。開始日および終了日はユーザーが定義します。ただし、リレーションオブジェクトはこれらを使用します。開始日および終了日の名前が異なる新しいリレーションオブジェクトを作成しないでください。すでに提供されています。

手順 4: エンティティタイプの作成

エンティティタイプは階層ツールで作成します。John は 2 つのエンティティタイプを作成することにしました (ProdGroup と Product Type)。

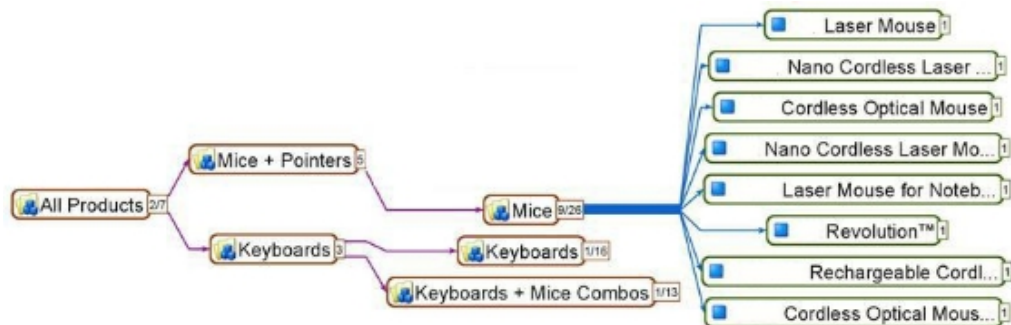
各エンティティタイプには、データ分析およびデザインから派生するコードがあります。John は Product を 1 つのタイプ、Product Group をもう 1 つのタイプとして使用することにしました。

このコードは、対応する RBO ベースオブジェクトテーブルで参照されます。この例では、コード Product は C_RBO_BO_CLASS テーブルで参照されます。BO_CLASS_CODE の値は「Product」です。

以下の表は、HM エンティティオブジェクトおよび HM リレーションオブジェクトと RBO テーブルとのリレーションを示しています。

オブジェクト	テーブル	リレーション
製品エンティティオブジェクト	BO_CLASS_CODE	多対 1 (オブジェクトからテーブル)
製品リレーションオブジェクト	C_RBO_BO_CLASS	多対 1 (オブジェクトからテーブル)
	C_RBO_HIERARCHY	1 対 1 (テーブルからオブジェクト)

John は、この節のすべての手順を実行して、パッケージなどの他の HM コンポーネントを作成し、HM でデータを表示する準備を行います。例えば、以下の図は John が階層ツールで設定し、階層マネージャで表示するリレーションを示しています。この例には、Mice デバイスに関する階層のみが表示されています。階層マネージャの使用の詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。



HM リポジトリベースオブジェクトの作成

ORS で階層ツールを使用するには、まずシステムで ORS 用のリポジトリベースオブジェクト (RBO テーブル) を作成する必要があります。RBO テーブルは、基本的にはシステムベースオブジェクトです。

これらは必須のベースオブジェクトであり、特定の列を含んでいる必要があります。

RBO テーブルには、クエリおよびパッケージ (およびその関連するクエリ) も作成されます。

注: これらの RBO テーブル、クエリ、またはパッケージは変更しないでください。

RBO を作成する手順

- 書き込みロックを取得します。
- 階層ツールを起動します。モデルワークベンチを展開し、**【階層】** をクリックします。

注: ナビゲーションパネルを右クリックして選択できるオプションは、階層ツールのメニューからも選択できます。

ORS に必要な RBO テーブルがない状態で階層ツールを起動すると、階層ツールによって RBO テーブルを作成するプロセスが順に案内されます。

以下の手順では、階層ツールで表示されるダイアログボックスでの選択内容を説明します。

1. [Hub コンソール] ダイアログで **【はい】** を選択して、ORS に HM 用にメタデータ (RBO テーブル) を作成します。
2. [RBO テーブルの作成] ダイアログで、テーブルスペース名を選択し、**【OK】** をクリックします。

デフォルトのエンティティアイコンのアップロード

階層ツールで、デフォルトのエンティティアイコンをアップロードするように求められます。

このアイコンはエンティティの作成時に役立ちます。

1. **【はい】** をクリックします。
2. 階層ツールがデフォルトのメタデータとともに表示されます。

エンティティアイコンの設定

階層ツールを使用して、後でエンティティタイプを設定するときに使用できる独自のエンティティアイコンを追加または設定することができます。

これらのエンティティアイコンは、階層マネージャ内でエンティティをグラフィック形式で表示する場合に使用されます。エンティティアイコンは、JAR ファイルまたは ZIP ファイルに格納する必要があります。

エンティティアイコンの追加

独自のアイコンをインポートするには、そのアイコンを格納する ZIP ファイルまたは JAR ファイルを作成します。各アイコンについて、16x16 の小さいアイコンと 48x48 の大きいアイコンを作成してください。

新しいエンティティアイコンを追加する手順

1. 書き込みロックを取得します。
2. 階層ツールを起動します。
3. ナビゲーションペインの任意の場所を右クリックして、**【エンティティアイコンの追加】** をクリックします。

注: ロックを取得していないと右クリックメニューは表示されません。

ファイルを参照するウィンドウが表示されます。

4. アイコンを格納している JAR ファイルまたは ZIP ファイルを参照します。
5. **【開く】** をクリックしてアイコンを追加します。

エンティティアイコンの変更

アイコンをコンソールから直接変更することはできません。

ZIP または JAR ファイルからダウンロードしてコンテンツを変更し、コンソールに再度アップロードします。

アイコングループは削除することも非アクティブ化することもできます。アイコンがすでにエンティティに関連付けられている場合、または今後アイコンをグループで使用する可能性がある場合は、削除する代わりに非アクティブ化することを検討します。

アイコンのグループは、アイコンパッケージを**非アクティブ**に設定することで非アクティブ化できます。非アクティブアイコンは UI に表示されず、エンティティタイプに割り当てできません。アイコンパッケージを再アクティブ化するには、**アクティブ**に設定します。

注: Informatica MDM Hub は、削除前にアイコンの割り当てを検証しません。現在エンティティタイプに割り当てられているアイコンを削除すると、編集内容を保存するときにエラーが発生します。

エンティティアイコンの削除

コンソールから ZIP ファイルまたは JAR ファイル内のアイコンを個別に削除することはできません。アイコンは、グループまたはパッケージとしてのみ削除できます。

エンティティアイコンのグループを削除する手順

1. 書き込みロックを取得します。
2. 階層ツールを起動します。
3. ナビゲーションペインでアイコンのコレクションを右クリックし、**[エンティティアイコンの削除]** を選択します。

エンティティ

階層マネージャでは、あらゆるオブジェクト、個人、場所、組織、またはビジネス上意味があるその他の要素がエンティティであり、データベース内で処理することができます。

例として、特定の個人の名前、特定の当座預金口座番号、特定の会社、特定の住所などがあります。

エンティティベースオブジェクト

エンティティベースオブジェクトは、階層の関係を維持するカラムを含んだベースオブジェクトです。ベースオブジェクトをエンティティベースオブジェクトとして作成できます。または、ベースオブジェクトをエンティティベースオブジェクトに変換することで、エンティティベースオブジェクトを作成することができます。

階層ツールを使用してエンティティベースオブジェクトを作成すると、階層ツールによって階層マネージャに必要なカラムが作成されます。階層ツールのオプションを使用して、既存のベースオブジェクトをエンティティベースオブジェクトに変換することもできます。

注: ベースオブジェクトにデータが含まれている場合は、ベースオブジェクトをエンティティベースオブジェクトに変換することはできません。

エンティティベースオブジェクトを追加した後は、スキーママネージャでそのエンティティベースオブジェクトの表示、編集、または削除を行います。

エンティティベースオブジェクトを作成するときには、次のいずれかの外部キーカラムの作成を選択します。
ROWID_BO_CLASS

行 ID オブジェクトを選択すると、階層ツールによって ROWID_BO_CLASS カラムが追加されます。

ROWID_BO_CLASS フィールドには行 ID の値が入力されます。

BO_CLASS_CODE または既存のカラム

ベースオブジェクトをエンティティベースオブジェクトとして作成した場合に、ベースオブジェクトのクラスコードを選択すると、階層ツールによって BO_CLASS_CODE カラムが追加されます。既存のベースオブジェクトをエンティティベースオブジェクトに変換する場合は、BO_CLASS_CODE カラムの作成を選択したり、既存のカラムを選択したりすることができます。

ベースオブジェクトのクラスコードは、レコードのエンティティタイプを定義します。このフィールドには、ユーザーが定義する値が入力されます。このカラムに入力するようにマッピングを設定します。

行 ID はシステムによって生成されて割り当てられますが、BO クラスコードはユーザーが作成できるため、覚えやすいコードにすることができます。

エンティティベースオブジェクトの例

この例の製品階層で、エンティティベースオブジェクトを 1 つ作成します。

エンティティベースオブジェクトを作成する場合、エンティティタイプの指定方法を決める必要があります。この例では、システム生成の行 ID 番号を使わず、わかりやすいコード「製品」と「製品グループ」を使ってエンティティタイプを指定します。

外部キーエンティティタイプのカラムを使用することを独自に選択するので、Product_Type カラムのあるベースオブジェクトを最初に作成します。新しいエンティティベースオブジェクトを作成する場合、カスタマイズしたカラム名を指定することはできません。このベースオブジェクトをエンティティベースオブジェクトに変換する際、Product_Type カラムをエンティティタイプ、外部キーとして選択します。この階層用に、2 つのエンティティタイプを作成します。Product_Type カラムの値により、レコードが属するエンティティタイプが決まります。

次の図に、サンプルのエンティティベースオブジェクトを変換するために行った選択を示します。

Convert to Entity Base Object

Following columns already exist in the "Product (Example)" Base Object:

- PRODUCT_TYPE

Constraints for following columns will be added to the "Product (Example)" Base Object:

- PRODUCT_TYPE

Foreign Key Column for Entity Types	
Foreign Key Column for Entity Types	Bo Class Code
Existing Base Object Column To Use	Product Type
Display name	Product Type
Physical name	PRODUCT_TYPE

OK Cancel

次に、外部キーカラムをマッピングに入力します。

エンティティベースオブジェクトの作成

エンティティベースオブジェクトであるベースオブジェクトを作成するには、Hub コンソールで階層ツールを使用します。

1. 書き込みロックを取得します。
2. [モデル] ワークベンチで、階層ツールを選択します。
3. [階層] > [新しいエンティティ/リレーションオブジェクトの作成] を選択します。
4. [新しいエンティティ/リレーションベースオブジェクトの作成] ダイアログボックスで [新しいエンティティベースオブジェクトを作成する] を選択して、[OK] をクリックします。

5. **【新しいエンティティベースオブジェクトを作成する】** ダイアログボックスで、エンティティベースオブジェクトの次のプロパティを指定します。

フィールド	説明
項目タイプ	ベースオブジェクト。読み取り専用。
表示名	Hub コンソールに表示されるこのベースオブジェクトの名前。
物理名	データベース内にあるテーブルの実際の名前。Informatica MDM Hub では、テーブルの物理名の候補として、入力した表示名に基づく名前が示されます。 行 ID はシステムによって生成されて割り当てられますが、BO クラスコードはユーザーが作成できるため、覚えやすいコードにすることができます。
データテーブルスペース	データテーブルスペースの名前。デフォルトは CMX_DATA です。
インデックステーブルスペース	インデックステーブルスペースの名前。デフォルトは CMX_INDX です。
テーブルスペースのロード	「テーブルスペースのロード」の名前。デフォルトは CMX_DATA です。
ユーザー一時テーブルスペース	ユーザー一時テーブルスペースの名前。デフォルトは CMX_USER_TEMP です。
説明	エンティティベースオブジェクトの説明。オプション。
セキュアリソース	セキュアリソースツールでのベースオブジェクトのステータスを判別します。有効な場合、ベースオブジェクトのステータスは「安全」になります。無効な場合、ベースオブジェクトのステータスは「非公開」になります。デフォルトでは有効になっています。
エンティティタイプの外部キーカラム	エンティティタイプの外部キーカラム。行 ID オブジェクトの場合、外部キーの関係は、MDM Hub によって生成された行 ID に基づきます。BO クラスオブジェクトの場合、外部キーの関係は、ユーザーが定義したコードに基づきます。デフォルトは行 ID オブジェクトです。
表示名	Hub コンソールに表示されるカラムの名前。
物理名	エンティティベースオブジェクトの外部キーカラムの名前。Hub コンソールには、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。

6. **【OK】** をクリックします。

MDM Hub により、階層マネージャが要求するカラムを持つエンティティベースオブジェクトが作成されます。

スキーマツールを使用して、エンティティベースオブジェクトにカラムを追加します。階層マネージャで使用される外部キーカラムは変更しないでください。外部キーカラムを変更すると、階層マネージャが予期せぬ動作を行い、データが失われる可能性があります。

ベースオブジェクトのエンティティベースオブジェクトへの変換

ベースオブジェクトを階層マネージャで使用するには、エンティティベースオブジェクトに変換する必要があります。ベースオブジェクトをエンティティベースオブジェクトに変換する前に、ベースオブジェクトで状態管理を有効にする必要があります。

ベースオブジェクトには、階層マネージャで必要とされるメタデータがありません。階層マネージャでベースオブジェクトを使用するには、変換プロセスを使ってこのメタデータを追加する必要があります。MDM Hub と階層マネージャの両方でエンティティベースオブジェクトを使用できます。

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションペインの任意の場所を右クリックし、**[BO をエンティティ/リレーションオブジェクトに変換]** を選択します。
注: 右クリックメニューで表示されるオプションは、階層ツールのメニューでも選択できます。
3. **[既存のベースオブジェクトの変更]** ダイアログで、**[エンティティベースオブジェクトに変換する]** を選択し、**[OK]** をクリックします。
注: **[ベースオブジェクトの変更]** フィールドに選択肢が表示されない場合は、利用できる非階層ベースオブジェクトがありません。この場合はスキーマツールで作成する必要があります。
4. **[OK]** をクリックします。
ベースオブジェクトにすでに階層マネージャのメタデータがある場合、階層マネージャのメタデータが存在することを示すメッセージが表示されます。
5. **[エンティティタイプの外部キーカラム]** フィールドで、追加するカラムを選択します。行 ID オブジェクトまたは BO クラスコードを選択できます。
BO クラスコードカラムを選択すると、MDM Hub で生成される行 ID ではなく、定義済みコードに基づいて外部キーリレーションを定義できるため、複雑さを軽減することができます。
6. **[使用する既存の BO カラム]** で、既存のカラムを選択するか、**[新しいカラムを作成する]** オプションを選択します。
BO カラムが存在しない場合は、**[新しいカラムを作成する]** オプションしか選択できません。
7. **[表示名]** フィールドと **[物理名]** フィールドでカラムの表示名と物理名を作成し、**[OK]** をクリックします。

ベースオブジェクトに階層マネージャで必要なカラムが追加されます。その他のカラムを追加する場合は、スキーママネージャを使用してください。

重要: スキーママネージャツールを使用してベースオブジェクトを変更するときに、階層ツールで追加されたカラムは変更しないでください。それらのカラムを変更すると、予期しない動作が発生し、データが失われる可能性があります。

エンティティタイプ

エンティティはエンティティタイプで分類されます。エンティティタイプにより、エンティティベースオブジェクト内でエンティティ进行分类できます。

エンティティタイプは特定のエンティティに属しています。エンティティベースオブジェクトには複数のエンティティタイプを指定できますが、ベースオブジェクト内の個々のエンティティは1つのエンティティタイプだけに属します。たとえば、製品エンティティベースオブジェクトには、製品エンティティタイプまたは製品グループエンティティタイプを指定できます。Product ベースオブジェクトのエンティティには、製品エンティティタイプまたは製品グループエンティティタイプのいずれかを指定できます。

例として、医師、当座預金口座、銀行などがあります。エンティティベースオブジェクトには、エンティティタイプテーブル（Rbo BO Class）への外部キーが必要です。外部キーは、行 ID または定義済みのコード値として定義できます。

適切に定義されたエンティティタイプには、以下の特性があります。

- エンティティタイプを使用して、実際の組織のエンティティを反映するようにレコードを編成します。現実世界でのエンティティの性質を反映して効果的にデータを分類します。
- 集合的に扱われることで、エンティティのセット全体をカバーします。つまり、どのエンティティにもエンティティタイプが1つだけ設定されています。
- 十分な粒度があるため、各エンティティタイプが持つことのできるリレーションのタイプを容易に定義できます。例えば、エンティティタイプ「医師」が持つことのできるリレーションには、医療グループとの「所属先」リレーション、病院との「スタッフ」（または「入院特権を持つ非スタッフ」）リレーションなどがあります。
- これよりも全般的なエンティティタイプ(例えば、看護師、ナースプラクティショナー、医師などが含まれる「介添人」など)は、粒度が不十分です。このような全般的なエンティティタイプが持つリレーションのタイプは、エンティティタイプ以外の要素に左右されます。このため、より粒度の高いエンティティタイプを定義する必要があります。

階層タイプを設定するときに、階層マネージャツールでそのタイプのエンティティの外観用のアイコンと色を選択します。







エンティティタイプを作成する際、次のプロパティを設定できます。

フィールド	説明
コード	エンティティタイプの一意のコード名。エンティティベースオブジェクトから外部キーとして使用できます。各エンティティタイプには独自のコード値を指定する必要があります。
表示名	階層ツールに表示される、このエンティティタイプの名前。
説明	エンティティタイプの説明。 オプション。
色	Hub コンソールの階層マネージャツールと Informatica Data Director 階層ビューに表示されるこのエンティティタイプに関連付けられたエンティティの色。
小さいアイコン	Hub コンソールの階層マネージャツールと Informatica Data Director 階層ビューに表示されるこのエンティティタイプに関連付けられたエンティティの小さいアイコン。小さいアイコンは、階層内で表示されるエンティティの数が多い場合に表示されます。
大きいアイコン	Hub コンソールの階層マネージャツールと Informatica Data Director 階層ビューに表示されるこのエンティティタイプに関連付けられたエンティティの大きいアイコン。







エンティティタイプの例

この例のエンティティタイプは、製品および製品グループです。

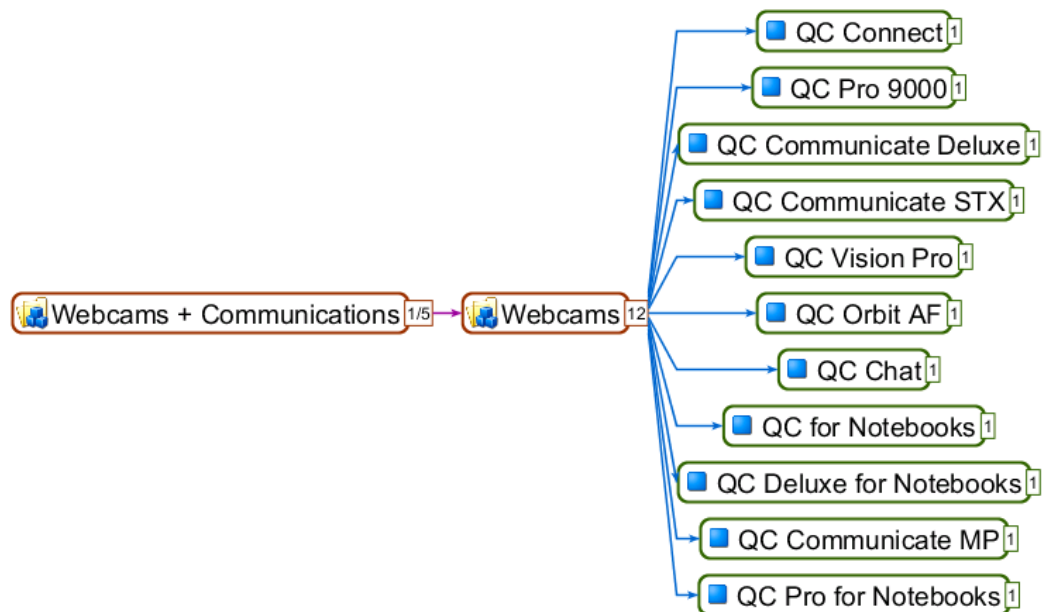
次の図に、製品エンティティタイプのプロパティを示します。

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... 

次の図に、製品グループエンティティタイプのプロパティを示します。

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 

次の図に、製品と製品グループのエンティティタイプのある階層を示します。



エンティティタイプの作成

エンティティタイプを作成するには、**階層ツール**でエンティティオブジェクトを選択してから、エンティティタイプを追加します。

1. 書き込みロックを取得します。
2. **【モデル】** ワークベンチで、**階層ツール**を選択します。
3. 階層ツールのナビゲーションパネルの**【エンティティオブジェクト】** ノード下で、エンティティタイプを作成するエンティティベースオブジェクトを選択します。
最初にエンティティベースオブジェクトを選択しないと、エンティティタイプを追加できません。
4. **【階層】** > **【エンティティタイプの追加】** を選択します。
階層ツールに、**【新しいエンティティタイプ】** というエンティティタイプが表示されます。
5. エンティティタイプのプロパティを編集して、**【保存】** をクリックします。

エンティティタイプの編集

エンティティタイプを編集する手順

1. ナビゲーションツリーの階層ツールで、編集するエンティティタイプをクリックします。
2. 編集するフィールドごとに、**【編集】** ボタンをクリックし、必要な変更を加えます。
3. 変更が完了したら、**【保存】** をクリックして変更を保存します。

注: エンティティオブジェクトでコードカラムが使用されている場合に、そのエンティティタイプのレコードがすでにあるときは、エンティティタイプコードを変更しないでください。

エンティティタイプの削除

どのリレーションタイプでも使用されていないエンティティタイプは削除できます。

1つ以上のリレーションタイプで使用されているエンティティタイプを削除しようとする、エラーが発生します。

エンティティタイプを削除する手順

1. 書き込みロックを取得します。
2. 階層ツールのナビゲーションツリーで、削除するエンティティタイプを右クリックし、**[エンティティタイプの削除]** を選択します。

エンティティタイプがどのリレーションタイプでも使用されていなければ、階層ツールから削除を確認するように求められます。

3. **[はい]** を選択します。

選択したエンティティタイプがリストから削除されます。

注: エンティティレコードですでに使用されているエンティティタイプは削除しないでください。エンティティオブジェクトが行 ID カラムではなくコードカラムを使用している場合に、そのエンティティオブジェクトに削除しようとするエンティティタイプのレコードが存在していると、エラーが返されます。

エンティティの表示オプション

エンティティの色やアイコンに加えて、フォントのサイズや最大幅を設定することもできます。

色とアイコンはエンティティタイプごとに指定できますが、フォントのサイズと幅は全タイプのエンティティに適用されます。

HM でフォントサイズを変更するには、[HM のフォントサイズおよびエンティティボックスサイズ] を使用します。デフォルトのエンティティフォントサイズ（38 ポイント）および最大エンティティボックス幅（600 ピクセル）は、cmxserver.properties ファイルを設定することで上書きできます。使用する設定は以下のとおりです。

```
sip.hm.entity.font.size=fontSize  
sip.hm.entity.max.width=maxWidth
```

fontSize には 6 から 100 までの値、maxWidth には 20 から 5000 までの値を指定できます。指定した値が範囲を外れている場合は、最小値または最大値が使用されます。指定した値が数値でない場合は、デフォルト値が使用されます。

エンティティベースオブジェクトをベースオブジェクトに戻す

ベースオブジェクトを誤ってエンティティオブジェクトに変換した場合、または階層マネージャでエンティティオブジェクトを操作する必要がある場合は、エンティティオブジェクトをベースオブジェクトに戻すことができます。

ベースオブジェクトに戻す場合、オブジェクトから HM メタデータを削除します。

エンティティベースオブジェクトをベースオブジェクトに戻す手順

1. 階層ツールで、書き込みロックを取得します。

2. エンティティベースオブジェクトで右クリックして、**[エンティティ/リレーションオブジェクトを BO に戻す]** を選択します。
3. 関連するリレーションオブジェクトを戻すことを求めるメッセージが表示された場合は、**[OK]** をクリックします。
注: エンティティオブジェクトを戻す場合、対応するリレーションオブジェクトも戻す必要があります。
4. **[エンティティ/リレーションオブジェクトを戻す]** ダイアログボックスで **[OK]** をクリックします。
エンティティが戻ると、ダイアログが表示されます。

階層タイプ

リレーションタイプを設定する場合は、1 つ以上の階層タイプに関連付ける必要があります。

これらのリレーションタイプはランク付けされず、相互に関連するとも限りません。これらは、単に分類や識別がしやすいようにグループ分けされたリレーションタイプです。同じリレーションタイプを複数の階層に関連付けることができます。階層タイプは、階層の論理的な分類です。

階層タイプを作成する際、次のプロパティを設定できます。

フィールド	説明
コード	階層の一意のコード名。階層マネージャのリレーションベースオブジェクトの外部キーとして使用できます。階層タイプの作成後にコードを変更することはできません。
表示名	Hub コンソールに表示されるこの階層の名前。
説明	この階層の説明。

初めて作成した階層タイプには、参照がありません。リレーションタイプに、作成した階層タイプを関連付けると、階層タイプの参照にエンティティタイプ、リレーションタイプ、およびプロファイルが階層ツールによって入力されます。

階層の追加

新しい階層を追加する手順

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションペインでエンティティオブジェクトを右クリックし、**[階層の追加]** を選択します。
ナビゲーションツリー内の **[階層]** ノードの下に新しい階層が表示されます（名前は **[新しい階層]** になります）。プロパティペインには、デフォルトのプロパティが表示されます。
3. この新しい階層の以下のプロパティを指定します。
4. **[保存]** をクリックして新しい階層を保存します。

階層の削除

階層を使用するリレーションレコードがすでにある場合は、その階層を削除しないでください。

リレーションオブジェクトが行 ID カラムではなく階層コードカラムを使用している場合に、削除しようとする階層のレコードがそのリレーションオブジェクトに存在する場合は、エラーが返されます。

階層を削除する手順

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションツリーで、削除する階層を右クリックし、**【階層の削除】**を選択します。
削除を確認するように求められます。
3. **【はい】**を選択します。
階層ツールによって、選択した階層がリストから削除されます。

注: リレーションタイプが関連付けられている階層は削除可能です。関連付けられたリレーションタイプのリストとともに警告が表示されます。階層を削除すると、その階層へのすべての参照が自動的に削除されます。

リレーションオブジェクト

2つの特定のエンティティ間のリレーションオブジェクトを表します。

リレーションオブジェクトは、次のいずれかのオブジェクトになります。

リレーションベースオブジェクト

2つのエンティティベースオブジェクト間の関係を維持します。リレーションベースオブジェクトを使って、多対多の関係を確立します。

外部キーベースオブジェクト

外部キーベースオブジェクトは、外部キーカラムのあるエンティティベースオブジェクトです。外部キーベースオブジェクトを使って、1対1または1対多の関係を確立します。たとえば、多くの製品は単一の製品グループに関連していることがあります。エンティティベースオブジェクトには複数の外部キーカラムを含めることができます。各外部キーカラムではリレーションが維持されています。

階層のリレーションは、リレーションタイプ、階層タイプ、リレーションの属性、およびリレーションがアクティブになる日付を指定することによって定義します。

スキーマツールのリレーションノードには、階層リレーションとデータモデルリレーションが表示されます。

リレーションベースオブジェクト

リレーションベースオブジェクトは、階層リレーションを格納するベースオブジェクトです。外部キーリレーションベースオブジェクトは、別のエンティティベースオブジェクトに対する外部キーを含むエンティティベースオブジェクトです。リレーションベースオブジェクトは、2つのエンティティベースオブジェクトを関連付けるテーブルです。外部キー関係タイプは、1つの階層にのみ関連付けることができます。

2つの特定エンティティ間にはリレーションが1つのみ存在できます。この2つのエンティティ間にリレーションをさらに追加しようとすると、新しく追加されるのではなく、既存のリレーションが更新されます。

スキーママネージャを使用してベースオブジェクトを変更するときに、階層マネージャで追加されたカラムは変更しないでください。それらのカラムを変更すると、予期しない動作が発生し、データが失われる可能性があります。

次の表では、リレーションベースオブジェクトを作成するときに設定できるプロパティについて説明します。

プロパティ	説明
項目タイプ	読み取り専用。すでに指定されています。
表示名	Hub コンソールに表示される際のこのベースオブジェクトの名前。
物理名	データベース内にあるテーブルの実際の名前。Informatica MDM Hub では、テーブルの物理名の候補として、入力した表示名に基づく名前が示されます。
データテーブルスペース	データテーブルスペースの名前。詳細については、『 <i>Multidomain MDM のインストールガイド</i> 』を参照してください。
インデックステーブルスペース	インデックステーブルスペースの名前。詳細については、『 <i>Multidomain MDM のインストールガイド</i> 』を参照してください。
説明	このベースオブジェクトの説明。
エンティティベースオブジェクト 1	このリレーションベースオブジェクトでリンクするエンティティベースオブジェクト。
表示名	エンティティベースオブジェクト 1 に対する FK のカラム名。
物理名	データベース内のカラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。
エンティティベースオブジェクト 2	このリレーションベースオブジェクトでリンクするエンティティベースオブジェクト。
表示名	エンティティベースオブジェクト 2 に対する FK のカラム名。
物理名	データベース内のカラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。
階層の FK カラム	階層の外部キーとして使用されるカラム。行 ID またはコードのいずれかを選択できます。 BO クラスコードカラムを選択すると、Informatica MDM Hub で生成される行 ID ではなく、定義済みコードに基づいて外部キー関係を定義できるため、複雑さを軽減することができます。
階層の FK の表示名	Hub コンソールに表示される際のこの FK カラムの名前。
階層の FK の物理名	テーブル内の階層の外部キーカラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。
リレーションタイプの FK カラム	リレーションの外部キーとして使用されるカラム。行 ID またはコードのいずれかを選択できます。
リレーションタイプの表示名	リレーションタイプのコードまたは行 ID を格納するために使用するカラム名。
リレーションタイプの物理名	テーブル内のリレーションタイプの FK カラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。

リレーションベースオブジェクトの作成

リレーションベースオブジェクトを作成するには、階層ツールを使用して、外部キーの制約を設定します。

1. 書き込みロックを取得します。
2. **【モデル】** ワークベンチで、**階層ツール**を選択します。
3. **【階層】 > 【新しいエンティティ/リレーションオブジェクトの作成】** を選択します。
4. **【新しいエンティティ/リレーションベースオブジェクトの作成】** ダイアログボックスで**【新しいリレーションベースオブジェクトを作成する】**を選択します。 **【OK】** をクリックします。
5. **【新しいリレーションベースオブジェクトの作成】** ダイアログボックスでリレーションベースオブジェクトのプロパティを指定します。 **【OK】** をクリックします。

ベースオブジェクトのリレーションベースオブジェクトへの変換

リレーションベースオブジェクトは、2つのエンティティベースオブジェクトに関する情報を格納するテーブルです。

ベースオブジェクトには、階層マネージャでリレーション情報のために必要とされるメタデータがありません。ベースオブジェクトを使用するには、最初にベースオブジェクトをエンティティオブジェクトに変換して、リレーションオブジェクトに関連付けるエンティティオブジェクトを選択します。

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションペインで右クリックし、**【BO をエンティティ/リレーションオブジェクトに変換】**を選択します。
3. **【OK】** をクリックします。
[リレーションベースオブジェクトへの変換] 画面が表示されます。
4. このベースオブジェクトの以下のプロパティを指定します。

フィールド	説明
エンティティベースオブジェクト 1	このリレーションベースオブジェクトでリンクするエンティティベースオブジェクト。
表示名	エンティティベースオブジェクト 1 に対する FK のカラムの名前。
物理名	データベース内のカラムの実際の名前。 Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。
エンティティベースオブジェクト 2	このリレーションベースオブジェクトでリンクするエンティティベースオブジェクト。
表示名	エンティティベースオブジェクト 2 に対する FK のカラムの名前。
物理名	データベース内のカラムの実際の名前。 Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。

フィールド	説明
階層の FK カラム	階層の外部キーとして使用されるカラム。行 ID またはコードのいずれかを選択できます。 BO クラスコードカラムを選択すると、Informatica MDM Hub で生成される行 ID ではなく、定義済みコードに基づいて外部キーのリレーションを定義できるため、複雑さを軽減することができます。
使用する既存の BO カラム	使用する既存の BO 内の実際のカラム。
階層の FK の表示名	Hub コンソールに表示されるこの FK カラムの名前。
階層の FK の物理名	テーブル内の階層の外部キーカラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。
リレーションタイプの FK カラム	リレーションの外部キーとして使用されるカラム。行 ID またはコードのいずれかを選択できます。
使用する既存の BO カラム	使用する既存の BO 内の実際のカラム。
リレーションタイプの FK の表示名	リレーションタイプのコードまたは行 ID を格納するために使用する FK カラムの名前。
リレーションタイプの FK の物理名	テーブル内のリレーションタイプの FK カラムの実際の名前。Informatica MDM Hub では、カラムの物理名の候補として、入力した表示名に基づく名前が示されます。

5. **[OK]** をクリックします。

注: スキーママネージャツールを使用してベースオブジェクトを変更するときに、階層マネージャで追加されたカラムは変更しないでください。これらのカラムを変更すると、予期しない動作が発生し、データが失われる可能性があります。

リレーションベースオブジェクトをベースオブジェクトに戻す

リレーションベースオブジェクトをベースオブジェクトに戻し、リレーションオブジェクトから階層マネージャのメタデータを削除します。リレーションオブジェクトは、ベースオブジェクトとして残されますが、階層マネージャではこのベースオブジェクトは表示されません。

戻すリレーションタイプカラムがルックアップ用のステージングテーブルにある場合、リレーションベースオブジェクトを戻す前に、このステージングテーブルカラムを空にする必要があります。

階層マネージャリレーションをアップグレードする場合は、リレーションベースオブジェクトを戻す前に、階層マネージャのリレーションをプロビジョニングツールにコピーしてください。

1. 階層ツールで、書き込みロックを取得します。
2. リレーションベースオブジェクトで右クリックして、**[エンティティ/リレーションオブジェクトを BO に戻す]** を選択します。
3. **[エンティティ/リレーションオブジェクトを戻す]** ダイアログボックスで **[OK]** をクリックします。
エンティティが戻ると、ダイアログボックスが表示されます。

外部キーリレーションベースオブジェクト

外部キーリレーションベースオブジェクトは、別のエンティティベースオブジェクトに対する外部キーを含むエンティティベースオブジェクトです。

外部キーリレーションベースオブジェクトを変換するには、エンティティベースオブジェクトを用意して、外部キーカラムを追加する必要があります。

外部キーリレーションベースオブジェクトを作成する際、外部キーカラムを追加するエンティティベースオブジェクトを選択します。

フィールド	説明
FK 制約エンティティ BO 1	リストから FK エンティティベースオブジェクトを選択します。
使用する既存の BO カラム	外部キーに使用する既存のベースオブジェクトカラムの名前。新しいカラムを作成することも可能です。
FK カラム表示名 1	Hub コンソールに表示される際の FK カラムの名前。
FK カラム物理名 1	データベース内の FK カラムの実際の名前。Hub コンソールには、テーブルの物理名の候補として、入力した表示名に基づく名前が表示されます。
FK カラムが表す対象	リレーションで FK カラムが表す対象に応じて、[エンティティ 1] または [エンティティ 2] を選択します。

外部キーリレーションを使用するには、関連付けるエンティティベースオブジェクトを作成します。エンティティ間のリレーションが 1 対多の場合、外部キーリレーションを使用します。

外部キーリレーションを作成する際、関連付ける各エンティティの外部キーカラムを追加する必要があります。

外部キーリレーションベースオブジェクトの作成

外部キーリレーションベースオブジェクトを作成するには、**階層ツール**を使用します。

- 書き込みロックを取得します。
- 【モデル】ワークベンチで、**階層ツール**を選択します。
- 【階層】 > 【外部キーリレーションオブジェクトの作成】を選択します。
- 【既存のベースオブジェクトの変更】ダイアログボックスで、外部キーを追加するエンティティベースオブジェクトを選択して、外部キーカラムの数を指定します。【OK】をクリックします。
- 外部キーカラムのプロパティを選択してから、【OK】をクリックします。

リレーションタイプ

リレーションタイプは、リレーションのクラスを記述し、このタイプのリレーションに含めることができるエンティティのタイプ、リレーションの方向（該当する場合）、および Hub コンソールでのリレーションの表示方法を定義します。リレーションベースオブジェクトまたは外部キーリレーション用のリレーションタイプを作成できます。

注: リレーションタイプは物理構造であり、構成が重くなります。これに対して、階層タイプは論理構造であり、一般的に構成が軽くなります。したがって、リレーションタイプを多くするよりも階層タイプを多くする方が、通常は簡単です。MDM Hub で階層タイプやリレーションタイプを定義する前に、環境内のデータ管理要件や階層管理要件を必ず確認します。

適切に定義された一連の階層のリレーションタイプには、次の特性があります。



- エンティティタイプ間の実際の関係を反映します。
- 各リレーションに対して、複数のリレーションタイプがサポートされています。

フィールド	説明
コード	リレーションタイプの一意のコード名。階層のリレーションベースオブジェクトの外部キーとして使用できます。
表示名	Hub コンソールに表示されるこのリレーションタイプの名前。一意のわかりやすい名前を指定します。
説明	このリレーションタイプの説明。
色	このリレーションタイプに関連付けられたリレーションが階層マネージャコンソールおよび Informatica Data Director の Hub コンソールに表示される際の色。
エンティティタイプ 1	この新しいリレーションタイプに関連付ける 1 つ目のエンティティタイプ。このタイプのすべてのエンティティに、このリレーションタイプのリレーションを含めることができます。
エンティティタイプ 2	この新しいリレーションタイプに関連付ける 2 つ目のエンティティタイプ。このタイプのすべてのエンティティに、このリレーションタイプのリレーションを含めることができます。
方向	<p>方向が設定された階層を新しいリレーションタイプでできるようにする場合に方向を選択します。選択できる方向は次のとおりです。</p> <ul style="list-style-type: none"> - エンティティ 1 からエンティティ 2 - エンティティ 2 からエンティティ 1 - 方向未設定 - 双方向 - 不明 <p>方向が設定された階層の例としては、報告系統 (reports to) のリレーションが一般社員から管理者、管理者からその上、最終的に組織の代表までの方向で示された組織図などがあります。</p>
FK リレーションの開始日	外部キーリレーションの開始日。
FK リレーションの終了日	外部キーリレーションの終了日。
階層	この新しいリレーションタイプに関連付ける階層の横にあるチェックボックスをチェックします。選択したすべての階層に、このリレーションタイプのリレーションを含めることができます。

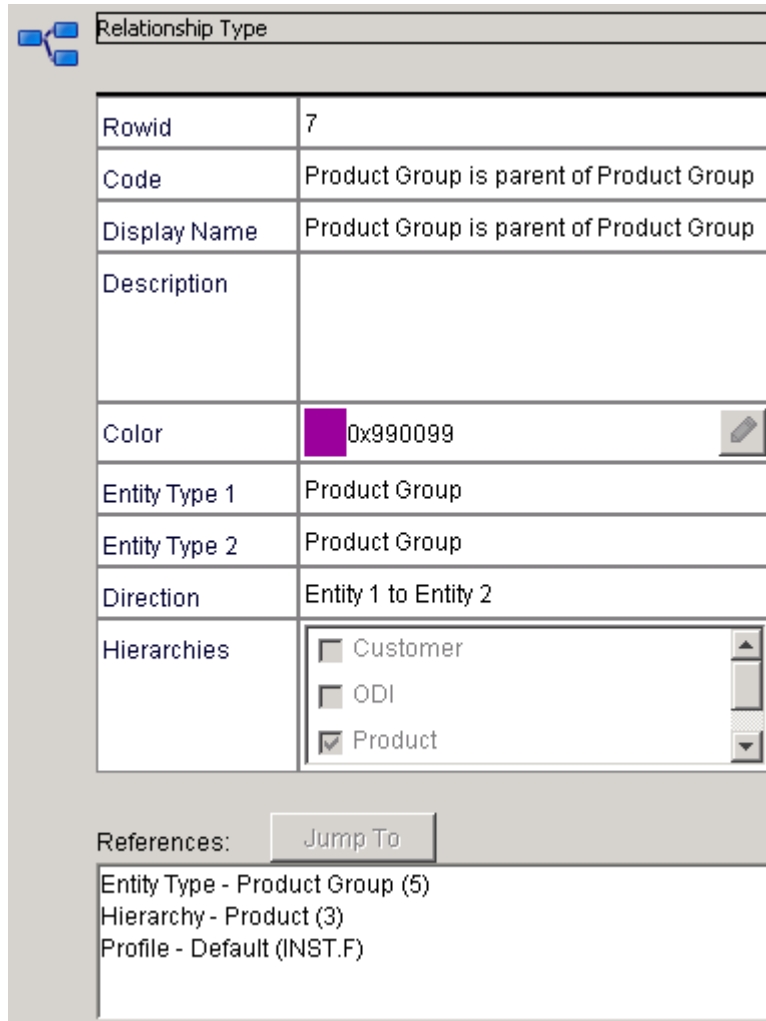
リレーションタイプの例



この例では、2つのタイプのリレーションを使用します。1つ目のリレーションでは、製品グループエンティティタイプが、製品エンティティタイプの親です。2つ目のリレーションでは、製品グループエンティティタイプが、製品グループエンティティタイプの親です。

次の図に、製品グループのプロパティが製品リレーションオブジェクトの製品リレーションタイプの親であることを示します。

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	<div><input type="checkbox"/> Customer</div> <div><input type="checkbox"/> ODI</div> <div><input checked="" type="checkbox"/> Product</div>
References: Jump To	
Entity Type - Product (4) Entity Type - Product Group (5) Hierarchy - Product (3) Profile - Default (INST.F)	

次の図に、製品グループのプロパティが製品リレーションオブジェクトの製品グループリレーションタイプの親であることを示します。



Relationship Type	
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)

リレーションタイプの作成

リレーションタイプを作成するには、**【階層】** ツールを使用して、リレーションタイプをリレーションベースオブジェクトに追加します。

1. 書き込みロックを取得します。
2. **【モデル】** ワークベンチで、**階層** ツールを選択します。
3. **階層** ツールのナビゲーションペインで、リレーションタイプを作成するリレーションベースオブジェクトを選択します。

最初に関係ベースオブジェクトを選択しないと、リレーションタイプを追加できません。

4. **【階層】** > **【リレーションタイプの追加】** を選択します。
階層ツールに、**【新しいリレーションタイプ】** というリレーションタイプが表示されます。
5. リレーションタイプのプロパティを編集して、**【保存】** をクリックします。

リレーションタイプの編集

リレーションタイプを編集する手順

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションツリーで、編集するリレーションタイプをクリックします。
3. 編集するフィールドごとに、**【編集】** をクリックし、必要な変更を加えます。
4. 変更が完了したら、**【保存】** をクリックして変更を保存します。

注: リレーションオブジェクトでコードカラムが使用されている場合に、そのリレーションタイプのレコードがすでにあるときは、リレーションタイプコードを変更しないでください。

この警告は、FK リレーションタイプには当てはまりません。

リレーションタイプの削除

注: リレーションタイプを使用するリレーションレコードがすでにある場合は、そのリレーションタイプを削除しないでください。リレーションオブジェクトが行 ID カラムではなくリレーションタイプコードカラムを使用している場合に、削除しようとするリレーションタイプのレコードがリレーションオブジェクトに存在する場合は、エラーが返されます。

上記の警告は、FK リレーションタイプには該当しません。階層に関連付けられたリレーションタイプは削除できます。確認ダイアログに、削除されるリレーションタイプに関連付けられている階層が表示されます。

リレーションタイプを削除する手順

1. 階層ツールで、書き込みロックを取得します。
2. ナビゲーションツリーで、削除するリレーションタイプを右クリックし、**【リレーションタイプの削除】** を選択します。
削除を確認するように求められます。
3. **【はい】** を選択します。
選択したリレーションタイプがリストから削除されます。

パッケージ

ここでは、階層ツールを使用してパッケージをスキーマに追加する方法について説明します。エンティティベースオブジェクト、リレーションベースオブジェクト、および外部キーリレーションベースオブジェクトのパッケージを作成することができます。パッケージでレコードの挿入や変更を行う場合は、必ず**【配置】** オプションを有効にします。

パッケージとは、Informatica MDM Hub の 1 つ以上の基本テーブルの公開されたビューです。パッケージは、それらのテーブル内のカラムのサブセットを、そのテーブルに結合された他のテーブルと一緒に示します。パッケージはクエリに基づきます。基になるクエリで、テーブルまたは別のパッケージからレコードのサブセットを選択することができます。パッケージは、基になるデータのユーザーのビューを設定するために使用されます。

最初にパッケージを作成してから、そのパッケージをエンティティタイプまたはリレーションタイプに関連付ける必要があります。外部キー関係オブジェクトの表示パッケージの場合は、パッケージに REL_START_DATE および REL_END_DATE カラムを含める必要があります。

注: ロードジョブを実行する前に、パッケージを設定して関連プロファイルを検証する必要があります。

階層マネージャデータの設定

ユーザーが階層マネージャツールでアクセスできるデータを設定できます。

階層パッケージを設定する際、ユーザーインターフェースにどのエンティティフィールドをどの順序で表示するかを設定します。パッケージのフィールドごとに、0 から始まる数を選択します。フィールドに値 0 を割り当てると、そのフィールドはユーザーインターフェースで、フィールドのリストの一番上に表示されます。

次の階層マネージャのユーザーインターフェース要素に表示するフィールドとその順序を設定できます。

ラベル

階層マネージャツールに表示するエンティティフィールドを指定するには、数字を使用します。

ツールチップ

最小値を割り当てるカラムは、階層マネージャツール内のエンティティにマウスを合わせたときにツールチップ内に表示されるフィールドです。

一般

各種タイプのエンティティとリレーションが同じリスト内に表示されるときに表示するフィールド。選択するフィールドは、すべての階層マネージャパッケージ内にある必要があります。

検索

レコードの検索に使用できるフィールド。

リスト

検索結果リストに表示されるフィールド。

詳細

選択したエンティティの詳細の表示を選択したときに表示されるフィールド。

Put

選択したエンティティを編集するときに値を変更できるフィールド。

追加

階層マネージャでエンティティを追加するときに表示されるフィールド。

エンティティ、リレーション、および FK リレーションオブジェクトのパッケージの作成

HM パッケージを作成する手順

1. 階層ツールで、ナビゲーションペイン内を右クリックし、**[新しいパッケージの作成]** を選択します。
2. 書き込みロックを取得します。

階層ツールによって新しいパッケージの作成ウィザードが開始され、最初のダイアログボックスが表示されます。

3. この新しいパッケージの以下の情報を指定します。

フィールド	説明
パッケージのタイプ	以下のいずれかのタイプです。 エンティティオブジェクト リレーションオブジェクト FK リレーションオブジェクト
クエリグループ	既存のクエリグループを選択するか、新しいクエリグループを作成します。 Informatica MDM Hub では、クエリの論理的なグループをクエリグループと呼びます。クエリを編成するクエリグループを選択します。
クエリグループ名	新しいクエリグループの名前。新しいグループの作成を選択した場合にのみ必要です。
説明	作成する新しいクエリグループの説明（オプション）。

4. [次へ] をクリックします。

新しいパッケージの作成ウィザードの次のダイアログボックスが表示されます。

5. この新しいパッケージの以下の情報を指定します。

フィールド	説明
クエリ名	クエリの名前。 Informatica MDM Hub のクエリは、Hub Store からデータを取得する要求です。
説明	必要に応じて入力する説明。
プライマリテーブルの選択	このクエリのプライマリテーブル。

6. [次へ] をクリックします。

新しいパッケージの作成ウィザードの次のダイアログボックスが表示されます。

7. この新しいパッケージの以下の情報を指定します。

フィールド	説明
表示名	このパッケージの表示名。パッケージツールでこのパッケージが表示されるときに使用されます。
物理名	このパッケージの物理名。入力した表示名に基づいた物理名が Hub コンソールに表示されます。
説明	必要に応じて入力する説明。

フィールド	説明
PUT を有効にする	レコードの挿入または変更を可能にする場合に選択します（オプション）。 選択しなかった場合は、パッケージが読み取り専用になります。外部キーリレーションオブジェクトパッケージを作成する場合は、この手順の手順 9 で追加の操作が必要となります。 注: 外部キーリレーションごとに、PUT および非 PUT パッケージの両方を持つ必要があります。同じ外部キーリレーションオブジェクト用に作成する PUT パッケージと非 PUT パッケージの両方に同じカラムを設ける必要があります。
セキュアリソース	セキュアリソースを作成する場合に選択します。（オプション）。

8. **【次へ】** をクリックします。

新しいパッケージの作成ウィザードの最後のダイアログボックスが表示されます。表示されるダイアログボックスは、作成するパッケージのタイプによって異なります。

エンティティまたはリレーションにいずれかのパッケージを作成するか、または FK リレーションに PUT パッケージを作成する場合は、以下のようなダイアログボックスが表示されます。必須カラム（グレーで表示）は自動的に選択されます。この選択は解除できません。

パッケージに関係のないカラムの選択を解除します。

注: 外部キーリレーションごとに、PUT および非 PUT パッケージの両方を持つ必要があります。同じ外部キーリレーションオブジェクト用に作成する PUT パッケージと非 PUT パッケージの両方に同じカラムを設ける必要があります。

外部キーリレーションに非 PUT 対応パッケージを作成する場合（手順 7 で PUT チェックボックスをオンにしなかった場合）は、以下のダイアログボックスが表示されます。

9. 外部キーリレーションに非 PUT 対応パッケージを作成する場合は、この新しいパッケージに以下の情報を指定します。

フィールド	説明
階層	このパッケージに関連付けられている階層。
リレーションタイプ	このパッケージに関連付けられているリレーションタイプ。

注: 外部キーリレーションごとに、PUT および非 PUT パッケージの両方を持つ必要があります。同じ外部キーリレーションオブジェクト用に作成する PUT パッケージと非 PUT パッケージの両方に同じカラムを設ける必要があります。

10. この新しいパッケージのカラムを設定します。

11. **【完了】** をクリックしてパッケージを作成します。

この新たに作成されたパッケージを表示、編集、または削除するには、パッケージツールを使用します。

階層マネージャで必要となるカラムは削除しないでください。これらのカラムは、ユーザーが階層ツールでパッケージを作成すると自動的に選択されて、グレーで表示されます。

エンティティタイプまたはリレーションタイプへのパッケージの割り当て

プロファイルを作成し、プロファイル内のエンティティ/リレーションタイプそれぞれのパッケージを作成したら、パッケージをエンティティタイプまたはリレーションタイプに割り当てる必要があります。これにより、

階層マネージャでエンティティが表示されるときにどのフィールドが表示されるかが決まります。リレーションタイプとエンティティタイプに同じパッケージを割り当てることもできます。

エンティティ/リレーションタイプにパッケージを割り当てる手順

1. 書き込みロックを取得します。
2. 階層ツールで、エンティティ/リレーションタイプを選択します。

階層マネージャに、そのタイプのパッケージのプロパティが表示されます。プロパティがない場合は、同じプロパティペインがフィールドが空の状態が表示されます。表示パッケージと Put パッケージを選択すると、階層マネージャのパッケージのカラム情報が下のパネルに表示されます。

セル内の数値は、属性が表示される順序を示します。

3. エンティティタイプまたはリレーションタイプのパッケージを設定します。

フィールド	説明
ラベル	階層マネージャのグラフィカルコンソールに表示されるエンティティ/リレーションのラベルの表示に使用するカラム。これらのカラムは、階層マネージャコンソールおよび Informatica Data Director でラベルパターンを作成するために使用されます。 ラベルを編集するには、ラベルの右側にあるラベルの値をクリックします。[パターンの編集] ダイアログで、新しいラベルを入力するか、パターンで使用するカラムをダブルクリックします。
ツールチップ	エンティティ/リレーションをスクロールしているときに表示される説明またはコメントの表示に使用するカラム。階層マネージャコンソールおよび Informatica Data Director でツールチップパターンを作成するために使用されます。 ツールチップを編集するには、ツールチップパターンのラベルの右側にあるツールチップパターンの値をクリックします。[パターンの編集] ダイアログで、新しいツールチップパターンを入力するか、パターンで使用するカラムをダブルクリックします。
共通	タイプが異なるエンティティ/リレーションを同じリストに表示する場合に使用するカラム。選択したカラムは、プロファイル内のすべてのエンティティ/リレーションタイプに関連付けられたパッケージに含まれている必要があります。
検索	検索ツールで利用できるカラム。
リスト	検索結果に表示するカラム。
詳細	画面の下部に表示されるエンティティ/リレーションの詳細ビューで使用するカラム。
Put	レコードを編集する際に表示されるカラム。
追加	新しいレコードを作成する際に表示されるカラム。

4. 変更が完了したら、[保存] をクリックして変更を保存します。

プロファイルの概要

階層プロファイルは、階層オブジェクトへのユーザーアクセスを定義します。

プロファイルによって、ユーザーが階層マネージャで表示、編集、または追加できるフィールドやレコードが決まります。例えば、2つのプロファイルを用意し、一方ではすべてのエンティティおよびリレーションに対する完全な読み取り/書き込みアクセスを許可し、もう一方は読み取り専用にする（追加や編集の操作を許可しない）ことも可能です。定義したプロファイルはセキュアリソースとして設定できます。

プロファイルの追加

HM にアクセスする前に、新しいプロファイル（名前は [デフォルト]）が自動的に作成されます。デフォルトのプロファイルをそのまま使用できるほか、別のプロファイルを追加することもできます。

注: Informatica Data Director では、エンティティラベルおよびリレーション/エンティティのツールチップの表示を定義する際にデフォルトのプロファイルが使用されます。追加のプロファイル、およびプロファイル内で定義した追加の情報は、階層マネージャコンソール内でのみ使用され、Informatica Data Director では使用されません。

新しいプロファイルを追加する手順

1. 書き込みロックを取得します。
2. 階層ツールで、ナビゲーションペインの任意の場所を右クリックし、**【プロファイルの追加】** を選択します。

ナビゲーションツリー内の [プロファイル] ノードの下に新しいプロファイルが表示されます（名前は **【新しいプロファイル】** になります）。プロパティペインには、デフォルトのプロパティが表示されます。これらのリレーションタイプを選択して [保存] をクリックすると、ツリーの [プロファイル] の下に、エンティティオブジェクト、エンティティタイプ、リレーションオブジェクト、およびリレーションタイプのノードが表示されます。リレーションタイプの選択を解除すると、ツリーからリレーションタイプだけが削除されます（エンティティタイプは削除されません）。

3. この新しいプロファイルの以下の情報を指定します。

フィールド	説明
名前	このプロファイルの一意のわかりやすい名前。
説明	このプロファイルの説明。
リレーションタイプ	このプロファイルに関連付けられるリレーションタイプを 1 つまたは複数選択します。

4. **【保存】** をクリックして新しいプロファイルを保存します。

画面の [参照] セクションに、選択したリレーションタイプに関する情報が表示されます。また、エンティティタイプも表示されます。この情報は、選択したリレーションタイプから取得されたものです。

プロファイルの編集

プロファイルを編集する手順

1. 書き込みロックを取得します。
2. ナビゲーションツリーの階層ツールで、編集するプロファイルをクリックします。
3. 必要に応じて、プロファイルを設定します（適切なプロファイル名、説明、リレーションタイプを指定し、パッケージを割り当てます）。
4. **【保存】** をクリックして変更を保存します。

プロファイルの検証

プロファイルを検証する手順

1. 書き込みロックを取得します。
2. 階層ツールのナビゲーションペインで、検証するプロファイルを選択します。

3. プロパティペインで、[検証] タブをクリックします。
注: プロファイルは、パッケージがエンティティタイプおよびリレーションタイプに割り当てられた後にのみ正常に検証できます。
階層ツールに [検証] タブが表示されます。
4. 使用するサンドボックスを選択します。
サンドボックスの作成と設定の詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。
5. データを検証するには、[データを検証する] チェックボックスをオンにします。多数のレコードがある場合は時間がかかることがあります。
6. 検証プロセスを開始するには、[HM 設定の検証] をクリックします。
検証プロセス中は、階層ツールに進捗状況ウィンドウが表示されます。検証の結果はボタンの下のウィンドウに表示されます。
7. 検証が完了したら、[保存] をクリックします。
8. 検証レポートを保存するディレクトリを選択します。
9. [クリア] をクリックして、検証結果の説明を示すボックスをクリアします。

プロファイルのコピー

プロファイルをコピーする手順

1. 書き込みロックを取得します。
2. 階層ツールで、コピーするプロファイルを右クリックし、[プロファイルのコピー] を選択します。
ナビゲーションツリー内の [プロファイル] ノードの下に新しいプロファイルが表示されます（名前は [新しいプロファイル] になります）。この新しいプロファイルは、選択してコピーしたプロファイルの完全なコピーです（名前は異なります）。プロパティペインには、デフォルトのプロパティが表示されます。
3. 必要に応じて、プロファイルを設定します（適切なプロファイル名、説明、リレーションタイプを指定し、パッケージを割り当てます）。
4. [保存] をクリックして新しいプロファイルを保存します。

プロファイルの削除

プロファイルを削除する手順

1. 書き込みロックを取得します。
2. 階層ツールで、削除するプロファイルを右クリックし、[プロファイルの削除] を選択します。
階層ツールに、このプロファイルを削除するとパッケージが削除されることを警告するウィンドウが表示されます。
3. [はい] をクリックします。
階層ツールにより、プロファイルが削除されます。

プロファイルからのリレーションタイプの削除

リレーションタイプを削除する手順

1. 書き込みロックを取得します。

2. 階層ツールで、リレーションタイプを右クリックし、**[プロファイルからのエンティティタイプ/リレーションタイプの削除]** を選択します。

削除するエンティティ/リレーションタイプを使用するリレーションタイプがプロファイルに含まれている場合は、まずそのリレーションタイプをプロファイルから削除しないと、目的のタイプを削除できません。

プロファイルからのエンティティタイプの削除

エンティティタイプを削除する手順

1. 書き込みロックを取得します。
2. 階層ツールで、エンティティタイプを右クリックし、**[プロファイルからのエンティティタイプ/リレーションタイプの削除]** を選択します。

削除するエンティティタイプを使用するリレーションタイプがプロファイルに含まれている場合は、まずそのリレーションタイプをプロファイルから削除しないと、目的のタイプを削除できません。

エンティティタイプおよびリレーションタイプへのパッケージの割り当て

プロファイルを作成したら、以下を行う必要があります。

- プロファイルに関連付けられたエンティティタイプおよびリレーションタイプにパッケージを割り当てる。
- パッケージをセキュアリソースとして設定する。

第 14 章

階層マネージャのチュートリアル

この章では、以下の項目について説明します。

- [階層の設定の概要, 229 ページ](#)
- [チュートリアル例について, 231 ページ](#)
- [手順 1. Product エンティティベースオブジェクトの作成, 232 ページ](#)
- [手順 2. エンティティタイプの作成, 232 ページ](#)
- [手順 3. 製品リレーションオブジェクトの作成, 236 ページ](#)
- [手順 4. リレーションタイプの作成, 237 ページ](#)
- [手順 5. 階層タイプの作成, 240 ページ](#)
- [手順 6. 階層プロファイルへのリレーションタイプの追加, 240 ページ](#)
- [手順 7. パッケージの作成, 241 ページ](#)
- [手順 8. パッケージの割り当て, 243 ページ](#)
- [手順 9. 階層マネージャにおけるデータの表示項目の設定, 246 ページ](#)
- [階層管理, 264 ページ](#)

階層の設定の概要

MDM 階層は MDM Hub 内のレコード間のリレーションを示します。リレーションは、同じエンティティベースオブジェクト内のレコード間、または別のエンティティベースオブジェクト内のレコード間でも可能です。階層にデータを入力する前に、階層を設定する必要があります。

階層を設定する前に、データを把握し、確立するリレーションを決定する必要があります。このチュートリアルのために、異なる製品グループに分類する製品レコードを用意しました。このチュートリアルでは、製品階層を設定する手順を説明します。

製品階層を設定するには、以下の手順を実行します。

1. Product エンティティベースオブジェクトを作成します。Product エンティティベースオブジェクトには、製品グループエンティティタイプと製品エンティティタイプのデータが保存されます。
2. エンティティタイプを作成します。
 - a. 製品エンティティタイプを作成します。各製品のデータを含むレコードが製品エンティティタイプとなります。
 - b. 製品グループエンティティタイプを作成します。階層内の製品カテゴリが製品グループエンティティタイプとなります。

3. Product Rel リレーションベースオブジェクトを作成します。Product Rel リレーションベースオブジェクトには、[製品グループは製品グループの親です] リレーションタイプと [製品グループは製品の親です] リレーションタイプのデータが保存されます。
4. リレーションタイプを作成します。
 - a. [製品グループは製品グループの親です] リレーションタイプを作成します。
 - b. [製品グループは製品の親です] リレーションタイプを作成します。
5. 階層設定を作成します。階層設定で、階層にエンティティタイプ、リレーションタイプ、および階層プロファイルに関連付けます。
6. デフォルトの階層プロファイルにリレーションタイプを追加します。リレーションタイプを階層プロファイルに追加すると、そのリレーションタイプに関連付けられているエンティティタイプも階層プロファイルに追加されます。
7. パッケージを作成します。パッケージと関連付けられているクエリは、ユーザーが階層マネージャでアクセスできるデータを定義します。
 - a. 製品エンティティオブジェクトパッケージを作成します。
 - b. Product Rel リレーションオブジェクトパッケージを作成します。
8. パッケージを割り当てます。
 - a. [PKG Product] パッケージを製品エンティティタイプに割り当てます。
 - b. [PKG Product] パッケージを製品グループエンティティタイプに割り当てます。
 - c. [PKG 製品リレーション] パッケージを [製品グループは製品グループの親です] リレーションタイプに割り当てます。
 - d. [PKG 製品リレーション] パッケージを [製品グループは製品の親です] リレーションタイプに割り当てます。
9. 階層マネージャでのデータの表示を設定します。
 - a. 各エンティティタイプにエンティティオブジェクトラベルを設定します。
 - b. 各エンティティタイプにエンティティオブジェクトツールチップテキストを設定します。
 - c. 各リレーションタイプにリレーションオブジェクトツールチップテキストを設定します。
 - d. 各エンティティタイプにエンティティリストを設定します。
 - e. 各リレーションタイプにリレーションリストを設定します。
 - f. 各エンティティタイプにエンティティ検索フィールドを設定します。
 - g. 各リレーションタイプにリレーション検索フィールドを設定します。
 - h. 各エンティティタイプにエンティティ検索結果を設定します。
 - i. 各エンティティタイプにリレーション検索結果を設定します。
 - j. 各エンティティタイプにエンティティの詳細を設定します。
 - k. 各リレーションタイプにリレーションの詳細を設定します。
 - l. 編集可能なエンティティフィールドを設定します。
 - m. 各エンティティタイプにエンティティ作成フィールドを設定します。
 - n. 各リレーションタイプにリレーション作成フィールドを設定します。

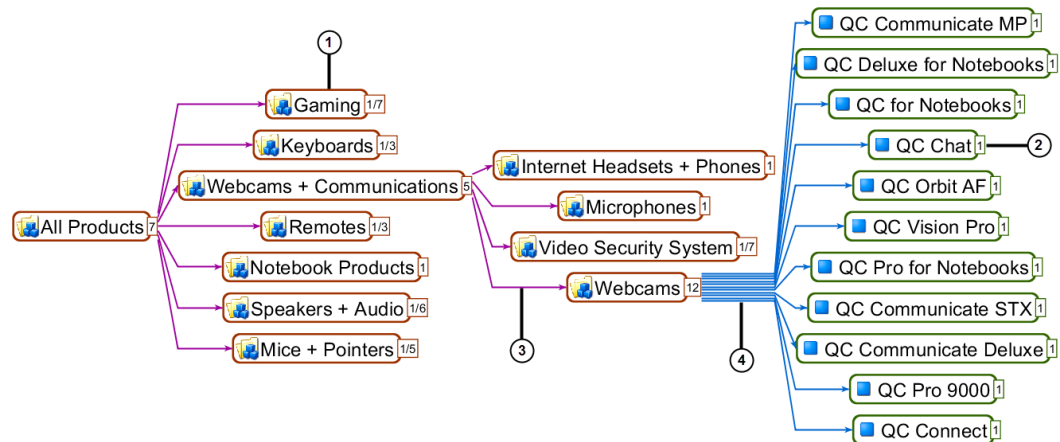
チュートリアル例について

このチュートリアルでは、製品階層を作成する手順について説明します。階層は、製品と、製品をまとめた製品グループで構成されます。

チュートリアルで設定する階層は、Resource Kit に付属するサンプルのオペレーショナル参照ストア内にあるデータを使用して、事前に設定およびデータが入力されています。設定した階層を参照する場合、サンプルのオペレーショナル参照ストアをインポートします。サンプルのオペレーショナル参照ストアの詳細については、『Multidomain MDM のサンプル ORS ガイド』を参照してください。

チュートリアルの階層には、エンティティタイプとリレーションタイプがそれぞれ2つあります。エンティティタイプには「製品グループ」と「製品」があります。リレーションタイプには、「製品グループは製品グループの親です」と「製品グループは製品の親です」の2つがあります。エンティティタイプのデータは、Product エンティティベースオブジェクトに保存されます。リレーションタイプのデータは、Product Rel リレーションベースオブジェクトに保存されます。

以下の図に、Hub コンソール階層マネージャの階層ビューに表示される製品階層の一部を示します。



1. 製品グループエンティティ
2. 製品エンティティ
3. 「製品グループは製品グループの親です」リレーション
4. 「製品グループは製品の親です」リレーション

外部キーリレーション

外部キーリレーションは、外部キーフィールドを使用して、2つのエンティティベースオブジェクト間のリレーションを保持します。

このチュートリアルでは、階層内のリレーションは Product Rel リレーションベースオブジェクトに保持されるため、外部キーフィールドに基づくリレーションは必要ありません。

手順 1. Product エンティティベースオブジェクトの作成

製品および製品グループのエンティティタイプのレコードを保存する Product エンティティベースオブジェクトを作成する必要があります。エンティティベースオブジェクトに変換でき、空状態の管理が有効な Product ベースオブジェクトがあると仮定します。ベースオブジェクトをエンティティベースオブジェクトに変換すると、Hub コンソールで必要な階層カラムがベースオブジェクトに追加されます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. **【階層】 > 【BO をエンティティ/リレーションオブジェクトに変換】** を選択します。
4. **【既存のベースオブジェクトの変更】** ダイアログボックスで、リストから Product ベースオブジェクトを選択します。**【エンティティベースオブジェクトに変換する】** を選択し、**【OK】** をクリックします。
5. **【OK】** をクリックします。
6. エンティティタイプフィールドのパラメータに以下の外部キーカラムを選択します。

エンティティタイプの外部キーカラム

【ベースオブジェクトクラスコード】 を選択すると、選択した値に基づいてリレーションが確立されます。

使用する既存のベースオブジェクトのカラム

【新しいカラムを作成する】 を選択します。

表示名

外部キーカラムの表示名。製品タイプと入力します。

物理名

外部キーカラムの名前。PRODUCT_TYPE と入力します。

手順 2. エンティティタイプの作成

製品エンティティタイプおよび製品グループエンティティタイプを作成します。製品エンティティは、製品データを含む製品レコードです。製品グループエンティティは、製品をグループに分類するのに使用します。例えば、**【Webcams】** 製品グループには、すべての Web カメラ製品レコードが含まれます。エンティティタイプを作成する場合は、外部キーと、エンティティタイプが階層マネージャに表示される方法を設定します。

以下のエンティティタイプのパラメータを設定できます。

コード

エンティティタイプの一意のコード名。

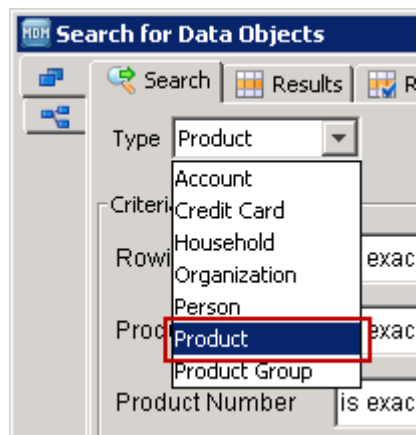
以下の図に、Web カメラ製品レコードのセルに表示される [製品] 外部キーコードを示します。

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

表示名

階層ツールに表示される、このエンティティタイプの名前。

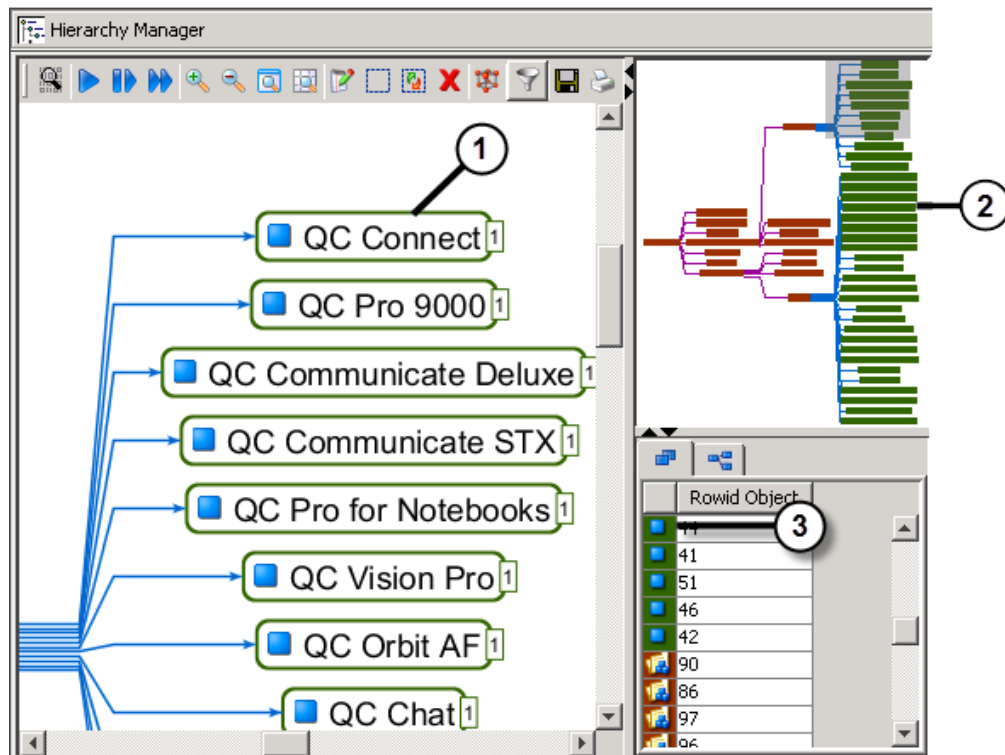
以下の図に、階層マネージャ検索ウィンドウに表示される [製品] 表示名を示します。



色

エンティティタイプに関連するエンティティの色で、階層マネージャツールに表示されます。

以下の図に、階層マネージャツールで表示されるエンティティタイプの色を示します。



1. 階層ビュー内にあるエンティティのアウトラインの色。
2. 階層の概要内にあるエンティティの色。
3. エンティティテーブル内のエンティティの背景色。

小さいアイコン

エンティティタイプに関連するエンティティのアイコンで、階層マネージャのテーブルに表示されます。

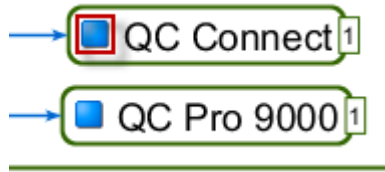
以下の図に、階層マネージャテーブルに表示される製品エンティティタイプの小さいアイコンを示します。

Rowid	Object
44	
41	
51	
46	
42	
90	
86	
97	
96	

大きいアイコン

エンティティタイプに関連するエンティティのアイコンで、階層マネージャの階層ビューペインに表示されます。

以下の図に、階層ビューに表示される製品エンティティタイプの大きいアイコンを示します。



製品エンティティタイプの作成

製品データが含まれる製品レコードは、製品エンティティタイプに属します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションパネルにある **【エンティティオブジェクト】** ノードの下で、Product エンティティベースオブジェクトを選択します。
4. **【階層】** > **【エンティティタイプの追加】** を選択します。
5. 以下のエンティティタイプのプロパティを入力します。

コード

製品と入力します

表示名

製品と入力します

色

【色の選択】 ダイアログから **【RGB】** タブを選択し、色のコード 336600 と入力します。

小さいアイコン

小さい青色の四角形の画像を選択します。

大きいアイコン

大きい青色の四角形の画像を選択します。

6. **【保存】** をクリックします。

製品グループのエンティティタイプの作成

製品グループエンティティは、製品をグループに分類するのに使用します。例えば、**【Webcams】** 製品グループには、すべての Web カメラ製品レコードが含まれます。

1. モデルワークベンチで、**【階層】** をクリックします。
2. 階層ツールのナビゲーションパネルにある **【エンティティオブジェクト】** ノードの下で、Product エンティティベースオブジェクトを選択します。
3. **【階層】** > **【エンティティタイプの追加】** を選択します。
4. 以下のエンティティタイプのプロパティを入力します。

コード

製品グループと入力します

表示名

製品グループと入力します。

色

【色の選択】ダイアログから【RGB】タブを選択し、色のコード 993300 と入力します。

小さいアイコン

小さい青色の四角形グループの画像を選択します。

大きいアイコン

大きい青色の四角形グループの画像を選択します。

5. 【保存】をクリックします。

手順 3. 製品リレーションオブジェクトの作成

製品リレーションオブジェクトは、製品エンティティオブジェクトに関連する階層リレーションを維持します。製品リレーションオブジェクトを作成するには、既存のベースオブジェクトを変換するか、または製品リレーションオブジェクトを作成します。このチュートリアルでは、製品リレーションオブジェクトを作成します。

1. モデルワークベンチで、【階層】をクリックします。
2. 【階層】 > 【新しいエンティティ/リレーションオブジェクトの作成】を選択します。
3. 【新しいリレーションベースオブジェクトを作成する】を選択します。【OK】をクリックします。
4. リレーションオブジェクトの以下のパラメータを入力します。

パラメータ	パラメータ値
表示名	製品リレーション
物理名	C_PRODUCT_REL
データテーブルスペース	CMX_DATA
インデックステーブルスペース	CMX_INDX
説明	オプション。
セキュアリソース	有効
エンティティベースオブジェクト 1	製品
表示名	Product ID1
物理名	PRODUCT_ID1
エンティティベースオブジェクト 2	製品
表示名	Product ID2

パラメータ	パラメータ値
物理名	PRODUCT_ID2
階層の FK カラム	行 ID オブジェクト
階層の FK の表示名	行 ID 階層
階層の FK の物理名	ROWID_HIERARCHY
リレーションタイプの FK カラム	行 ID オブジェクト
リレーションタイプの FK の表示名	行 ID リレーションタイプ
リレーションタイプの FK の物理名	ROWID_REL_TYPE

5. **[OK]** をクリックします。

手順 4.リレーションタイプの作成

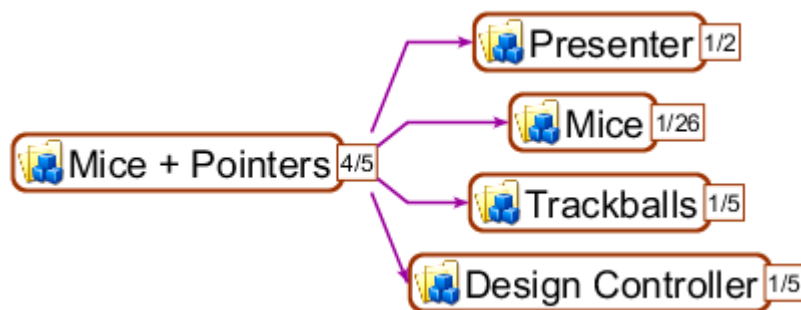
リレーションタイプでは、エンティティタイプ間のリレーションと、階層を表示する場合のリレーションの外観を定義します。

以下のリレーションタイプを作成します。

製品グループは製品グループの親です

ある製品グループエンティティが別の製品グループエンティティの親である場合のリレーションタイプ。例えば、製品グループ [Mice + Pointers] は、[Presenter]、[Mice]、[Trackballs]、および [Design Controller] の親です。

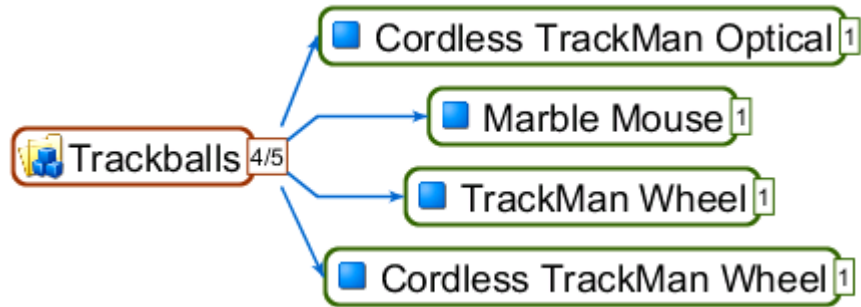
以下の図に、製品グループが他の製品グループの親である場合のリレーションを示します。



製品グループは製品の親です

製品グループエンティティが製品エンティティの親である場合のリレーションタイプ。例えば、[Trackballs] 製品グループは、[Cordless TrackMan Optical]、[Marble Mouse]、[TrackMan Wheel]、および [Cordless TrackMan Wheel] 製品の親です。

以下の図に、製品グループが製品の親である場合のリレーションを示します。



以下のリレーションタイプのパラメータを設定できます。

コード

階層設定内のリレーションのコード。

表示名

階層ツールに表示されるリレーション名。

説明

オプション。

色

2つのエンティティ間のリレーションの矢印の色。

エンティティタイプ 1

リレーション内にある2つのエンティティタイプのうちの最初の1つ。

エンティティタイプ 2

リレーション内にある別のエンティティタイプ。

方向

リレーションの方向。例えば、[エンティティ 1 からエンティティ 2] を選択すると、エンティティ 1 はエンティティ 2 の親になります。

階層

リレーションが属する階層。

リレーションタイプを作成する場合、リレーション、階層マネージャでのリレーションタイプの表示方法、およびリレーションが属する階層を設定します。

【製品グループは製品グループの親です】 リレーションタイプの作成

【製品グループは製品グループの親です】 リレーションタイプは、製品グループエンティティ間のリレーションを維持します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションパネルにある [リレーションオブジェクト] の下で、Product Rel リレーションオブジェクトを選択します。[階層] > [リレーションの追加] を選択します。
4. 以下のリレーションタイプのパラメータを入力します。

コード

製品グループは製品グループの親ですと入力します。

表示名

製品グループは製品グループの親ですと入力します。

説明

オプション。

色

階層マネージャのリレーションを表す矢印の色。[色の選択] ダイアログから [RGB] タブを選択し、色のコード 0066CC と入力します。

エンティティタイプ 1

[製品グループは製品グループの親です] を選択します。

エンティティタイプ 2

[製品グループは製品グループの親です] を選択します。

方向

[エンティティ 1 からエンティティ 2] を選択します。

階層

製品階層を選択します。

5. [保存] をクリックします。

[製品グループは製品の親です] リレーションタイプの作成

[製品グループは製品の親です] リレーションタイプは、製品グループエンティティと製品エンティティの間のリレーションを維持します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションパネルにある [リレーションオブジェクト] の下で、Product Rel リレーションオブジェクトを選択します。[階層] > [リレーションの追加] を選択します。
4. 以下のリレーションタイプのパラメータを入力します。

コード

製品グループは製品の親ですと入力します。

表示名

製品グループは製品の親ですと入力します。

説明

オプション。

色

階層マネージャのリレーションを表す矢印の色。[色の選択] ダイアログから [RGB] タブを選択し、色のコード 990099 と入力します。

エンティティタイプ 1

[製品グループ] を選択します。

エンティティタイプ 2

「製品」を選択します。

方向

「エンティティ 1 からエンティティ 2」を選択します。

階層

製品階層を選択します。

5. 「保存」をクリックします。

手順 5. 階層タイプの作成

階層にエンティティタイプ、リレーションタイプ、およびプロファイルに関連付けるには、階層タイプを作成します。リレーションタイプを作成するには、階層タイプを作成する必要があります。

1. 書き込みロックを取得します。
2. モデルワークベンチで、「階層」をクリックします。
3. 「階層」 > 「階層の追加」を選択します。
4. 次の階層プロパティを入力します。

コード

階層の一意のコード名。製品と入力します。

表示名

Hub コンソールに表示される階層の名前。製品と入力します。

説明

オプション。

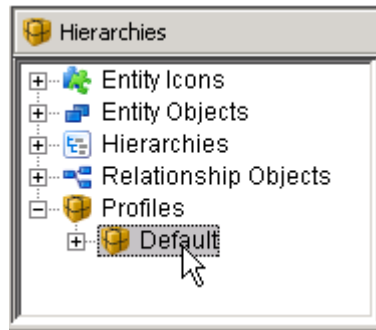
5. 「保存」をクリックします。

手順 6. 階層プロファイルへのリレーションタイプの追加

階層プロファイルにリレーションタイプを追加すると、リレーションタイプに関連するエンティティタイプも階層プロファイルに追加されます。このチュートリアルでは、デフォルトのプロファイルにリレーションタイプを追加します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、「階層」をクリックします。

3. 階層ツールのナビゲーションペインで、デフォルトのプロファイルを選択します。



4. [プロファイル] ペインの [全般] タブにある [リレーションタイプ] セクションで、**【製品グループは製品グループの親です】** および **【製品グループは製品の親です】** の各リレーションタイプを有効にします。
5. **【保存】** をクリックします。

手順 7. パッケージの作成

エンティティオブジェクトおよびリレーションのパッケージを作成します。パッケージでは、ユーザーがアクセスできる階層マネージャ内のデータを定義します。

階層ツールを使用して、階層パッケージおよび関連するクエリを作成します。パッケージを作成する場合は、アクセス設定が可能なカラムを選択します。カラムがパッケージに含まれない場合は、階層マネージャで利用可能なカラムを設定できません。

階層パッケージを作成したら、パッケージはパッケージツールに、クエリはクエリツールに表示されます。パッケージツールまたはクエリツールを使用して、階層パッケージまたはクエリを作成することはできません。

クエリとパッケージの詳細については、[第 9 章、「クエリとパッケージ」 \(ページ 128\)](#)を参照してください。

製品エンティティオブジェクトのパッケージの作成

[PKG Product] パッケージを作成すると、製品エンティティオブジェクトにこのパッケージを割り当てることができます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. **【階層】** > **【新しいパッケージの作成】** を選択します。
4. **【新しいパッケージの作成】** ダイアログボックスの手順 1 で、**【エンティティオブジェクトのパッケージを作成する】** を選択します。
5. **【クエリグループ】** セクションで、クエリグループを選択または作成し、クエリツールでパッケージを整理します。**【次へ】** をクリックします。
6. **【新しいパッケージの作成】** ダイアログボックスの手順 2 で、**【クエリ名】** フィールドに製品と入力します。**【プライマリテーブルを選択】** フィールドで、Product エンティティベースオブジェクトを選択します。**【次へ】** をクリックします。
7. **【新しいパッケージの作成】** ダイアログボックスの手順 3 で、PKG Product と入力します。階層マネージャツールでエンティティレコードフィールドの値を変更できるように、**【put を有効にする】** チェックボックスを有効にします。**【次へ】** をクリックします。

8. [新しいパッケージの作成] ダイアログボックスの手順 4 で、クエリで使用する以下のカラムを選択します。
 - 行 ID オブジェクト
 - 製品名
 - 製品番号
 - 製品の説明
 - 製品タイプコード
 - 製品タイプ
 - Hub 状態インジケータ

注: [行 ID オブジェクト]、[Hub 状態インジケータ]、または [製品タイプ] は無効にできません。
9. [完了] をクリックします。

製品リレーションのパッケージの作成

[PKG 製品リレーション] パッケージを作成すると、製品リレーションのリレーションオブジェクトにこのパッケージを割り当てることができます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. [階層] > [新しいパッケージの作成] を選択します。
4. [新しいパッケージの作成] ダイアログボックスの手順 1 で、[リレーションオブジェクトのパッケージを作成する] を選択します。
5. [クエリグループ] セクションで、クエリグループを選択または作成し、RL クエリなどのクエリツールでパッケージを整理します。[次へ] をクリックします。
6. [新しいパッケージの作成] ダイアログボックスの手順 2 で、[クエリ名] フィールドに **RL 製品** と入力します。[プライマリテーブルを選択] フィールドで、Product Rel エンティティベースオブジェクトを選択します。[次へ] をクリックします。
7. [新しいパッケージの作成] ダイアログボックスの手順 3 で、PKG 製品リレーションと入力します。[put を有効にする] チェックボックスを有効にします。[次へ] をクリックします。
8. [新しいパッケージの作成] ダイアログボックスの手順 4 で、クエリで使用する以下のカラムを選択します。
 - 行 ID オブジェクト
 - 行 ID 階層
 - 行 ID リレーションタイプ
 - Product ID1
 - Product ID2
 - 階層レベル
 - Hub 状態インジケータ

注: [行 ID オブジェクト]、[行 ID 階層]、[行 ID リレーションタイプ]、[製品 ID1]、[製品 ID2]、または [Hub 状態インジケータ] を無効にすることはできません。
9. [完了] をクリックします。

手順 8.パッケージの割り当て

エンティティタイプまたはリレーションタイプのデータ表示方法を設定できるように、エンティティタイプとリレーションタイプにパッケージを割り当てます。

ユーザーによる割り当ての可能なパッケージには、次の種類があります。

表示パッケージ

データの読み取りアクセス用パッケージ。必須。

Put パッケージ

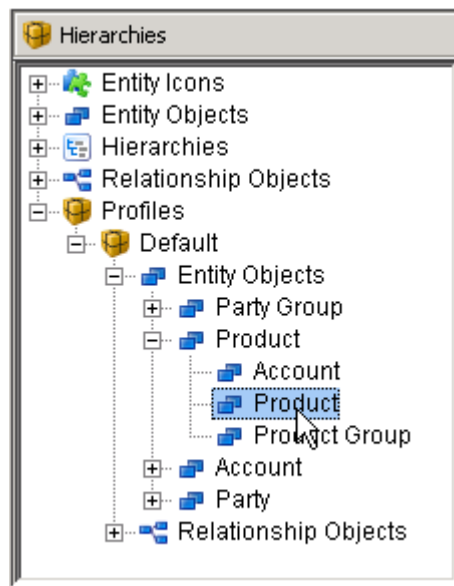
データの書き込みアクセス用パッケージ。オプション。

エンティティタイプまたはリレーションタイプに Put パッケージを割り当てない場合は、そのタイプを作成または編集できません。例えば、製品エンティティタイプに Put パッケージを割り当てない場合は、階層マネージャで新規製品レコードを作成したり、既存の製品レコードを編集したりできません。Put に対応している Put パッケージしか割り当てることができません。

[PKG Product] パッケージを製品エンティティタイプに割り当てる

階層マネージャで製品エンティティデータを表示する方法を設定できるように、製品エンティティタイプに [PKG Product] パッケージを割り当てます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。



4. [MDM 表示パッケージ] リストから [PKG Product] パッケージを選択します。
5. [MDM Put パッケージ] リストから [PKG Product] パッケージを選択します。

[HM パッケージ] セクションで、製品エンティティタイプのデータの表示項目を設定できるようになりました。

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								

[PKG Product] パッケージを製品グループのエンティティタイプに割り当てる

階層マネージャで製品グループのエンティティデータを表示する方法を設定できるように、製品グループのエンティティタイプに [PKG Product] パッケージを割り当てます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品グループのエンティティタイプのノードを選択します。
4. [MDM 表示パッケージ] リストから [PKG Product] パッケージを選択します。
5. [MDM Put パッケージ] リストから [PKG Product] パッケージを選択します。

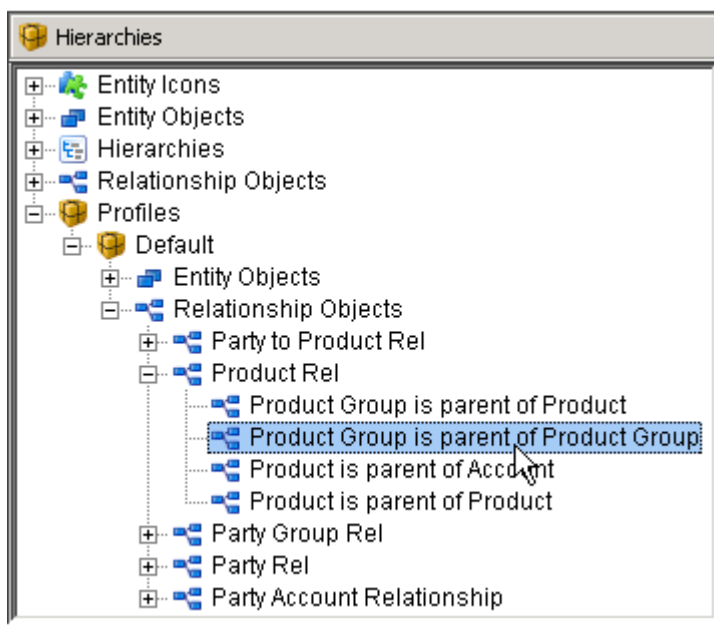
[HM パッケージ] セクションで、製品グループエンティティタイプのデータの表示項目を設定できるようになりました。

[PKG 製品リレーション] パッケージを [製品グループは製品グループの親です] リレーションタイプに割り当てる

階層マネージャでリレーションデータを表示する方法を設定できるように、[PKG 製品リレーション] パッケージを [製品グループは製品グループの親です] リレーションタイプに割り当てます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。

3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある [製品グループは製品グループの親です] リレーションタイプのノードを選択します。



4. [MDM 表示パッケージ] リストから [PKG 製品リレーション] パッケージを選択します。
5. [MDM Put パッケージ] リストから [PKG 製品リレーション] パッケージを選択します。

[HM パッケージ] セクションで、[製品グループは製品グループの親です] リレーションタイプのデータの表示項目を設定できるようになりました。

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	▼	▼	▼	▼	▼	▼	▼
ProductID1	▼	▼	▼	▼	▼	▼	▼
ProductID2	▼	▼	▼	▼	▼	▼	▼
Hub State Indicator	▼	▼	▼	▼	▼	▼	▼
Hierarchy Level	▼	▼	▼	▼	▼	▼	▼

[PKG 製品リレーション] パッケージを [製品グループは製品の親です] リレーションに割り当てる

階層マネージャでリレーションデータを表示する方法を設定できるように、[PKG 製品リレーション] パッケージを [製品グループは製品の親です] リレーションタイプに割り当てます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある [製品グループは製品の親です] リレーションタイプのノードを選択します。
4. [MDM 表示パッケージ] リストから [PKG 製品リレーション] パッケージを選択します。
5. [MDM Put パッケージ] リストから [PKG 製品リレーション] パッケージを選択します。

[HM パッケージ] セクションで、[製品グループは製品の親です] リレーションタイプのデータの表示項目を設定できるようになりました。

手順 9. 階層マネージャにおけるデータの表示項目の設定

ユーザーがレコードを検索、表示、編集、および追加した場合に、階層マネージャで表示するエンティティフィールドとリレーションフィールドを設定できます。また、階層マネージャのラベルとツールチップに表示するフィールドも設定できます。

階層マネージャでのデータの表示項目を設定するには、各リレーションタイプとエンティティタイプで階層マネージャパッケージを設定します。複数の階層プロファイルがある場合は、各プロファイルに対してデータの表示項目を設定する必要があります。

次の階層マネージャのユーザーインタフェース要素に表示するフィールドとその順序を設定できます。

ラベル

階層マネージャツールのエンティティに対するラベルテキスト。

ツールチップ

階層マネージャツール内のエンティティまたはリレーションにマウスポインタを合わせてしばらく待った際に表示されるテキスト。

一般

各種タイプのエンティティとリレーションが同じリスト内に表示されるときに表示するフィールド。選択するフィールドは、すべての階層マネージャパッケージ内にある必要があります。

検索

エンティティレコードまたはリレーションレコードの検索に使用できるフィールド。

リスト

エンティティレコードまたはリレーションレコードの検索結果のリストに表示されるフィールド。

詳細

選択したエンティティまたはリレーションの詳細の表示を選択したときに表示されるフィールド。

Put

選択したエンティティまたはリレーションを編集するときに値を変更できるフィールド。

追加

階層マネージャでエンティティまたはリレーションを追加する際に表示されるフィールド。

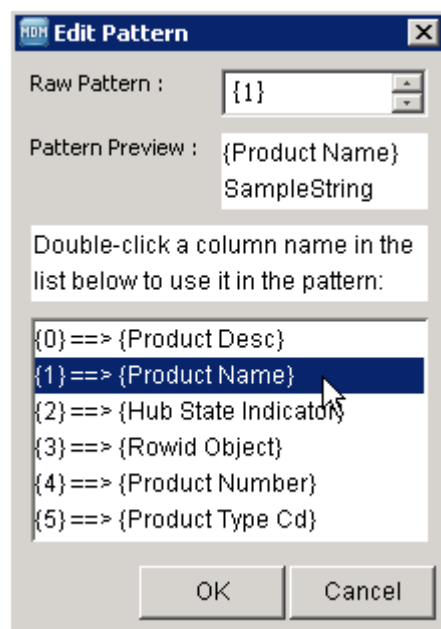
製品エンティティタイプのラベルの設定

このチュートリアルでは、製品エンティティのラベルテキストとして表示する製品名を設定します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
4. [HM パッケージ] セクションの [ラベル] カラムで、[製品名] 行に番号が割り当てられていることを確認します。

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. [エンティティタイプの HM パッケージ] セクションで、[ラベルパターン] フィールドの [編集] ボタンをクリックします。
6. [パターンの編集] ダイアログボックスで [{#} ==> {Product Name}] をダブルクリックします。



[Raw パターン] フィールドで、カラムを連結したり、カスタムテキストを追加したりできます。ここでは、カスタムテキストを追加せずに [製品名] カラムを使用します。

7. [保存] をクリックします。

以下の図に、階層マネージャに表示される製品名 [Media Desktop Laser] の製品エンティティを示します。

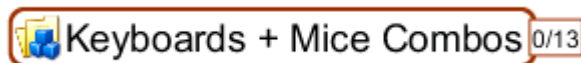


製品グループエンティティタイプのラベルの設定

このチュートリアルでは、製品グループエンティティのラベルテキストとして表示する製品名を設定します。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品グループのエンティティタイプのノードを選択します。
4. [HM パッケージ] セクションの [ラベル] カラムで、[製品名] 行に番号が割り当てられていることを確認します。
5. [エンティティタイプの HM パッケージ] セクションで、[ラベルパターン] フィールドの [編集] ボタンをクリックします。
6. [パターンの編集] ダイアログボックスで [{#} ==> {Product Name}] をダブルクリックします。
[Raw パターン] フィールドで、カラムを連結したり、カスタムテキストを追加したりできます。ここでは、カスタムテキストを追加せずに [製品名] カラムを使用します。
7. [保存] をクリックします。

以下の図に、階層マネージャに表示される製品名 [Keyboards + Mice Combos] の製品グループエンティティを示します。



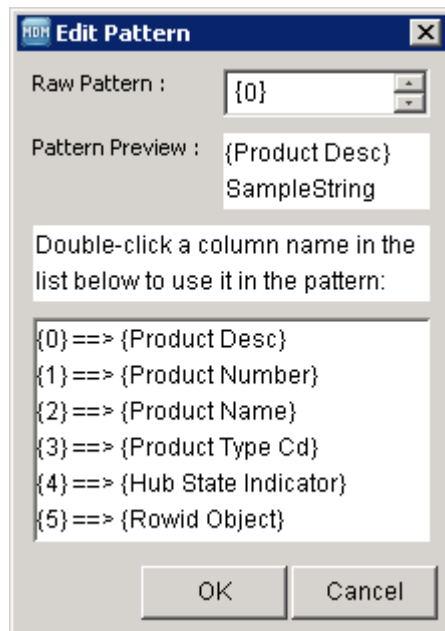
製品エンティティタイプのツールチップテキストの設定

このチュートリアルでは、製品エンティティのツールチップテキストとして表示する製品の説明を設定します。階層マネージャの製品エンティティにマウスポインタを合わせると、製品の説明がツールチップとして表示されます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
4. [HM パッケージ] セクションの [ツールチップ] カラムで、[製品の説明] 行に番号が割り当てられていることを確認します。

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. **【エンティティタイプの HM パッケージ】** セクションで、**【ツールチップパターン】** フィールドの **【編集】** ボタンをクリックします。
6. **【パターンの編集】** ダイアログボックスで **[[#]==> {Product Desc}]** をダブルクリックします。



【Raw パターン】 フィールドで、カラムを連結したり、カスタムテキストを追加したりできます。ここでは、カスタムテキストを追加せずに **【製品の説明】** カラムを使用します。

7. **【保存】** をクリックします。

以下の図に、ツールチップとして表示される製品の説明を設定した **【LS1 Laser Mouse】** 製品エンティティを示します。



製品グループエンティティタイプのツールチップテキストの設定

このチュートリアルでは、製品グループエンティティタイプのツールチップテキストとして表示する製品の説明を設定します。階層マネージャの製品グループエンティティにマウスポインタを合わせると、製品の説明がツールチップとして表示されます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションペインで、**【デフォルト】** 階層プロファイルの下にある製品グループのエンティティタイプのノードを選択します。
4. **【HM パッケージ】** セクションの **【ツールチップ】** カラムで、**【製品の説明】** 行に番号が割り当てられていることを確認します。
5. **【エンティティタイプの HM パッケージ】** セクションで、**【ツールチップパターン】** フィールドの **【編集】** ボタンをクリックします。
6. **【パターンの編集】** ダイアログボックスで **[[#]==> {Product Desc}]** をダブルクリックします。

【Raw パターン】フィールドで、カラムを連結したり、カスタムテキストを追加したりできます。ここでは、カスタムテキストを追加せずに【製品の説明】カラムを使用します。

7. 【保存】をクリックします。

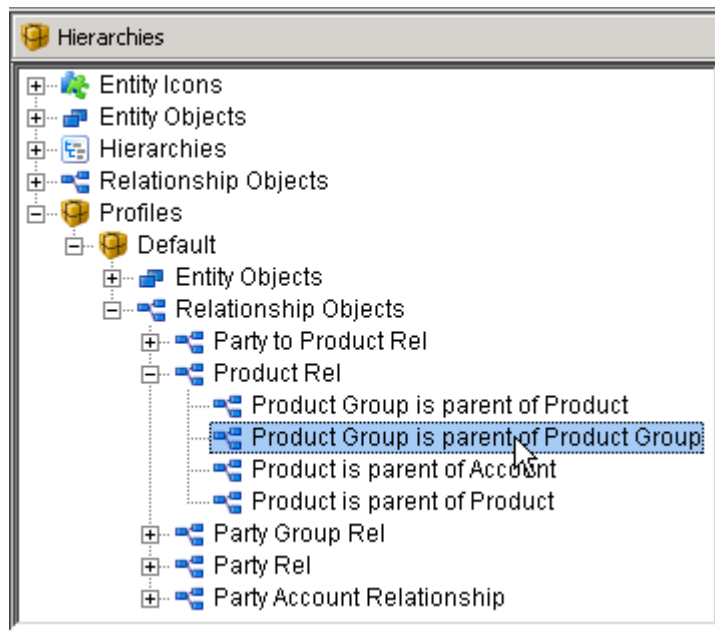
以下の図に、ツールチップとして表示される製品の説明を設定した【Webcams】製品グループエンティティを示します。



【製品グループは製品グループの親です】リレーションタイプのツールチップテキストの設定

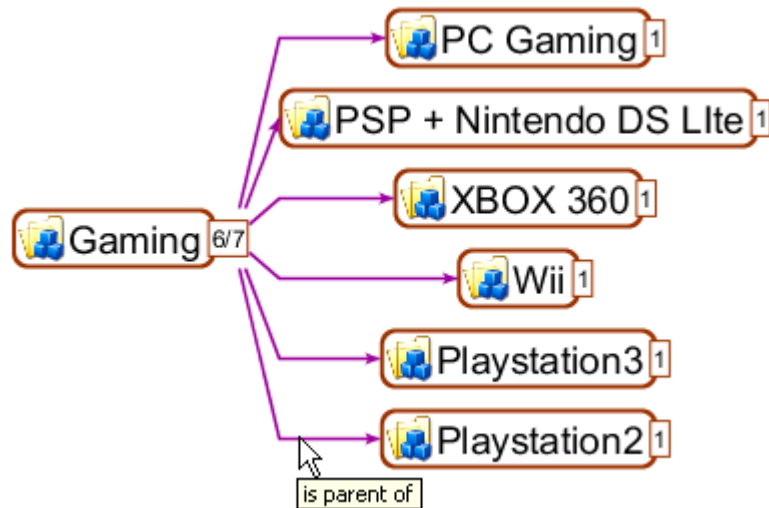
このチュートリアルでは、【製品グループは製品グループの親です】のツールチップテキストとして表示するテキスト【次の項目の親です】を設定します。階層マネージャでリレーションを表す矢印にマウスポインタを合わせると、ツールチップテキストが表示されます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、【階層】をクリックします。
3. 階層ツールのナビゲーションペインで、【デフォルト】階層プロファイルの下にある【製品グループは製品グループの親です】リレーションタイプのノードを選択します。



4. 【エンティティタイプの HM パッケージ】セクションで、【ツールチップパターン】フィールドの【編集】ボタンをクリックします。
5. 【パターンの編集】ダイアログボックスの【Raw パターン】フィールドで、次の項目の親ですと入力します。
6. 【保存】をクリックします。

以下の図に、[ゲーム] 製品グループと [Playstation 2] 製品グループとのリレーションのツールチップを示します。

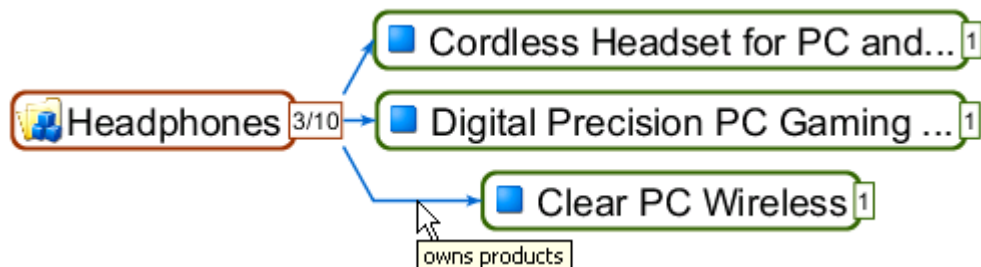


「製品グループは製品の親です」 リレーションタイプのツールチップテキストの設定

このチュートリアルでは、「製品グループは製品の親です」のツールチップテキストとして表示するテキスト【所有する製品】を設定します。階層マネージャでリレーションを表す矢印にマウスポインタを合わせると、ツールチップテキストが表示されます。

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある [製品グループは製品の親です] リレーションタイプのノードを選択します。
4. [エンティティタイプの HM パッケージ] セクションで、[ツールチップパターン] フィールドの [編集] ボタンをクリックします。
5. [パターンの編集] ダイアログボックスの [Raw パターン] フィールドで、所有する製品と入力します。
6. [保存] をクリックします。

以下の図に、[Headphones] 製品グループと [Clear PC Wireless] 製品とのリレーションのツールチップを示します。



エンティティのリストの設定

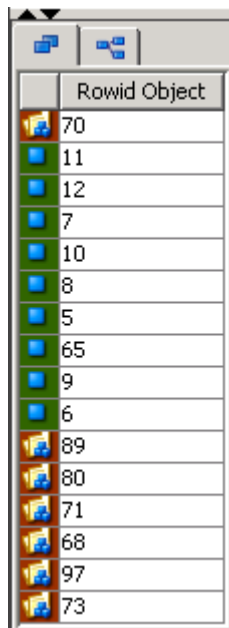
このチュートリアルでは、階層マネージャで表示されるエンティティのリストに表示する [行 ID オブジェクト] フィールドを設定します。エンティティリストには、タイプが異なるエンティティを含めることができま

す。選択できるのは、すべてのエンティティタイプのパッケージを対象とするクエリすべてに含めるフィールドのみです。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションペインで、**【デフォルト】** 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
4. **【HM パッケージ】** セクションの **【共通】** カラムで、**【行 ID オブジェクト】** 行に「0」を割り当てます。

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. **【保存】** をクリックします。
 6. 製品グループエンティティタイプについてもこの手順を繰り返します。
- 以下の図に、階層マネージャに表示される製品と製品グループのエンティティを含むリストを示します。



リレーションのリストの設定

このチュートリアルでは、階層マネージャで表示されるリレーションのリストに表示する **【行 ID オブジェクト】** フィールドを設定します。リレーションのリストには、異なるエンティティタイプのリレーションを含めることができます。選択できるのは、すべてのリレーションタイプのパッケージを対象とするクエリすべてに含めるフィールドのみです。

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションペインで、**【デフォルト】** 階層プロファイルの下にある **【製品グループは製品グループの親です】** リレーションタイプのノードを選択します。

4. [HM パッケージ] セクションの [共通] カラムで、[行 ID オブジェクト] 行に「0」を割り当てます。

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. [保存] をクリックします。
6. [製品グループは製品の親です] リレーションタイプについてもこの手順を繰り返します。

以下の図に、階層マネージャに表示されるリレーションを含むリストを示します。

Rowid Object
24
25
30
32
29
28
105
26
31
27
103
106
102
104
112

エンティティ検索フィールドの設定

このチュートリアルでは、ユーザーが製品エンティティを検索する場合に、以下のフィールドが【データオブジェクトの検索】ダイアログボックスの【検索】タブで降順に表示されるように設定します。

- 製品名
- 製品番号
- 製品の説明
- 製品タイプコード
- 行 ID オブジェクト

1. 書き込みロックを取得します。
2. モデルワークベンチで、[階層] をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。

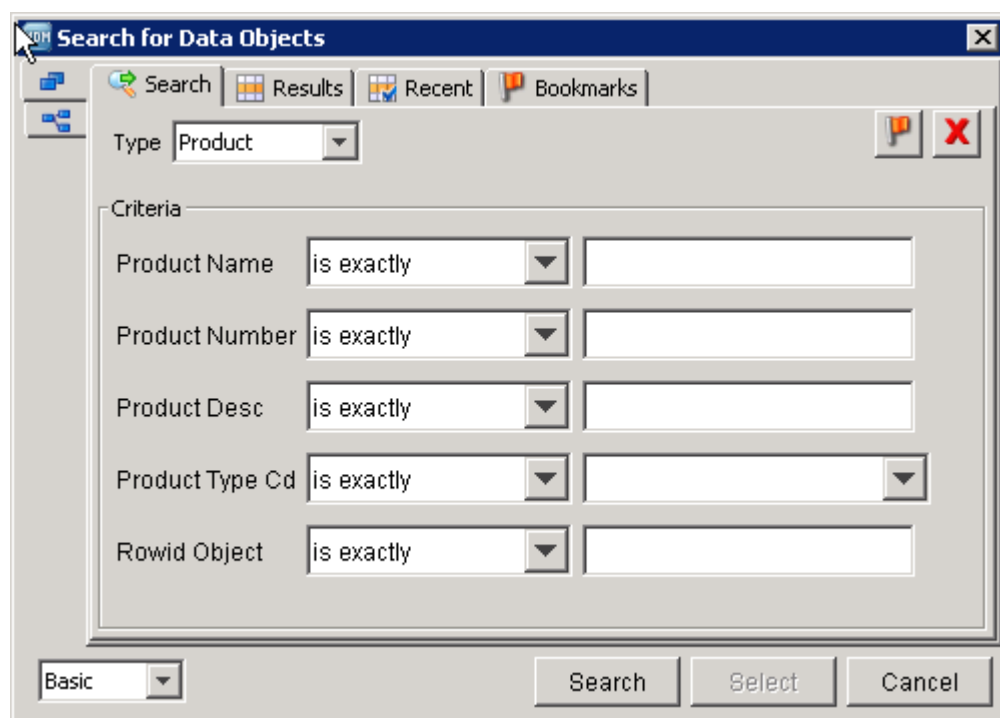
4. [HM パッケージ] セクションの [検索] カラムで、検索フィールドとして表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. [保存] をクリックします。
6. 製品グループエンティティタイプについてもこの手順を繰り返します。

この例で、[製品タイプ] フィールドと [Hub 状態インジケータ] フィールドには番号が割り当てられていません。このため、これらのフィールドは [データオブジェクトの検索] ダイアログボックスに表示されません。[製品名] フィールドがフィールドリストの上部に表示されます。これは、[製品名] に最も小さい値を割り当てたためです。[行 ID オブジェクト] フィールドがリストの下部に表示されます。これは、[行 ID オブジェクト] に最も大きい値を割り当てたためです。

以下の図に、階層マネージャに表示される [検索] タブを示します。



リレーション検索フィールドの設定

このチュートリアルでは、ユーザーがリレーションを検索する場合に、以下のフィールドが [データオブジェクトの検索] ダイアログボックスの [検索] タブで降順に表示されるように設定します。

- 行 ID オブジェクト
- 行 ID 階層
- Product ID1

- Product ID2
- Hub 状態インジケータ

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションペインで、**【デフォルト】** 階層プロファイルの下にある **【製品グループは製品の親です】** リレーションタイプのノードを選択します。
4. **【HM パッケージ】** セクションの **【検索】** カラムで、検索フィールドとして表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. **【保存】** をクリックします。
6. **【製品グループは製品グループの親です】** リレーションタイプについてもこの手順を繰り返します。

以下の図に、階層マネージャに表示される **【検索】** タブを示します。

エンティティ検索結果の設定

このチュートリアルでは、ユーザーがエンティティを検索した場合に、以下の製品フィールドが【データオブジェクトの検索】ダイアログボックスの【結果】タブで左から右に順番に表示されるように設定します。

- 行 ID オブジェクト
- 製品名
- 製品タイプコード
- 製品の説明
- 製品番号
- 製品タイプ

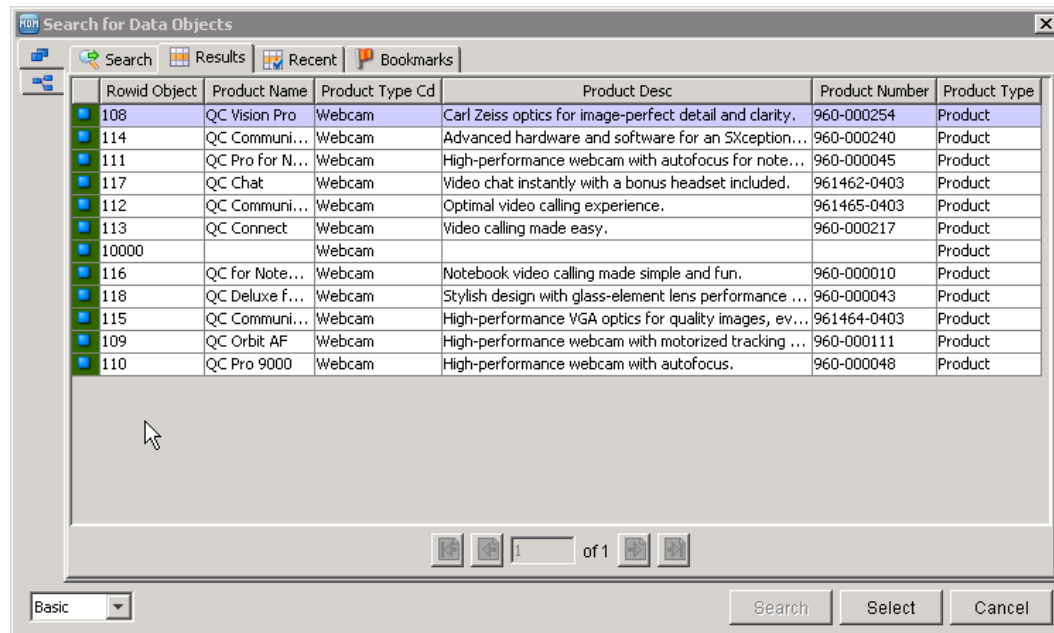
1. 書き込みロックを取得します。
2. モデルワークベンチで、【階層】をクリックします。
3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
4. 【HM パッケージ】セクションの【リスト】カラムで、検索結果に表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. 【保存】をクリックします。
6. 製品グループエンティティタイプについてもこの手順を繰り返します。

この例で、[Hub 状態インジケータ] フィールドには番号が割り当てられていません。このため、このフィールドは【結果】タブに表示されません。[行 ID オブジェクト] フィールドが右に表示されます。これは [行 ID オブジェクト] に最も小さい値を割り当てたためです。[製品タイプ] フィールドが左に表示されます。これは [製品タイプ] に最も大きい値を割り当てたためです。

次の図は、この例で構成設定を使用する際に表示される【結果】タブを示しています。



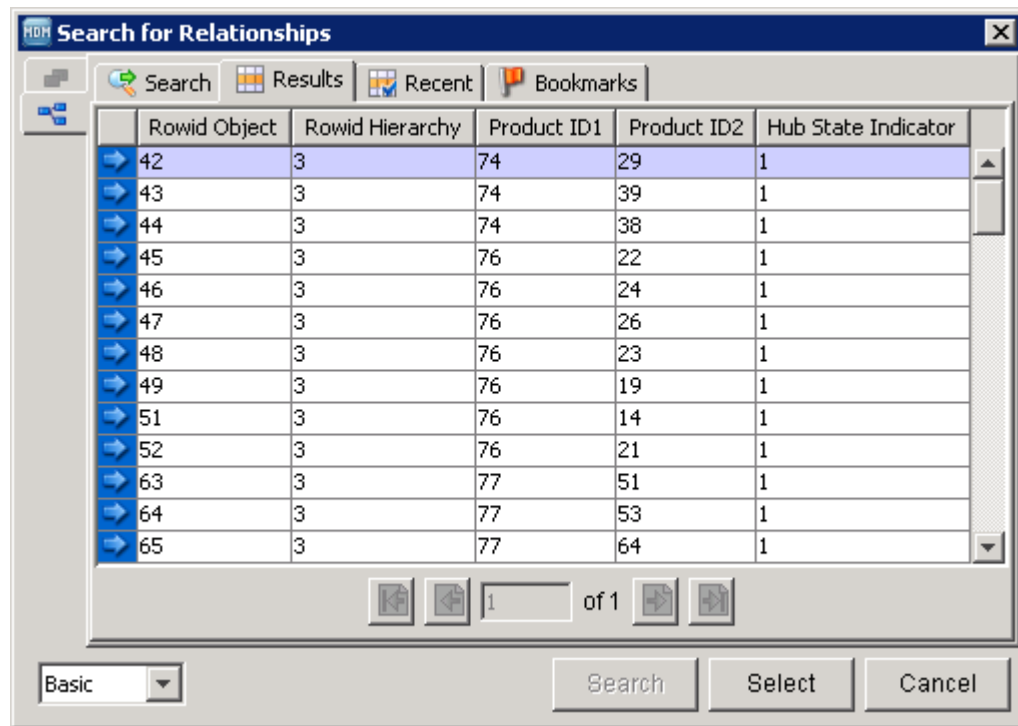
リレーション検索結果の設定

このチュートリアルでは、ユーザーがエンティティを検索した場合に、[製品グループは製品の親です] リレーションタイプの以下のフィールドが、【データオブジェクトの検索】ダイアログボックスの【結果】タブで左から右に順番に表示されるように設定します。

- 行 ID オブジェクト
 - 行 ID 階層
 - Product ID1
 - Product ID2
 - Hub 状態インジケータ
1. 書き込みロックを取得します。
 2. モデルワークベンチで、【階層】をクリックします。
 3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある [製品グループは製品の親です] リレーションタイプのノードを選択します。
 4. 【HM パッケージ】 セクションの 【リスト】 カラムで、検索結果に表示するカラムに番号を割り当てます。最小値を割り当てたカラムが、検索結果の左側に表示されます。

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. **【保存】** をクリックします。
 6. **【製品グループは製品グループの親です】** リレーションタイプについてもこの手順を繰り返します。
- 次の図は、この例で構成設定を使用する際に表示される**【結果】** タブを示しています。



Hub コンソール階層マネージャのエンティティの詳細の設定

このチュートリアルでは、ユーザーが Hub コンソール階層マネージャでエンティティの詳細を確認するときに、**【詳細】** ダイアログボックスに次のフィールドが降順で表示されるように設定します。

- 製品名
- 製品の説明
- 製品番号
- 製品タイプコード
- 行 ID オブジェクト
- Hub 状態インジケータ

1. 書き込みロックを取得します。
2. モデルワークベンチで、**【階層】** をクリックします。
3. 階層ツールのナビゲーションペインで、**【デフォルト】** 階層プロファイルの下にある製品エンティティタイプのノードを選択します。

4. **【HM パッケージ】** セクションの **【詳細】** カラムで、**【詳細】** ダイアログボックスに表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. **【保存】** をクリックします。
 6. 製品グループエンティティタイプについてもこの手順を繰り返します。
- 次の図に、階層マネージャに表示される **【詳細】** ダイアログボックスを示します。

The image shows a 'Details' dialog box with a title bar and a close button. Inside, there's a section 'Entity properties' containing a table with two columns: 'Name' and 'Value'. The table lists the following properties and values:

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

At the bottom of the dialog box, there are two icons (a pencil and a trash can) and an 'OK' button.

IDD 階層マネージャのエンティティの詳細の設定

このチュートリアルでは、ユーザーが IDD 階層マネージャで Person オブジェクトの詳細を確認するときに表示される顧客名を設定します。

1. Informatica Data Director コンフィギュレーションマネージャにログインします。
http://<host: ホスト>:<port: ポート>/bdd/config
2. IDD アプリケーションを選択して、**【編集】** をクリックします。
3. [アプリケーションの編集] 画面の [サブジェクト領域] タブで、**【サブジェクト領域グループ】** > **【顧客】** > **【個人】** > **【名前】** を選択します。
4. **【サブジェクト領域の子の編集】** をクリックします。
5. [サブジェクト領域の子] ダイアログボックスの [レイアウト] タブで、テーブルの [名前] カラム名を選択して、**【レイアウトの編集】** をクリックします。
6. **【HM で表示】** を有効にします。**【OK】** をクリックします。
7. **【OK】** をクリックしてから **【保存】** をクリックします。

リレーションの詳細の設定

このチュートリアルでは、ユーザーがリレーションの詳細を表示する場合に、以下のフィールドが **【詳細】** ダイアログボックスで降順に表示されるように設定します。

- 行 ID オブジェクト

- 行 ID 階層
- Product ID1
- Product ID2
- Hub 状態インジケータ

1. 書き込みロックを取得します。
2. モデルワークベンチで、**[階層]** をクリックします。
3. 階層ツールのナビゲーションペインで、**[デフォルト]** 階層プロファイルの下にある **[製品グループは製品の親です]** リレーションタイプのノードを選択します。
4. **[HM パッケージ]** セクションの **[詳細]** カラムで、**[詳細]** ダイアログボックスに表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. **[保存]** をクリックします。
 6. **[製品グループは製品グループの親です]** リレーションタイプについてもこの手順を繰り返します。
- 以下の図に、階層マネージャに表示される **[詳細]** ダイアログボックスを示します。

Details

Relationship properties

Name	Value
Rowid Object	26
Rowid Hierarchy	3
Product ID1	70
Product ID2	9
Hub State Indicator	1

OK

編集可能なエンティティフィールドの設定

ユーザーが編集できるエンティティフィールドを設定するには、エンティティタイプに Put パッケージを割り当てておく必要があります。このチュートリアルでは、ユーザーがエンティティを編集する場合に、以下のフィールドが **[エンティティレコードエディタ]** ダイアログボックスで降順に表示されるように設定します。

- 製品名
- 製品番号

1. 書き込みロックを取得します。
2. モデルワークベンチで、**[階層]** をクリックします。

3. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
4. **[HM パッケージ]** セクションの **[Put]** カラムで、編集可能フィールドとして表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. **[保存]** をクリックします。
 6. 製品グループエンティティタイプについてもこの手順を繰り返します。
- 以下の図に、階層マネージャに表示される **[エンティティレコードエディタ]** ダイアログボックスを示します。

The dialog box titled "Entity Record Editor" contains the following fields and controls:

- Type:** A dropdown menu with "Product" selected.
- Product Name:** A text input field containing "QC Pro for Notebooks".
- Product Number:** A text input field containing "960-000045".
- Show only editable cells:** An unchecked checkbox.
- Buttons:** "OK", "Cancel", and "Set Default Tru..." at the bottom.

リレーションフィールドの編集について

[PKG 製品リレーション] リレーションパッケージにあるリレーションフィールドは、階層マネージャで編集できません。リレーションパッケージの Put カラムは設定する必要がありません。編集するカラムに番号を割り当てても、ユーザーはリレーションパッケージのフィールドを編集できません。

階層マネージャのリレーションレコードエディタに関する詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

エンティティ作成フィールドの設定

このチュートリアルでは、ユーザーが階層マネージャでエンティティを作成する場合に、以下のフィールドが **[エンティティレコードエディタ]** ダイアログボックスで降順に表示されるように設定します。

- 製品名
- 製品番号
- 製品の説明

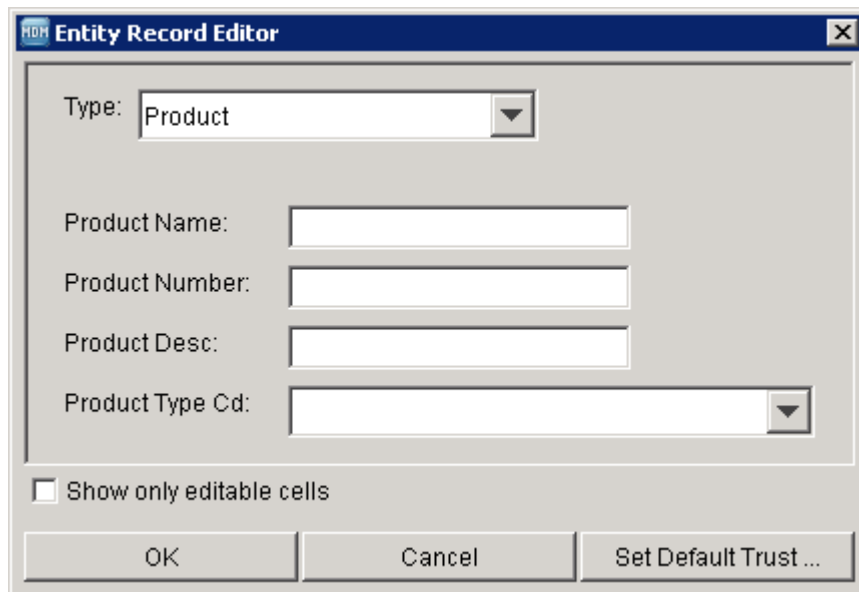
- 製品タイプコード

- 書き込みロックを取得します。
- 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある製品エンティティタイプのノードを選択します。
- [HM パッケージ] セクションの [追加] カラムで、[エンティティレコードエディタ] ダイアログボックスのフィールドとして表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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		

- [保存] をクリックします。
- 製品グループエンティティタイプについてもこの手順を繰り返します。

以下の図に、ユーザーが階層マネージャで製品エンティティを作成する場合の [エンティティレコードエディタ] ダイアログボックスを示します。



The image shows a dialog box titled "Entity Record Editor". It contains the following fields and controls:

- Type:** A dropdown menu with "Product" selected.
- Product Name:** A text input field.
- Product Number:** A text input field.
- Product Desc:** A text input field.
- Product Type Cd:** A dropdown menu.
- Show only editable cells:** An unchecked checkbox.
- Buttons:** "OK", "Cancel", and "Set Default Trust ...".

リレーション作成フィールドの設定

このチュートリアルでは、ユーザーが階層マネージャで2つのエンティティ間のリレーションを作成する場合に、以下のフィールドが [リレーションレコードエディタ] ダイアログボックスで降順に表示されるように設定します。

- Hub 状態インジケータ
- 階層レベル

- 書き込みロックを取得します。

2. 階層ツールのナビゲーションペインで、[デフォルト] 階層プロファイルの下にある [製品グループは製品の親です] リレーションタイプのノードを選択します。
3. [HM パッケージ] セクションの [追加] カラムで、[リレーションレコードエディタ] ダイアログボックスのフィールドとして表示するカラムに番号を割り当てます。最も小さい値を割り当てるカラムがフィールドリストの上部に表示されます。

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. [保存] をクリックします。
5. [製品グループは製品グループの親です] リレーションタイプについてもこの手順を繰り返します。

以下の図に、ユーザーが階層マネージャで2つのエンティティ間のリレーションを作成する場合の [リレーションレコードエディタ] ダイアログボックスを示します。

Relationship Record Editor

77 Mice

61 S150 Laser Mouse for Notebooks

Hierarchy: Product

Relationship Type: Product Group is parent of Pr...

Start Date: [Red X] [Def 31]

End Date: [Red X] [Def 31]

Hub State Indicator: [Dropdown]

Hierarchy Level: [Text Field]

☐ Show only editable cells

OK Cancel Set Default ...

階層管理

階層を設定すると、MDM Hub コンソールまたは Informatica Data Director で階層マネージャツールを使って、レコードとレコード間のリレーションを階層に追加できるようになります。このチュートリアルでは、階層マネージャツールを使って設定済みの階層にデータを入力する方法については説明しません。

MDM Hub コンソールの階層マネージャツールの詳細については、*Multidomain MDM* のデータスチュワードガイドを参照してください。

Informatica Data Director の階層マネージャツールの詳細については、*Multidomain MDM Data Director* のユーザーガイドを参照してください。

パート IV: データフローの設定

この部には、以下の章があります。

- [MDM Hub のプロセス, 266 ページ](#)
- [ランディングプロセスの設定, 295 ページ](#)
- [MDM Hub ステージング, 302 ページ](#)
- [物理削除の検出, 324 ページ](#)
- [データクレンジングの設定, 341 ページ](#)
- [ロードプロセスの設定, 362 ページ](#)
- [一致プロセスの設定, 384 ページ](#)
- [一致ルールの設定例, 444 ページ](#)
- [Elasticsearch による検索, 468 ページ](#)
- [統合プロセスの設定, 492 ページ](#)
- [保留中の制御テーブル, 497 ページ](#)
- [パブリッシュプロセスの設定, 498 ページ](#)

第 15 章

MDM Hub のプロセス

この章では、以下の項目について説明します。

- [MDM Hub のプロセスの概要, 266 ページ](#)
- [Informatica MDM Hub のプロセスについて, 266 ページ](#)
- [ランディングプロセス, 269 ページ](#)
- [ステージプロセス, 271 ページ](#)
- [ロードプロセス, 272 ページ](#)
- [トークン化プロセス, 279 ページ](#)
- [一致プロセス, 286 ページ](#)
- [統合プロセス, 290 ページ](#)
- [パブリッシュプロセス, 292 ページ](#)

MDM Hub のプロセスの概要

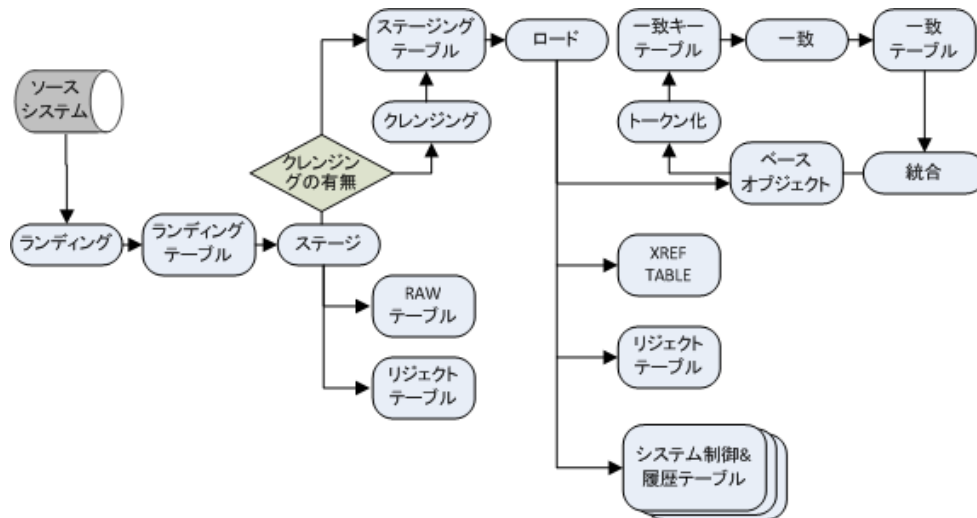
データを MDM Hub にロードして、複数のバッチプロセスで処理できます。ランドプロセスにより、データを MDM Hub に移動できます。その後、このステージプロセスを使用して、データをベースオブジェクトにロードするロードプロセス用にデータをクレンジングして準備することができます。ロードプロセス後、トークン化と一致プロセスを実行することで、重複するレコードを特定して、統合プロセスによって統合することができます。

Informatica MDM Hub のプロセスについて

Informatica MDM Hub のバッチ処理では、データが Informatica MDM Hub の個々のプロセスに順番に渡されていきます。

バッチプロセスの全体的なデータフロー

以下の図に、個別のプロセス、ソースシステム、ベースオブジェクト、およびサポートテーブルなどのバッチプロセスを使用する、Informatica MDM Hub を介した全体的なデータフローの詳細を示します。



注: パブリッシュプロセスはバッチプロセスではないため、この図では説明していません。

ベースオブジェクトのレコードの統合ステータス

ここでは、ベースオブジェクトのレコードの統合ステータスについて説明します。

統合インジケータ

すべてのベースオブジェクトに、CONSOLIDATION_IND という名前のシステムカラムがあります。

この統合インジケータは、MDM Hub の複数のプロセスを進行するにつれての、個々のレコードの統合ステータスを表します。

以下の表に、統合インジケータの値を示します。

値	状態名	説明
1	CONSOLIDATED	レコードが統合され、一意であると判断されていることを示すとともに、最善のデータであることを示します。
2	QUEUED_FOR_MERGE	レコードの一致プロセスがすでに行われている可能性があり、マージが可能な状態であることを示します。また、レコードのマージプロセスがすでに行われている可能性があり、他のマージが可能であることを示します。
3	Not MERGED または MATCHED	一致プロセスが完了し、統合できる状態にあります。また、データスチュワードがマッチングを行って手動でのレコードのマージプロセスを完了すると、統合インジケータは 2 になります。
4	NEWLY_LOADED	レコードが新規に挿入されたものであること、または一致プロセスを実行するためのフラグがレコードに設定されていることを示します。
9	ON_HOLD	通知があるまで、データスチュワードがレコードを保留状態にしていることを示します。統合インジケータの値に関係なく、任意のレコードを保留状態にすることができます。一致プロセスと統合プロセスでは、保留中のレコードは無視されます。

統合インジケータの変化

Informatica MDM Hub では、ベースオブジェクトレコードの統合インジケータが以下の順序で更新されます。

1. ロードプロセスで新しいレコードがベースオブジェクトにロードされると、Informatica MDM Hub によってレコードに統合インジケータ 4 が割り当てられます。これは、レコードの一致が必要であることを示します。
2. マッチプロセスの冒頭でレコードが一致候補として選択されると、マッチプロセスによってレコードの統合インジケータが 3 に変更されます。
注: 一致またはマージの構成設定を変更すると、一致をリセットするダイアログが表示されて、ベースオブジェクト内のレコードをリセット（統合インジケータを 4（一致準備完了）に変更）するかどうかを尋ねられます。
3. 完了前に、マッチプロセスによってマッチ候補レコードの統合インジケータが 2（統合準備完了）に変更されます。手動でマージを行う場合、データスチュワードによって手動のマッチングが行われ、マージされたレコードの統合インジケータは 2 になります。
注: マッチプロセスでレコードの一致が検出されているとは限りません。
統合インジケータが 2 のレコードが、マージマネージャに表示されます。
4. [一致しないすべての行を一意とする] が有効であり、かつレコードがマッチプロセスで処理されてマッチが検出されなかった場合は、Informatica MDM Hub でそのレコードの一致インジケータが自動的に 1（一意）に変更されます。
5. レコードが統合プロセスで処理された後、[一致しないすべての行を一意とする] が有効で、かつレコードにマージする重複レコードがなくなった場合、Informatica MDM Hub によってそのレコードの統合インジケータが 1 に変更されます。これは、このレコードがベースオブジェクト内で一意であり、ベースオブジェクトでのそのエンティティのマスターレコード（ベストバージョンオブトゥールズ）であることを意味します。
注: レコードの統合インジケータが 1 に設定されると、Informatica MDM Hub ではそのレコードが他のレコードと直接一致されなくなります。新しいレコードまたは更新されたレコード（統合インジケータ 4）は、統合レコードと一致できます。

セルデータの存続性と優先順位

2 つのレコードからマージするセルを評価するとき、MDM Hub は存続するセルデータと破棄するセルデータを決定します。MDM Hub は、存続セルデータまたは優先されるセルが、2 つのセルの中でより適切であると判断します。最終的には、1 つの統合されたレコードに最適な存続セルデータが含まれ、最善データとなります。

存続性は、信頼が有効なカラムおよび信頼が有効ではないカラム両方に適用されます。MDM Hub は、2 つの異なるレコードのセルを比較するとき、優先度が高い方から、次の要因に基づいて存続性を決定します。

1. 信頼スコア。信頼スコアは、カラムで信頼が有効になっている場合にのみ適用されます。ROWID_OBJECT が高い方のデータが勝ちます。信頼スコアが同等、またはカラムで信頼が有効になっていない場合、MDM Hub は次の比較に進みます。
2. 相互参照レコードの SRC_LUD。相互参照 SRC_LUD 値がより新しいデータが勝ちます。SRC_LUD 値が等しい場合、MDM Hub は次の比較に進みます。
3. ベースオブジェクトの ROWID_OBJECT。ROWID_OBJECT 値は、数値の降順で評価されます。ROWID_OBJECT 値が等しい場合、MDM Hub は次の比較に進みます。
4. 相互参照の ROWID_XREF。ROWID_XREF 値は、数値の降順で評価されます。ROWID_XREF が高い方のデータが勝ちます。

ROWID_OBJECT の存続性

レコードがマージされるとき、MDM Hub は、どの ROWID_OBJECT が存続して、マージされたレコードの ROWID_OBJECT になるかを決定します。ROWID_OBJECT の存続性は、レコードがマージされる方法と場所に依存します。例えば、バッチマージジョブ後にレコードが存続する方法と、Informatica Data Director (IDD) でレコードが存続する方法は異なります。

MDM Hub は、次のシナリオごとに異なる方法で ROWID_OBJECT の存続性を処理します。

バッチマージジョブ

バッチマージジョブの実行時、MDM Hub はすべてのレコードの統合インジケータを考慮します。1つの新しいレコード (CONSOLIDATION_IND = 4) が照合され、統合レコード (CONSOLIDATION_IND = 1) とマージされる場合、統合 ROWID_OBJECT が優先されます。(CONSOLIDATION_IND = 4) の2つの新しいレコードが照合される場合、最小値の ROWID_OBJECT を持つレコードが優先されます。

Merge SIF API

Merge SIF API を使用してレコードをマージするときは、ターゲットの ROWID_OBJECT が優先されます。ターゲットは SIF 要求に表示される1番目のレコードです。

Informatica Data Director

IDD でリアルタイムにレコードをマージするときは、ターゲットの ROWID_OBJECT が優先されます。ターゲットは、IDD の [一致マージ比較] ビューの [マージのプレビュー] カラムに表示されるレコードです。

レコードをマージ用にキューに格納する場合、IDD に表示されている現在のレコードの ROWID_OBJECT が優先されます。ただし IDD での存続性は、ベースオブジェクトレコードの統合インジケータにも依存します。例えばレコード (CONSOLIDATION_IND = 4) が照合され、統合レコード (CONSOLIDATION_IND = 1) とマージされる場合、統合 ROWID_OBJECT が優先されます。

注: レコードがマージ用にキューに格納される場合、IDD は基礎となる _MTCH テーブルのエントリを使用します。そのため、存続する ROWID_OBJECT を正確に知るには、_MTCH テーブルを表示します。ROWID_OBJECT_MATCHED カラムに、存続する ROWID_OBJECT の一覧が表示されます。

ランディングプロセス

ここでは、Informatica MDM Hub のランディングプロセスに関する概念とタスクについて説明します。

ランディングプロセスについて

データのランディングは、Informatica MDM Hub にデータをロードするための最初の手順です。

ソースシステムおよびランディングテーブル

ランディングデータには、1つ以上のソースシステムから Informatica MDM Hub ランディングテーブルへのデータの転送が含まれます。

- ソースシステムは、Informatica MDM Hub にデータを供給する外部システムです。ソースシステムには、組織内にある、または外部から取得あるいは購入したアプリケーション、データストアおよびその他のシステムを使用できます。
- ランディングテーブルは、最初にソースシステムからロードしたデータを含む Hub ストアにあるテーブルです。

Informatica MDM Hub の外部で行われるランディングプロセス

ランディングプロセスは、Informatica MDM Hub の外部で、Hub ストアにランディングテーブルを直接取り込む外部バッチプロセスまたは外部アプリケーションを使用して実行されます。データを管理する後続のプロセスは、Informatica MDM Hub の内部で実行されます。

ランディングテーブルへの入力方法

ランディングテーブルには、以下の方法でデータを入力できます。

ロード方法	説明
外部バッチプロセス	ETL（抽出-変換-ロード）ツールまたはその他の外部プロセスによって、データがソースシステムから Informatica MDM Hub にコピーされます。バッチロードは、Informatica MDM Hub の外部で行われます。バッチロードの結果のみが、データが入力されたランディングテーブルの形式で Informatica MDM Hub に表示されます。 注: このプロセスは、選択した別の ETL ツールで処理されます。この ETL ツールは Informatica MDM Hub の製品スイートには含まれていません。
リアルタイム処理	オンラインのリアルタイムモードで外部アプリケーションからランディングテーブルにデータを入力できます。このようなアプリケーションは Informatica MDM Hub の製品スイートには含まれていません。

どのソースシステムでも、使用する方法は、それが特定のソースシステムからデータを取得する最も効率的な方法（または、唯一の方法）かどうかによって決まります。また、初期データロード（ビジネスデータを Hub ストアに初めてロードすること）にはバッチ処理がよく使用されます。ランディングテーブルに多数のレコードを入力する最も効率的な方法であるためです。

注: ランディングテーブルのデータは、ベースオブジェクトのロードプロセスが実行され、正常に完了するまで削除できません。

ランディングプロセスの管理

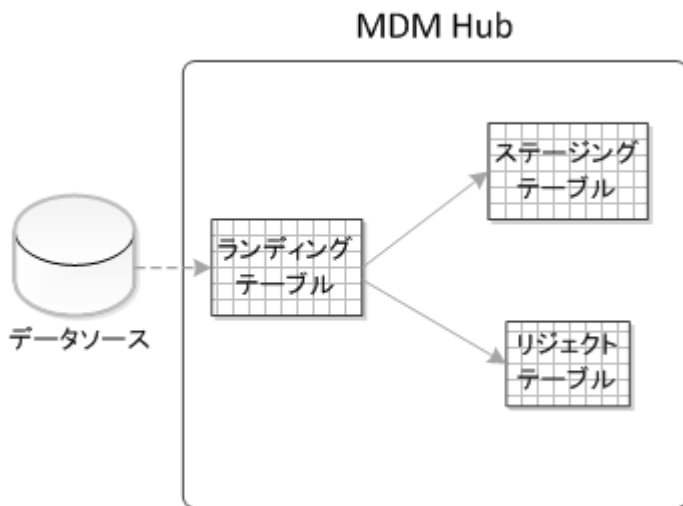
ランディングプロセスを管理する方法については、このマニュアルの以下のトピックを参照してください。

タスク	トピック
設定	第 16 章, 「ランディングプロセスの設定」 (ページ 295): - 「ソースシステムの設定」 (ページ 295) - 「ランディングテーブルの設定」 (ページ 298)
実行	ランディングプロセスの実行は、Informatica MDM Hub の外部で行われ、ランディングテーブルへデータを入力するために使用する方法 (「ランディングテーブルへの入力方法」 (ページ 270) を参照) に依存します。
アプリケーション開発	外部アプリケーションを使用してランディングテーブルへの入力を行う場合は、そのアプリケーションで使用する API に関する開発者向けマニュアルを参照してください。

ステージプロセス

MDM Hub ステージプロセスでは、ソースデータがランディングテーブルから、特定のベースオブジェクトに関連付けられたステージングテーブルに転送されます。完全なステージプロセスは MDM Hub で実行されます。

次の図に、ソースデータがランディングテーブルからステージングテーブルとリジェクトテーブルに転送される MDM Hub ステージプロセスを示します。



MDM Hub ステージプロセスを実行する前に、外部データソースからランディングテーブルにデータをロードします。ランディングテーブルとステージングテーブル間のマッピングを定義します。マッピングは、ランディングテーブルのソースカラムをステージングテーブルのターゲットカラムにリンクします。MDM Hub によってデータがステージングテーブルに移動される前に、データをクレンジングする場合は、マッピングでデータクレンジングを設定します。ステージジョブを実行すると、MDM Hub により、データがマッピングを基に、ランディングテーブルのカラムからステージングテーブルのカラムに転送されます。

ステージプロセス時、MDM Hub では 250 レコードのブロックを一度に 1 つ処理します。ブロックのレコードに問題がある場合、MDM Hub ではそのレコードを拒否テーブルに移動します。セルの値が長すぎる場合、またはレコードの更新日が現在の日付より後の場合、レコードは拒否されます。MDM Hub が拒否されたレコードを移動したら、MDM Hub はそのブロックの残りのレコードの処理を停止し、他のブロックに移行します。ステージプロセスが完了したら、ジョブを再度実行します。処理されなかったレコードは再度ピックアップされて処理されます。

ランディングテーブルのデータ履歴を保持できます。ステージングテーブルに対して監査証跡を有効にすると、ランディングテーブルのデータが RAW テーブルにアーカイブされます。MDM Hub は、ユーザーが設定したステージジョブの実行数または保持期間の間、ランディングテーブルデータを RAW テーブルに保持します。指定のステージジョブの実行数または保持期間に達すると、MDM Hub はソースオブジェクトのプライマリキーごとに 1 つのレコードを RAW テーブルに保持します。

ランディングテーブルの新しいレコードまたは更新されたレコードを識別するように MDM Hub を設定できます。ステージングテーブルに対して差分検出を有効にすると、MDM Hub で新しいレコードと更新されたレコードが処理され、変更されていないレコードは無視されます。

MDM Hub は、1 つのランディングテーブルから複数のステージングテーブルにデータを転送できます。ただし、ステージングテーブルは、それぞれ 1 つのランディングテーブルからのみデータを受け取ります。

ステージプロセスは、ロードプロセス用にデータを準備するプロセスです。ロードプロセスによって、ステージングテーブルのデータがターゲットベースオブジェクトにロードされます。

ロードプロセス

ここでは、Informatica MDM Hub のロードプロセスに関する概念とタスクについて説明します。

ロードプロセスについて

Informatica MDM Hub のロードプロセスでは、ステージングテーブルのデータを Hub ストア内の対応するターゲットテーブル（ベースオブジェクト）に移動します。

ロードプロセスでのステージングテーブルのデータの処理方法は、以下に基づいて決まります。

- ターゲットテーブルにすでに対応するレコードが存在するかどうか。存在する場合は、前回のロードプロセスの実行後にステージングテーブルのレコードが更新されたかどうか。
- 特定のカラムに対して信頼が有効になっているかどうか（ベースオブジェクトのみ）。有効になっている場合は、セルデータの信頼スコアがロードプロセスで計算されます。
- ロードできる有効なデータであるかどうか。有効でない場合は、ロードされずに却下されます。
- その他の構成設定。

ロードプロセスに関連付けられているテーブル

ベースオブジェクトに加えて、Hub ストアにあるステージングテーブル、相互参照テーブル、履歴テーブル、リジェクトテーブルがロードプロセスに関連付けられています。

次のテーブルがロードプロセスに関連付けられています。

ステージングテーブル

ステージングプロセス中にランディングテーブルから受け取られてコピーされたデータが含まれます。

相互参照テーブル

データ（ベースオブジェクトの各レコードのソースシステム）のリネージを追跡するために使用されます。Informatica MDM Hub では、ベースオブジェクトにロードされるソースシステムレコードごとに、次のものが含まれるレコードが相互参照テーブルに保持されます。

- レコードの提供元のシステムの識別子
- ソースシステム内のそのレコードのプライマリキー値
- そのシステムから提供された最新のセルの値

各ベースオブジェクトレコードには、1 つ以上の相互参照レコードがあります。

履歴テーブル

ベースオブジェクトに対して履歴が有効になっている場合に、レコードが更新または挿入されると、この情報がロードプロセスによって次の 2 つのテーブルに書き込まれます。

- ベースオブジェクト履歴テーブル
- 相互参照履歴テーブル

リジェクトテーブル

特定の理由でロードプロセスによって却下されたステージングテーブルのレコードが含まれます。却下されたレコードはベースオブジェクトにロードされません。リジェクトテーブルは、ステージングテーブル（stagingTableName_REJ と呼ばれる）に関連付けられています。却下されたレコードはロードジョブの実行後に調査できます。

Informatica MDM Hub は、レコードの却下理由を最初に検出したときにそのレコードを却下して、パフォーマンスを向上させます。リジェクトテーブルには Informatica MDM Hub がレコードを却下した理由が 1 つ示されます。Informatica MDM Hub がレコードを却下する理由が複数ある場合、リジェクトテーブルには Informatica MDM Hub が検出した最初の理由が示されます。

初期データロードおよび増分ロード

初期データロード（IDL）によって、新たに作成された空のベースオブジェクトに初めてデータがロードされます。

初期データロードでは、ステージングテーブルのすべてのレコードが、新規のレコードとしてベースオブジェクトに挿入されます。

ベースオブジェクトに対して初期データロードが行われた後のロードプロセスは、新規データまたは更新されたデータのみがベースオブジェクトにロードされることから、増分ロードと呼ばれます。

重複データは無視されます。

信頼設定および検証ルール

Informatica MDM Hub では、信頼と検証ルールを使用して最も信頼できるデータを特定します。

信頼設定

ベースオブジェクトのカラムのデータが複数のソースシステムから引き出されている場合は、Informatica MDM Hub で、異なるソースシステムからのカラムデータの相対的信頼性を比較するために信頼が使用されます。例えば、注文システムは、請求先住所のソースとしてダイレクトマーケティングシステムよりも信頼できる可能性があります。

信頼は、カラムレベルで有効にして設定します。例えば、注文システムの顧客名および請求システムの電話番号の信頼レベルを高く指定することができます。

以下の表に、統合される 2 つのベースオブジェクトレコードを示します。

ROWID_OBJECT	名前	電話番号
100	Doug McDougal Grp	1-555-901-4670
200	The Doug McDougal Group	201-10810

以下の表に、各カラムの計算された信頼スコアを示します。

ROWID_OBJECT	名前	電話番号
100	62	56（[電話番号] カラムで優先される信頼スコア）
200	71（[名前] カラムで優先される信頼スコア）	37

信頼スコアが最も高いデータが統合されたレコードに残ります。以下の表に、統合されたレコードを示します。

ROWID_OBJECT	名前	電話番号
100	The Doug McDougal Group	1-555-901-4670

信頼は、ソースシステム、変更履歴、およびその他のビジネスルールに基づいて、各セルに関連付けられた相対的な信頼度を測定するメカニズムです。信頼では、セルデータの品質と経過時間、およびその信頼性が時間の経過と共にどの程度減衰（低下）するかが考慮されます。信頼は、2つのレコードを統合する際の存続性を判断する場合や、ソースシステムの更新を信頼してマスタレコードに反映してかまわないかどうかを判断する場合に使用されます。

特定の値が正しい場合、データスチュワードは、その値で計算された信頼設定を上書きすることができます。また、データスチュワードは、ベースオブジェクトのレコードに値を直接入力することもできます。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

検証ルール

多くの場合、信頼は検証ルールと組み合わせて使用され、場合によっては設定されている条件およびアクションに従って信頼スコアがダウングレード（減点）されます。

データが検証ルールで指定された基準を満たすと、そのデータの信頼値が検証ルールで指定された割合だけダウングレードされます。以下に例を示します。

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'

【最小信頼度の保持】 フラグがカラムに対して有効になっている（チェックボックスが選択されている）場合は、カラムの最小信頼度設定よりも下に信頼をダウングレードすることはできません。

ロードプロセスのランタイム実行フロー

この節では、ロードプロセスの実行中に、構成済みの設定および処理されるデータの特性に基づいて発生する処理について、詳しく説明します。また、Informatica MDM Hub ロードプロセスのデフォルトの挙動について説明します。増分ロードの場合は、RowID でのロードによって、ロード、一致、およびマージの各プロセスを効率化できます。

レコードがすでに存在するかどうかの確認

ロードプロセスでは、Informatica MDM Hub がレコードに同じソースシステムからの既存のレコードと同じプライマリキーがないかどうかを最初にチェックします。ステー징テーブルの各レコードがターゲットテーブルのレコードと比較されて、レコードがターゲットテーブルに存在していないかが確認されます。

次に行われる処理は、この比較の結果によって異なります。

ロード操作	説明
ロードの挿入	ステージングテーブルのレコードがターゲットテーブルにまだ存在していない場合は、Informatica MDM Hub がその新しいレコードをターゲットテーブルに挿入します。
ロードの更新	<p>ステージングテーブルのレコードがターゲットテーブルにすでに存在する場合は、Informatica MDM Hub が適切なアクションを実行します。ターゲットのベースオブジェクトがステージングテーブルのレコードのデータで更新される場合は、ロードの更新が行われます。ロードプロセスでは、レコードがソースシステムから前回提供された後に更新されている場合にのみ、そのレコードが更新されます。</p> <p>ロードの更新は、Informatica MDM Hub の現在の構成設定およびステージングテーブルの各レコードのデータの特性によって左右されます。例えば、更新の強制が有効になっている場合は、すでにロードされているかどうかにかかわらずレコードが更新されます。</p>

ロードプロセスでは、ロードの更新が先に実行され、その後にロードの挿入が実行されます。

ロードの挿入

MDM Hub は、ロードの挿入プロセス中に次の手順を実行します。

1. セルデータの信頼を計算します。
2. 検証ルールを実行します。
3. 外部キーのルックアップを実行します。
4. ROWID_OBJECT 値を作成します。
5. レコードをターゲットベースオブジェクトおよびその他のテーブルに挿入します。

ロード挿入中の動作は、ターゲットベースオブジェクトおよびその他の要因によって異なります。

ロードの挿入とターゲットベースオブジェクト

ステージングテーブルのレコードにロードの挿入を実行する手順

- ロードプロセスにより、新しいレコードに一意の ROWID_OBJECT 値が生成されます。
- ロードプロセスでは、外部キールックアップが実行され、参照整合性を維持するために必要な外部キー値が置換されます。
- ロードプロセスによってレコードがベースオブジェクトに挿入され、この新しいレコードに、生成された ROWID_OBJECT 値（ベースオブジェクトでのこのレコードのプライマリキーになります）、外部キールックアップ値、および NULL 値を含むステージングテーブルのすべてのカラムデータ（PKEY_SRC_OBJECT を除く）がコピーされます。
ベースオブジェクトには、同じオブジェクトに複数のレコードが存在する場合があります（例えばソースシステム A からのレコードとソースシステム B からのレコード）。Informatica MDM Hub では、両方の新しいレコードに新規フラグが設定されます。
- ベースオブジェクト内の新しいレコードごとに、ロードプロセスによって、関連付けられている未処理テーブルに ROWID_OBJECT 値が挿入されます。これにより、トークン化プロセスで一致キーが再生成されません。
- ロードプロセスにより、ベースオブジェクトの新しいレコードごとに、その CONSOLIDATION_IND が 4（一致準備完了）に設定されます。これにより、新しいレコードをベースオブジェクトの他のレコードと一致させることができます。

- ロードプロセスにより、ベースオブジェクトに関連付けられた相互参照テーブルにレコードが挿入されます。ロードプロセスにより、相互参照テーブルのプライマリキー値が生成され、この新しいレコードに、生成されたキー、ソースシステムの識別子、およびステージングテーブルのカラム（PKEY_SRC_OBJECTを含む）がコピーされます。

注: ベースオブジェクトにソースシステムからのプライマリキー値は含まれません。代わりに、生成された ROWID_OBJECT 値がベースオブジェクトのプライマリキーとなります。ソースシステムからのプライマリキー（PKEY_SRC_OBJECT）は、相互参照テーブルに格納されます。

- ベースオブジェクトに対して履歴が有効になっている場合は、ロードプロセスにより履歴テーブルおよび相互参照履歴テーブルにレコードが挿入されます。
- ベースオブジェクト内の 1 つ以上のカラムに対して信頼が有効になっている場合は、ロードプロセスにより信頼アルゴリズムをサポートする制御テーブルにもレコードが挿入され、信頼された各セルの信頼の要素および検証ルールに、信頼の計算に使用される値が入力されます。この情報は、後から信頼を計算する必要が生じた場合に使用できます。
- ベースオブジェクトに対して [ロード時に一致トークンを生成する] が有効になっている場合は、ロードプロセス完了後に自動的にトークン化プロセスが開始されます。

ロードの更新

MDM Hub は、ロードの更新プロセス中に次の手順を実行します。

1. レコードが変更されているかどうかを確認します。
2. セルデータの信頼を計算します。
3. 検証ルールを実行します。
4. 外部キーのルックアップを実行します。
5. ターゲットベースオブジェクトおよびその他のテーブルで、ターゲットレコードを更新します。

ロード更新中の動作は、ターゲットベースオブジェクトおよびその他の要因によって異なります。

ロードの更新とターゲットベースオブジェクト

ターゲットベースオブジェクトでのロードの更新時に、次の変更が発生します。

1. デフォルトでは、ステージングテーブル内のレコードごとに、ロードプロセスで LAST_UPDATE_DATE カラムの値と、関連する相互参照テーブル内のソース最終更新日（SRC_LUD）が比較されます。
 - ステージングテーブル内のレコードが、ソースシステムから前回レコードが供給された後に更新されている場合は、ロードプロセスによってロードの更新が続行されます。
 - ステージングテーブル内のレコードが、ソースシステムから前回レコードが供給された後に変更されていない場合、日付が同じで信頼が有効になっていなければロードプロセスでそのレコードが無視され（アクションが実行されず）、重複レコードであればそのレコードは却下されます。
管理者は、このデフォルトの動作を変更して、ロードプロセスでこの LAST_UPDATE_DATE のチェックをバイパスし、すでにロードされているかどうかに関係なくレコードが強制的に更新されるようにすることができます。
2. ロードプロセスでは、外部キールックアップが実行され、参照整合性を維持するために必要な外部キー値が置換されます。
3. ターゲットベースオブジェクトに信頼が有効なカラムがある場合は、ロードプロセスで以下の処理が行われます。
 - 更新対象のレコードにある信頼が有効なカラムごとに、そのカラムの信頼設定に基づいて信頼スコアが計算されます。
 - 検証ルール（定義されている場合）が適用されて、信頼スコアが適宜ダウングレードされます。

4. ロードプロセスでベースオブジェクト内のレコードが更新される場合、相互参照テーブル、履歴テーブル、およびその他制御テーブル内の関連するレコードが適宜更新されます。ロードプロセスでは、ベースオブジェクトに関連付けられている未処理テーブルに対してレコードの ROWID_OBJECT 値の挿入も行われます。これにより、トークン化プロセスで一致キーが再生成されます。ベースオブジェクトは統合インジケータ値を保持します。ロードプロセスでは、以下のルールに従ってベースオブジェクト内のターゲットレコードが更新されます。
 - ステージングテーブルレコード内のセルの信頼スコアが、ターゲットベースオブジェクトレコード内の対応するセルの信頼スコアよりも高い場合は、ターゲットレコード内のセルが更新されます。
 - ステージングテーブルレコード内のセルの信頼スコアが、ターゲットベースオブジェクトレコード内の対応するセルの信頼スコアよりも低い場合は、ターゲットレコード内のセルは更新されません。
 - ステージングテーブルレコード内のセルの信頼スコアが、ターゲットベースオブジェクトレコード内の対応するセルの信頼スコアと同じである場合、またはコラムで信頼が有効になっていない場合は、LAST_UPDATE_DATE が新しい方のレコードのセル値が優先されます。
 - ステージングテーブルレコードの LAST_UPDATE_DATE の方が新しい場合は、ターゲットベースオブジェクトレコード内の対応するセルが更新されます。
 - ベースオブジェクト内のターゲットレコードの LAST_UPDATE_DATE の方が新しい場合は、セルは更新されません。
5. ベースオブジェクトに対して [ロード時に一致トークンを生成する] が有効になっている場合は、ロードプロセス完了後に自動的にトークン化プロセスが開始されます。

外部キーのルックアップ

バッチロードまたは PUT API がレコードを挿入または更新すると、Informatica MDM Hub はステージングテーブルのルックアップ設定を使用してソースシステムの外部キーを Informatica MDM Hub の外部キーに変換します。

参照整合性制約の無効化

初期ロード/更新の実行時（つまりリアルタイムの同時アクセスがない場合）に、パフォーマンスを向上させるために、ベースオブジェクトに対する参照整合性の制約を無効にすることができます。

未定義のルックアップ

ルックアップテーブルやルックアップコラムを生成せずに子オブジェクトのルックアップを定義する場合、データを正しくロードするためには、ロードプロセスを実行する前に子オブジェクトに対してステージプロセスを繰り返す必要があります。

NULL の外部キーの許可

ステージングテーブルのコラムを設定する場合、ターゲットベースオブジェクトに対して、NULL 外部キーを許可するかどうかを指定できます。[NULL の外部キーを許可する] ステージングテーブルコラムのプロパティで、NULL の外部キー値を許可するかどうかを指定します。

注: MDM Hub では、空の文字列は、その空の文字列に関与するデータベースタイプに関係なく、NULL 値と同等です。

デフォルトでは、[NULL の外部キーを許可する] チェックボックスはチェック解除されており、NULL 外部キーは許可されません。ロードプロセスは以下の処理を行います。

- 有効なルックアップ値のレコードを受け入れます。
- NULL 外部キーがあるレコードを却下します。
- 無効な外部キーの値を持つレコードを却下します。

[NULL の外部キーを許可する] を有効に（選択）した場合、ロードプロセスは以下の処理を行います。

- 有効なルックアップ値のレコードを受け入れます
- NULL 外部キーを持つレコードを受け入れます（また、これらのレコードに対するロードの挿入とロードの更新を許可します）
- 無効な外部キーの値を持つレコードを却下します。

ロードプロセスは、受け入れたレコードに対してのみ、ロードの挿入とロードの更新を許可します。却下されたレコードはターゲットテーブルにロードされずにリジェクトテーブルに挿入されます。

注: 初期データロードの実行中にのみ、ターゲットベースオブジェクトが空の場合、ロードプロセスは NULL 外部キーを許可します。

ロードジョブで却下されたレコード

ロードプロセスの実行中にステージングテーブルのレコードが却下される理由は以下のとおりです。

- LAST_UPDATE_DATE カラムに将来の日付または NULL の日付がある。
- LAST_UPDATE_DATE が 1900 よりも小さい。
- NULL 値がステージングテーブルの PKEY_SRC_OBJECT にマップされている。
- PKEY_SRC_OBJECT に重複がある。
- HUB_STATE_IND フィールドに無効な値がある（状態が有効なオブジェクトのみ）。
- 無効な外部キーまたは NULL 外部キー。

却下されたレコードはベースオブジェクトにロードされません。却下されたレコードはロードジョブの実行後に調査できます。

注: レコードを却下するには、ロードプロセスからランディングテーブルに戻るトレーサビリティが必要です。ステージングテーブルからレコードをロードする場合、関連するランディングテーブルから対応レコードが削除されていると、ロードプロセスはそのレコードを却下テーブルに挿入します。

ロードプロセスに関するその他の考慮事項

ここでは、ロードプロセスのその他の考慮事項について説明します。

ロードプロセスでの親子レコードの処理方法

子テーブルに親テーブルからの生成キーが含まれている場合は、ロードプロセスで適切なプライマリキー値が親テーブルから子テーブルにコピーされます。例えば、以下のデータがありますとします。

親テーブル:

PARENT_ID	FNAME	LNAME
101	Joe	Smith
102	Jane	Smith

子テーブル: 親 PKEY_SRC_OBJECT とのリレーションあり

ADDRESS	CITY	STATE	FKEY_PARENT
1893	my city	CA	101
1893	my city	CA	102

この例では、ROWID_OBJECT、PKEY_SRC_OBJECT、またはテーブルルックアップ用の一意のカラムを指すリレーションを設定することができます。

状態が有効なベースオブジェクトのロード

ロードプロセスには、状態が有効なベースオブジェクトのレコードを処理する場合の特別な考慮事項があります。

注: ロードプロセスでは、HUB_STATE_IND カラムに無効な値が入っているステー징テーブルのレコードがすべて却下されます。

一致トークンの生成（オプション）

一致プロセスを実行する前に、一致トークンを生成する必要があります。スキーママネージャでベースオブジェクトを設定する際に、ロードジョブ完了直後に一致トークンを生成するか、一致ジョブを実行するまでデータのトークン化を遅らせるかを指定できます。[ロード時に一致トークンを生成する] チェックボックスの設定によって、トークン化プロセスがいつ行われるかが決まります。

トークン化プロセス

トークン化プロセスでは、マッチトークンを生成し、ベースオブジェクトに関連付けられたマッチキーテーブルにそれらを格納します。マッチトークンは、以降の一致プロセスで、一致する候補を特定するために使用されます。

一致トークンおよび一致キー

一致トークンは、ベースオブジェクトレコードのデータのエンコード済み表現およびエンコードされていない表現です。一致トークンには以下のものがあります。

- 一致キー。固定長です。あいまい一致ベースオブジェクトのあいまい一致キーのすべてのカラムから作成されたエンコード済みの値で構成される圧縮文字列です。一致キーは、同じ一致キー値を持つ関連する差異など、名前またはアドレスの語や数値の組み合わせを含みます。
- エンコードされていない文字列は、一致カラムからのフラットデータで構成されます（あいまい一致キーおよびすべてのあいまい一致カラムならびに完全一致カラム）。

一致キーテーブル

一致キーテーブルには、MDM Hub でベースオブジェクトレコードに対して生成された一致トークンと一致キーが含まれます。一致キーテーブルは各ベースオブジェクトに関連付けられています。

一致キーテーブル名の形式は C_<base object name>_STRP です。例えば、ベースオブジェクト名が PARTY の場合、関連付けられた一致キーテーブルの名前は C_PARTY_STRP になります。トークン化プロセスでは、ベースオブジェクトの各レコードに対して一致キーテーブルに 1 つ以上のレコードが生成されます。

一致およびマージプロセスが完了した後に、無効な一致トークンを一致キーテーブルから削除する必要があります。また、再トークン化中に、無効な一致トークンは削除され、有効な一致トークンに置き換えられます。無効な一致トークンを削除する操作は、MDM Hub のパフォーマンスに影響を与えることがあります。IBM DB2 環境でパフォーマンスを向上させるには、一致トークンを削除する代わりに [無効] とマークする `cmx.server.stripDML.useUpdate=true` プロパティを設定します。

次の表に、一致キーテーブルの重要な列を示します。

カラム名	データ型 (サイズ)	説明
ROWID_OBJECT	CHAR (14)	この一致トークンが生成されたレコードを識別します。
SSA_KEY	CHAR (8)	このレコードの一致キー。関連するベースオブジェクトレコードのあいまい一致キーカラム (名前、住所、組織名など) の値をエンコードしたものです。文字列は、名前や住所に含まれる単語と数字の組み合わせから構築された固定長の圧縮エンコード値で構成されます。
SSA_DATA	VARCHAR2 (500)	関連するベースオブジェクトレコードで定義される連結された一致カラムを、エンコードされていないプレーンテキストの文字列で表したものです。連結された一致カラムには、あいまい一致キーとすべてのあいまい一致カラムおよび完全一致カラムが含まれます。

一致キーテーブル内の各レコードには、一致トークン (SSA_KEY と SSA_DATA の両方のデータ) が含まれています。

一致キーの例

生成される一致キーは、各自で構成した一致設定およびベースオブジェクト内のデータの特性によって異なります。

以下の例は、あいまい一致/検索ストラテジを使用して文字列から生成された一致キーを示しています。

レコード内の文字列	生成された一致キー
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

この例では、文字列 BETH O'BRIEN と LIZ O'BRIEN の一致キー値が同じになります (PCOG\$\$\$\$)。一致プロセスでは、一致候補が検索される際にこれらが一致候補と見なされます。

あいまい一致ベースオブジェクトにのみ適用されるトークン化プロセス

トークン化プロセスはあいまい一致ベースオブジェクトにのみ適用され、完全一致ベースオブジェクトには適用されません。あいまい一致ベースオブジェクトの場合にトークン化プロセスを実行すると、Informatica MDM Hub で、ある程度あいまいに行を一致させることができます。同一である必要はなく、十分に類似していれば一致と見なされます。

トークン化プロセスの主要な概念

この節では、トークン化プロセスに適用される主要な概念について説明します。

一致トークンの生成

MDM Hub では、バッチジョブまたは SIF API が実行されると一致トークンが生成または更新されます。一致トークンは一致キーテーブルに保存され、一致プロセスのために最新である必要があります。MDM Hub では、一致トークンは一致プロセスと無関係に保持されます。

バッチジョブまたは SIF API が実行されると、ベースオブジェクトレコードは「未処理」とマークされることがあります。「未処理」とマークされたベースオブジェクトレコードには一致トークンが生成または更新されません。

ベースオブジェクトレコードは、次の条件をすべて満たす場合に「未処理」とマークされます。

- 更新がベースオブジェクトの一致カラムに影響を与える。
- 更新後の一致カラムのベストバージョン オブ トゥールズ (BVT) が古い値と異なる。

未処理テーブル

すべてのベースオブジェクトには `<base object name>_DRTY` という名前の関連付けられた未処理テーブルがあり、これはシステムテーブルです。この未処理テーブルには ROWID_OBJECT カラムがあり、一致トークンを生成する必要があるベースオブジェクトレコードを識別します。MDM Hub は、一致トークンを一致キーテーブルに保存します。

未処理テーブルの一意の ROWID_OBJECT ごとに、トークン化プロセスにより一致トークンが生成され、その後未処理テーブルがクリーンアップされます。

次のバッチプロセスのシーケンス中に、MDM Hub が未処理テーブルを更新します。

1. ロード。MDM Hub が新しいレコードをロードするか、既存のレコードを更新します。MDM Hub が未処理テーブルに、新しいレコードまたは一致カラムの値が変更された更新済みレコードの ROWID_OBJECT 値を入力します。
2. トークン化。MDM Hub が一致キーを生成します。MDM Hub が未処理テーブルから、トークン化されたレコードの ROWID_OBJECT 値を削除します。
3. 一致。MDM Hub が一致を識別します。未処理テーブルは元のままです。
4. 統合。MDM Hub が一致したレコードを統合します。MDM Hub が未処理テーブルに、新しいレコードまたは一致カラムの値が変更された更新済みレコードの ROWID_OBJECT 値を入力します。
5. トークン化。MDM Hub が一致キーを生成します。MDM Hub が未処理テーブルから、トークン化されたレコードの ROWID_OBJECT 値を削除します。

あいまい一致ベースオブジェクトでのキータイプおよびキー幅

あいまい一致ベースオブジェクトに対しては、以下の設定に基づいて一致キーが生成されます。

プロパティ	説明
キータイプ	このベースオブジェクトについてトークン化されるプライマリタイプの情報（Person_Name、Organization_Name、または Address_Part1）を特定します。一致プロセスでは、名前と住所の特性に関する情報を使用して、一致キーが生成され、検索が実施されます。利用可能なキータイプは、使用されるポピュレーションセットによって異なります。
キー幅	あいまい一致キーの分析の完全性、予想される一致候補の数、およびキーが消費するディスク容量を決定します。指定可能なキー幅は、限定、標準、拡張、および優先です。

一致キーは、データ内のエラー、差異、語句の入れ替えに対応できる必要があるため、Informatica MDM Hub では名前、住所、または組織ごとに複数の一致トークンが生成されます。ベースオブジェクトレコードごとに生成されるキーの数は、使用するデータおよび一致キー幅によって異なります。

一致キーの分布およびホットスポット

スキーママネージャの「一致/マージ設定の詳細」ペインにある「一致キーの分布」タブでは、一致キーテーブル内の一致キーの分布を調べることができます。このツールは、データ内の潜在的なホットスポットの特定に役立ちます。ホットスポットでは、一致キーが集中しているために一致過多が起きる可能性があり、関係のない一致を含む多数の一致が一致プロセスで生成されます。

トークン化率

変更されたレコードの割合が指定された割合を超えたときは必ずトークン化プロセスを繰り返すように一致プロセスを設定できます（ベースオブジェクトの詳細プロパティとして設定）。

トークン化およびマージプロセスのパフォーマンスの最適化

MDM Hub 環境のトークン化およびマージプロセスのパフォーマンスを最適化するには、cmxcleanse.properties ファイルに最適化プロパティを追加します。

- 次のディレクトリにある cmxcleanse.properties ファイルを開きます。
UNIX の場合: <infamdm installation directory>/hub/cleanse/resources
Windows の場合: <infamdm installation directory>\hub\cleanse\resources
- トークン化およびマージプロセスのパフォーマンスを最適化するには、cmxcleanse.properties ファイルに最適化プロパティを追加します。

次の表に、設定可能な最適化プロパティを示します。

プロパティ	説明
cmx.server.stripDML.useUpdate	IBM DB2 のみ。true に設定すると、必要でない一致トークンを一致キーテーブルから削除する代わりに、ステータスを無効に更新するように MDM Hub を設定します。デフォルトは false です。 無効な一致トークンのステータスの更新は、無効な一致トークンを削除する操作よりも効率的です。 注: このプロパティを true に設定した場合、無効なレコードを削除するには、定期的に一致キーテーブルをクリーンアップする必要があります。
cmx.server.stripDML.blockSize	MDM Hub が各ブロックで処理するレコード数。デフォルトは 100 です。
cmx.server.stripDML.noOfThreadsForInsert	MDM Hub が一致キーテーブルにレコードを挿入するために使用するスレッド数。デフォルトは 50 です。
cmx.server.stripDML.noOfThreadsForUpdate	MDM Hub が一致キーテーブルのレコードを更新するために使用するスレッド数。デフォルトは 30 です。
cmx.server.stripDML.noOfThreadsForDelete	MDM Hub が一致キーテーブルからレコードを削除するために使用するスレッド数。デフォルトは 30 です。

注: さまざまな操作のブロックサイズおよびスレッド数は、MDM Hub 環境の要件に基づいて設定します。

一致キーテーブルのクリーンアップ

一致キーテーブルに有効でない一致トークンが含まれている場合、一致キーテーブルをクリーンアップする必要があります。

一致キーテーブルをクリーンアップするには、次のいずれかのタスクを実行します。

- すべての一致トークンを再生成する。
- CleanStrp バッチジョブを実行する。
- バックグラウンドクリーンアッププロセスを実行する。

一致トークンの再生成

一致キーテーブルからすべての無効な一致トークンを削除するには、ベースオブジェクトのすべての一致トークンを再生成できます。

1. Hub コンソールで、バッチビューアツールを起動します。
2. すべての一致トークンを再生成するベースオブジェクトを展開します。
3. **一致トークンの生成**を展開します。
一致トークンの生成に使用できる、ベースオブジェクトに関連付けられたバッチジョブが表示されます。
4. 一致トークンの生成に使用するバッチジョブを選択します。
一致トークンの生成バッチジョブのプロパティが表示されます。
5. **すべての一致トークンを再生成** オプションを有効にします。

6. **【バッチの実行】** をクリックします。

MDM Hub によりベースオブジェクト全体の一致トークンが再生成されます。

CleanStrp バッチジョブの実行

CleanStrp バッチジョブを実行して、無効な一致トークンを一致キーテーブルから削除できます。CleanStrp バッチジョブでは、invalid_ind=1 が設定されている一致トークンが識別され、それらの一致トークンが削除されます。さらに、ステータスが invalid_ind=1 になっている一致トークンが削除された後、一致キーテーブルのインデックスが再構築されます。

1. Hub コンソールで、バッチビューアツールを起動します。
2. すべての無効な一致トークンをクリーンアップするベースオブジェクトを展開します。
3. **【CleanStrp】** を展開します。
一致キーテーブルから無効な一致トークンをクリーンアップするために使用できるバッチジョブが表示されます。
4. 一致トークンのクリーンアップに使用するバッチジョブを選択します。
CleanStrp バッチジョブのプロパティが表示されます。
5. **【バッチの実行】** をクリックします。
ベースオブジェクトに関連付けられた一致キーテーブルから無効な一致トークンがクリーンアップされます。

バックグラウンドクリーンアッププロセスの実行

バックグラウンドクリーンアッププロセスを実行して、無効な一致トークンを一致キーテーブルから削除できます。バックグラウンドクリーンアップバッチプロセスでは、invalid_ind=1 が設定されている一致トークンが識別され、それらの一致トークンが削除されます。バックグラウンドクリーンアッププロセスでは、一致トークンの削除後に一致キーテーブルのインデックスは再構築されません。

- ▶ バックグラウンドクリーンアッププロセスの実行を有効にするには、関連する Hub サーバプロパティをプロパティファイルに追加します。
 - a. 次のディレクトリにある cmxserver.properties ファイルを開きます。
UNIX の場合: <infamdm install directory>/hub/server/resources
Windows の場合: <infamdm install directory>\hub\server\resources

b. 次のプロパティを cmxserver.properties ファイルに追加します。

プロパティ	説明
cmx.server.strp_clean.execution_mode	<p>一致キーテーブルのバックグラウンドクリーンアッププロセスの操作範囲を設定します。</p> <p>操作範囲に次の値のいずれかを指定します。</p> <ul style="list-style-type: none"> - ALL。すべての登録されているオペレーショナル参照ストアのすべての一致キーテーブルから、invalid_ind=1 がある一致トークンを削除します。 - CONFIGURED_ORs。指定したオペレーショナル参照ストアのすべての一致キーテーブルから invalid_ind=1 がある一致トークンを削除します。操作範囲を CONFIGURED_ORs に設定する場合は、cmx.server.strp_clean.orc プロパティを cmxserver.properties ファイルに追加します。 - CONFIGURED_STRP。特定のオペレーショナル参照ストアの特定のベースオブジェクトの一致キーテーブルから invalid_ind=1 がある一致トークンを削除します。操作範囲を CONFIGURED_STRP に設定する場合は、cmx.server.strp_clean.strp プロパティを cmxserver.properties ファイルに追加します。
cmx.server.strp_clean.orc	<p>無効な一致トークンを削除するために、バックグラウンドクリーンアッププロセスを実行する必要があるオペレーショナル参照ストアの名前を指定します。例えば、invalid_ind=1 がある一致トークンを cmx_orc1 および cmx_orc2 のすべての一致キーテーブルから削除するには、cmx.server.strp_clean.orc=cmx_orc1,cmx_orc2 を追加します。</p>
cmx.server.strp_clean.strp	<p>一致キーテーブルをクリーンアップするために、バックグラウンドクリーンアッププロセスを実行する必要があるオペレーショナル参照ストアとベースオブジェクトの組み合わせを指定します。例えば、invalid_ind=1 がある一致トークンを、cmx_orc1 の BO1 および cmx_orc2 の BO2 の一致キーテーブルから削除するには、cmx.server.strp_clean.strp=cmx_orc1.C_BO1,cmx_orc2.C_BO2 を追加します。</p>
cmx.server.strp_clean.delete_records_count	<p>一致キーテーブルからクリーンアップするレコード数を指定します。</p>
cmx.server.strp_clean.retry_sec	<p>MDM Hub が、一致キーテーブルで無効な一致トークンがあるレコードを検索する時間（秒）を指定します。デフォルトは 60 です。</p>
cmx.server.strp_clean.threads_count	<p>MDM Hub が、一致キーテーブルで無効な一致トークンがあるレコードを検索するときに使用するスレッド数を指定します。デフォルトは 20 です。</p>

一致プロセス

ベースオブジェクトのレコードを統合する前に、Informatica MDM Hub で、重複（一致）している可能性があるレコードを特定する必要があります。

一致プロセスでは、照合ルールに基づいて以下の作業が実行されます。

- ベースオブジェクト内にある重複している可能性がある（同一または類似の）レコードを特定する
- 類似性が高く自動的に統合できるレコードと、統合する前にデータスチュワードによる手動での確認が必要なレコードを決定する

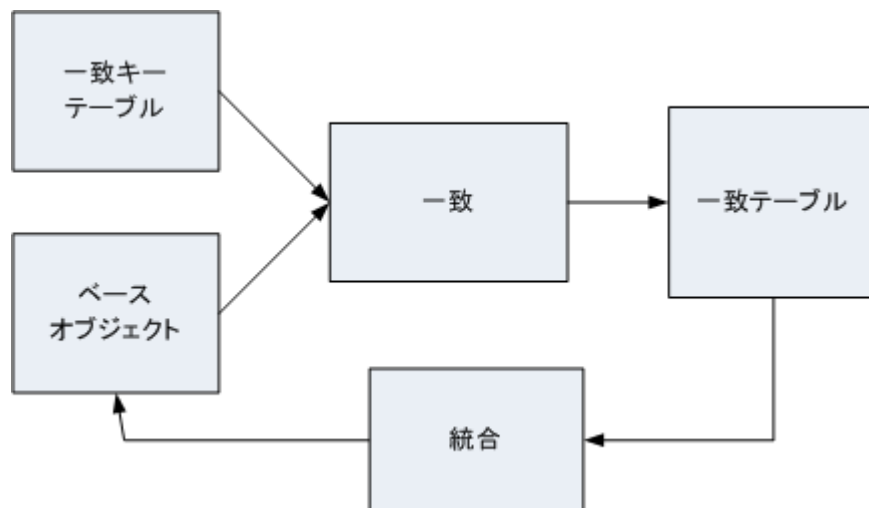
Informatica MDM Hub には、一致プロセスでレコードを比較して重複を特定するための方法が大きく分けて 2 つあります。

- あいまい一致は、ベースオブジェクトのレコードを一致させるときに Informatica MDM Hub で最もよく使用される方法です。あいまい一致では、レコード間の十分な類似点を探し、データパターンの差異（スペルの誤り、文字の入れ替え、単語の連結や分割、省略、切り詰め、音声変化など）の可能性を考慮した確率的な手法で一致を判定します。
- 完全一致は、一致カラムの値が同じであるレコードを一致させる方法であり、それほど頻繁には使用されません。完全ストラテジの方が処理は高速ですが、完全一致では、データに誤りがあると一致が見落とされる可能性があります。

どちらの方法が適しているかは、データの特性、データについて把握できている情報、一致や統合に関する具体的な要件などによって異なります。

一致プロセス中に、Informatica MDM Hub でベースオブジェクトのレコードの類似性が比較されます。一致プロセスで 2 つのレコード間に十分な類似点（同一または類似する一致）が見つかり、2 つのレコードが相互に重複している可能性があると思なされると、以下の処理が行われます。

- 一致するレコードのペアに対する ROWID_OBJECT 参照を、一致を特定した一致ルール、および一致するレコードを自動統合の対象に含めるかどうかに関する情報とともに一致テーブルに入力する。



- それらのレコードの統合インジケータを 2（統合準備完了）に変更して統合のフラグを設定する

一致ルール

一致ルールは、ベースオブジェクト内の 2 つのレコードが重複している可能性があるかどうかを Informatica MDM Hub で判別するための条件を定義したものです。一致ルールは、一致カラムまたはプライマリキーに基づくことができます。

以下の表に、作成できる一致ルールのタイプを示します。

ルールタイプ	説明
完全一致	値が完全に一致するか、NULL が NULL に一致するなど、特殊なケースが完全に一致する必要があります。完全一致ルールの処理では、データベースに対して SQL 文が使用されます。
あいまい一致	値は完全には一致しませんが、照合する値と類似しています。一致するかどうかは、いくつかの値で共有される一致トークンによって判断されます。これはポピュレーションによって変わります。例えば、一致目的が名前の場合、英語を話すポピュレーションでは、Robert、Rob、Bob の一致トークン値を同じです。あいまい一致ではあいまい一致キーを使用します。これは、すべての一致トークンの範囲に対応するインデックスです。
フィルター一致	あいまい一致キーを使用して一致候補を特定した後、候補に対して完全一致を実行します。 注: データベースサーバーに関連してパフォーマンスが制約される場合、完全一致ルールではなくフィルター一致ルールを使用することを検討してください。

次の表に、一致カラムおよびプライマリキーに基づく一致ルールを示します。

一致ルールの基本	説明
一致カラム	一致カラムとして定義したカラム（姓、名、住所 1、住所 2 など）の値に基づいてベースオブジェクトレコードを一致させるために使用されます。一致を特定するための最も一般的な方法です。
プライマリキー	レコードに同じプライマリキーを使用する 2 つのシステムのレコードを一致させるために使用されます。プライマリキーの一致は、すばやく、かつ非常に正確です。ただし、2 つのソースシステムが同一のプライマリキーを使用することは一般的ではありません。

同じベースオブジェクトに対して、一致カラムとプライマリキーの両方を使用できます。

完全一致およびあいまい一致ベースオブジェクト

ベースオブジェクトは、以下の一致タイプのいずれかを使用するように設定されます。

ベースオブジェクトのタイプ	説明
完全一致ベースオブジェクト	完全一致カラムのみを使用できます。
あいまい一致ベースオブジェクト	あいまい一致カラムと完全一致カラムの両方を含めることができます。 <ul style="list-style-type: none">- あいまい一致のみ- 完全一致のみ- または、あいまい一致と完全一致の組み合わせ

ベースオブジェクトのタイプによって、定義できる一致のタイプおよび一致カラムのタイプが決まります。ベースオブジェクトのタイプは、ベースオブジェクトに対して選択された一致/検索ストラテジによって決まります。

一致プロセスで使用されるサポートテーブル

一致プロセスは以下のサポートテーブルを使用します。

テーブル	説明
一致キーテーブル	すべてのベースオブジェクトレコードに対して生成された一致キーを含みます。一致キーテーブルは以下の命名規則を使用します。 <i>C_baseObjectName_STRP</i> ここで、 <i>baseObjectName</i> はベースオブジェクトのルート名です。 例: C_PARTY_STRP
一致テーブル	このベースオブジェクトに対して実行された一致プロセスの結果であるベースオブジェクトの一致レコードのペアを含みます。 一致テーブルは以下の命名規則を使用します。 <i>C_baseObjectName_MTCH</i> ここで、 <i>baseObjectName</i> はベースオブジェクトのルート名です。 例: C_PARTY_MTCH
一致フラグ監査テーブル	マージマネージャで、自動マージ用に手動一致レコードをキューに追加したユーザーのユーザー ID を含みます。 一致フラグ監査テーブルは以下の命名規則を使用します。 <i>C_baseObjectName_FHMA</i> ここで、 <i>baseObjectName</i> はベースオブジェクトのルート名です。 一致フラグ監査テーブルがこのベースオブジェクトに対して有効な場合にのみ使用します。

ポピュレーションセット

あいまい一致/検索ストラテジを持つベースオブジェクトに対して、一致プロセスは標準ポピュレーションセットを使用して、国、地域、および言語の違いを示します。ポピュレーションセットは、トークン化、一致/検索ストラテジ、および一致目的を一致プロセスがどのように処理するかに影響します。

ポピュレーションセットは、名前、アドレス、および特定のポピュレーションに関するその他の識別情報をカプセル化します。例えば、番地や通りの名称、郵便番号などの住所のフォーマットは、国ごとに異なります。同様に、姓の分布も地域ごとに異なります。例えば、「Smith」という姓は米国では極めて一般的ですが、世界の他の地域では一般的ではありません。

ポピュレーションセットは、特定のポピュレーションのデータに現れやすい差異およびエラーについて調整することで、一致精度を向上させることができます。

重複データの一致

すべての非システムベースオブジェクトカラムの重複に対して一致を生成するには、重複データの一致機能を使用します。これらの一致は、完全な重複がベースオブジェクトカラムに設定数以上出現した場合に生成されます。ほとんどのデータに対して、最適な値は 2 です。

一致は生成されても、これらのレコードに対する統合インジケータは 4（未統合）のままになるため、標準の一致ルールを使用して後で一致させることができます。

注: 重複データの一致ジョブは、しきい値が 2 以上で、対応するベースオブジェクトに NON_EQUAL 一致ルールが定義されていない場合、バッチビューアに表示されます。

一致グループの構築と遷移一致

一致グループの構築（BMG）プロセスでは、統合プロセスの前に不要な一致処理を排除します。例えば、あるベースオブジェクトに以下の一致ペアがあるとします。

- レコード 1 はレコード 2 に一致
- レコード 2 はレコード 3 に一致
- レコード 3 はレコード 4 に一致

一致プロセスを実行して一致グループの構築を作成した後、統合プロセスの実行前に、以下のレコードが表示される場合があります。

- レコード 2 はレコード 1 に一致
- レコード 3 はレコード 1 に一致
- レコード 4 はレコード 1 に一致

この例では、レコード 4 をレコード 1 に一致させる明示的なルールはありません。他の一致の動作の結果として（レコード 1 をレコード 2 に、2 を 3 に、3 を 4 に一致させた結果）、間接的に一致したものです。間接一致は**遷移一致**とも呼ばれます。マージマネージャやデータマネージャで完全な一致履歴を表示すると、遷移一致の詳細を確認できます。

一致グループの構築（BMG）プロセスを使用する場合は、ベースオブジェクトの一致のプロパティで**【一致しないすべての行を一意とする】**を有効にします。**【一致しないすべての行を一意とする】**を有効にしないと、推移的一致中に、HMRG テーブルの rowid_match_rule に不正な値が表示されます。

手動統合の最大一致数

バッチジョブの実行中に処理される手動一致の最大数を設定できます。

制限値を設定することによって、データスチュワードが多数の統合プロセスを手動で処理する手間を軽減することができます。この制限値に到達すると、手動統合用に準備するレコード数が減少するまで、一致プロセスは実行を停止します。

外部一致ジョブ

Informatica MDM Hub では、新しいデータを既存のベースオブジェクトに実際にロードすることなく、そのデータをベースオブジェクトと一致させることができます。一致ジョブ全体を実行する代わりに、外部一致ジョブを実行して一致をテストし、結果を確認できます。外部一致ジョブは、あいまい一致ルールと完全一致ルールの両方を処理できるため、あいまい一致ベースオブジェクトおよび完全一致ベースオブジェクトで使用できます。

分散型プロセスサーバー

Informatica MDM Hub の実装では、複数のプロセスサーバーを同時に実行することにより、一致プロセスのスループットを向上させることができます。

アプリケーションサーバーまたはデータベースサーバーの障害の処理

一致バッチサイズが大きい非常に大きな一致ジョブの実行中にアプリケーションサーバーまたはデータベースで障害が発生した場合は、バッチ全体を再実行する必要があります。複数の一致バッチが 1 つの単位となっています。増分チェックポイントはありません。この問題に対処するため、データベースまたはアプリケーションサーバーに障害が発生することが考えられる場合は、一致バッチをより小さなサイズに設定して、一致バッチの再実行にかかる時間を削減します。

統合プロセス

ここでは、Informatica MDM Hub の統合プロセスに関する概念とタスクについて説明します。

統合プロセスについて

統合は、一致プロセスで一致ペアが特定された後に、一致したレコードのデータを 1 つのマスターレコードに統合するプロセスです。

次の図は、3 つの異なるソースシステムのレコードのセルデータを 1 つのマスターレコードに統合する例を示しています。

Informatica MDM Hub	マスタID	名	MN	姓	住所	市区町村	都道府県	郵便番号
	M-0001	Abel	Noel	Willan	161 Washington Ave.	Buffalo	NY	14263
Sales	SFA_ID	名	MN	姓	住所	市区町村	都道府県	郵便番号
	12345	Abel		Willan	161 Washington Ave.	Buffalo	NY	14263
Accounts	Cust_ID	名	MN	姓	住所	市区町村	都道府県	郵便番号
	502068	Abel	Noel	Willan	161 Washington Ave.	Buffalo	NY	14263
Marketing	Target_ID	名	MN	姓	住所	市区町村	都道府県	郵便番号
	willan05	Abel	N	Willan	Elm & Carlston Streets	Buffalo	NY	14263

レコードの自動統合と手動統合

一致ルールで一致テーブル内の AUTOMERGE_IND カラムを設定して、一致するレコードの統合方法を指定します。自動または手動を選択できます。

- 手動統合のフラグが設定されたレコードは、データスチュワードによってマージマネージャツールを使用して確認されます。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。
- 自動統合のフラグが設定されたレコードは、自動的にマージされます。また、ベースオブジェクトに対して自動一致とマージジョブを実行することもできます。この場合、一致ジョブの後に自動マージジョブが繰り返し呼び出され、ベースオブジェクトのすべてのレコードの一致の確認が完了するか、手動統合のレコード数の上限に達するまで続けられます。

トレーサビリティ

Informatica MDM Hub の目標は、すべての重複データを特定して取り除き、完全なトレーサビリティを保ちながら、それらをまとめて 1 つの統合されたレコードにマージまたはリンクすることです。

トレーサビリティとは、統合されたレコードのソースとなっているのはどのシステムか、また、そのシステム内のどのレコードかに関する情報を管理する Informatica MDM Hub 機能です。Informatica MDM Hub では、相互参照テーブルおよび履歴テーブルを使用してトレーサビリティが維持されます。

統合プロセスの主要な構成設定

統合プロセスには、以下の構成可能な設定が影響します。

オプション	説明
ベースオブジェクトのスタイル	統合プロセスでマージとリンクのどちらが使用されるかを決定します。
不変ソース	ソースシステムを不変と指定することができます。つまり、そのソースシステムからのレコードは一意と見なされ、一度完全に統合されたら変更されることはありません。
個別システム	ソースシステムを個別と指定することができます。つまり、そのシステムからのデータは、ベースオブジェクトに挿入されても統合されません。
子ベースオブジェクトに対するカスケードマージ解除	子ベースオブジェクトに対してカスケードマージ解除を有効にするとともに、親ベースオブジェクトのレコードがマージ解除されたときの動作を指定することができます。
親がマージされたときの子ベースオブジェクトレコード	親子関係にある2つのベースオブジェクトで、子ベースオブジェクトに対して有効にすると、親レコードが統合された場合に、子レコードが一致プロセスのために再送信されます。

統合オプション

一致したレコードをマージで統合することができます。マージ（物理的な統合）では、一致するレコードが結合され、ベースオブジェクトが更新されます。マージは、マージスタイルベースオブジェクトに行われます。

デフォルトでは、ベースオブジェクトの統合は物理的に保存されます（つまり、マージがデフォルトの動作です）。

マージでは、ベースオブジェクトテーブル内の複数のレコードが結合されます。2つのレコードの類似度に応じて、自動または手動でマージが実行されます。

- 完全に一致するレコードは、自動的にマージされます（自動マージプロセス）。
- 完全には一致しない近似するレコードは、手動確認のキューに追加され（手動マージプロセス）、データスチュワードによってマージマネージャツールを使用して確認されます。データスチュワードは、一致候補を調べ、それらの中からマージする一致を選択します。手動マージの一致ルールは、近似する一致を特定するように設定されます。
- 上記以外のレコードはすべて、Informatica MDM Hub によって手動確認のキューに追加され、データスチュワードによってマージマネージャツールを使用して確認されます。

一致ルールは、自動統合の対象となる完全な一致と、手動マージの対象となる近似する一致を特定するように設定されます。

特定されたレコードのステータスが Informatica MDM Hub で自動的に統合済みに変更される（レコードがデータスチュワードのキューから削除される）ようにするには、**【一致しないその他のすべての行を一意とする】**チェックボックスをチェック（選択）します。

最善データ

ベースオブジェクトの最善データ（BVT と略されることもあります）とは、ソースレコードのデータのうちの最適なセルと統合されているレコードです。

ベースオブジェクトレコードは BVT レコードで、対応するソースレコードのうちの最も信頼度が高いセルの値と統合して構築されます。

統合とワークフロー統合

状態が有効なベースオブジェクトでは、ベースオブジェクト内のレコードの現在のシステム状態が統合の動作に影響します。例えば、自動的に統合できるのはアクティブなレコードだけであり、システム状態が保留や削除済みのレコードは自動的に統合できません。統合時のシステム状態の影響については、以下のトピックを参照してください。

- [第 11 章, 「状態管理および BPM ワークフローツール」 \(ページ 173\) \(特に「状態遷移」 \(ページ 176\) と「レコードの状態およびベースオブジェクトレコード値の存続性」 \(ページ 177\)\)](#)
- 『*Multidomain MDM のデータスチュワードガイド*』の「データの統合」。

パブリッシュプロセス

Informatica MDM Hub は、Hub ストア内のデータ変更についての XML メッセージを生成し、そのメッセージをアウトバンドの Java Messaging System (JMS) メッセージキューにパブリッシュすることで、外部システムと統合します。

Informatica MDM Hub 実装では、指定されているビジネス要件および技術要件のためにパブリッシュプロセスが使用されます。外部の他のシステム、プロセス、またはアプリケーションでは、JMS メッセージキューをリスンして XML メッセージを取得し、そのメッセージに応じた処理を行うことができます。

すべての組織でこの機能が活用されるわけではありません。Informatica MDM Hub 実装でのこの機能の使用は任意です。

Informatica MDM Hub でサポートされている JMS モデル

Informatica MDM Hub では、以下の JMS モデルがサポートされています。

point-to-point (P2P)

ターゲット外部システムに特定の送信先。

パブリッシュ/サブスクライブ

Enterprise Service Bus (ESB) への point-to-point の後に、ESB から他のシステムへのパブリッシュ/サブスクライブ

調整済みデータをアウトバウンド配信するためのパブリッシュプロセス

パブリッシュプロセスは Informatica MDM Hub の主要なアウトバウンドフローです。ランド、ステージ、ロード、マッチ、および統合の各プロセスはすべて、Informatica MDM Hub の主要なインバウンドフローの調整に関連付けられています。

パブリッシュプロセスは Informatica MDM Hub の主要なアウトバウンド配信に属しています。調整を行なった後、Informatica MDM Hub はマスターのレコードデータをその他のアプリケーションやデータベースに配信できます。

パブリッシュプロセスのメッセージトリガ

Informatica MDM Hub メッセージトリガはパブリッシュプロセスを起動します。

Informatica MDM Hub は、MDM Hub ストアのデータが変更されるとメッセージトリガを生成します。メッセージトリガは、Java メッセージサービスのメッセージキューで Informatica MDM Hub がパブリッシュする XML メッセージを作成します。パブリッシュプロセスは XML メッセージを受信すると実行されます。

送信 JMS メッセージキュー

Informatica MDM Hub は送信メッセージキューを通信チャネルとして使用して、データの変更を外部システムにフィードバックします。

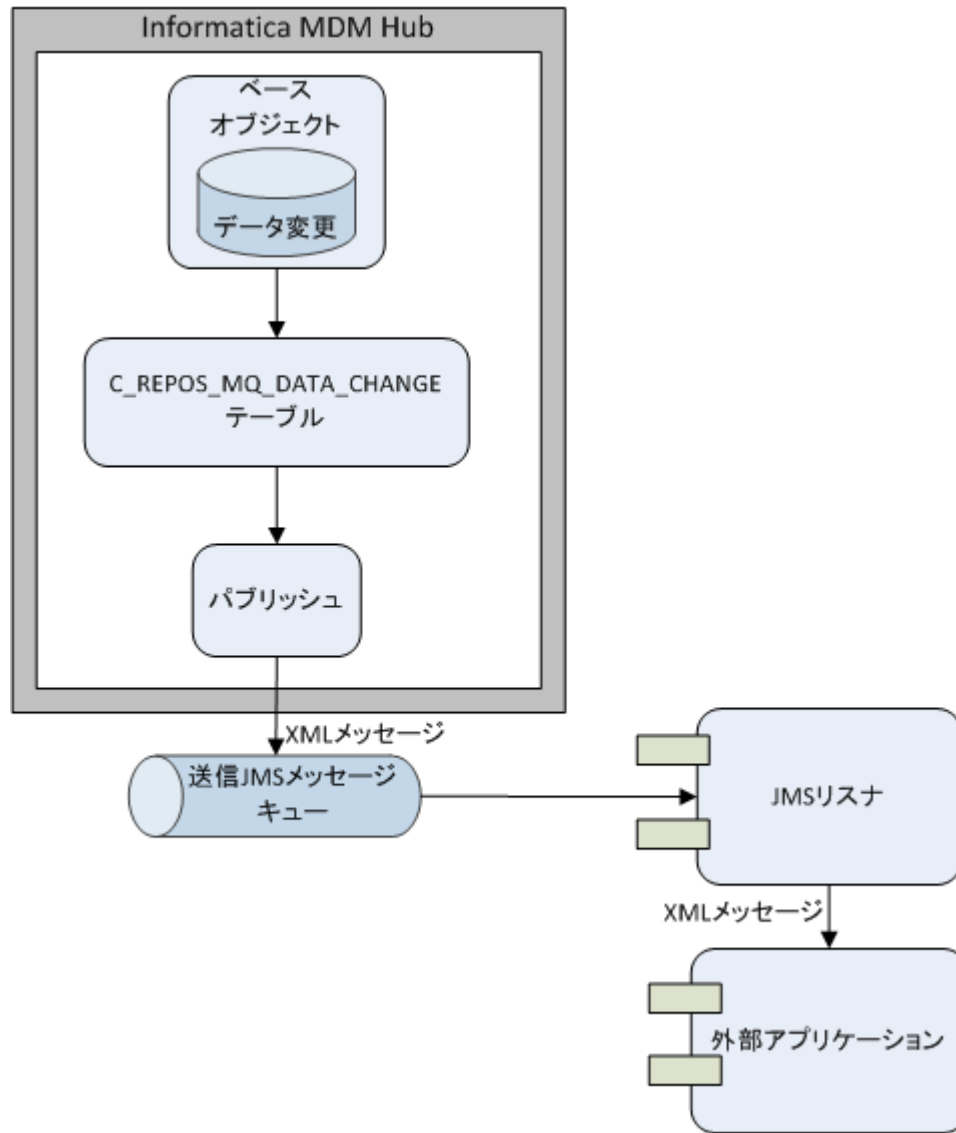
Informatica は埋め込みメッセージキューをサポートしています。これはアプリケーションサーバーに搭載されている JMS プロバイダを使用します。埋め込みメッセージキューは ConnectionFactory の JNDI 名および接続先の JMS キューの名前を使用します。これらの JNDI 名は、アプリケーションサーバーで設定しておく必要があります。Hub コンソールで、アプリケーションサーバー環境で設定済みのメッセージキューサーバーとメッセージキューを登録できます。

ORS 固有の XML メッセージスキーマ

XML メッセージは、共通の XML スキーマ (siperian-mrm-events.xsd) を基にした ORS 専用スキーマファイル (<ors-name>-siperian-mrm-event.xsd) を使用して作成されます。この ORS 専用スキーマは、JMS イベントスキーママネージャを使用して生成します。これは、パブリッシュプロセスを設定するのに必要なタスクです。

パブリッシュプロセスのランタイムフロー

以下の図は、パブリッシュプロセスのランタイムフローを示しています。



このとき、

1. バッチロードまたはリアルタイムの SIF API 要求（SIF put または cleanse_put 要求）により、ベースオブジェクトに対する挿入または更新が行われます。
C_REPOS_MQ_DATA_CHANGE テーブルに渡されるデータを制御するメッセージルールを設定できます。
2. Hub サーバーは、C_REPOS_MQ_DATA_CHANGE テーブルから定期的にデータをポーリングします。
3. 送信されていないデータに対して、Hub サーバーはそのデータに基づいて XML メッセージを生成し、メッセージキューに対して設定済みの送信キューに送信します。
4. 外部アプリケーションが、送信キューからメッセージを取得して処理します。

第 16 章

ランディングプロセスの設定

この章では、以下の項目について説明します。

- [ランディングプロセスの設定の概要, 295 ページ](#)
- [ソースシステムの設定, 295 ページ](#)
- [ランディングテーブルの設定, 298 ページ](#)

ランディングプロセスの設定の概要

ランディングプロセスを設定するには、次のタスクを Hub コンソールで実行します。

- [「ソースシステムの設定」 \(ページ 295\)](#)
- [「ランディングテーブルの設定」 \(ページ 298\)](#)

ソースシステムの設定

ここでは、Informatica MDM Hub 実装のソースシステムを定義する方法について説明します。

ソースシステムの概要

ソースシステムは、Informatica MDM Hub にデータを提供する外部のアプリケーションまたはシステムです。Informatica MDM Hub で各種のソースシステムからの入力を管理するには、各ソースシステムに一意的内部名が必要です。Informatica MDM Hub 実装のソースシステムを定義するときは、モデルワークベンチのシステムと信頼ツールを使用します。

ソースシステムの信頼の設定

ベースオブジェクトの同一のカラムに対して複数のソースシステムからデータを渡す場合、カラムごとに信頼を設定して、そのカラムに対するデータの提供元として信頼度が（他のソースシステムと比べて）高いソースシステムを指定することができます。信頼は、2 つのレコードを統合する際の存続性を判断する場合や、ソースシステムの更新を信頼して「最善データ」レコードに反映してかまわないかどうかを判断する場合に使用されます。

管理ソースシステム

Informatica MDM Hub では、データマネージャツールまたはマージマネージャツールからの手動による信頼のオーバーライドやデータの編集 Informatica MDM Hub uses an administration s (『*Multidomain MDM のデータスチュワードガイド*』を参照) に、管理ソースシステムを使用します。

この管理ソースシステムから、信頼が有効な任意のカラムにデータを渡すことができます。管理ソースシステムの名前はデフォルトでは Admin ですが、この名前は必要に応じて変更できます。

状態管理オーバーライドシステム

状態管理オーバーライドシステムは、他のあらゆるソースシステムからレコードの状態をオーバーライドして、レコードの状態を削除済みとマークできるソースシステムです。相互参照によってレコードがアクティブ状態と示される場合でも、削除済みと指定できます。

データが複数のソースシステムからベースオブジェクトレコードに提供される場合、提供レコードの 1 つ以上がアクティブ状態にあると、MDM Hub は状態管理オーバーライドシステムから削除済みの状態でレコードを挿入します。レコードの全体の状態は削除済みに設定されます。

状態管理オーバーライドシステムとして設定できるのは 1 つのソースシステムのみです。ソースシステムを状態管理オーバーライドシステムとして有効にするには、モデルワークベンチのシステムと信頼ツールを使用します。

注: 状態管理オーバーライドシステムは、バッチジョブには適用できません。

Informatica システムのリポジトリテーブル

システムと信頼ツールで定義したソースシステムは、Informatica MDM Hub の特別なパブリックリポジトリテーブル (C_REPOS_SYSTEM。表示名は MDM System) に格納されます。このテーブルは、[システムテーブルを表示する] オプションが選択されていれば、スキーママネージャに表示されます。C_REPOS_SYSTEM は、パッケージでも使用できます。

注: C_REPOS_SYSTEM テーブルには Informatica MDM Hub メタデータが含まれています。Informatica MDM Hub のシステムテーブルと同様に、C_REPOS_SYSTEM テーブルの構造やデータは変更しないでください。変更すると、Informatica MDM Hub が予測不能な動作をして、データが消失するおそれがあります。

システムと信頼ツールの起動

システムと信頼ツールを開始する手順

- Hub コンソールでモデルワークベンチを展開して、[システムと信頼ツール] をクリックします。

システムと信頼ツールが表示されます。

システムと信頼ツールでは以下のペインが表示されます。

ペイン	説明
ナビゲーション	システム データを Informatica MDM Hub に提供する各ソースシステムのリスト。管理ソースシステムを含みます。 信頼 ツリーを展開すると以下の情報が表示されます。 - 1 つ以上の信頼が有効なカラムを含むベースオブジェクト - 信頼が有効なカラム（のみ）
プロパティ	選択されているソースシステムのプロパティ。ベースオブジェクトが選択されている場合は、そのベースオブジェクトのカラムに対する信頼設定が表示されます。

ソースシステムのプロパティ

MDM Hub にデータを渡す必要のあるソースシステムを定義します。ソースシステムの定義は MDM Hub の外部で行います。モデルワークベンチのシステムと信頼ツールで、MDM Hub 用のソースシステムを定義します。

次の表では、MDM Hub のソースシステムの定義のプロパティについて説明します。

プロパティ	説明
名前	ソースシステムのわかりやすい一意の名前。
プライマリキー	MDM Hub によってプレフィックスとしてソースシステムのプライマリキーの値に追加される、ソースシステムの一意の ID。この値は読み取り専用です。
状態管理オーバーライドシステム	MDM Hub にデータを渡す他のすべてのソースシステムのレコードの状態を上書きするかどうかを指定します。他のすべてのソースシステムのレコードの状態を上書きするプロパティを有効にします。MDM Hub にデータを渡す他のすべてのソースシステムのレコードの状態を上書きしない場合は、このプロパティを無効にします。デフォルトでは無効になっています。
説明	オプション。ソースシステムの説明。

ソースシステムの追加

システムと信頼ツールを使用して、Informatica MDM Hub 実装にデータを渡す各ソースシステムを定義します。

注: ソースシステムのプライマリキーは、ソースシステム名の最初の 14 文字を大文字で表したものです。プライマリキーは、C_REPOS_SYSTEM リポジトリテーブルの ROWID_SYSTEM フィールドに格納されます。

1. システムと信頼ツールを起動します。
2. 書き込みロックを取得します。
3. ソースシステムのリストを右クリックし、**[システムの追加]** を選択します。
[新しいシステム] ダイアログボックスが表示されます。
4. ソースシステムのプロパティを指定します。

5. **[OK]** をクリックします。
リストにソースシステムが表示されます。

ソースシステムのプロパティの編集

管理システムを含むあらゆるソースシステムの名前を変更できます。ソースシステムの名前を変更すると、名前は Hub コンソールのコンテキスト内で変更されます。

注: このソースシステムが Informatica MDM Hub の実装にデータを提供した場合、Informatica MDM Hub は、このソースシステムのデータに対するリネージュの追跡を継続します。

1. システムと信頼ツールを起動します。
2. 書き込みロックを取得します。
3. ソースシステムのリストで、ソースシステムを選択します。
画面が更新され、ソースシステムのプロパティが表示されます。編集可能フィールドの隣に **[編集]** アイコンが表示されます。
4. プロパティを編集するには、**[編集]** アイコンをクリックして編集を行います。
5. 必要に応じて、信頼設定を編集します。
6. **[保存]** をクリックします。

ソースシステムの削除

ソースシステムがステージングテーブルへのデータ提供を開始する前に、ソースシステムを削除できます。ソースシステムを削除すると、ソースシステムの定義と関連付けられているメタデータはすべて削除されます。Informatica MDM Hub の外部への影響はありません。

注: 次のソースシステムは削除できません。

- 管理システム。
 - ベースオブジェクトのソースとして設定されている、任意のソースシステム。ソースシステムを参照する、ベースオブジェクトに関連付けられたステージングテーブル。
 - ステージングテーブルにデータを提供したことのある（つまり、ステージングプロセスが、ステージングテーブルにデータを移入したことのある）任意のソースシステム。
1. システムと信頼ツールを起動します。
 2. 書き込みロックを取得します。
 3. ソースシステムのリストで、ソースシステムを右クリックして **[システムの削除]** を選択します。
 4. アクションの確認が求められたら、**[はい]** をクリックします。
システムと信頼ツールによって、リストからソースシステムが削除されます。

ランディングテーブルの設定

ここでは、Informatica MDM Hub 実装でランディングテーブルを設定する方法について説明します。

ランディングテーブルについて

ランディングテーブルは、ソースシステムから Informatica MDM Hub へのデータのフローにおける中間の格納領域となります。つまり、ランディングテーブルは、ソースシステムから Hub ストアにデータを取り込むときに「データがランディングする場所」です。ランディングテーブルを定義するには、モデルワークベンチのスキーママネージャを使用します。

ソースシステムからのランディングテーブルへのデータの取り込みは、完全に Informatica MDM Hub の外部で行われます。各種のソースシステムからランディングテーブルのデータを収集するために使用するデータモデルも、Informatica MDM Hub の外部のものです。1つのソースシステムから複数のランディングテーブルにデータを取り込むことができます。また、1つのランディングテーブルで、異なるソースシステムからデータを受け取ることができます。使用するデータモデルは、それぞれの実装の要件に応じて決まります。

一方、Informatica MDM Hub の内部では、ランディングテーブルがステージングテーブルにマップされます。ランディングテーブルにマップされたステージングテーブルで、ベースオブジェクトにデータを提供するソースシステムが特定されます。ロードプロセス時に、Informatica MDM Hub は、ランディングテーブルからターゲットのステージングテーブルにデータをコピーし、そのデータにソースシステム ID のタグを付け、必要に応じてデータのクレンジングも行います。ランディングテーブルは、1つ以上のステージングテーブルにマップできます。ステージングテーブルは、1つのランディングテーブルにのみマップできます。

ランディングテーブルには、Informatica MDM Hub の外部で、バッチまたはリアルタイムのいずれかの方法でデータが取り込まれます。ランディングテーブルにデータが取り込まれた後、ステージプロセスでランディングテーブルからデータを取得し、必要に応じてデータをクレンジングしてから、適切なステージングテーブルにデータを取り込みます。

ランディングテーブルのカラム

ランディングテーブルには、ユーザーによって追加されるカラムである、ユーザー定義カラムがあります。また、ランディングテーブルには、ランディングテーブルレコードを一意に識別する SRC_ROWID システムカラムがあります。

レコードがステージプロセスによってロードされた場合は、SRC_ROWID の値がトレースに役立ちます。SRC_ROWID カラムの値は一意にする必要があります。SRC_ROWID カラムに重複する値が存在していると、ステージジョブが失敗します。SRC_ROWID カラムに重複する値が存在する場合は、Informatica グローバルカスタマサポートに連絡します。

注: ソースシステムテーブルに複数カラムのキーがある場合は、それらのカラムを連結して1つの一意な VARCHAR 値をプライマリキーカラムとして生成します。

ランディングテーブルのプロパティ

ランディングテーブルには以下のプロパティがあります。

プロパティ	説明
項目タイプ	追加するテーブルのタイプ。[ランディングテーブル] を選択します。
表示名	Hub コンソールに表示されるこのランディングテーブルの名前。
物理名	データベース内のランディングテーブルの実際の名前。Informatica MDM Hub では、ランディングテーブルの物理名の候補として、入力した表示名に基づく名前が示されます。
データテーブルスペース	このランディングテーブルのデータテーブルスペースの名前。詳細については、『 <i>Multidomain MDM のインストールガイド</i> 』を参照してください。

プロパティ	説明
インデックステーブルスペース	このランディングテーブルのインデックステーブルスペースの名前。詳細については、『 <i>Multidomain MDM のインストールガイド</i> 』を参照してください。
説明	このランディングテーブルの説明。
作成日	このランディングテーブルが作成された日時。
データセット全体を含む	<p>このランディングテーブルに、ソースシステムからのデータセット全体が格納されるか、更新のみが格納されるかを指定します。</p> <ul style="list-style-type: none"> - オン（デフォルト）の場合は、このランディングテーブルにソースシステムのデータセット全体が格納されることを示します（初期データロードなど）。このチェックボックスが有効になっている場合は、ステージプロセスにおいて、変更されたレコードのみがステージングテーブルにコピーされるように、Informatica MDM Hub の差分検出機能を設定できます。 - オンでない場合は、このランディングテーブルにソースシステムからの変更されたデータのみが格納されることを示します（増分ロードなど）。この場合、Informatica MDM Hub は、変更されていないレコードが、ランディングテーブルへの入力前にフィルタで除外されたものと見なします。このため、ステージプロセスでは、ランディングテーブルのすべてのレコードが、ステージングテーブルに直接挿入されます。このチェックボックスが無効になっている場合、Informatica MDM Hub の差分検出機能は使用できません。 <p>注: このプロパティは、ソースシステムのプロパティを編集する場合にのみ変更できます。</p>

ランディングテーブルの追加

ランディングテーブルを追加することができます。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. **【ランディングテーブル】** ノードを選択します。
4. [ランディングテーブル] ノードを右クリックし、**【項目の追加】** を選択します。
[テーブルの追加] ダイアログが表示されます。
5. この新しいランディングテーブルのプロパティを指定します。
6. **【OK】** をクリックします。
新しいランディングテーブルとそのサポートテーブルがオペレーショナル参照ストア（オペレーショナル参照ストア）内に作成され、スキーマのツリーに新しいランディングテーブルが追加されます。
7. ランディングテーブルのカラムを設定します。
8. このランディングテーブルにソースシステムの変更されたデータだけを含めるように設定する場合は、ランディングテーブルのプロパティ（[完全なデータセットを含める]）を編集します。

ランディングテーブルのプロパティの編集

ランディングテーブルのプロパティを編集する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. 編集するランディングテーブルを選択します。
スキーママネージャで、選択したテーブルを示す [ランディングテーブル ID] ペインが表示されます。

4. ランディングテーブルのプロパティを必要に応じて変更します。
5. **【保存】** ボタンをクリックして変更を保存します。
6. 必要に応じて、ランディングテーブルのカラムの設定を変更します。

ランディングテーブルの削除

ランディングテーブルを削除する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマツリーで、**【ランディングテーブル】** ノードを展開します。
4. 削除するランディングテーブルを右クリックして、**【削除】** を選択します。
削除を確認するように求められます。
5. **【はい】** を選択します。

ランディングテーブルがデータベースから削除され、このランディングテーブルとステージングテーブル間のマッピングも削除されます（ただしステージングテーブルは削除されません）。さらに、削除したランディングテーブルもスキーマツリーから削除されます。

第 17 章

MDM Hub ステージング

この章では、以下の項目について説明します。

- [MDM Hub ステージングの概要, 302 ページ](#)
- [MDM Hub ステージングテーブル, 303 ページ](#)
- [MDM Hub ステージングの要件, 305 ページ](#)
- [ステージングテーブルを追加する, 305 ページ](#)
- [ランディングテーブルとステージングテーブル間のカラムのマッピング, 308 ページ](#)
- [監査証跡と差分検出を設定する, 317 ページ](#)
- [ステージングテーブルの管理, 321 ページ](#)
- [マッピングの管理, 322 ページ](#)

MDM Hub ステージングの概要

MDM Hub ステージングでは、ランディングテーブルのソースデータを、ベースオブジェクトに関連付けられたステージングテーブルに移動します。MDM Hub ステージングを実行するには、ランディングテーブルとステージングテーブルを作成する必要があります。MDM Hub ステージングを実行する前に、データをランディングテーブルにロードする必要があります。

注: MDM Hub は、単一のランディングテーブルから複数のステージングテーブルにデータを移動できます。ただし、ステージングテーブルは、それぞれ 1 つのランディングテーブルからのみデータを受け取ります。

ステージジョブを実行する前に、ランディングテーブルとステージングテーブル間のマッピングを定義します。マッピングは、ランディングテーブルのソースカラムをステージングテーブルのターゲットカラムにリンクします。ステージジョブを実行すると、MDM Hub により、データがマッピングを基に、ランディングテーブルのカラムからステージングテーブルのカラムに移動します。

ステージプロセス時、MDM Hub では 250 レコードのブロックを一度に 1 つ処理します。ブロックのレコードに問題がある場合、MDM Hub ではそのレコードを拒否テーブルに移動します。セルの値が長すぎる場合、またはレコードの更新日が現在の日付より後の場合、レコードは拒否されます。MDM Hub が拒否されたレコードを移動したら、MDM Hub はそのブロックの残りのレコードの処理を停止し、他のブロックに移行します。ステージプロセスが完了したら、ジョブを再度実行します。処理されなかったレコードは再度ピックアップされて処理されます。

ステージプロセス中にクレンジング操作を実行する場合は、MDM Hub クレンジング機能を使用します。マッピングでデータクレンジングを設定します。MDM Hub ステージングを実行するときに、差分検出および監査証跡を設定できます。ステージプロセスによって、データのロードプロセスが準備されます。

MDM Hub ステージングテーブル

MDM Hub ステージジョブを実行すると、MDM Hub はランディングテーブルから MDM Hub ステージングテーブルにデータをロードします。ステージングテーブルの構造は、統合されたデータを含んだターゲットオブジェクトの構造を基にします。ステージングテーブルを設定するには、モデルワークベンチのスキーマツールを使用します。

次のタスクを実行して、MDM Hub ステージングテーブルを設定します。

1. MDM Hub ステージングの前提条件を満たしていることを確認します。
2. ステージングテーブルを追加します。
3. ランディングテーブルとステージングテーブル間のカラムのマッピングを行います。
4. 監査証跡と差分検出を設定します。

MDM Hub ステージプロセステーブル

MDM Hub ステージプロセスでは、ランディングテーブル、ステージングテーブル、RAW テーブル、リジェクトテーブルを使用して、ソースシステムから MDM Hub にデータを移動します。

MDM Hub ステージプロセスでは次の MDM Hub テーブルを使用します。

ランディングテーブル

ソースシステムから最初にロードされるデータを含んだ MDM Hub テーブル。MDM Hub は、MDM Hub ステージプロセス用にランディングテーブルのデータを使用します。

ステージングテーブル

ステージプロセス中に MDM Hub によって許可されるデータを含んだ MDM Hub テーブル。MDM Hub ステージプロセス中、データはランディングテーブルからステージングテーブルに移動します。

RAW テーブル

MDM Hub によってランディングテーブルからアーカイブされる RAW データを含んだ MDM Hub テーブル。各 RAW テーブルはステージングテーブルに関連付けられ、<staging table name>_RAW という名前が付いています。データの特定のロード回数後または一定の時間後に RAW データをアーカイブするように、MDM Hub を設定できます。MDM Hub では、ランディングテーブルが変更されていない場合でも Null プライマリキーが挿入されます。

リジェクトテーブル

MDM Hub によって却下されたレコードと却下ごとの理由の説明を含んだ MDM Hub テーブル。各リジェクトテーブルはステージングテーブルに関連付けられ、<staging table name>_REJ という名前が付いています。作成したステージングテーブルごとに、MDM Hub によってリジェクトテーブルが生成されます。MDM Hub では、ランディングテーブルが変更されていない場合でも Null プライマリキーが挿入されます。

ステージプロセス中、MDM Hub は次の理由のためにレコードを却下します。

- LAST_UPDATE_DATE カラムに将来の日付または Null の日付がある。
- LAST_UPDATE_DATE カラムの値が 1900 未満になっている。
- Null 値がステージングテーブルの PKEY_SRC_OBJECT カラムにマップされている。
- PKEY_SRC_OBJECT カラムに重複が含まれている。MDM Hub で同じ PKEY_SRC_OBJECT 値を持つレコードが複数見つかった場合、最も高い SRC_ROWID 値を持つレコードが MDM Hub によってロードされます。MDM Hub は、他のレコードをリジェクトテーブルに移動します。

- HUB_STATE_IND フィールドに、状態管理が有効なベースオブジェクトに対して無効な値が含まれている。
- 一意のカラムに重複が含まれている。

ステージングテーブルのプロパティ

ステージングテーブルを Hub コンソールで作成および管理することができます。ステージングテーブルの作成時に、ステージングテーブルのプロパティを一部設定します。

次の表では、ステージングテーブルの作成時に設定するステージングテーブルのプロパティについて説明します。

プロパティ	説明
表示名	Hub コンソールに表示されるステージングテーブルの名前。
物理名	データベース内のステージングテーブルの名前。MDM Hub では、ステージングテーブルの物理名の候補として、入力した表示名に基づく名前が示されます。
データテーブルスペース	ステージングテーブルのデータテーブルスペースの名前。
インデックステーブルスペース	ステージングテーブルのインデックステーブルスペースの名前。
説明	ステージングテーブルの説明。
テーブルタイプ	テーブルのタイプ。デフォルトは Staging です。
作成日	ステージングテーブルが作成された日。
システム	ステージングテーブルデータのソースシステム。
ソースシステムキーを保持する	MDM Hub でソースシステムのキー値を使用するか、MDM Hub によって生成されたキー値を使用するかを指定します。ソースシステムのキー値の使用を有効化します。MDM Hub によって生成されたキー値の使用を無効化します。デフォルトでは無効になっています。 注: 複数のレコードが同一の PKEY_SRC_OBJECT を含んでいる場合、ステージプロセス中に残るレコードは、LAST_UPDATE_DATE が最新です。他のレコードはリジェクトテーブルに送信されます。
ギャップを埋める	新しいレコードバージョンを追加したときに、各レコードバージョンの有効期間が連続するよう調整するかどうかを指定します。有効に設定すると、ベースオブジェクトに新しいレコードバージョンを追加可能な場合に、MDM Hub によって各レコードバージョンの有効期間が連続するよう調整されます。有効に設定すると、各レコードバージョンの有効期間にギャップが発生するようなレコードバージョンの追加は MDM Hub によって拒否されます。デフォルトでは無効になっています。
最高予約キー	初回ロード後に増やす必要のあるキーの数。このプロパティは、 【ソースシステムキーを保持する】 チェックボックスを有効にした場合に表示されます。
セルの更新	ステージングテーブルの入力レコードの値と同じ場合に、MDM Hub によってターゲットテーブルのセルを更新させます。
カラム	ステージングテーブルのカラム。

マッピングのプロパティ

ランディングテーブルとステータステーブル間のマッピングを定義する場合は、そのマッピングのプロパティを設定します。

次の表に、マッピングプロパティを示します。

フィールド	説明
名前	Hub コンソールに表示されるマッピング名前です。
説明	マッピングの説明です。
ランディングテーブル	マッピングのソースとなるランディングテーブルを選択します。
ステージングテーブル	マッピングのターゲットとなるステージングテーブルを選択します。
セキュアリソース	マッピングをセキュアリソースにする場合に有効にします。この結果、マッピングに対するアクセスを制御できるようになります。マッピングをセキュアリソースとして指定すると、セキュアリソースツールでそのマッピングに特権を割り当てることができます。

MDM Hub ステージングの要件

開始する前に、次のインストールと設定タスクを実行します。

1. データのクレンジングに必要な外部クレンジングエンジンを設定します。
2. データをランディングテーブルにロードするランドプロセスを完了させます。

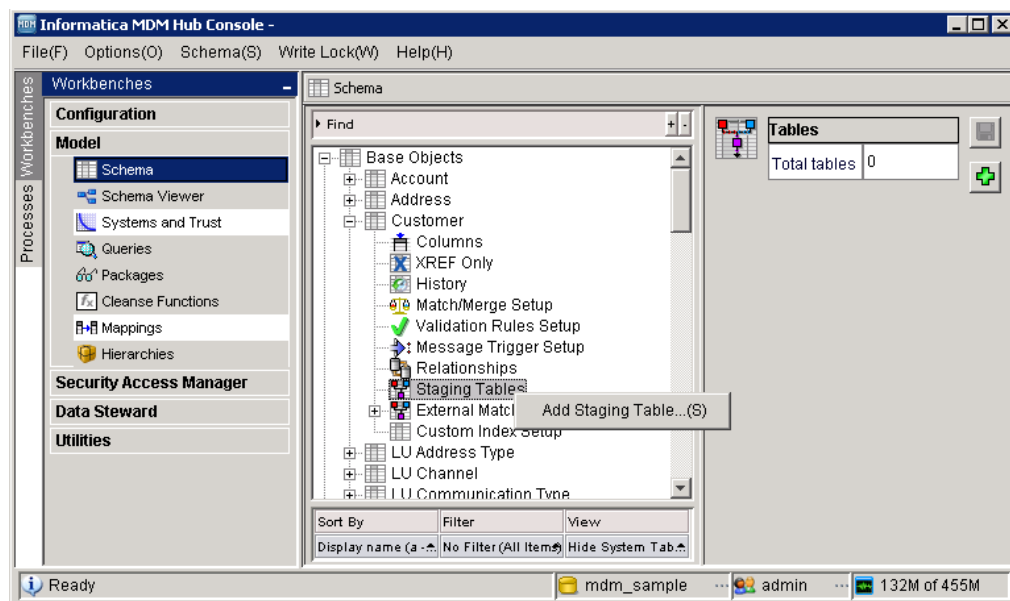
ステージングテーブルを追加する

ランディングテーブルのデータの移動先にするステージングテーブルを追加します。

1. Hub コンソールでスキーマツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、ステージングテーブルを追加するベースオブジェクトのノードを展開します。
4. [ステージングテーブル] ノードを右クリックします。

[ステージングテーブルの追加] オプションが表示されます。

次の図に、ステージングテーブルを Customer ベースオブジェクトに追加するための【ステージングテーブルの追加】オプションを示します。



5. 【ステージングテーブルの追加】をクリックします。
【ベースオブジェクトへのステージングの追加】ダイアログボックスが表示されます。
6. ステージングテーブルのプロパティを指定します。

次の図に、[表示名] フィールドが S_CRM_CUST に設定された、[Customer ベースオブジェクトへのステージングの追加] ダイアログボックスを示します。

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

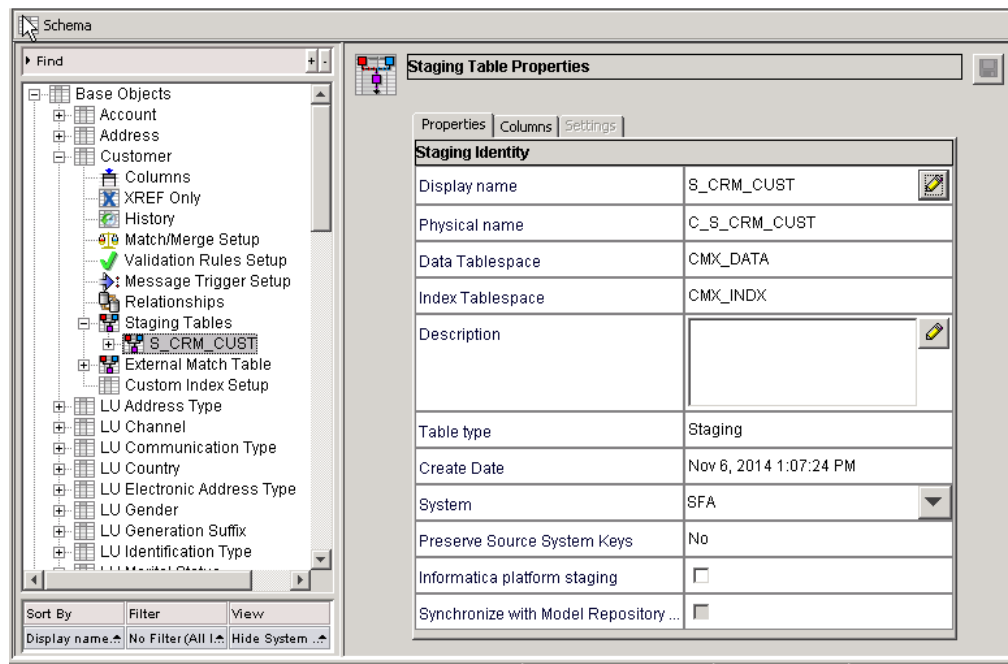
OK Cancel

7. ベースオブジェクトのカラムのリストで、ソースシステムで提供するカラムを選択します。
ベースオブジェクトのカラムをすべて選択する場合は、[すべてのカラムを選択] をクリックします。
8. カラムのプロパティを指定します。

9. **[OK]** をクリックします。

ステージングテーブルとそのサポートテーブルがオペレーショナル参照ストア内に作成され、ナビゲーションツリーにステージングテーブルが追加されます。

次の図に、ナビゲーションツリーの S_CRM_CUST ステージングテーブルを示します。



ランディングテーブルとステージングテーブル間の カラムのマッピング

1つのランディングテーブルから1つ以上のステージングテーブルにカラムをマッピングします。カラムのマッピングでは、ランディングテーブルのカラムからステージングテーブルのカラムにデータを転送する方法を定義します。少なくとも1つの一意のプライマリーキーをマッピングし、ソースデータからマスターデータへのトレーサビリティを確保する必要があります。また、他のソースデータをマスターデータにマッピングすることもできます。

各カラムマッピングには、入力と出力があります。入力はランディングテーブルのカラムです。出力はステージングテーブルのカラムです。データを処理するクレンジング関数で入力をパススルーできます。例えば、データの標準化、データの検証、またはデータの集計を行うことができます。また、カラムマッピングをセキュアリソースまたはプライベートリソースにするかどうかを定義することもできます。

カラムマッピングを設定したら、ステージジョブを実行してステージングテーブルにソースデータを取り込みます。

クレンジングされたデータとパススルーされた（変更なし）データ

ステージングテーブルの各データカラムには、次のいずれかの方法でランディングカラムからデータが入力されます。

コピーの方法	説明
パススルー	Informatica MDM Hub によって、データが変更されることなくそのままコピーされます。データはランディングテーブルのカラムから直接入力されます。
クレンジング	Informatica MDM Hub が、クレンジング関数を使用してデータを標準化および検証します。クレンジング関数の出力が、ステージングテーブルのターゲットカラムに対する入力になります。

注: ステージングテーブルでは、ランディングテーブルのすべてのカラムまたはクレンジング関数のすべての出力文字列を使用する必要はありません。同じランディングテーブルから複数のステージングテーブルに入力を提供することが可能です。また、同じクレンジング関数を複数のランディングテーブルの複数のカラムに再利用することも可能です。

分解と集計

クレンジング関数では、データの分解や集計を行うこともできます。

データを分解するクレンジング関数

分解クレンジング関数は、1つのフィールドからデータを取得し、そのデータを分解し、そのそれぞれをステージングテーブル内の異なるカラムに割り当てます。

例えば、敬称フィールドを5つの名前タイプフィールドに分解する場合、分解クレンジング関数は1つの入力文字列と5つの出力文字列で構成されます。マッピングでは、入力文字列をクレンジング関数にマッピングし、各出力文字列をステージングテーブル内のターゲットカラムにマッピングします。

データを集計するクレンジング関数

集計クレンジング関数は、複数のフィールドからデータを取得し、そのデータを集計し、集計データをステージングテーブル内の1つのカラムに割り当てます。

例えば、5つの名前タイプフィールドを1つの敬称フィールドに集計する場合、集計クレンジング関数は5つの入力文字列と1つの出力文字列で構成されます。マッピングでは、入力文字列をクレンジング関数にマッピングし、出力文字列をステージングテーブル内のターゲットカラムにマッピングします。

Informatica Data Quality のバッチジョブとカラムのマッピング

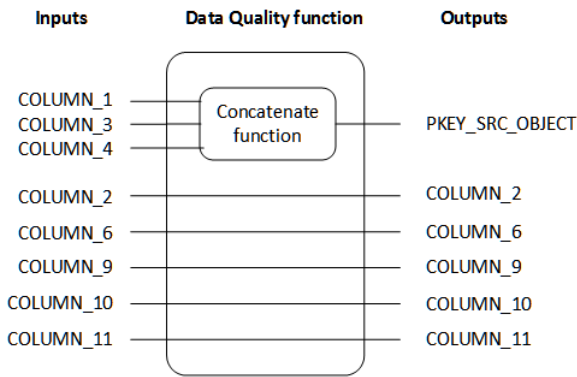
Informatica Data Quality のバッチジョブを使用して、Informatica MDM マスターレコードを処理することができます。Informatica Data Quality バッチジョブを使用するには、ランディングテーブルからマッピングするすべてのカラムに同じ Data Quality 関数をパススルーさせる必要があります。カラムは、その関数内でクレンジングされるか、関数がパススルーされ変更されません。

注: マッピングで、すべての入力が1つの Data Quality 関数をパススルーしない場合、Informatica Data Quality のバッチジョブを使用できません。代わりに Informatica Data Quality では、一度に1つずつレコードを処理します。

例えば、ランディングテーブルの複数のカラムからプライマリキーを作成するとします。マッピングを作成し、Data Quality 関数を追加して、ランディングテーブルからステージングテーブルにマッピングするすべてのカラムを追加します。Data Quality 関数内で、プライマリキー用のカラムを連結する関数を追加し、連結関数の

出力を PKEY_SRC_OBJECT カラムにマッピングします。残りのカラムについては、他の関数を使用するか、カラムを変更せずにパススルーさせることができます。

次の図に、必要な入力とマッピングされた出力のすべてを含む Data Quality 関数を示します。



ステージングテーブルで物理削除検出を有効にする場合は、物理削除の検出テーブルでプライマリキーを作成するために使用したカラムを指定します。

マッピングツールの起動

マッピングツールを開始するには、Hub コンソールでモデルワークベンチを展開して、**【マッピング】**をクリックします。

マッピングツールが以下のパネルとともに表示されます。

カラム	説明
マッピングリスト	ランディングからステージングへの定義済みの各マッピングのリスト。
プロパティ	選択されているマッピングのプロパティ。

マッピングリストでマッピングを使用すると、そのプロパティが表示されます。

マッピングツールのタブ

マッピングが選択されると、マッピングツールに以下のタブが表示されます。

タブ	説明
全般	このマッピングの全般的なプロパティ。
マッピング図	ランディングテーブルとステージングテーブルのカラム間のマッピングを定義できる対話型の図。
クエリパラメータ	このマッピングのクエリパラメータを指定可能。
テスト	マッピングをテスト可能。

マッピング図

マッピングの **【図】** タブをクリックすると、現在のカラムマッピングが表示されます。

マッピングラインは、ランディングテーブルのソースカラムからステージングテーブルのターゲットカラムへのマッピングを示します。マッピングラインの両端の円の色は、データタイプを示します。

マッピングの作成

ランディングテーブルとステージングテーブル間のマッピングを作成します。

1. **モデルワークベンチ**で、**【マッピング】** をクリックします。
2. 書き込みロックを取得します。
3. マッピング領域で右クリックし、**【マッピングの追加】** をクリックします。
マッピングツールに **【マッピング】** ダイアログボックスが表示されます。
4. マッピングのプロパティを指定します。

フィールド	説明
名前	Hub コンソールに表示されるマッピング名前です。
説明	マッピングの説明です。
ランディングテーブル	マッピングのソースとなるランディングテーブルを選択します。
ステージングテーブル	マッピングのターゲットとなるステージングテーブルを選択します。
セキュアリソース	マッピングをセキュアリソースにする場合に有効にします。この結果、マッピングに対するアクセスを制御できるようになります。マッピングをセキュアリソースとして指定すると、セキュアリソースツールでそのマッピングに特権を割り当てることができます。

次の図に、サンプルマッピングを示します。

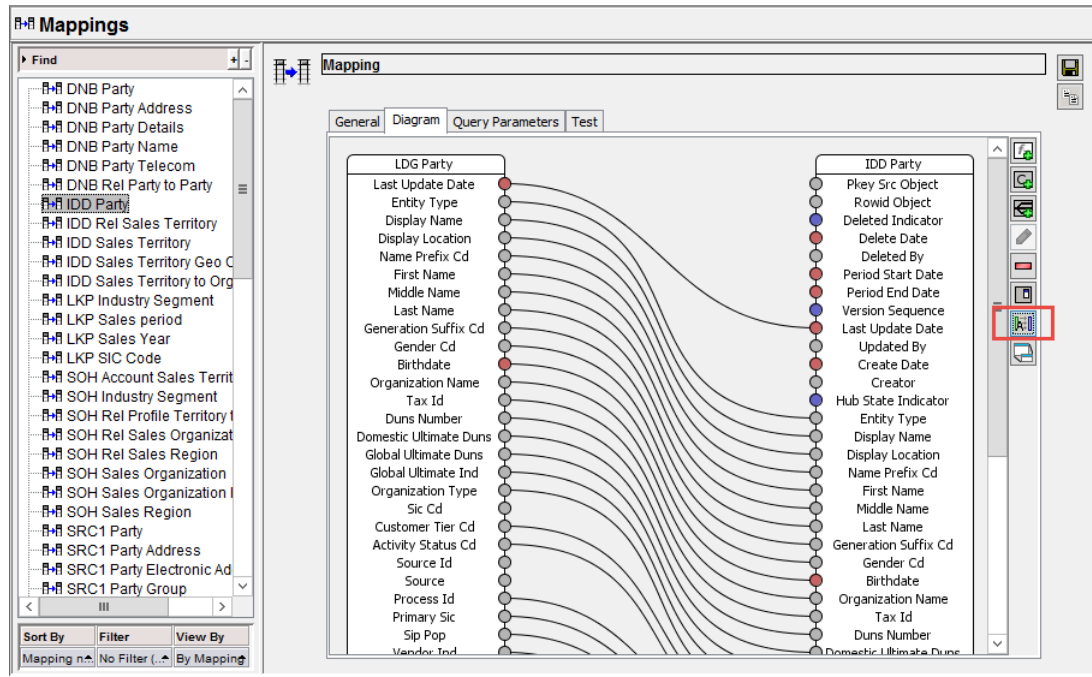
The image shows a 'Mapping' dialog box with the following fields and values:

Mapping	
Name	IDD Party
Description	Example mapping
Landing table	LDG Party ▼
Staging table	IDD Party ▼
Secure Resource	<input checked="" type="checkbox"/>

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

5. **【OK】** をクリックします。
6. **【マッピング図】** タブをクリックします。
マッピングツールのワークスペースに、ランディングテーブルとステージングテーブルが表示されます。
7. ランディングテーブルからステージングテーブルに同じ名前のカラムをマッピングするには、**【自動マッピング】** ボタンをクリックします。

次の図に、サンプルマッピングでの自動マッピングの結果を示します。接続は変更できます。



8. **【保存】** をクリックします。

基本マッピングが作成されます。次に、プライマリキーをマッピングします。

プライマリキーのマッピング

ソースシステム内のレコードを識別するために、MDM Hub にはプライマリキーと呼ばれる一意のキーが必要です。ソースシステムに一意のキーを持つカラムが含まれている場合は、そのカラムをランディングテーブル

からステージングテーブルの PKEY_SRC_OBJECT カラムにマップします。ソースシステムに一意のキーが含まれていない場合は、複数のカラムの値を連結して一意のキーを作成できます。

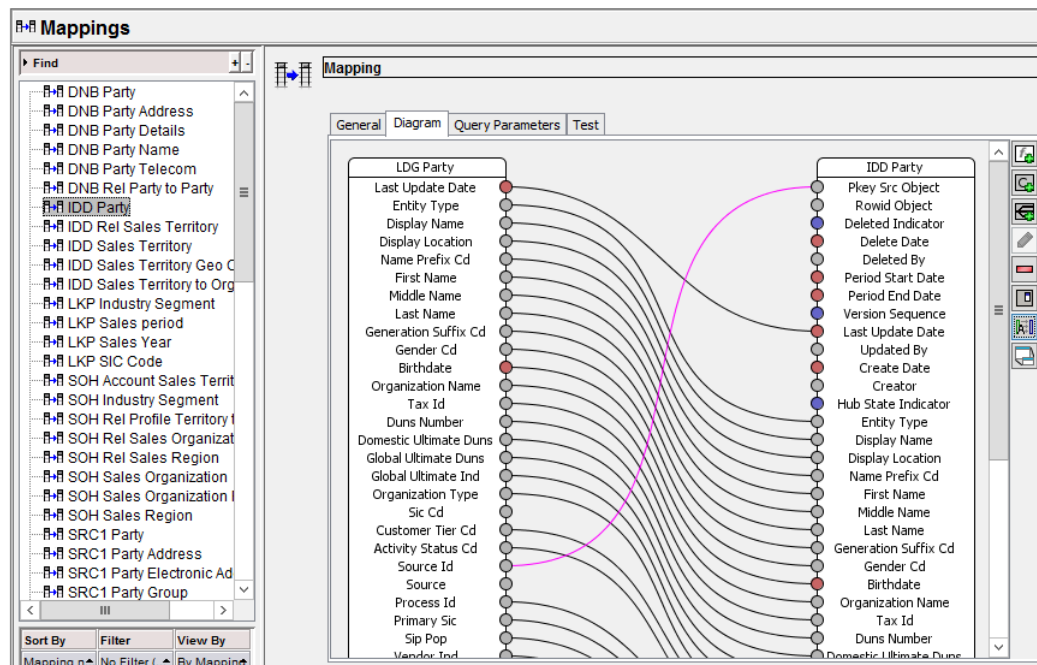
単一カラムをプライマリーキーとして使用

ソースシステムに一意のキーを持つカラムが含まれている場合は、そのカラムをランディングテーブルからステージングテーブルの PKEY_SRC_OBJECT カラムにマップします。

この手順では、[マッピング図] ワークスペースでマッピングが開いていることを想定しています。

1. **[マッピング図]** ワークスペースのランディングテーブルで、一意のカラム名を見つけます。その出力コネクタを、ステージングテーブルの PKEY_SRC_OBJECT カラムの入力コネクタにドラッグします。

カラムが線で接続されます。



2. **[保存]** をクリックします。

複数カラムをプライマリーキーとして使用

ソースデータに一意の値を持つカラムが含まれていない場合、複数のカラムの値を連結して、一意のプライマリーキーを形成します。MDM Hub または Informatica Data Quality から文字列の連結関数を使用するか、カスタム関数を作成します。

[マッピング図] ワークスペースでマッピングが開いていることを確認します。

注: ステージングテーブルの PKEY_SRC_OBJECT カラムにランディングテーブルのカラムを 4 つ以上マップすることは可能ですが、インデックスの作成は 3 カラムまでに制限されます。

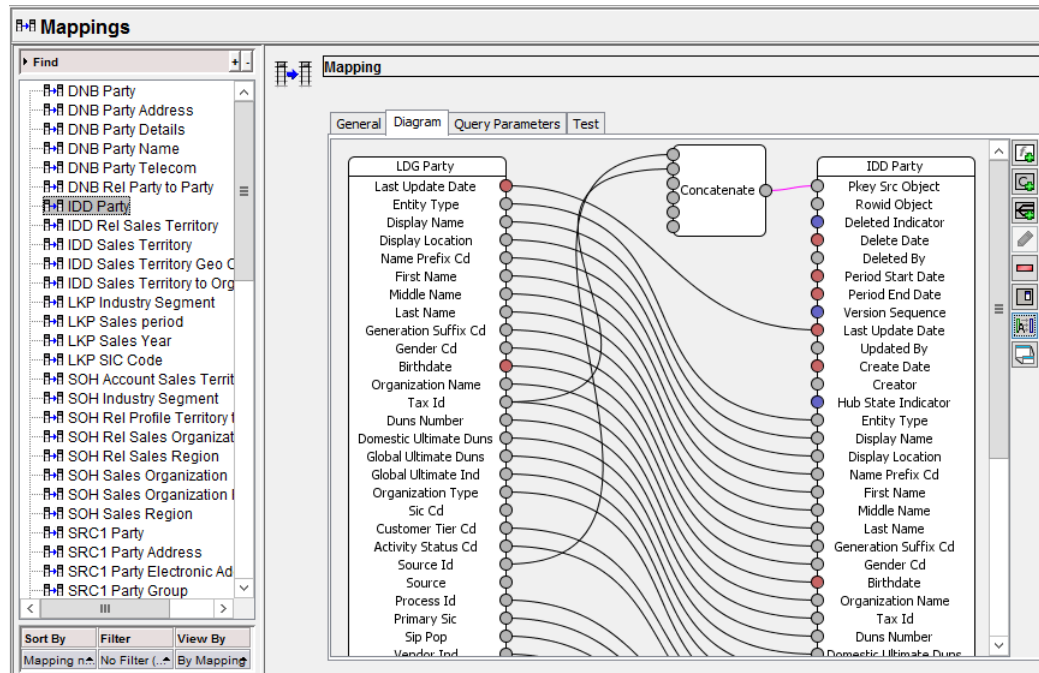
1. プライマリーキー用に連結するカラムを特定します。
2. 連結関数を追加します。
 - a. **[マッピング図]** ワークスペースで、ワークスペースを右クリックして **[関数の追加]** を選択します。
 - b. **[文字列関数]** を展開します。
 - c. **[連結]** をクリックします。

d. **[OK]** をクリックします。

連結関数がワークスペースに表示されます。

3. 結合したいカラムごとに、その出力コネクタを、ランディングテーブルから連結関数の入力コネクタにドラッグします。
4. 出力コネクタを、関数からステージングテーブルの PKEY_SRC_OBJECT カラムの入力コネクタにドラッグします。

プライマリキーの接続が完了します。



5. **[保存]** をクリックします。

マッピングカラム

直接接続の場合、ランディングテーブルのカラムをステージングテーブルのカラムに接続します。ランディングテーブルのデータをクレンジングするには、関数を追加します。関数への入力、ランディングテーブルのカラムから取得されます。関数からの出力は、ステージングテーブルのカラムに渡されます。

この手順では、[マッピング図] タブでマッピングが開いていることを想定しています。自動マッピングプロセスを使用する場合は、同じ名前のカラムが直接接続されます。

1. **[マッピング図]** ワークスペースで、自動的に作成された接続を確認します。自動接続のいずれかが適切でない場合は、その接続を削除します。
 - a. コネクタ線をクリックします。
 - b. 右クリックして、**[リンクの削除]** を選択します。
2. ランディングテーブル内の未接続のカラムをステージングテーブル内のカラムに接続する必要があるかどうかを判断します。

接続を予定している場合は、次の要件を考慮します。

- データ型。カラムは同じデータ型であるか、暗黙的に変換できるデータ型のいずれかである必要があります。

- クレンジング。データをクレンジングまたは変換してから、ステージングテーブルにロードする場合、使用する関数を決める必要があります。
 - 文字列長。ステージングテーブルでは、CHAR または VARCHAR データ型のカラムの長さは、ランディングテーブル内のカラムの長さ以上である必要があります。
- 警告:** ロード時、文字列がステージングテーブル内のカラムに対して長すぎると、ロードプロセスはレコード全体を拒否テーブルに送信します。
3. データをクレンジングする必要がない場合は、カラムを直接接続します。カラムを接続するには、ランディングテーブルカラムの出力コネクタをステージングテーブルカラムの入力コネクタにドラッグします。
 4. データをクレンジングする必要がある場合は、関数をワークスペースに追加してカラムを接続します。
 - a. ワークスペースで、右クリックして **関数の追加** を選択します。
 - b. 関数を選択して **[OK]** をクリックします。
関数が **[マッピング図]** ワークスペースに表示されます。
 - c. ランディングテーブルでカラムを検索します。その出力コネクタを関数の入力コネクタにドラッグします。関数への入力として、追加カラムの追加を繰り返します。
 - d. 関数の出力コネクタを、ステージングテーブルカラムの入力コネクタにドラッグします。
 5. 接続の追加が完了したら、**[保存]** をクリックします。

マッピングのレコードのフィルタリング

デフォルトでは、ランディングテーブルからすべてのレコードが取得されます。

オプションとして、ランディングテーブル内のレコードをフィルタリングするマッピングを設定することができます。フィルタには、個別フィルタと条件フィルタの 2 種類があります。この設定は、マッピングツールの **[クエリパラメータ]** タブで行います。

個別マッピング

[クエリパラメータ] タブで **[個別を有効にする]** チェックボックスをクリックすると、ステージジョブでランディングテーブルから個別レコードのみが選択されます。Informatica MDM Hub では、以下の SELECT 文を使用してステージングテーブルにデータが取り込まれます。

```
select distinct * from landing_table
```

個別マッピングは、1 つのランディングテーブルから複数のステージングテーブルにデータを提供するときに、そのランディングテーブルが正規化されていない場合（例えば、顧客データと住所データの両方を含む場合）などに役立ちます。例えば、1 人の顧客が 3 つの住所を持っているとします。この場合、個別マッピングを使用すると、2 つの余分な顧客レコードが却下テーブルに書き込まれなくなります。

もう 1 つの例として、ランディングテーブルに以下のデータが含まれているとします。

LUD	CUST_ID	NAME	ADDR_ID	ADDR
7/24	1	JOHN	1	1 MAIN ST
7/24	1	JOHN	2	1 MAPLE ST

Customer ステージングテーブルには LUD、CUST_ID、および NAME のみがマッピングされているため、顧客テーブルへのマッピングでは **[個別を有効にする]** をオンに（選択）して重複レコードが作成されないようにします。個別を有効にすると、1 つのレコードのみが顧客テーブルに取り込まれ、却下は発生しません。

あるいは、住所のマッピングで個別を無効にして ADDR_ID と ADDR をマッピングします。これにより、2 つのレコードが得られ、却下は発生しません。

条件マッピング

[条件を有効にする] チェックボックスを選択すると、クレンジングで SQL WHERE 句を適用してデータをアンロードできます。例えば、ランディングテーブルのデータが米国のすべての州から取得されているとします。WHERE 句を使用して、ある 1 つの州（カリフォルニアなど）のデータのみを取り込むように、ステージングテーブルに書き込まれるデータをフィルタリングすることができます。そのためには、WHERE 句（WHERE キーワードは省略）で「STATE='CA'」と入力します。クレンジングジョブを実行すると、レコードがアンロードされ、SELECT * FROM LANDING WHERE STATE='CA' として処理されます。条件マッピングを指定する場合は、[検証] ボタンをクリックして SQL 文を検証してください。

マッピングのクエリパラメータの設定

マッピングのクエリパラメータを設定する手順

1. マッピングツールを起動します。
2. 書き込みロックを取得します。
3. 設定するマッピングを選択します。
4. [クエリパラメータ] タブをクリックします。
このマッピングの [クエリパラメータ] タブが表示されます。
5. 個別マッピングを設定するかどうかに応じて、[個別を有効にする] チェックボックスをオンまたはオフにします。
6. 条件マッピングを設定するかどうかに応じて、[条件を有効にする] チェックボックスをオンまたはオフにします。
有効にした場合は、SQL WHERE 句（WHERE キーワードを除く）を入力し、[検証] をクリックして検証します。
7. [保存] ボタンをクリックして変更を保存します。

クレンジング関数を使用するマッピングの保持

クレンジング関数を含むマッピングは、クレンジングエンジンの状態の影響を受けます。

次の状況の場合、MDM Hub では、マッピングから関数を削除します。

- 環境内のクレンジングエンジンを変更すると、マッピングで使用する関数は新しいクレンジングエンジンで使用できなくなります。
- プロセスサーバーのアプリケーションサーバーを再起動すると、クレンジングエンジンでは初期化に失敗します。

関数をマッピングに復元するには、マッピングを編集し、アクティブなクレンジングエンジンで使用可能な関数を追加します。

行 ID によるロード

Informatica MDM Hub に行 ID によるロードを設定して、ロード、一致、マージの各処理を合理化することができます。

ステージプロセスを設定するときには、ランディングテーブルのカラムをステージングテーブルのカラムに対応付けする必要があります。行 ID によるロードを設定するには、ROWID_OBJECT 値を含むランディングテーブルのカラムをステージングテーブル内の ROWID_OBJECT カラムに対応付けします。例えば、ランディングテーブル内の INPUT_ROWID_OBJECT カラムをステージングテーブル内の ROWID_OBJECT カラムに対応付けることによって MDM Hub に ROWID_OBJECT 値を提供できます。

ロードプロセスでは、ステージングテーブル内の ROWID_OBJECT カラムからの値が、ベースオブジェクトに関連付けられた相互参照テーブル内の ORIG_ROWID_OBJECT カラムの値と比較されます。値が一致する場合、ロード処理が相互参照テーブルのレコードを更新します。

ロードプロセスでは、PKEY_SRC_OBJECT カラムと ROWID_SYSTEM カラムで値が一致する場合に、既存の相互参照レコードが入力レコードに更新されます。それ以外の条件では、ロードプロセスによって、入力レコードの値が入った新しい相互参照レコードが挿入され、既存の相互参照レコードとマージされます。ロードプロセスでは、ベースオブジェクトレコードが相互参照レコードからの優先されるセル値に更新されます。

ソースと行 ID を指定して HUB_STATE_IND の値「-1」をロードすると、ソースにかかわらず、指定した行 ID に一致するすべての相互参照レコードの HUB_STATE_IND 値が「-1」に設定されます。すべての相互参照レコードの HUB_STATE_IND 値が「-1」になるため、MDM Hub はベースオブジェクトレコードが削除されると見なします。

ベースオブジェクトの初期データロードでは、すべてのレコードがターゲットベースオブジェクトに挿入されます。このため、初期データロード後に行われる増分ロードでは、行 ID によるロードを有効にします。

監査証跡と差分検出を設定する

ステージングテーブルへのカラムのマッピングを終了したら、ステージングテーブルデータの監査証跡および差分検出を設定できます。

監査証跡を有効にすると、レコードは MDM Hub によって RAW テーブルに保持されます。ユーザーが設定したステージジョブの実行数または保持期間の間、MDM Hub はレコードが保持されます。ランディングテーブルの重複レコードは、対応する RAW テーブル内にあります。MDM Hub でステージジョブの実行回数または保持期間に達すると、ステージジョブの実行ごとに、ソースオブジェクトの各プライマリキーのレコードが 1 つ、MDM Hub によって保持されます。たとえば、設定した保持期間が 2 つのステージジョブの実行数である場合、最初の 2 つのステージジョブの実行で、ソースオブジェクトのプライマリキーのすべての重複が MDM Hub によって RAW テーブルに保持されます。2 つのステージジョブの実行後、ステージジョブの実行ごとに、ソースオブジェクトの各プライマリキーのレコードが 1 つ、RAW テーブルに保持されます。

差分検出を有効にすると、ランディングテーブルで新しいレコードまたは更新されたレコードが MDM Hub によって検出されると、そのレコードは変更されないまま、対応する RAW テーブルにコピーされます。それ以外の場合は、MDM Hub によって、レコードがターゲットテーブルにコピーされます。

監査証跡と差分検出を設定するには、**[設定]** タブをクリックします。

ステージングテーブルの監査証跡の設定

Informatica MDM Hub では、ロード数やタイムスタンプに基づいて RAW テーブルにデータの履歴を保持する監査証跡を設定できます。

この監査証跡は、例えば HDD（物理削除検出）を使用する場合などに便利です。デフォルトでは、監査証跡は無効になっていて、RAW テーブルは空です。有効にすると、ステージジョブの実行数が設定した数に達するか指定した保持期間が経過するまで、レコードが RAW テーブルに保持されます。

注: 監査証跡は、監査マネージャツールとはまったく別の機能です。混同しないように注意してください。

ステージングテーブルの監査証跡を設定する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. まだ追加していない場合は、ステージングテーブルのマッピングを追加します。
4. 設定するステージングテーブルを選択します。

5. [プロパティ] パネルの下部にある **[RAW テーブルに監査証跡を保持する]** をクリックして、RAW データの監査証跡を有効にします。
監査テーブルの保持期間を選択するように求められます。
6. 監査の保持期間として、以下のいずれかのオプションを選択します。

オプション	説明
ロード数	データを保持するバッチロードの数。
期間	データを保持する期間。

7. **[保存]** をクリックして変更を保存します。

監査証跡の設定が完了すると、指定した保持期間にわたってデータが保持されます。例えば、2つのロード（ステージジョブの実行）を保持するように監査証跡を設定したとします。この場合、監査証跡では、ステージングテーブルに対する最新の2つのロードのデータが保持されます。ランディングテーブルに各ロードのレコードが10個あったとすると、RAW テーブルのレコードの総数は20個になります。

ステージジョブが複数回実行される場合、ROWID_JOB に基づいて最新の2つのセットのデータが RAW テーブルに保持されます。古い ROWID_JOB のデータは削除されます。例えば、最初のステージジョブの ROWID_JOB の値が1で、2回目のステージジョブの値が2であるとします。この場合に3回目のステージジョブを実行すると、ROWID_JOB=1のレコードが破棄されます。

注: プロセスの初回実行後にバッチビューアの [履歴のクリア] ボタンを使用する場合は、次の点に注意してください。ステージングテーブルの監査証跡を有効にしている場合に、関連付けられたステージジョブを選択した状態でバッチビューアの [履歴のクリア] ボタンを選択すると、次にステージングジョブを実行したときに RAW テーブルと REJ テーブルのレコードがクリアされます。

ステージングテーブルの差分検出の設定

ステージングテーブルに対して差分検出を有効にすると、Informatica MDM Hub で新しいレコードと変更されたレコードだけが処理され、変更されていないレコードは無視されます。差分検出用に選択したカラムに NULL 値または将来の日付を持つレコードは、ステージングプロセスによって拒否されます。タイムラインが有効なベースオブジェクトの場合、PERIOD_START_DATE または PERIOD_END_DATE カラムに基づいて差分検出を実行すると、PERIOD_START_DATE または PERIOD_END_DATE カラムに NULL 値または将来の日付を持つレコードは拒否されます。

特定のカラムに対する差分検出の有効化

特定のカラムに対して差分検出を有効にする手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. 設定するステージングテーブルを選択します。
[ステージングテーブル] プロパティペインが表示されます。
4. **[差分検出を有効にする]** チェックボックスをオンにします。
5. **[特定のカラムを使用して差分を検出する]** オプションを選択します。
使用可能なカラムのリストが表示されます。
6. 差分検出に使用するカラムを選択し、**[追加]** >または **[すべて追加]** >>をクリックして、差分検出用のすべてのカラムを追加します。

ステージングテーブルにデータをロードする際に、定義したセットのカラムに以前のロード値と異なる値が含まれている場合は、その行が変更されたものとして処理されます。定義したセットのすべてのカラムが同じである場合は、その行が変更されていないものとして処理されます。 マップされていないカラムは無視されます。

ステージングテーブルに対する差分検出の有効化

ステージングテーブルに対して差分検出を有効にする手順

1. スキーママネージャを起動します。
2. 設定するステージングテーブルを選択します。
3. 書き込みロックを取得します。
4. **【差分検出を有効にする】** チェックボックスを選択（オン）して、テーブルの差分検出を有効にします。場合によっては、このオプションまで画面をスクロールする必要があります。
5. 差分の検出方法を指定します。

オプション	説明
マッピングのすべてのカラムを比較して差分を検出する	最終更新日を含むすべてのカラムが、差分比較の対象として選択されます。
日付カラムによって差分を検出する	スキーマに適切な日付カラムがある場合は、このオプションを選択し、差分比較に使用する日付カラムを選択します。適切な日付カラムがある場合は、このオプションの使用が推奨されます。

6. ステージプロセスまたはロードプロセスで前の重複が却下された場合に、ステージングを許可するかどうかを指定します。
 - このオプションを選択（オン）すると、前回ステージングされた重複が却下されている場合に、この次のステージプロセスの実行で、ステージングされる重複レコードの差分検出をバイパスできません。
注: このオプションを有効にした場合に、関連するステージジョブが選択された状態でユーザーがバッチビューアで **【履歴のクリア】** ボタンをクリックすると、この機能が依存している前回の却下の履歴が破棄されます。これは、ステージジョブが次回実行されるときに REJ テーブル内のレコードがクリアされるためです。
 - このオプションをクリア（オフ）すると（デフォルト）、前回ステージングされた重複が却下されている場合に、この次のステージプロセスの実行で、ステージングされる重複レコードの差分検出をバイパスできなくなります。差分検出では、次のステージプロセスの実行で後から処理される、対応する重複ランディングレコードが除外されます。

Informatica MDM Hub での差分検出の処理の仕組み

差分検出が有効になっている場合は、ステージジョブで、ランディングテーブルの内容（選択したステージングテーブルにマップされているもの）が前回のステージジョブの実行で処理されたデータセットと比較されます。

この比較によって、前回の実行後にデータが変更されたかどうかは判別されます。変更されたレコード、新規のレコード、および却下されたレコードが、ステージングテーブルに追加されます。重複するレコードは無視されます。

注: 却下されたレコードは、2 回目のステージジョブの実行後にクレンジングからロードに移行されます。

差分検出の使用に関する考慮事項

差分検出を使用する際、注意しなければならないことがいくつかあります。

次の点を考慮します。

- 差分検出は、レコード全体を比較するか、日付カラムに基づいて実行できます。差分検出の最も効率的な方法は、最終更新日に基づく方法です。この場合、Informatica MDM Hub では、各入力レコードの最終更新日カラムを、そのレコードの前の最終更新日と比較するだけで済みます。
- 差分検出には、ステージングテーブル内の `last_update_date` カラムにマップされたランディングテーブル内のカラムを差分検出に含めないようにするオプションがあります。
注: 差分検出の設定時に `last_update_date` カラムを対象に含めると、変更されたのが `last_update_date` カラムだけだった場合に、MDM Hub で、差分検出やロードに関する不要な作業が多数実行されます。
- 最終更新日に基づいてレコードを処理する場合、最終更新値の比較（例えば、ソースレコードの最終更新日が現在のシステム日付よりも前であるかどうかをテストする場合など）に `Now` クレンジング関数は使用しないでください。この方法で `Now` を使用すると、予期しない結果になる可能性があります。
- 差分検出は、最終更新日が変更の有無を正確に表していないソースのカラムに対してのみ実行してください。Informatica MDM Hub のステージジョブでは、ソースレコード全体が PRL（前回のロード）テーブル内の対応する最新のレコードと比較されます。異なるセルが見つかったら、そのレコードがステージングテーブルに渡されます。差分検出は PRL テーブルから実行されます。
- 差分検出が日付カラム（`last_update_date`、システム定義の日付カラム、またはカスタム定義の日付カラム（DOB など））に基づいている場合、その日付カラムで（RAW テーブル内で最大の日付ではなく PRL テーブル内の対応するレコードと比較して）日付値が新しいレコードのみがステージングテーブルに挿入されます。
- 差分検出が特定カラム（DOB や `last_update_date` などの日付カラムを含む）に基づいている場合、その指定したカラムで（RAW テーブル内で最大の日付ではなく PRL テーブル内の対応するレコードと比較して）日付値に何らかの変更があるレコードがステージングテーブルに挿入されます。
- 差分検出ですべてのカラムの差分を確認する場合、プライマリキーが NULL のレコードだけが却下されます。これは想定された動作です。差分プロセスで処理されなかったその他のレコードは、その後のステージプロセスで却下されます。
- 最終更新日に基づいて差分検出を実行する場合、最終更新日またはプライマリキーに対する変更が検出されます。最終更新日以外の値や連結されたプライマリキーの一部でない値に対する変更は検出されません。
- 重複したプライマリキーは、マップされたカラムに基づく差分検出を使用している場合は以降のステージプロセスで考慮されません。
- 却下処理では以下を行うことができます。
 - バッチジョブに関して特定のステージングテーブルのすべての却下レコードを表示する
 - すべてのステージングテーブルのすべての却下レコードを日付別に表示する
 - クエリフィルタに基づいて却下テーブルをクエリする

ステージングテーブルの管理

MDM Hub ステージングテーブルの設定後、ステージングテーブルのプロパティを変更する必要があることがあります。また、ステージングテーブルに関連付けられたソースシステムを表示したい場合があります。また、必要のないステージングテーブルを削除することもできます。

ステージングテーブルのプロパティの変更

ステージングテーブルのプロパティは、必要なときに変更できます。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、[ベースオブジェクト] ノードを展開し、このステージングテーブルに関連付けられているベースオブジェクトのノードを展開します。
ステージングテーブルがベースオブジェクトに関連付けられている場合は、[ステージングテーブル] ノードを展開して表示します。
4. 設定するステージングテーブルを選択します。
選択したテーブルのプロパティが表示されます。
5. ステージングテーブルのプロパティを指定します。
編集するプロパティごとに、その横にある **[編集]** ボタンをクリックし、新しい値を指定します。
注: ステージングテーブルとその関連するサポートテーブル (RAW およびプライマリランディングテーブルなど) が空の場合、ソースシステムを変更できます。
ステージングテーブルまたはその関連テーブルにデータが含まれる場合は、ソースシステムを変更しないでください。
6. ベースオブジェクトのカラムのリストで、ソースシステムで提供するカラムを変更します。
 - **[すべて選択]** ボタンをクリックすると、カラムを1つずつクリックしなくても、すべてのカラムを選択できます。
 - **[すべてクリア]** ボタンをクリックすると、すべてのカラムのチェックを解除できます。**注:** [行 ID オブジェクト] と [最終更新日] が自動的に選択されます。これらのカラムのチェックを解除したり、プロパティを変更したりすることはできません。
7. 必要に応じて、カラムのプロパティを変更します。
8. 必要に応じて、外部キーカラムのルックアップを変更します。カラムを選択し、**[編集]** ボタンをクリックしてルックアップカラムを設定します。
9. セルの更新を変更する場合は、**[セルの更新]** チェックボックスをクリックします。
10. 必要に応じて、ステージングテーブルのカラムの設定を変更します。
11. 必要に応じて、このステージングテーブルの監査証跡と差分検出を設定します。
12. **[保存]** ボタンをクリックして変更を保存します。

ステージングテーブルのソースシステムへの移動

ステージングテーブルに関連付けられたソースシステムを表示するには、ステージングテーブルを右クリックし、**[ソースシステムに移動]** を選択します。

Hub コンソールでシステムと信頼ツールが起動され、このステージングテーブルに関連付けられたソースシステムが表示されます。

ステージングテーブルの削除

ステージングテーブルを削除する手順

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、**[ベースオブジェクト]** ノードを展開し、このステージングテーブルに関連付けられているベースオブジェクトのノードを展開します。
4. 削除するステージングテーブルを右クリックして、**[削除]** を選択します。
削除を確認するように求められます。
5. **[はい]** を選択します。

ステージングテーブルがオペレーショナル参照ストア（オペレーショナル参照ストア）から削除されます。関連する制御テーブルも削除され、また削除したステージングテーブルがスキーマツリーから削除されます。

マッピングの管理

場合によっては、マッピングのプロパティを変更したり、マッピングを削除したり、マッピングに関連付けられたスキーマを表示する必要があります。また、ステージングの実行前にマッピングをテストする可能性があります。

マッピングのプロパティの編集

既存のマッピングをコピーして新しいマッピングを作成する手順

1. マッピングツールを起動します。
2. 書き込みロックを取得します。
3. 編集するマッピングを選択します。
4. マッピングのプロパティ、ダイアグラム、設定を、必要に応じて編集します。
5. **[保存]** ボタンをクリックして変更を保存します。

マッピングのコピー

既存のマッピングをコピーして新しいマッピングを作成する手順

1. マッピングツールを起動します。
2. 書き込みロックを取得します。
3. コピーするマッピングを右クリックし、**[マッピングのコピー]** を選択します。
[マッピング] ダイアログが表示されます。
4. マッピングのプロパティを指定します。
5. **[OK]** をクリックします。
6. **[保存]** ボタンをクリックして変更を保存します。

スキーマへの移動

マッピングツールを使用してスキーママネージャを素早く起動し、選択したマッピングに関連付けられているスキーマを表示することができます。

注: [スキーマに移動] コマンドは、ワークベンチビューだけにあり、プロセスビューにはありません。

マッピングのスキーマに移動する手順

1. マッピングツールを起動します。
2. スキーマを表示するマッピングを選択します。
3. ナビゲーションペインの下部にある [表示方法] リストで、次のいずれかのオプションを選択します。
 - ステージングテーブル別
 - ランディングテーブル別
 - マッピング別
4. ナビゲーションペイン内で右クリックし、[スキーマに移動] を選択します。
マッピングツールで、選択したマッピングのスキーマが表示されます。

マッピングのテスト

設定したマッピングをテストする手順

1. マッピングツールを起動します。
2. 書き込みロックを取得します。
3. 設定するマッピングを選択します。
4. [テスト] タブをクリックします。
このマッピングの [テスト] タブが表示されます。
5. [入力名] でカラムの入力値を指定します。
6. [テスト] をクリックします。
マッピングがテストされ、[出力名] の下のカラムに結果が入力されます。

マッピングの削除

マッピングを削除する手順

1. マッピングツールを起動します。
2. 書き込みロックを取得します。
3. 削除するマッピングを右クリックして、[マッピングの削除] を選択します。
削除を確認するように求められます。
4. [はい] をクリックします。
サポートされるテーブルが削除され、メタデータからマッピングが削除されてマッピングのリストが更新されます。

第 18 章

物理削除の検出

この章では、以下の項目について説明します。

- [物理削除の検出の概要, 324 ページ](#)
- [物理削除の検出のタイプ, 325 ページ](#)
- [ベースオブジェクトの削除フラグ値, 325 ページ](#)
- [信頼および検証ルール, 325 ページ](#)
- [物理削除の検出テーブル, 326 ページ](#)
- [物理削除の検出の設定, 327 ページ](#)
- [物理削除の検出用のプライマリーキーカラムの指定, 330 ページ](#)
- [直接削除, 331 ページ](#)
- [合意削除, 334 ページ](#)
- [ユーザーイグジット内での物理削除の検出の使用, 339 ページ](#)

物理削除の検出の概要

ベースオブジェクトに値を供給するソースシステムは絶えず更新され、レコードがそれぞれのソースシステムから物理削除されることがあります。物理削除されたレコードとは、ソースシステムから削除されたレコードです。MDM Hub は、ソースシステムで物理削除されたレコードを検出でき、関連するベースオブジェクトに変更内容を反映することができます。

注: Oracle および Microsoft SQL Server 環境の MDM Hub では、ソースシステムで物理削除されたレコードを検出できます。

この MDM Hub ステージジョブは、最新のソースシステムファイルの中のすべてのレコードを、前回のランディングテーブルの中のすべてのレコードと比較します。MDM Hub は、最新のソースシステムで欠けているレコードを検出して、フルロードに対して物理削除を示すフラグを付けます。次に MDM Hub は、削除されたレコードのカラム値を、終了日（任意）と削除フラグとともにランディングテーブルに再挿入します。最後に MDM Hub が、関連するベースオブジェクトを更新します。

MDM Hub は、増分ロードやトランザクションによるロードでは直前のソースシステムファイルの中に入っていないレコードは検出しません。物理削除の検出テーブルをリポトリテーブル内に作成して、物理削除を設定する必要があります。MDM Hub は、物理削除を検出するときに、削除済みのフラグが付いているレコードの数をジョブメトリックテーブル内に挿入します。ステージプロセスの一部である Java ユーザーイグジットとして、物理削除の検出を実装します。

物理削除の検出のタイプ

選択する物理削除の検出のタイプは、ベースオブジェクトレコードに関連付けられたソースシステムと相互参照レコードに応じて異なります。

レコードの削除状態の決定を単一のソースシステムが行った場合、MDM Hub は直接削除のベースオブジェクトレコードにフラグを付けます。レコードの削除状態を決定したのが複数のソースシステムだった場合、MDM Hub は合意削除を行います。MDM Hub は、すべてのソースシステムのレコードが削除済みの状態になっているか調べ、削除フラグをベースオブジェクトレコードに割り当てます。

ベースオブジェクトの削除フラグ値

ベースオブジェクトのデータを統合する方法を定義するときは、削除フラグを特定し、削除フラグ値を決定する必要があります。

MDM Hub には、デフォルトの削除フラグ値があります。ベースオブジェクトレコードに対して削除フラグと共に終了日の値も提供するように MDM Hub を設定することができます。

以下に、ベースオブジェクトの削除フラグとそのデフォルト値を示します。

アクティブ

すべてのソースシステムでそのレコードがアクティブであることを示します。値を変更するには、DELETE_FLAG_VAL_ACTIVE カラムの値を別の値に設定します。デフォルトは A です。

非アクティブまたは完全に削除済み

すべてのソースシステムでそのレコードが物理削除されていることを示します。値を変更するには、DELETE_FLAG_VAL_INACTIVE カラムの値を別の値に設定します。デフォルト値は I です。

部分的に削除済み

一部のソースシステムでそのレコードが削除されていることを示します。値を変更するには、DELETE_FLAG_VAL_PARTIAL カラムの値を別の値に設定します。デフォルトは P です。

削除フラグおよび想定される終了日を必要とするベースオブジェクトを決定します。ベースオブジェクトに関連するランディングテーブルとステージングテーブル内の適切なカラムが含まれているか確認してください。また、リポジトリ内の物理削除の検出テーブルを、ステージングテーブルから物理削除を検出するように設定したかも検証してください。

信頼および検証ルール

MDM Hub がソースシステムからの削除済みレコードにフラグを設定した後は、信頼および検証のルールによって、削除されたフラグと終了日の動作が決まります。信頼ルールと検証ルールは、最も信頼できるデータの決定に役立ちます。

MDM Hub は、ベースオブジェクトレコード内のカラムの値がソース内で物理削除されていることを検出するときに、信頼ルールと検証ルールを使用します。信頼ルールと検証ルールは、これらのベースオブジェクトカラムが他のソースシステムからの値によって上書きされるべきタイミングを規定しています。MDM Hub は検証ルールを使用して、削除済みソースレコードに関してすべての信頼済みカラムの信頼率をダウングレードします。削除済みソースレコードによって提供されたベースオブジェクトレコードの値が、ベースオブジェクトレコード内に残ります。MDM Hub は、別のソースシステムが信頼度のより高い更新を削除済みソースレコードからの値を上書きするために提供した後に、削除された値を置き換えます。

物理削除の検出テーブル

物理削除の検出テーブルは、MDM Hub がソースシステムでの物理削除の検出に必要とするメタデータを格納するために作成するリポジトリテーブルです。物理削除の検出テーブルの名前は C_REPOS_EXT_HARD_DEL_DETECT です。

次の表で、物理削除の検出テーブルのカラムについて説明します。

カラム名	データ型とサイズ (単位: バイト)	説明
ROWID_TABLE	CHAR (14)	物理削除されたレコードを検出する必要があるステージングテーブルの、ROWID_OBJECT カラムの値を含みます。ROWID_TABLE カラムには NULL 値を使用できません。
HDD_ENABLED	INTEGER	MDM Hub によるステージングテーブルへの物理削除の検出を有効または無効にします。HDD_ENABLED カラムには NULL 値を使用できません。 以下のいずれかの値を使用します。 <ul style="list-style-type: none">- 0. 物理削除が検出できるように MDM Hub を有効にします。- 1. 物理削除を検出しないように MDM Hub を無効にします。 デフォルトは 0 です。
HDD_TYPE	VARCHAR2 (14)	ステージングテーブルの物理削除のタイプを示します。次の値のうちいずれかを物理削除タイプに使用してください。 <ul style="list-style-type: none">- DIRECT。ベースオブジェクトレコードの直接物理削除を実行します。- CONSENSUS。合意によるベースオブジェクトレコードの物理削除を実行します。
DELETE_FLAG_COLUMN_NAME	VARCHAR2 (30)	MDM Hub が検出する物理削除されたレコードに関する削除フラグの値が含まれているステージングテーブルカラムの名前。
DELETE_FLAG_VAL_ACTIVE	VARCHAR2 (14)	アクティブなレコードの削除フラグカラムの値。
DELETE_FLAG_VAL_INACTIVE	VARCHAR2 (14)	アクティブでないレコードの削除フラグカラムの値。
DELETE_FLAG_VAL_PARTIAL	VARCHAR2 (14)	合意削除シナリオにおける部分削除の削除フラグカラムの値。
HAS_END_DATE	INTEGER	ステージングテーブルに終了日カラムがあるかどうかを示します。HAS_END_DATE カラムには NULL 値を使用できません。 以下のいずれかの値を使用します。 <ul style="list-style-type: none">- 0。ステージングテーブルに終了日カラムがありません。- 1。ステージングテーブルに終了日カラムがあります。 デフォルトは 0 です。

カラム名	データ型とサイズ（単位：バイト）	説明
END_DATE_COLUMN_NAME	VARCHAR2 (30)	終了日カラムの名前。 HAS_END_DATE=1 の場合、MDM Hub はこのカラムを使用します。
END_DATE_VAL	DATE	終了日に使用する値。 終了日の値をハードコードで実装することが必要な場合、その値は END_DATE_VAL カラムに MM/DD/YYYY 形式で保存されます。 デフォルトは SYSDATE です。
TRAN_HDD_ENABLED	INTEGER	物理削除を検出する処理がトランザクショナルかどうかを示します。TRAN_HDD_ENABLED カラムには NULL 値を使用できません。 以下のいずれかの値を使用します。 - 0. 物理削除の検出処理はトランザクショナルではありません。 - 1. 物理削除の検出処理はトランザクショナルであり、前回のランディングとランディングテーブルに対してフルロードを比較しません。 デフォルトは 0 です。

物理削除の検出の設定

ソースシステム内の物理削除を検出するように MDM Hub を設定することができます。

物理削除を検出できるように MDM Hub を設定するには、物理削除の検出テーブルを作成してジョブメトリックタイプを登録する必要があります。さらに、ランディングテーブルとステー징テーブルを設定し、マッピングを作成する必要もあります。ランディングテーブルとステー징テーブルの設定が終わったら、物理削除の検出テーブルにステー징テーブルの情報を指定します。最後に、物理削除されたレコードを検出するための物理削除の検出機能呼び出すユーザーイグジットを実装します。

注: 物理削除の検出テーブルの設定のために提供されている SQL ステートメントをすべて結合して、1 組の SQL ステートメントとして実行することができます。

1. 物理削除の検出テーブルを設定します。
 - a. 物理削除の検出テーブルを作成するには、次の SQL ステートメントを実行します。

Oracle の場合

```
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
);

```

Microsoft SQL Server の場合

```

CREATE TABLE [dbo].[C_REPOS_EXT_HARD_DEL_DETECT]
(
    [ROWID_TABLE]                [nvarchar](14)          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. 物理削除の検出テーブルに制約を追加するには、次の SQL 文を実行します。

Oracle の場合

```

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));

```

Microsoft SQL Server の場合

```

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. C_REPOS_TABLE の ROWID_TABLE カラムへの物理削除の検出テーブルの ROWID_TABLE カラムに外部キー制約を作成するには、次の SQL ステートメントを実行します。

Oracle の場合

```

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);

```

Microsoft SQL Server の場合

```
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
```

親レコードが C_REPOS_TABLE から削除された場合に MDM Hub が適切なメタデータを削除できるように、ON DELETE CASCADE 句で外部キーを定義します。

2. ジョブメトリックタイプのコードをジョブメトリックタイプテーブルに登録します。

次の SQL ステートメントを実行して、ジョブメトリックタイプのコードを 100 として登録します。

Oracle の場合

```
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);
```

Microsoft SQL Server の場合

```
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. ランディングテーブルとステージングテーブルを設定します。

- a. 削除フラグと終了日の必要なベースオブジェクトレコードを決定します。
- b. ベースオブジェクトにデータを供給するランディングカラムとステージングテーブルカラムをすべて含めたか確認します。
- c. Hub コンソールのスキーマツールで、物理削除の検出テーブルの中の DELETE_FLAG_COLUMN_NAME に指定した値が表示されているか確認します。また、デフォルトの値が A か I か P になっていることを確認します。
- d. そのレコードがアクティブな場合、およびレコードに終了日がある場合は、その終了日が NULL またはシステムの日付になっているか確認します。

終了日に NULL 値を指定している場合、終了日カラムに対して **【NULL の更新を許可する】** チェックボックスにチェックマークを付けます。

- e. Hub コンソールの **【マッピング】** ツールで、ランディングテーブルとステージングテーブルの間でカラム同士を割り付けます。
- f. Hub コンソールの **【スキーマ】** ツールで、物理削除の検出に使用なステージングテーブルの差分検出を有効にします。

特定のカラムを使用することによる差分検出のオプションを有効にする場合は、削除されたレコードが検出されるようにカラムリストに削除フラグカラムの名前を含めます。

4. 物理削除の検出テーブルでメタデータを定義するには、次の SQL ステートメントを実行します。

Oracle の場合

```
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>'

```

Microsoft SQL Server の場合

```

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. 物理削除の検出用のユーザーイグジットを実装します。

- a. ランディング後およびステージ後の Java ユーザーイグジットを実装し、物理削除機能呼び出すロジックを実装先に追加します。
実装するユーザーイグジットのインターフェースは、mdm-ue.jar にあります。
- b. 実装されたランディング後およびステージ後のユーザーイグジットクラスを、JAR ファイルにパッケージ化します。
- c. パッケージ化された JAR ファイルを、ユーザーイグジットを登録することによって MDM Hub にアップロードします。

物理削除の検出用のプライマリキーカラムの指定

関数を使用して複数のソースカラムからプライマリキーを作成する場合、物理削除の検出プロセスは、関数への入力を参照してソースカラムを特定します。物理削除の検出プロセスは、関数への入力であるすべてのソースカラムに対して実行されます。

注: 関数へのすべての入力がプライマリキーに渡される場合、この手順を省略できます。

関数への入りにプライマリキーで使えないソースカラムが含まれる場合、プライマリキーに渡されるカラム名のリストを作成します。カラムのリストを C_REPOS_EXT_HARD_DEL_DETECT リポジトリテーブルに追加します。物理削除の検出プロセスは、リスト内のカラムに対して実行されます。

1. MDM Hub がプライマリキーの作成に使用する、ランディングテーブル内のカラムを特定します。カラム名のカンマ区切りリストを作成します。

2. SQL ツールで、物理削除の検出を設定したときに作成した C_REPOS_EXT_HARD_DEL_DETECT リポジトリテーブルを開きます。
3. リポジトリテーブルを作成したときに HDD_LANDING_PKEY_COLUMNS カラムを追加しなかった場合は、カラムを追加します。カラムタイプを VARCHAR に設定し、カラム名のカンマ区切りリストを含めることができる長さを入力します。
4. カラムに、カラム名のカンマ区切りリストを追加します。

ステージジョブが実行されると、検証プロセスがカラム名のリストを検証します。検証プロセスはスペースを無視し、重複したカラム名を削除します。また、カラム名がランディングテーブル内のカラム名に一致することを確認します。

直接削除

MDM Hub は、物理削除されたレコードを単一のソースシステムに基づいて検出します。ソースシステムのレコードが削除されているか、その相互参照レコードの 1 つが削除されている場合、MDM Hub はベースオブジェクトレコードに、アクティブではないことを示すフラグを付けます。

ソースデータが単一ソースのシステムから来ている場合に、直接削除を実行します。レコードの削除状態の決定を単一ソースのシステムで行う場合にも、直接削除を実行します。MDM Hub は、そのソースシステムの直近のステータスに基づいて削除ステータスを決定します。ランディング後のユーザーイグジットは直接削除を実行できます。

直接削除フラグ設定用の物理削除の検出設定

直接削除フラグ設定を設定するには物理削除の検出テーブルを設定する必要があります。直接物理削除の検出を実行するときは、削除フラグ値、レコードの終了日（任意）、そして一部のベースオブジェクトプロパティが必要です。

直接削除フラグ設定用に次のメタデータを設定することができます。

削除フラグ

直接削除を使って物理削除を検出する必要があるときは、以下の削除フラグを使用します。

- DELETE_FLAG_VAL_ACTIVE = A
- DELETE_FLAG_VAL_INACTIVE = I
- HDD_ENABLED = 1
- HDD_TYPE = DIRECT

終了日

HAS_END_DATE=1 に設定した場合、終了日の値 (END_DATE_VAL) を指定します。レコードの終了日は、過去の日付にすることも将来の日付にすることもできます。終了日の形式は MM/DD/YYYY です。

ベースオブジェクトのカラム

削除フラグカラムの名前や終了日カラムの名前など、ベースオブジェクトのカラムを指定します。

終了日による直接削除用の物理削除の検出の設定

物理削除を検出するための一般的な設定を完了したら、ステージングテーブルの終了日による直接削除用のプロセスを設定します。

1. 物理削除が検索できるように MDM Hub を設定します。
2. MDM Hub が物理削除を必ず検出しなければならないステージジョブを特定します。
3. ステージングテーブルの詳細とそのほかの必須メタデータを、物理削除の検出テーブルに挿入します。

終了日コードのある直接削除の例

自分の会社で C_CUSTOMER_STG というステージングテーブルに関して物理削除を検出する必要があったとします。社員たちは、ベースオブジェクト内の物理削除されたレコードの直接削除を実行して終了日を削除済みレコードに割り付けようと考えています。

物理削除の検出に必要なメタデータを、C_CUSTOMER_STG ステージングテーブルの物理削除の検出テーブルに挿入する必要があります。物理削除を検出する場合、MDM Hub が、削除されたレコードの終了日に加え、削除フラグを提供する必要があります。例の中で、削除フラグカラムの名前は DEL_CODE で、ステージングテーブルの終了日カラムの名前は END_DATE です。

次のサンプルコードでは、削除済みレコードの終了日で直接削除を行うためにメタデータを物理削除の検出テーブルに挿入しています。

Oracle の場合

```
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';
```

Microsoft SQL Server の場合

```
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
```

直接削除用の物理削除の検出の例

自分の会社では単一のソースシステムからベースオブジェクトレコードにデータが供給されているとします。この場合、ソースシステムで物理削除を検出して、ベースオブジェクトレコードに削除済みを表すフラグを付

ける必要があります。ソースシステムは、レコードを削除し、そのレコードを再アクティブ化し、それから再アクティブ化されたレコードを削除します。MDM Hub はそれに合わせて、削除済みで終了日が過ぎたことを示すフラグをソースレコードに付けます。

物理削除の検出前のレコード

MDM Hub がソースシステムで物理削除を検出する前は、ベースオブジェクトレコードと相互参照レコードに対し削除フラグがアクティブです。

ソースで物理削除が検出される前の相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Robert	James	Jones	A	null

ソースで物理削除が検出される前のベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	A	null

このソースで物理削除が検出されたときのレコード

ソースシステムに物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングテーブルを前回のランディングテーブルと比較することで、削除されたレコードを検出します。
2. 削除されたレコードのカラム値を、削除フラグと終了日とともにランディングテーブルに再挿入します。
3. 相互参照レコードを更新して終了日に非アクティブになるようにします。

終了日には、システムの日付やユーザーが終了日の値を決めるときに指定した日付が含まれます。

ソースで物理削除が検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Robert	James	Jones	I	04/05/13

ソースで物理削除が検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	I	04/05/13

ソースのレコードが再アクティブ化された後のレコード

レコードがソースシステムで再アクティブ化されている場合、MDM Hub はベースオブジェクトレコードと相互参照レコードに、アクティブであることを示すフラグを付けます。ソースレコードがアクティブなとき、終了日は NULL です。

ソースシステムで物理削除されたレコードが再びアクティブになった後の相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Robert	James	Jones	A	null

ソースシステムで物理削除されたレコードが再びアクティブになった後のベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	A	null

このソースで物理削除が再び検出されたときのレコード

ソースシステムに物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングテーブルを前回のランディングテーブルと比較することで、削除されたレコードを検出します。
2. 削除されたレコードのカラム値を、削除フラグと終了日とともにランディングテーブルに再挿入します。
3. 相互参照レコードを更新して終了日に非アクティブになるようにします。

終了日には、システムの日付やユーザーが終了日の値を決めるときに指定した日付が含まれます。

ソースで物理削除が検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Robert	James	Jones	I	04/05/13

ソースで物理削除が検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	I	04/05/13

合意削除

MDM Hub は、すべての提供ソースシステムでレコードの削除ステータスに基づいて、物理削除されたレコードを検出します。直接削除と併せて、合意削除を使用します。

すべての提供ソースシステムからレコードが削除されている場合、MDM Hub はベースオブジェクトレコードにアクティブではないことを示すフラグを付けます。ベースオブジェクトレコードに関連付けられたすべての相互参照レコードが、部分的に非アクティブな状態である必要があります。

合意削除は、ソースデータが複数のソースシステムから来た場合に実行します。レコードを削除することについてすべてのソースシステム間で合意がある場合、MDM Hub は削除のレコードにフラグを付けます。MDM Hub は、ソースシステムからのレコードに、部分的に非アクティブであることを示すフラグを付けます。次に MDM Hub は、信頼ルールと検証ルールに基づいて、ベースオブジェクトレコードを上書きする他のソースシステムレコードを決めます。削除日が存在する場合、MDM Hub は削除日に基づいて、ベースオブジェクトレコードを上書きする他のソースシステムレコードを決めます。

ソースシステムからのレコードの削除状態に関して合意が必要なときは、ランディング後のユーザーイグジットを実装して直接削除を呼び出してください。また、ステージング後のユーザーイグジットでベースオブジェクトレコードの合意削除用の合意削除ロジックを呼び出す必要もあります。

合意削除フラグ設定用の物理削除の検出設定

合意削除フラグ設定を設定するには物理削除の検出テーブルを設定する必要があります。合意物理削除の検出を実行するとき、削除フラグ値、レコードの終了日、そして一部のベースオブジェクトプロパティが必要です。

合意削除フラグ設定用に次のメタデータを設定することができます。

削除フラグ

合意削除を実行する必要があるときは、以下の削除フラグを使用します。

- DELETE_FLAG_VAL_ACTIVE = A
- DELETE_FLAG_VAL_INACTIVE = I

- DELETE_FLAG_VAL_PARTIAL = P
- HDD_ENABLED = 1
- HDD_TYPE = CONSENSUS

終了日

HAS_END_DATE=1 に設定した場合、終了日の値 (END_DATE_VAL) を指定します。レコードの終了日は、過去の日付にすることも将来の日付にすることもできます。形式は MM/DD/YYYY です。

ベースオブジェクトのカラム

削除フラグカラムの名前や終了日カラムの名前など、ベースオブジェクトのカラムを指定します。

終了日による合意削除用の物理削除の検出の設定

物理削除を検出するための一般的な設定を完了したら、ステージングテーブルの終了日による合意削除用のプロセスを設定します。

1. 物理削除が検索できるように MDM Hub を設定します。
2. MDM Hub が物理削除を必ず検出しなければならないステージジョブを特定します。
3. ステージングテーブルの詳細とそのほかの必須メタデータを、物理削除の検出テーブルに挿入します。
4. **【スキーマ】** ツールで、物理削除されたレコードに関する削除フラグの値を含んだカラム名に対して、**【信頼】** と **【検証】** のチェックボックスを有効にします。
5. 終了日を使用する場合は、終了日のカラム名に対して **【信頼】** と **【検証】** のチェックボックスを有効にします。
6. **システムと信頼** ツールで、削除フラグのカラム名に対する信頼設定と各ソースシステムに対する終了日のカラム名を指定します。

減衰の値は、ベースオブジェクトのすべてのソースシステムについて同じでなければなりません。

7. **スキーマ** ツールで、物理削除の検出に構成した各ベースオブジェクトに対する削除フラグのカラム名について検証ルールを設定します。部分削除フラグを、アクティブなフラグまたは完全に削除されたフラグよりも低い信頼値に、設定する必要があります。
 - a. ベースオブジェクトに対して **【検証ルールのセットアップ】** を選択します。
【検証ルール】 ページが表示されます。
 - b. **【検証ルールの追加】** ボタンをクリックします。
【検証ルールの追加】 ダイアログボックスが表示されます。
 - c. ルール名を入力し、リストから **【カスタム】** ルールタイプを選択します。
注: ルール名にはカンマを挿入しないでください。カンマを挿入すると、検証ルールの順序が乱れる場合があります。
 - d. ダウングレード率を選択し、**【ルール SQL】** セクションに次の形式を使用してルールを指定します。
WHERE S.<Delete flag column name> LIKE '<Value of delete flag column for partial delete>'
カラムに対して **【最小信頼度の保持】** を有効にしてはなりません。

終了日コードのある合意削除の例

自分の会社で C_CUSTOMER_STG というステージングテーブルに関して物理削除を検出する必要があったとします。社員たちは、ベースオブジェクト内の物理削除されたレコードの合意削除を実行して終了日を削除済みレコードに割り付けようと考えています。

物理削除の検出に必要なメタデータを、C_CUSTOMER_STG ステージングテーブルの物理削除の検出テーブルに挿入する必要があります。物理削除を検出する場合、MDM Hub が、削除されたレコードの終了日に加え、削除フラグを提供する必要があります。また、この処理では、MDM Hub がベースオブジェクトレコードに削除済みを示すフラグを付ける前に、関連するすべての相互参照レコードに削除済みのフラグが付いていることを検証する必要があります。削除フラグカラムの名前が DEL_CODE で、ステージングテーブルの終了日カラムの名前が END_DATE です。

次のサンプルコードでは、削除済みレコードの終了日で合意削除するためにメタデータを物理削除の検出テーブルに挿入しています。

Oracle の場合

```
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;
```

Microsoft SQL Server の場合

```
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
```

合意削除用の物理削除の検出の例

自分の会社では複数のソースシステムからベースオブジェクトレコードにデータが供給されているとします。各ソースシステムのレコードが次から次へと削除され、ソースシステム 1 からのレコードが再アクティブ化されてから、再び物理削除されます。MDM Hub はソースシステムの物理削除を検出する必要があります。レコードがすべてのソースシステムで物理削除された場合、ベースオブジェクトレコードに削除済みを示すフラグが付けられます。

MDM Hub は、削除されたソースレコードに関する信頼度のダウングレードに基づいて、ベースオブジェクトレコードのカラム値を更新します。すべてのソースシステムのレコードは物理削除される必要があります、MDM

Hub は、すべての関連する相互参照レコードに、非アクティブまたは部分的に非アクティブであることを示すフラグを付ける必要があります。

物理削除の検出前のレコード

ソースシステムで物理削除を検出する前は、ベースオブジェクトレコードと相互参照レコードに対する削除フラグがアクティブになっています。MDM Hub は、3 つのソースレコードを 1 つのベースオブジェクトレコードにマージします。個人の名前はソースシステム 2 から、ミドルネームはソースシステム 3 から、姓はソースシステム 1 から来ています。

ソースで物理削除が検出される前の相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Bob	J	Jones	A	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	A	null

ソースで物理削除が検出される前のベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	A	null

ソース 3 で物理削除が検出されたときのレコード

ソースシステム 3 に物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングページから削除されたソースレコードを検出します。
2. 削除されたレコードのカラム値を、アクティブではない削除フラグと終了日とともにランディングテーブルに再挿入します。
3. 相互参照レコードのすべてが非アクティブまたは部分的に非アクティブであるか確認し、関連する相互参照レコードを部分的に非アクティブに更新します。
4. ベースオブジェクト内にソースシステム 3 からのレコードによって供給されたミドルネームを、信頼スコアの高いソースレコードがそのカラムに更新を行うまで、保持します。

ソースシステム 3 で物理削除が検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Bob	J	Jones	A	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	P	null

ソースシステム 3 で物理削除が検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	James	Jones	A	null

ソース 1 で物理削除が検出されたときのレコード

ソースシステム 1 に物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングページから削除されたソースレコードを検出します。
2. 削除されたレコードのカラム値を、削除フラグと終了日とともにランディングテーブルに再挿入します。
3. 相互参照レコードのすべてが非アクティブまたは部分的に非アクティブであるか確認し、関連する相互参照レコードを部分的に非アクティブに更新します。
4. BVT が計算され、ベースオブジェクトがソース 1 からの値をミドルネーム (John) に反映します (そのレコードは部分的に削除されているものの信頼スコアが高いため)。

ソースシステム 1 で物理削除が検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Bob	John	Jones	P	null
2	163	Robert	J	Jones	A	null
3	163	Bob	James	Jones	P	null

ソースシステム 1 で物理削除が検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	John	Jones	A	null

ソース 2 で物理削除が検出されたときのレコード

ソースシステム 2 に物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングページから削除されたソースレコードを検出します。
2. カラム値を、削除フラグと終了日とともにランディングテーブルに再挿入します。
3. 相互参照レコードのすべてが非アクティブまたは部分的に非アクティブであるかを確認し、関連する相互参照レコードを非アクティブとして更新します。
この時点で、他のどのソースシステムもアクティブなステータスを提供することができません。
4. 他のどのソースシステムもアクティブなレコードを提供できない場合、ベースオブジェクトレコードは非アクティブ状態を終了日で反映します。

終了日には、システムの日付やユーザーが終了日の値を決めるときに指定した日付が含まれます。

ソースシステム 2 で物理削除が検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Bob	John	Jones	P	null
2	163	Robert	J	Jones	I	04/05/13
3	163	Bob	James	Jones	P	null

ソースシステム 2 で物理削除が検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	John	Jones	I	04/05/13

ソース 1 のレコードが再アクティブ化された後のレコード

レコードがソースシステムで再アクティブ化されている場合、MDM Hub はベースオブジェクトレコードと相互参照レコードに、アクティブであることを示すフラグを付けます。ソースレコードがアクティブなとき、終了日は NULL です。

ソースシステム 1 のソースレコードが再アクティブ化され、MDM Hub が関連する相互参照レコードにアクティブであることを示すフラグを付けます。レコードがアクティブなとき、終了日は NULL です。

ソースシステム 1 で物理削除されたレコードが再びアクティブになった後の相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
1	25	Bob	John	Jones	A	null
2	163	Robert	J	Jones	I	04/05/13
3	163	Bob	James	Jones	P	null

ソースシステム 1 で物理削除されたレコードが再びアクティブになった後のベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	John	Jones	A	null

ソース 1 で物理削除が再び検出されたときのレコード

ソースシステム 1 に物理削除があるとき、MDM Hub は以下のタスクを実行します。

1. ランディングページから削除されたソースレコードを検出します。
 2. 削除されたレコードのカラム値を、削除フラグと終了日とともにランディングテーブルに再挿入します。
 3. 相互参照レコードのすべてが非アクティブまたは部分的に非アクティブであるかを確認し、関連する相互参照レコードを非アクティブとして更新します。
この時点で、他のどのソースシステムもアクティブなステータスを提供することができません。
 4. ベースオブジェクトレコードは、終了日を使って非アクティブな状態を反映します。
- 終了日には、システムの日付やユーザーが終了日の値を決めるときに指定した日付が含まれます。

ソースシステム 1 で物理削除が再び検出されたときの相互参照レコード。

ソース	顧客 ID	名	ミドルネーム	姓	削除コード	終了日
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

ソースシステム 1 で物理削除が再び検出されたときのベースオブジェクトレコード。

顧客 ID	名	ミドルネーム	姓	削除コード	終了日
25	Robert	John	Jones	I	04/05/13

ユーザーイグジット内での物理削除の検出の使用

ランディング後およびステージ後のユーザーイグジットを実装すると、ソースシステム中の物理削除されたレコードを検出することができます。直接削除の検出を行う場合は、ランディング後のユーザーイグジットが必要です。合意物理削除の検出を行う場合は、ランディング後とステージ後の両方のユーザーイグジットが必要です。

1. ランディング後のユーザーイグジットの実装内部で使用する `HardDeleteDetection` クラスのインスタンスを作成します。

`mdm-ue.jar` 内で `HardDeleteDetection` クラスを利用できます。これは次のディレクトリにあります。

Windows の場合:<infamdm installation directory>\hub\server\lib

UNIX の場合:<infamdm installation directory>/hub/server/lib

2. ランディング後やステージ後のユーザーイグジットの Java コードに次の行を追加して、物理削除されたレコードを検出できるようにします。

- ランディング後のユーザーイグジットの場合

```
public void processUserExit(UserExitContext userExitContext, String stagingTableName, String
landingTableName, String previousLandingTableName)
    throws Exception
{
    HardDeleteDetection hdd = new HardDeleteDetection(userExitContext.getBatchJobRowid(),
stagingTableName);
    hdd.startHardDeleteDetection(userExitContext.getDBConnection());
}
```

- ステージ後のユーザーイグジットの場合

```
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. ユーザーイグジットを JAR にパッケージ化し、それを MDM Hub にアップロードします。
4. ステージジョブを実行します。
ステージジョブでユーザーイグジットが呼び出されると、MDM Hub により物理削除を検出するための入力パラメータ値が提供されます。

関連項目：

- [「ユーザーイグジット JAR ファイルの実装」 \(ページ 588\)](#)
- [「MDM Hub へのユーザーイグジットのアップロード」 \(ページ 588\)](#)

第 19 章

データクレンジングの設定

この章では、以下の項目について説明します。

- [データクレンジングの設定の概要, 341 ページ](#)
- [MDM Hub でデータクレンジングを設定する, 341 ページ](#)
- [プロセスサーバーでデータクレンジングを設定する。 , 342 ページ](#)
- [クレンジング関数の設定, 347 ページ](#)
- [関数のテスト, 355 ページ](#)
- [クレンジング関数での条件の使用, 355 ページ](#)
- [クレンジングリストの設定, 356 ページ](#)

データクレンジングの設定の概要

ステージプロセス中にデータをクレンジングおよび標準化するように、MDM Hub でデータクレンジングを設定できます。クレンジングしたデータにマッチングを実行すると、MDM Hub により、信頼できる一致が多数生成されます。

MDM Hub でデータクレンジングを実行する前に、クレンジングエンジンをインストールして設定します。また、MDM Hub 環境用に Process サーバーを設定する必要があります。Process Server は、データをクレンジングしてバッチジョブを処理するサプレットです。レコード内の値を標準化または検証するクレンジング関数を設定できます。

MDM Hub でデータクレンジングを設定する

MDM Hub でデータクレンジングを設定するには、次のタスクを実行します。

- プロセスサーバーでデータクレンジングを設定する。
- クレンジング関数を設定する。
- クレンジングリストを設定する。

プロセスサーバーでデータクレンジングを設定する。

MDM Hub 実装にプロセスサーバーを設定できます。

プロセスサーバーは、データをクレンジングしてバッチジョブを処理するサブレットです。このプロセスサーバーをアプリケーションサーバー環境にデプロイします。

プロセスサーバーは、各インスタンスで同時に複数の要求を処理できるようにマルチスレッド対応になっています。

MDM Hub 内のオペレーショナルリファレンスストアごとに、複数のプロセスサーバーを実行できます。クレンジングプロセスは、一般に CPU 依存のプロセスです。このスケーラブルなアーキテクチャは、データ量の増加に合わせて MDM Hub 実装の規模を拡張するために利用できます。プロセスサーバーを複数のホストにデプロイすることで、処理の負荷を複数の CPU に分散し、クレンジング操作を同時に実行できます。さらに、外部アダプタの中には本質的にシングルスレッドのものもあるため、MDM Hub アーキテクチャでは、アプリケーションサーバーインスタンスごとに 1 つの処理スレッドを実行してマルチスレッド操作をシミュレートできるようになっています。

注: 複数のプロセスサーバーインスタンスを、同じホスト上または別のホスト上の別のアプリケーションサーバーインスタンス上にインストールし、それらを MDM Hub に登録できます。プロセスサーバーの単一インストールを複数のプロセスサーバーインスタンスとして MDM Hub 内の別のポートに登録できません。

クレンジング操作のモード

クレンジング操作は、以下のモードに従って分類できます。

- オンラインおよびバッチ
- オンラインのみ
- バッチのみ

モードを使用して、特定のプロセスサーバーが動作する操作クラスを指定することができます。2 つのプロセスサーバーをデプロイする場合、片方をバッチのみ、もう片方をオンラインのみに指定したり、両方のサーバーを両方のクラスの要求を受け取るように指定したりすることができます。指定しないと、プロセスサーバーはバッチ要求とオンライン要求の両方をデフォルトで実行します。

分散型データクレンジング

複数のプロセスサーバーを並行に実行し、クレンジング操作およびあいまい一致プロセスのスループットを向上させることができます。指定したジョブのサイズに一致またはサイズを超えた場合、MDM Hub はプロセスサーバーの間でジョブを分散します。

分散データクレンジングを設定するには、各 Process サーバーで、`cmxcleanse.properties` ファイルで次のプロパティを設定します。

`cmx.server.match.distributed_match`

オプション。手動で追加する必要があります。Process サーバーがクレンジング操作およびあいまい一致プロセスのための分散処理環境に参加するかどうかを指定します。デフォルトは 1 で、有効になります。分散処理を無効にするには 0 に設定します。

複数のプロセスサーバーの設定の詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

`cmx.server.cleansize.min_size_for_distribution`

オプション。手動で追加する必要があります。ジョブがプロセスサーバーの間で分散されるときにサイズを指定します。デフォルトは 1000 です。

Hub コンソールで、プロセスサーバープロパティを通常どおり設定します。

クレンジング要求

クレンジング要求はすべてバッチ API によって発行されます。

バッチ API はクレンジング要求を XML ペイロードとしてパッケージ化してプロセスサーバーに送ります。プロセスサーバーは、要求を受け取ると、XML を解析して該当するコードを呼び出します。

モードの種類	説明
オンラインの操作	結果は 1 つの XML 応答としてパッケージ化され、HTTP POST 接続を介して送り返されます。
バッチジョブ	プロセスサーバーは、処理するデータをフラットファイルに抽出して処理した後、一括ロード機能を使用してデータを返します。 <ul style="list-style-type: none">- Oracle 用バルクローダーは SQL*Loader ユーティリティです。- IBM DB2 用バルクローダーは IBM DB2 のロードユーティリティです。- Microsoft SQL Server 用バルクローダーは Java Database Connectivity ドライバを使用します。

プロセスサーバーは、各インスタンスで同時に複数の要求を処理できるようにマルチスレッド化されており、プロセスサーバーへのバッチ要求のデフォルトのタイムアウトは 1 年です。また、オンライン要求のデフォルトのタイムアウトは 1 分です。

ステージジョブまたは一致ジョブの実行中に複数のプロセスサーバーが登録され、ステージングまたはマッチングの必要なレコードの総数が 500 を超えると、そのジョブは有効なプロセスサーバーの間で同時に分散されます。

プロセスサーバーツールの起動

プロセスサーバーの情報、例えば、名前、ポート、サーバータイプ、プロセスサーバーがオンラインとオフラインのどちらかなどを表示するには、プロセスサーバーツールを起動します。

プロセスサーバーツールを起動するには、Hub コンソールでユーティリティワークベンチを展開して、**[Process Server]** をクリックします。プロセスサーバーツールにより、設定されたプロセスサーバーのリストが表示されます。

プロセスサーバーのプロパティ

選択した Process サーバーのサーバーおよびポートを識別します。その他のプロパティは、サーバーが処理するプロセスのタイプや、使用するスレッドの数などの、プロセスサーバーの動作を制御します。

Process サーバーの次のタイプのプロセスを有効または無効にすることができます。

- クレンジング操作
- あいまい一致およびクエリ検索プロセス
- ロードおよびマージのバッチプロセス
- Elasticsearch プロセス

ベストプラクティス: 単純にします。すべてのプロセスサーバー上のすべての操作およびプロセスを有効にします。パフォーマンスが問題になる場合、サーバー間で負荷を分散させる方法を決定できます。例えば、クレンジングの負荷が重い場合、オンラインクレンジング操作用に 1 つの Process サーバーを設定し、バッチクレンジング操作用に別の 1 つを設定します。クレンジングおよび一致のバッチジョブが同時に実行する場合、別のプロセスサーバーでジョブを実行することを検討してください。

プロパティを変更した場合、Process サーバーを再起動します。スレッド数は例外です。再起動の必要はありません。次の表に、指定可能なプロパティを示します。

プロパティ	説明
サーバー	この Process サーバーをデプロイしたアプリケーションサーバーの IP アドレスまたは完全修飾ホスト名。 注: localhost をホスト名として使用しないでください。
ポート	この Process サーバーをデプロイしたアプリケーションサーバーの HTTP または HTTPS ポート。
クレンジング操作	この Process サーバーがクレンジング操作を処理するかどうかを示します。デフォルトは true です。 【クレンジングモード】 オプションを使用して、このサーバーがバッチジョブ、リアルタイムクレンジング要求、あるいはその両方を処理するかどうかを指定します。
クレンジングおよびあいまい一致のバッチスレッド	クレンジング、トークン化、および一致のバッチジョブで使用するスレッドの数。デフォルトは 1 です。 ベストプラクティス: <ul style="list-style-type: none"> - あいまい一致またはアドレス検証を使用している場合、CPU ごとに 1 つのスレッドを有効にします。例えば、クアッドコアマシンでは、この値は 4 に設定します。 - あいまい一致またはアドレス検証を使用しておらず、ほとんどのデータクレンジングに文字列が含まれる場合、CPU ごとに 4 スレッドを有効にします。例えば、クアッドコアマシンでは、この値は 16 に設定します。 - Process サーバーをオペレーショナル参照ストアとは別のマシンで実行している場合、リモートデータベースで発生する可能性のある待ち時間に対応できるように、さらに 1 スレッドを追加します。 - メモリを多く消費するプロセスを実行している場合、JVM のこれらすべてのスレッドに割り当てるメモリの合計容量を 1 GB に制限します。
クレンジングモード	このプロセスサーバーが処理するクレンジング操作のタイプを指定します。 <ul style="list-style-type: none"> - バッチのみ。バッチジョブからのクレンジング要求のみを処理します。クレンジング関数は、ステージプロセスでのマッピングによって呼び出されます。 - オンラインのみ。リアルタイムのクレンジング要求のみを処理します。要求は、CleansePut SIF の API から暗黙的に呼び出されるクレンジング関数、または IDD サブジェクト領域のクレンジング関数で明示的に設定されているクレンジング関数から送信されます。 - 両方。バッチとオンライン両方のクレンジング要求を処理します。
あいまい一致およびクエリ検索処理	この Process サーバーがあいまい一致を処理するかどうかを示します。デフォルトは true です。 【一致モード】 オプションを使用して、このサーバーがバッチジョブ、リアルタイム一致要求、あるいはその両方を処理するかどうかを指定します。
一致モード	この Process サーバーが処理するあいまい一致処理のタイプを指定します。 <ul style="list-style-type: none"> - バッチのみ。要求がバッチジョブから送信された場合にのみ、マッチングに参加します。 - オンラインのみ。リアルタイム要求の場合にのみ、マッチングに参加します。リアルタイムのマッチング要求は、searchMatch SIF の API 呼び出し、IDD 拡張検索から送信されます。 - 両方。バッチ要求およびオンライン要求でのマッチングに参加します。
オフライン	この Process サーバーのステータス。チェックボックスを選択または選択解除しても Process サーバーのステータスは変わりません。
ロードおよびマージのバッチ処理	この Process サーバーがデータのロードのバッチジョブおよびマージのバッチジョブを処理するかどうかを示します。デフォルトは false です。

プロパティ	説明
ロードおよびマージのスレッド	データのロードのバッチジョブおよび自動マージのバッチジョブに使用するスレッドの最大数。 ベストプラクティス: マシンの CPU ごとに 4 スレッドを有効にします。例えば、クアドコアマシンでは、値を 16 に設定します。デフォルトは 20 です。
CPU 評価	クレンジングサーバープール内のマシンの相対的な CPU パフォーマンスを指定します。 ベストプラクティス: プロセスサーバーを実行するマシンのパフォーマンスが同じ位であれば、デフォルト値の 1 のままにします。この Process サーバーマシンのパフォーマンスが別のマシンの 2 倍であれば、この値を 2 に設定します。
Elasticsearch 処理	この Process サーバーがバッチジョブ「スマート検索データの初期インデックス処理」を実行するかどうかを示します。このバッチジョブは、あるビジネスエンティティについて、検索可能なフィールドのすべての値のインデックスを作成します。
保護された接続 (HTTPS)	この Process サーバーが HTTPS プロトコルを使用するかどうかを示します。選択した場合、 【ポート】 オプションが HTTPS ポート番号に設定されていることを確認します。

プロセスサーバーの追加

プロセスサーバーツールを使用して、プロセスサーバーを追加できます。オペレーショナル参照ストアごとに複数のプロセスサーバーを設定できます。

1. プロセスサーバーツールを起動します。
2. 書き込みロックを取得します。
3. プロセスサーバーツールの右側のペインで、**【プロセスサーバーの追加】** ボタンをクリックしてプロセスサーバーを追加します。
【プロセスサーバーの追加と編集】 ダイアログボックスが表示されます。
4. プロセスサーバーのプロパティを設定します。
5. **【OK】** をクリックします。
6. **【保存】** をクリックします。

プロセスサーバーの保護された通信の有効化

各プロセスサーバーには、署名済み証明書が必要です。Hub コンソールを使用して HTTPS プロトコルを有効化し、プロセスサーバーごとに保護されたポートを指定します。

1. 証明書ストアで、プロセスサーバー向けの署名済み証明書を作成します。
2. アプリケーションサーバーが証明書ストアにアクセスできることを確認します。
3. Hub コンソールにログインします。
4. オペレーショナルリファレンスストアデータベースを選択します。
5. 書き込みロックを取得します。
6. **【ユーティリティ】** ワークベンチで、**【プロセスサーバー】** を選択します。
7. プロセスサーバーを選択して、**プロセスサーバーの編集** アイコンをクリックします。
【プロセスサーバーの追加/編集】 ダイアログボックスが表示されます。

8. **【ポート】** が保護されたポートであることを確認します。
9. **【保護された接続 (HTTPS)】** チェックボックスを選択します。
10. **【OK】** をクリックします。
11. リストに表示されるその他のプロセスサーバーを確認します。

プロセスサーバープロパティの編集

プロセスサーバーのプロパティを編集することができます。

1. プロセスサーバーツールを起動します。
2. 書き込みロックを取得します。
3. 編集するプロセスサーバー を選択します。
4. **【プロセスサーバーの編集】** ボタンをクリックします。
選択されたプロセスサーバーの **【プロセスサーバーの追加と編集】** ダイアログボックスが表示されます。
5. プロセスサーバーに必要なプロパティを変更します。
6. **【OK】** をクリックします。
7. **【保存】** をクリックします。

プロセスサーバーの削除

プロセスサーバーツールを使用して、プロセスサーバーを削除できます。

1. プロセスサーバーツールを起動します。
2. 書き込みロックを取得します。
3. 削除するプロセスサーバーを選択します。
4. **【プロセスサーバーの削除】** ボタンをクリックします。
プロセスサーバーツールにより、削除を確認するように求められます。
5. **【OK】** をクリックします。

プロセスサーバーの設定のテスト

プロセスサーバー情報を追加または変更したときは、設定をチェックして接続が適切に機能するかどうかを確認することをお勧めします。

1. プロセスサーバーツールを起動します。
2. テストするプロセスサーバーを選択します。
3. **【プロセスサーバーのテスト】** ボタンをクリックして設定をテストします。
テストに成功した場合は、プロセスサーバーツールに、接続情報と成功メッセージを示すウィンドウが表示されます。
問題が発生すると、接続の問題についての情報を示すウィンドウが Informatica MDM Hub に表示されます。
4. **【OK】** をクリックします。

クレンジング関数の設定

MDM Hub では、データをクレンジングするクレンジング関数を構築して実行することができます。

クレンジング関数は、標準化や検証のために、MDM Hub によってレコード内のデータ値に適用される関数です。例えば、挨拶の文句に使用するデータの列がある場合、クレンジング関数を使用して「Doctor」という語をすべて「Dr.」に標準化することができます。クレンジング関数は継続的に適用することも、ステージングテーブルの列に単に出力値を割り当てることもできます。

クレンジング関数のタイプ

MDM Hub には、次のタイプのいずれかのクレンジング関数を指定できます。

- MDM Hub によって定義されたクレンジング関数
- クレンジングエンジンで定義されたクレンジング関数
- ユーザーが定義するカスタムクレンジング関数

事前に定義された関数では、名前や住所の標準化、住所の分解、性別の判定など、専門的なクレンジング機能を利用できます。クレンジング関数ツールの詳細についてはコンソールを参照してください。

ライブラリ

関数はライブラリ（Java ライブラリおよびユーザーライブラリ）にまとめられます。ライブラリは、モデルワークベンチのクレンジング関数ツールで使用できる関数を整理するためのフォルダーです。

クレンジング関数はセキュアリソース

クレンジング関数はセキュアリソースとして設定でき、セキュアまたはプライベートにすることができます。

使用できる関数はクレンジングエンジンで決まる

Hub コンソールに表示される関数は、使用しているクレンジングエンジンによって異なります。MDM Hub には、クレンジングエンジンで使用可能なクレンジング関数が表示されます。使用するクレンジングエンジンに関係なく、MDM Hub でのデータクレンジングの全体的なプロセスは同じです。

クレンジング関数ツールの起動

クレンジング関数ツールは、データをクレンジングする方法を定義するインターフェースを提供します。

クレンジング関数ツールを起動する手順

- Hub コンソールで、モデルワークベンチ展開して、**[クレンジング関数]** をクリックします。

Hub コンソールにクレンジング関数ツールが表示されます。

クレンジング関数ツールは以下の 2 つのペインに分かれています。

ペイン	説明
ナビゲーションペイン	ツリービューにクレンジング関数を表示します。ツリー内のノードをクリックすると、対応するプロパティページが右側のペインに表示されます。
プロパティペイン	選択されている関数のプロパティを表示します。カスタムクレンジング関数の場合、右側のペインでプロパティを編集できます。

左ペインに表示される関数は、使用しているクレンジングエンジンによって異なります。したがって、前の図のとおり表示されるとは限りません。

クレンジング関数のタイプ

クレンジング関数は、ツリーではタイプごとにグループ化されます。

クレンジング関数のタイプは、管理やアクセスが簡単になるように同様のクレンジング関数をグループ化するための上位カテゴリです。

クレンジング関数のプロパティ

ナビゲーションペインでクレンジング関数のタイプのリストを展開すると、クレンジング関数を選択してそのプロパティを表示できます。

特定のクレンジング関数に加え、[その他の関数] には、データ管理の効率化に役立つ [データベースの読み取り] 関数と [却下] 関数が含まれています。

フィールド	説明
データベースの読み取り	マップでデータベーステーブルのレコードを直接ルックアップできます。 注: この関数は、限られた数の同じデータ項目に対する参照が多数ある場合に使用すると便利です。
却下	マップの作成者が、正しくないデータを特定してそのレコードを却下し、理由を記録できます。

クレンジング関数の設定の概要

クレンジング関数を定義するには、以下のタスクを実行します。

注: Multidomain MDM 10.5 には、Apache Axis2 のバージョン 1.8.0 へのアップグレードが含まれています。クレンジング関数のカスタムコードを更新して、すべての Apache Axis2 ライブラリがバージョン 1.8.0 となるようにします。次の手順 1 から 3 を実行して、既存の SOAP クレンジング関数を更新します。

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. **[更新]** をクリックしてクレンジングライブラリを更新します。
4. ユーザー自身のクレンジングライブラリを作成します。このフォルダーにカスタムクレンジング関数を保存すると管理が簡素化されます。
5. 必要に応じて、新しいライブラリに正規表現関数を定義します。
6. 必要に応じて、新しいライブラリにグラフ関数を定義します。
7. カスタムクレンジング関数をグラフ関数に追加します。
8. カスタム関数をテストします。

ユーザークレンジングライブラリの設定

既存の内部または外部のクレンジング関数からクレンジング関数をカスタマイズして作成する場合、ユーザークレンジングライブラリまたは Java クレンジングライブラリを追加できます。グラフ関数、正規表現関数、クレンジングリストを追加するように、ユーザークレンジングライブラリを設定します。

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. **[更新]** をクリックして、クレンジングライブラリを更新します。

4. [クレンジング関数] ナビゲータで [クレンジング関数] を右クリックして、[ユーザーライブラリの追加] を選択します。
[ユーザーライブラリの追加] ダイアログボックスが表示されます。
5. 以下のプロパティを指定します。
名前
ライブラリの一意のわかりやすい名前。
説明
ライブラリの説明（省略可能）。
6. [OK] をクリックします。
追加したライブラリは [クレンジング関数] ナビゲータに表示されます。

Java クレンジングライブラリの設定

既存の内部または外部のクレンジング関数からクレンジング関数をカスタマイズして作成する場合、ユーザークレンジングライブラリまたは Java クレンジングライブラリを追加できます。MDM Hub 外部で生成したカスタムクレンジング関数を追加するように、Java クレンジングライブラリを設定します。

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. [更新] をクリックしてクレンジングライブラリを更新します。
4. [クレンジング関数] ナビゲータで [クレンジング関数] を右クリックして、[Java ライブラリの追加] を選択します。
[Java ライブラリの追加] ダイアログボックスが表示されます。
5. [ブラウズ] ボタンをクリックして、Java ライブラリ JAR ファイルの場所に移動します。
6. 以下のプロパティを指定します。
名前
ライブラリの一意のわかりやすい名前。
説明
ライブラリの説明（省略可能）。
7. ライブラリのパラメータを適宜設定します。
 - a. [パラメータ] ボタンをクリックします。
[パラメータ] ダイアログボックスが表示されます。
 - b. パラメータをライブラリに必要なだけ追加します。
 - パラメータを追加するには、[パラメータの追加] ボタンをクリックします。
[値の追加] ダイアログボックスが表示されます。
名前と値を入力し、[OK] をクリックします。
 - パラメータをインポートするには、[パラメータのインポート] ボタンをクリックします。
[開く] ダイアログボックスが表示されます。
必要なパラメータを含んだプロパティファイルを選択します。

このライブラリに必要な数だけパラメータを追加できます。

- パラメータを追加するには、**【パラメータの追加】** ボタンをクリックします。 **【値の追加】** ダイアログが表示されます。
名前と値を入力し、**【OK】** をクリックします。
- パラメータをインポートするには、**【パラメータのインポート】** ボタンをクリックします。 **【開く】** ダイアログが表示され、必要なパラメータが格納されたプロパティファイルを選択するように求められます。

ファイルからインポートした名前と値のペアは、実行時にユーザー定義の Java 関数で Java のプロパティの要素として使用できます。ユーザー ID やターゲット URL などのカスタム値を汎用関数に指定します。

- 【OK】** をクリックします。
追加したライブラリは **【クレンジング関数】** ナビゲータに表示されます。

正規表現関数の追加

クレンジング操作に対して正規表現関数を追加できます。正規表現は、一般的な構文規則と記号パターンに従ってテキストデータをマッチングおよび操作するために使用できる計算式です。正規表現の構文とパターンの詳細については、Javadoc の `java.util.regex.Pattern` を参照してください。

- クレンジング関数ツールを起動します。
- 書き込みロックを取得します。
- ユーザークレンジングライブラリ名を右クリックして、**【正規表現関数の追加】** をクリックします。
【正規表現の追加】 ダイアログボックスが表示されます。
- 以下のプロパティを指定します。

フィールド	説明
名前	正規表現関数の一意のわかりやすい名前。
説明	正規表現関数の説明（省略可能）。

- 【OK】** をクリックします。
追加する正規表現関数は、**【クレンジング関数】** ナビゲートで選択したユーザーライブラリ下に表示されます。
- 追加した正規表現関数の **【詳細】** タブをクリックします。
- 入力または出力式を入力するには、**【編集】** をクリックしてから、**【編集を許可】** をクリックして変更を適用します。
- 【保存】** をクリックします。

グラフ関数を設定する

グラフ関数とは、Hub コンソールのクレンジング関数ツールを使用してグラフィック形式で視覚化および設定できるクレンジング関数です。グラフ関数には、事前に定義された関数を追加することができます。

グラフ関数には 1 つ以上の入力と出力があります。グラフ関数ごとに、必要なすべての入力と出力を設定します。

次のタスクを実行して、グラフ関数を設定します。

- グラフ関数を追加する。

2. グラフ関数に関数を追加する。

入力と出力のプロパティ

グラフ関数には 1 つ以上の入力と出力があります。

すべてのグラフ関数の入力と出力の次のプロパティを設定します。

名前

入力または出力の一意のわかりやすい名前。

説明

入力または出力の説明（省略可能）。

データタイプ

入力または出力パラメータのデータタイプ。

以下の値を使用する。

- ブール。ブール値を受け入れます。
- 日付。日付値を受け入れます。
- 浮動小数点。浮動小数点を受け入れます。
- 整数。整数値を受け入れます。
- 文字列。どのようなデータでも受け入れます。

グラフ関数の追加

グラフ関数を追加できます。

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. ユーザーライブラリの名前を右クリックし、**[グラフ関数の追加]** を選択します。
[グラフ関数の追加] ダイアログボックスが表示されます。
4. 以下のプロパティを指定します。

フィールド	説明
名前	グラフ関数の一意のわかりやすい名前。
説明	グラフ関数の説明（省略可能）。

5. **[OK]** をクリックします。

グラフ関数が表示されます。この時点でグラフ関数は空です。グラフ関数を設定するには、[「グラフ関数への関数の追加」](#)（ページ 352）を参照してください。

グラフ関数への関数の追加

グラフ関数には、必要に応じていくつでも関数を追加できます。

グラフ関数は、再利用することもできます。グラフ関数を定義し、それらを他の関数と同じようにクレンジングライブラリで使します。例えば、グラフ関数を別のグラフ関数内に追加することが可能です。

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. グラフ関数をクリックし、次に【詳細】タブをクリックします。

関数のグラフィカル表示は、ワークスペースと呼ばれます。ワークスペースに入力と出力の両方が表示されないときは、ウィンドウのサイズを変更してください。

デフォルトのグラフ関数には、文字列型の入力と出力が1つずつ追加されます。グレーの円は、入力と出力を表します。関数によっては、さまざまなデータ型の入力と出力がさらに必要になる場合があります。

4. ワークスペースを右クリックして【関数の追加】を選択します。

クレンジング関数ツールによって、【追加する関数の選択】ダイアログボックスが表示されます。

5. 追加する関数が格納されているフォルダを展開し、関数を選択して【OK】をクリックします。

注: 使用可能な関数は、クレンジングエンジンの設定によって異なります。

関数がワークスペースに表示されます。関数を移動するには、新しい場所にドラッグします。

6. 関数を右クリックし、【拡張モード】を選択します。

拡張モードでは、すべての入力と出力のラベルが表示されます。円の色は、入力や出力のデータ型を示します。これらのデータ型は一致しなければなりません。

7. 入力コネクタ（入力ボックスの右側にある小さい円）にマウスを合わせます。使用可能な状態になると円の色が赤に変わります。
8. ノードをクリックし、いずれかの関数の入力ノードまで線を引きます。
9. いずれかの関数の出力ノードから出力ボックスのノードまで線を引きます。
10. 【保存】をクリックします。

グラフ関数のテストについては、[「関数のテスト」](#)（ページ 355）を参照してください。

関数モード

関数モードは、ワークスペースでの関数の表示方法を決定します。各関数には以下のモードがあり、関数を右クリックして選択することができます。



オプション	説明
コンパクト	関数を小さなボックスとして表示し、関数名のみを示します。
標準	関数をやや大きなボックスとして表示し、名前および入力と出力のノードを示しますが、ノードのラベルはありません。これはデフォルトのモードです。
拡張	関数を大きなボックスとして表示し、名前、入力と出力のノード、およびノードの名前を示します。

オプション	説明
ロギング対応	<p>デバッグに使用されます。このオプションを選択すると、ステージジョブを実行するときにこの関数のログファイルが生成されます。ログファイルには、ステージジョブで関数が呼び出されるたびに入出力が記録されます。ステージジョブごとに新しいログファイルが作成されます。</p> <p>ログファイルは、<jobID><graph function name>.log と命名されて次の場所に格納されます。</p> <p>\<infamdm_install_dir>\hub\cleanse\tmp\<オペレーショナル参照ストア></p> <p>注: このオプションは、ディスク容量を消費し、ディスク I/O に伴うパフォーマンスオーバーヘッドを発生させるため、プロダクション環境では使用しないでください。このロギングを無効にするには、関数を右クリックし、[ロギングの有効化] をオフにします。</p>
オブジェクトの削除	関数をグラフ関数から削除します。

関数をダブルクリックすることで、表示モード（コンパクト、標準、拡張）を順番に切り替えることができます。

ワークスペースのボタン

ワークスペースの右側のツールバーには、次のボタンがあります。

ボタン	説明
	変更を保存します。
	関数の入力を編集します。
	関数の出力を編集します。
	関数を追加します。
	定数を追加します。
	条件付き実行コンポーネントを追加します。
	選択したコンポーネントを編集します。
	選択したコンポーネントを削除します。
	グラフを展開します。このボタンを使用して左ペインを非表示にすると、画面上のワークスペースが広くなります。

定数の使用

定数は、標準化された入力があることがわかっている場合に役立ちます。

例えば、医師のみで構成されていることがわかっているデータセットがある場合は、定数を使用してタイトルに「Dr.」と入力できます。グラフ関数で定数を使用すると、背景色が灰色になって他の関数と視覚的に区別されます。

入力の設定

入力を追加する手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. 設定するクレンジング関数を選択します。
4. **【詳細】** タブをクリックします。
5. 入力を右クリックし、**【入力の編集】** を選択します。

[入力] ダイアログが表示されます。

注: 入力を作成した後に、入力を編集して入力のタイプを変更することはできません。入力のタイプを変更する必要がある場合は、正しいタイプの新しい入力を作成し、古い入力を削除します。

6. **【追加】** ボタンをクリックして別の入力を追加します。

[パラメータの追加] ダイアログが表示されます。

7. 以下のプロパティを指定します。

フィールド	説明
名前	このパラメータの一意のわかりやすい名前。
データ型	このパラメータのデータ型。
説明	このパラメータの説明（省略可能）。

8. **【OK】** をクリックします。

関数で必要な数だけ入力を追加します。

出力の設定

出力を追加する手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. 設定するクレンジング関数を選択します。
4. **【詳細】** タブをクリックします。
5. 出力を右クリックし、**【出力の編集】** を選択します。

[出力] ダイアログが表示されます。

注: 出力を作成した後に、出力を編集して出力のタイプを変更することはできません。出力のタイプを変更する必要がある場合は、正しいタイプの新しい出力を作成し、古い出力を削除します。

6. **【追加】** ボタンをクリックして別の出力を追加します。

[パラメータの追加] ダイアログが表示されます。

フィールド	説明
名前	このパラメータの一意のわかりやすい名前。
データ型	このパラメータのデータ型。
説明	このパラメータの説明（省略可能）。

7. **【OK】** をクリックします。
関数で必要な数だけ出力を追加します。

関数のテスト

グラフ関数または正規表現関数を追加して設定したら、その関数をテストして予期したとおりに動作するかどうかを確認することをお勧めします。

このテストプロセスは、関数に組み込まれた単一のレコードを模倣しています。

関数をテストする手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. テストするクレンジング関数を選択します。
4. **【テスト】** タブをクリックします。
テストスクリーンが表示されます。
5. 入力ごとに、[値] カラムのセルをクリックして入力のデータ型に準拠する値を入力し、テストする値を指定します。
 - ブール値入力の場合は、クレンジング関数ツールに true/false のドロップダウンリストが表示されます。
 - カレンダー入力の場合は、クレンジング関数ツールに [カレンダー] ボタンが表示されます。このボタンをクリックすると、[日付] ダイアログから日付を選択できます。
6. **【テスト】** をクリックします。
テストが正常に完了した場合は、出力が出力セクションに表示されます。

クレンジング関数での条件の使用

この節では、グラフ関数に条件を追加する方法について説明します。

条件付き実行コンポーネントについて

条件付き実行コンポーネントは、プログラミング言語の case 文（または switch 文）の構造に似ています。

このクレンジング関数は、条件を評価し、その評価に基づいて、条件に一致する case に関連付けられた適切なグラフ関数を適用します。条件に一致する case がない場合は、デフォルトの case（アスタリスク（*）が付いた case）が使用されます。

条件付き実行コンポーネントを使用する状況

条件付き実行コンポーネントは、例えば、データをセグメント化している場合に役立ちます。

テーブルに個別のデータグループ（顧客や見込み客など）がいくつかあるとします。この場合、レコードがメンバーとして属しているグループを示すカラムを作成できます。各グループはセグメントと呼ばれます。この例では、このカラムに顧客の場合は C、見込み客の場合は P と示すことができます。条件付き実行コンポーネントを使用すると、セグメントごとに異なる方法でデータをクレンジングできます。条件値が指定された条件を満たさない場合は、デフォルトのケースが実行されます。

条件付き実行コンポーネントの追加

条件付き実行コンポーネントを追加する手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. 設定するクレンジング関数を選択します。
4. ワークスペースを右クリックし、**[条件の追加]** を選択します。
[条件の編集] ダイアログが表示されます。
5. **[追加]** ボタンをクリックして値を追加します。
[値の追加] ダイアログが表示されます。
6. 条件の値を入力します。顧客（customer）と見込み客（prospect）の例の場合、C または P と入力します。**[OK]** をクリックします。
左側にある条件のリストと入力ボックスに新しい条件が表示されます。
必要な数だけ条件を追加します。デフォルトの条件は指定する必要はありません。新しい条件付き実行コンポーネントを作成するとデフォルトの case が自動的に作成されます。ただし、アスタリスク（*）を使用してデフォルトの case を指定することができます。デフォルトの case は、指定した case に該当しないすべての場合に実行されます。
7. すべての条件を処理するために必要な数だけ関数を追加します。
8. 各条件（デフォルトの条件を含む）について、入力ノードと関数の入力の間にリンクを設定します。また、関数の出力とクレンジング関数の出力の間のリンクも設定します。

注: グラフ関数では、ネストされた処理ロジックを指定できます。例えば、条件付きコンポーネントを他の条件付きコンポーネント内にネストすることができます（ネストされた case 文など）。つまり、複数の条件のテストを含む複雑なプロセス全体を定義し、それぞれのテストにもある程度複雑な条件を含めることができます。

クレンジングリストの設定

ここでは、Informatica MDM Hub 実装でクレンジングリストを設定する方法について説明します。

クレンジングリストについて

クレンジングリストは、実行時に定義済みの順序で実行される文字列関数の論理的なグループです。

クレンジングリストは、既知の文字列値を標準化したり、入力文字列から余分な文字（句読点など）を削除したりする場合に使用します。

クレンジングリストの追加

新しいクレンジングリストを追加する手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. **【更新】** をクリックしてクレンジングライブラリを更新します。これは外部のクレンジングエンジンで使用されます。
重要: 書き込みロックの取得後、レコードを処理する前に、**【更新】** を選択する必要があります。そうしないと、外部のクレンジングエンジンからエラーがスローされます。
4. **【クレンジング関数】** のリストでクレンジング関数を右クリックし、**【クレンジングリストの追加】** を選択します。
【クレンジングリストの追加】 ダイアログが表示されます。
5. 以下のプロパティを指定します。

フィールド	説明
名前	このクレンジングリストの一意のわかりやすい名前。
説明	このクレンジングリストの説明（省略可能）。

6. **【OK】** をクリックします。
画面の右側に新しい（空の）クレンジングリストの詳細ペインが表示されます。

クレンジングリストのプロパティ

ここでは、クレンジングリストの入力と出力のプロパティについて説明します。

入力のプロパティ

以下の表に、クレンジングリストの入力プロパティを示します。

プロパティ	説明
Input string	ソースシステムの文字列値。検索のターゲットとして使用されます。
searchType	<p>入力文字列に対して実行される一致（クレンジングリストの項目と入力文字列の比較）のタイプを指定します。次のうち 1 つの値になります。</p> <p>ENTIRE</p> <p>クレンジングリストの項目を文字列全体と比較します。入力文字列全体がクレンジングリストの項目と同じである場合にのみ一致と見なされます。このパラメータが指定されていない場合のデフォルトの設定です。</p> <p>WORD</p> <p>クレンジングリストの項目を、入力文字列内の各単語サブストリングと比較します。クレンジングリストの項目が、入力文字列内の次の単語境界の横にあるサブストリングである場合にのみ、一致と見なされます。文字列の先頭、文字列の末尾、またはスペース文字。</p> <p>ANYWHERE</p> <p>クレンジングリストの項目を、入力文字列内の部分と比較します。クレンジングリストの項目が、入力文字列のサブストリング（入力文字列での位置は無関係）である場合に、一致と見なされます。</p> <p>注: 文字列の比較では、大文字と小文字が区別されます。</p>
replaceAllOccurrences	<p>入力文字列内の一致したサブストリングが、一致するクレンジングリスト項目に置換される程度を指定します。以下のいずれかの値を指定します。</p> <p>TRUE</p> <p>入力文字列内の一致したすべてのサブストリングを、一致するクレンジングリスト項目で置換します。</p> <p>FALSE</p> <p>入力文字列内の一致した最初のサブストリングのみを、一致するクレンジングリスト項目で置換します。replaceAllOccurrences が指定されていない場合のデフォルトの設定です。</p> <p>注: Strip パラメータが TRUE である場合は、一致したサブ文字列が置換されずに除去されます。</p>
stopOnHit	<p>入力文字列に一致する項目が 1 つ見つかった時点で残りのクレンジングリストの処理を続行するかどうかを指定します。以下のいずれかの値を指定します。</p> <p>TRUE</p> <p>入力文字列に最初のクレンジングリスト項目が見つかった時点で（searchType の条件が満たされている場合）、クレンジングリストの処理を停止します。stopOnHit が指定されていない場合のデフォルトの設定です。</p> <p>FALSE</p> <p>入力文字列で引き続きクレンジングリストの残りの項目を検索します（さらに一致するサブストリングを検出するため）。</p>

プロパティ	説明
Strip	<p>入力文字列内の一致したテキストを、入力文字列から除去するか、または入力文字列内で置換するかを指定します。以下のいずれかの値を指定します。</p> <p>TRUE</p> <p>一致したテキストを置換するのではなく入力文字列から除去します。</p> <p>FALSE</p> <p>入力文字列内の一致したテキストを置換します。Strip が指定されていない場合のデフォルトの設定です。</p> <p>注: replaceAllOccurrences パラメータによって、置換または除去の対象が、入力文字列内のすべての一致箇所か、最初の一致箇所のみが決まります。</p>
defaultValue	<p>入力文字列にクレンジングリスト項目が見つからなかった場合の出力に使用される値。このプロパティが指定されていない場合に一致が検出されなかったときは、元の入力文字列が出力として使用されます。</p>

出力のプロパティ

以下の表に、クレンジングリストの出力プロパティを示します。

プロパティ	説明
output	クレンジングリスト関数の出力値。
一致	クレンジングリストの最後に一致した値。
一致フラグ	一致がリストにあるか (true) またはないか (false) を指定します。

クレンジングリストのプロパティの編集

新しいクレンジングリストは空の状態です。クレンジングリストを編集して、一致文字列と出力文字列を追加する必要があります。

クレンジングリストを編集して一致文字列と出力文字列を追加する手順

1. クレンジング関数ツールを起動します。
2. 書き込みロックを取得します。
3. 設定するクレンジングを選択します。
クレンジング関数ツールの右ペインに、クレンジングリストの情報が表示されます。
4. 必要に応じて、右ペインで表示名と説明を変更します。そのためには、変更する値の横の【編集】ボタンをクリックします。
5. 【詳細】タブをクリックします。
クレンジング関数ツールに、クレンジングリストの詳細が表示されます。
6. 右側のペインで【追加】ボタンをクリックします。
[出力文字列] ダイアログが表示されます。
7. 検索文字列、出力文字列、および一致タイプを指定して、【OK】をクリックします。
検索文字列はクレンジングする入力であり、結果的に出力文字列となります。

重要: Informatica MDM Hub では、入力されている順に文字列が検索されます。したがって、項目を指定する順序が結果に影響することがあります。指定可能な一致のタイプの詳細については、[「文字列の一致のタイプ」 \(ページ 360\)](#)を参照してください。

注: クレンジングリストに文字列を追加すると直ちにクレンジングリストが保存されます。

指定した文字列が [クレンジングリストの詳細] セクションに表示されます。

- 文字列の追加と削除を行うことができます。また、クレンジングリスト内で文字列を前後に移動することもできます。この操作は、ランタイムの実行シーケンスにおける文字列の順序に影響するため、結果にも影響します。
- どの検索文字列にも一致しない入力文字列に対する「デフォルト値」を指定することもできます。
デフォルト値を指定しない場合は、検索文字列に一致しなかったすべての入力文字列が、変更されないまま出力文字列に渡されます。


文字列の一致のタイプ

出力文字列には、以下のいずれかの一致タイプを指定できます。

一致タイプ	説明
完全に一致する	テキスト文字列 (「IBM」など)。文字列の一致では大文字と小文字が区別されません。例えば、文字列 test は TEST や Test と一致します。
正規表現	正規表現に Java 構文を使用するパターン (例えば、「I.M.*」は「IBM」、「IB Corp」、「IXM Inc.」と一致します)。ファーストネーム、ミドルネーム、ラストネームから構成される名前フィールドを解析するには、正規表現(\S+\$)を使用できます。どのような名前を指定しても、この正規表現ではラストネームが返されます。 パラメータとして入力した正規表現が文字列に対して使用され、一致した出力が出口に送信されます。また、グループ番号を指定して、正規表現の内部グループを一致することもできます。正規表現の構成とグループのしくみに関するマニュアルについては、 <code>java.util.regex.Pattern</code> の Javadoc を参照してください。
SQL 一致	SQL の LIKE 演算子に SQL 構文を使用するパターン (例えば、「I_M%」は「IBM」、「IBM Corp」、および「IXM Inc.」と一致します)。パイプライン記号 () などのメタ文字を使用する場合、そのメタ文字はバックスラッシュ (\\uff09 などのエスケープシーケンスで区切る必要があります)。

一致文字列のインポート




一致文字列 (ファイルやデータベーステーブルなど) をインポートする手順

-  ボタン (右側のペイン) をクリックします。
一致文字列のインポートウィザードが開きます。
- データのソースの接続プロパティを指定して、**[次へ]** をクリックします。
クレンジング関数ツールにインポート可能なテーブルのリストが表示されます。
- インポートするテーブルを選択し、**[次へ]** をクリックします。
クレンジング関数ツールにインポート可能なカラムのリストが表示されます。
- インポートするカラムをクリックし、**[次へ]** をクリックします。
クレンジング関数ツールにインポート可能な一致文字列のリストが表示されます。

サンプルデータのレコードを、句（レコードごとに1つのエントリ）または単語（各レコードの単語ごとに1つのエントリ）としてインポートできます。一致文字列を単語と句のどちらとしてインポートするかを選択し、**【完了】** をクリックします。


〔クレンジングリストの詳細〕ボックスに、指定したソースからデータが表示されます。

注: インポートされた一致文字列は、一致リストには追加されません。一致文字列を一致リストに追加するには、右側の〔検索文字列〕にそれらの文字列を移動する必要があります。

- 一致文字列を〔検索文字列〕と〔出力文字列〕の両方の一致文字列値とともに一致リストに追加するには、〔一致文字列〕リストで  ボタンをクリックします。
- 一致文字列を、定義する出力文字列値とともに一致リストに追加する場合は、追加したレコードをクリックし、新しい検索文字列と出力文字列を指定します。
- すべての一致文字列を一致リストに追加するには、  ボタンをクリックします。
- 一致リストからすべての一致文字列をクリアするには、  ボタンをクリックします。
- 一致リストが完成するまで、上記の手順を繰り返します。

一致出力文字列のインポート

ファイルやデータベーステーブルなどの一致出力文字列をインポートする手順

1. ボタン  （右側のペイン）をクリックします。
一致出力文字列のインポートウィザードが開きます。
2. データのソースの接続プロパティを指定します。
3. **【Next（次へ）】** をクリックします。
クレンジング関数ツールにインポート可能なテーブルのリストが表示されます。
4. インポートするテーブルを選択します。
5. **【Next（次へ）】** をクリックします。
クレンジング関数ツールにインポート可能なカラムのリストが表示されます。
6. インポートするカラムを選択します。
7. **【Next（次へ）】** をクリックします。
クレンジング関数ツールにインポート可能な一致文字列のリストが表示されます。
8. **【完了】** をクリックします。
〔クレンジングリストの詳細〕ボックスに、指定したソースからデータが表示されます。

第 20 章

ロードプロセスの設定

この章では、以下の項目について説明します。

- [概要, 362 ページ](#)
- [作業を開始する前に, 362 ページ](#)
- [データのロードに関する設定タスク, 363 ページ](#)
- [ステージングテーブルの設定, 363 ページ](#)
- [初期データロードの設定, 370 ページ](#)
- [ソースシステムの信頼の設定, 370 ページ](#)
- [検証ルールの設定, 376 ページ](#)

概要

この章では、Informatica MDM Hub 実装にロードプロセスを設定する方法について説明します。

作業を開始する前に

ロードプロセスの設定を開始する前に、以下のタスクが完了している必要があります。

- Informatica MDM Hub をインストールし、Hub Store を作成する
- スキーマを構築する
- ソースシステムを定義する
- ランディングテーブルを作成する
- ステージングテーブルを作成する
- ロードプロセスについて理解する

データのロードに関する設定タスク

Informatica MDM Hub 実装でのデータをロードするプロセスを設定するには、Hub コンソールで以下のタスクを実行する必要があります。

- ステージングテーブルを設定する。
- 初期データロードを設定する。
- ソースシステムの信頼を設定する。
- 検証ルールを設定する。

ロードプロセスに影響する可能性があるその他の構成設定については、以下を参照してください。

- [「行 ID によるロード」 \(ページ 316\)](#)
- [「個別システム」 \(ページ 493\)](#)
- [「一致トークンの生成 \(オプション\)」 \(ページ 279\)](#)
- [「ロードプロセス」 \(ページ 272\)](#)

ステージングテーブルの設定

MDM Hub は、ランディングテーブルからベースオブジェクトへのデータのフローの中で、一時的な中間ストレージ用にステージングテーブルを使用します。

ステージングテーブルは、Hub ストア内の 1 つのベースオブジェクトテーブルに対し、1 つのソースシステムのデータを格納します。バッチステージジョブはランディングテーブルからステージングテーブルに入力します。その後、バッチロードジョブはステージングテーブルからベースオブジェクトに入力します。

ステージングテーブルの構造は、統合されたデータを含んだターゲットオブジェクトの構造に基づいています。ステージングテーブルを設定するには、モデルワークベンチのスキーママネージャを使用します。ステージングテーブルを定義する前に、ソースシステムを少なくとも 1 つ定義する必要があります。

ステージングテーブルのカラム

ステージングテーブルにはシステムカラムとユーザー定義のカラムがあります。

システムステージングテーブルカラム

スキーママネージャを使用して、システムステージングテーブルカラムの作成と保持を行います。

次の表に、ステージングテーブルのシステムカラムを示します。

物理名	MDM Hub データ型 (サイズ)	説明
PKEY_SRC_OBJECT	VARCHAR(255)	ソースシステムのプライマリキー。PKEY_SRC_OBJECT の値は一意である必要があります。ソースレコードが単一の一意なカラムでない場合は、複数のカラムの値を結合することでレコードを一意なものにします。
ROWID_OBJECT	CHAR (14)	MDM Hub のプライマリキー。MDM Hub はステージプロセス中に一意の ROWID_OBJECT 値を割り当てます。

物理名	MDM Hub データ型 (サイズ)	説明
DELETED_IND	INT	将来の使用のために予約済み。
DELETED_DATE	TIMESTAMP	将来の使用のために予約済み。
DELETED_BY	VARCHAR (50)	将来の使用のために予約済み。
LAST_UPDATE_DATE	TIMESTAMP	ソースシステムが最後にレコードを更新した日。ベースオブジェクトの場合、LAST_UPDATE_DATE は相互参照テーブルに LAST_UPDATE_DATE と SRC_LUD を入力します。また信頼設定によってはベースオブジェクトテーブルに LAST_UPDATE_DATE も入力します。
UPDATED_BY	VARCHAR (50)	最後に更新を行ったユーザーまたはプロセス。
CREATE_DATE	TIMESTAMP	レコードが作成された日付。
PERIOD_START_DATE	TIMESTAMP	レコードの有効期間の開始日。タイムラインが有効なベースオブジェクトには、PERIOD_START_DATE 値が必要です。
PERIOD_END_DATE	TIMESTAMP	レコードの有効期間の終了日。タイムラインが有効なベースオブジェクトには PERIOD_END_DATE 値が必要です。
CREATOR	VARCHAR (50)	レコードの作成を行ったユーザーまたはプロセス。
SRC_ROWID	VARCHAR (30)	データベースの内部行 ID カラム。ステー징テーブルからランディングテーブルにレコードをトレースバックします。
HUB_STATE_IND	INT	状態が有効なベースオブジェクト向け。このレコードの状態を示す整数値。有効な値は次のとおりです。 <ul style="list-style-type: none"> - 0=保留中 - 1=アクティブ - -1=削除済み デフォルト値は 1 です。
VERSION_SEQ	INT	タイムラインが有効なベースオブジェクトの場合、同じプライマリソースキーを持つレコードがステー징テーブルにロードされた順に値が表示されます。タイムラインが有効でないベースオブジェクトの場合、この値は 1 でなければなりません。 VERSION_SEQ カラムはユーザー定義ランディングテーブルカラムからマッピングされます。ランディングテーブルカラムには、プライマリキーが同じで開始日および終了日が異なるレコードのバージョンシーケンスが含まれています。

ユーザー定義のステー징テーブルカラム

不必要な信頼の計算によるパフォーマンスの低下を避けるため、ステー징テーブルに追加するカラムは特定のソースシステムから選択してください。

ソースシステムからカラムを追加するときに、信頼カラムを複数のステー징テーブルからのデータを持つカラムに制限できます。ステー징テーブルに特定のソースシステムからのカラムがある場合、個々のカラムをすべてのソースシステムからのカラムとして扱うことはありません。各カラムに信頼を追加する必要はありません。また、カラムで値を使用していないすべてのソースには、NULL 値の信頼をダウングレードする検証ルールは必要ありません。

ソースシステムキーを保持する

ステージジョブの間、MDM Hub はキーを生成するか、ソースシステムのプライマリキーカラムのキーを使用できます。MDM Hub は、初期データロードを実行するために、1つのソースシステムに対してソースシステムキーを保持できます。

MDM Hub でソースシステムのキー値を使用するか、MDM Hub によって生成されたキー値を使用するかを指定できます。ソースシステムのキー値を使用する場合は、**【ソースシステムキーの保持】** オプションを有効にします。MDM Hub によって生成されたキー値の使用を無効化します。デフォルトでは無効になっています。

ステージジョブを実行する前に、ソースシステムのキーを保持するオプションを有効にすることができます。このオプションを有効にすると、MDM Hub によって内部キーは生成されませんが、ステージジョブの間、ソースシステムのプライマリキーが代わりに使用されます。ロードジョブを実行すると、ステージングテーブルの PKEY_SOURCE_OBJECT カラムの値が MDM Hub によって取得されて、ターゲットベースオブジェクトの ROWID_OBJECT カラムに挿入されます。

注: ベースオブジェクトのロード後、ソースシステムのキーを保持する設定は変更できなくなります。

最高予約キーの指定

最高予約キーは最高ソースシステムキーです。MDM Hub によって生成されるキーがソースシステムキーと競合しないようにするには、最高ソースシステムキーを予約します。ソースシステムキーを予約するときには、レコードに使用する最高ソースシステムキーを指定できます。

ソースキーと、MDM Hub によって生成されるキーの間にギャップを挿入するには、初期ロードの後のキー値の増分数を指定する必要があります。

ソースシステムキーの上限の境界に最高予約キー値を設定します。マージンを許可するには、ソースシステムキーの予測される範囲にバッファを追加し、この値を少し高く設定します。

このロード操作では、ソースシステムキーが含まれないベースオブジェクトにレコードを追加できます。ベースオブジェクトレコードがソースシステムキーを含まない場合、最高予約キー値よりも高いキーが MDM Hub によって割り当てられます。

最高予約キーを指定すると、MDM Hub はベースオブジェクトにロードするレコードの ROWID_OBJECT 値を次のように処理します。

1. 初期データロードで、MDM Hub はステージングテーブルの PKEY_SOURCE_OBJECT カラムにある値を取得します。MDM Hub は、内部キーを生成する代わりに、この値をベースオブジェクトの ROWID_OBJECT カラムに挿入します。
2. 初期データロードの後、MDM Hub はキーの開始位置を 1 増えた最高予約キー値にリセットします。
3. MDM Hub は、このステージングテーブルからの後続のロードにこの最高予約キー値を使用します。他のステージングテーブルからのロードには、MDM Hub はそれ自体が生成するキーを使用します。MDM Hub によって生成されるキーは、最高予約キー値と連続した状態になります。

注: キーが同じソースシステムから取得される場合でも、ベースオブジェクトに関連付けられた 1 つのステージングテーブル用としてソースシステムキーを予約できます。予約キーの範囲は初期ロードで設定します。

最高予約キーの例

ステージングテーブルのソースシステムキーを保持して、レコードに使用する最高ソースシステムキーを 100 に設定します。ソースシステムキーを保持したステージングテーブルから、さらにレコードをロードします。

その後、ソースシステムキーを保持したステー징テーブル以外のステーディングテーブルから、レコードをロードします。

次の各ロード操作中、ROWID_OBJECT カラムのキー値の変更を確認できます。

1. ソースシステムキーを保持するステーディングテーブルからのプライマリソースシステムキー 20、21、および 22 の初期データロード。
MDM Hub は 3 つのレコードを、ROWID_OBJECT 値 20、21、22 で関連するベースオブジェクトにロードします。
2. 初期データロード後、プライマリソースシステムキーのある 5 つのレコードを、ソースシステムキーを保持するステーディングテーブルからロードします。これらのレコードのプライマリソースシステムキーは 23、24、25、AB、YZ です。
MDM Hub は 5 つのレコードを、ROWID_OBJECT 値 23、24、25、101、102 で関連するベースオブジェクトにロードします。
3. 任意のプライマリキーの 3 つのレコードを、ソースシステムキーを保持していないステーディングテーブルからロードします。
MDM Hub は 3 つのレコードを、ROWID_OBJECT 値 103 から 105 で関連するベースオブジェクトにロードします。

セルの更新の有効化

パフォーマンスを向上するために、および MDM Hub が変更された値でセルを更新するように、セルを更新するオプションを有効にします。デフォルトでは、ロードプロセスにおいて、信頼レベルの高い受信レコードごとに、ターゲットベースオブジェクト内のセル値が置換されます。ロードプロセスでは、置換された値が同一であってもセル値が置換されます。

値が変更されていなくても、MDM Hub はセルの最終更新日を入力レコードに関連付けられた日付に更新し、新しい値と同じ信頼レベルをセルに割り当てます。動作を変更するには、ステーディングテーブルを設定するときに、セルの更新を有効にします。セルの更新を有効にした場合、ロードジョブの中で、MDM Hub はベースオブジェクト内のターゲットレコードを更新する前に、セル値を相互参照テーブルの現在の内容と比較します。システムの相互参照レコードにこのセルと同じ値がある場合、MDM Hub は Hub ストア内のセルを更新しません。

セルの更新を有効にすると、MDM Hub にターゲットベースオブジェクトレコードの最終更新日と信頼値への更新が要求されない場合に、ロードジョブのパフォーマンスが向上します。

ステーディングテーブルのカラムのプロパティ

ステーディングテーブルのカラムのプロパティは、外部キーのルックアップに関する情報を提供します。このプロパティではまた、ステーディングテーブルカラムに NULL 値が含まれている場合のバッチロードと PUT API の動作を設定することができます。

注: MDM Hub では、空の文字列は、その空の文字列に関連するデータベースタイプに関係なく、NULL 値と同等です。

ステーディングテーブルカラムには次のプロパティがあります。

カラム

関連するベースオブジェクトで定義されているカラムの名前。

ルックアップシステム

ルックアップシステムの名前（ルックアップテーブルが相互参照テーブルの場合）。

ルックアップテーブル

ステーディングテーブルの外部キーカラムの場合、ルックアップカラムを含むテーブルの名前。

ルックアップカラム

ステージングテーブルの外部キーカラムの場合、ルックアップテーブルのルックアップカラムの名前。

NULL の外部キーを許可する

有効な場合、ロードバッチジョブまたは PUT API では、ルックアップカラムに NULL 値が含まれている場合にデータをロードできます。外部キーのリレーションが必要な場合には **【NULL の外部キーを許可する】** を有効にしないでください。

無効な場合、ロードバッチジョブまたは PUT API では、ルックアップカラムに NULL 値が含まれている場合にデータをロードできません。Hub コンソールはレコードを拒否し、レコードをロードしません。

NULL の更新を許可する

ソースからカラムに取り込まれる値が NULL 値で、別のソースから同じカラムに取り込まれる値が NULL 以外の値の場合の動作を制御します。ベストバージョンオブトゥールズ (BVT) の計算を実行するすべてのプロセス (ロード、ロード更新、PUT、Cleanse Put、マージ、マージ解除、BVT の再計算、再検証) で、このプロパティが使用されます。

- True。有効にした場合、NULL 値がこのカラムの最も信頼できる値であれば、プロセスはベースオブジェクトレコードに NULL 値を書き込むことができます。
- False。デフォルト。無効にした場合、別のソースからカラムに取り込まれる値が NULL 以外の値であれば、プロセスはベースオブジェクトレコードに NULL 値を書き込むことができません。

プロセスが実行される時、NULL 値を提供するソースごとに、プロセスはそのソースのステージングテーブルを確認します。カラムで **【NULL の更新を許可する】** プロパティが false に設定されている場合、プロセスはそのカラムの信頼をゼロ未満にダウングレードします。その後、調整された信頼スコアを使用して BVT を計算します。この方法により、プロセスは、ベースオブジェクトレコードに書き込む最も信頼できる NULL 以外の値を選択できるようになります。

次の特別なケースでは、プロセスはステージングテーブルの **【NULL の更新を許可する】** プロパティを無視し、代わりにベースオブジェクトテーブルのカラムで **【NULL 値を適用する】** プロパティを使用します。

- プロセスはソースに関連付けられた複数のステージングテーブルを見つけます。このステージングテーブルでは、カラムに対する **【NULL の更新を許可する】** の複数の設定が混在しています。
- プロセスはソースのステージングテーブルを見つけますが、カラムがステージングテーブルで設定されていません。例えば、サービス呼び出しによって常にカラム値が更新されるため、カラムはステージングテーブルで設定されません。
- プロセスはソースのステージングテーブルを見つけることができません。例えば、相互参照レコードの STG_ROWID_TABLE カラムに値がなく、ステージングテーブルを判断する代替方法が確定されていません。

プロセスは、次のシナリオで **【NULL の更新を許可する】** プロパティおよび **【NULL 値を適用する】** プロパティの両方を無視します。

- すべてのソースから NULL 以外の値が取り込まれる場合は、最も信頼できるソースからの値が採用されます。
- すべてのソースから NULL 値が取り込まれる場合は、NULL 値が採用されます。
- ベースオブジェクトのソースシステムが 1 つの場合は、NULL かどうかに関係なく、そのソースの値がベースオブジェクトに書き込まれます。

NULL の更新を許可する例

3 つの提供元ソースを持つ顧客ベースオブジェクトがあるとします。ロード更新プロセスは、ミドルネームが削除された、つまり値が NULL のソース A からデータをロードします。ソース B とソース C には顧客のミドルネームがあります。

次の表では、3つのソース、ステージングテーブルのミドルネームカラムの設定、信頼の調整、BVT 計算の結果を示します。

ソース	ステージング テーブル ミドルネーム 信頼	ステージング テーブル ミドルネーム NULL の更新 を許可する	XREF レコード ミドルネーム 値	調整後の信頼	ベースオブジ ェクトレコー ド BVT 値
ソース A	90	false	NULL	< 0	-
ソース B	60	false	Edward	60	-
ソース C	80	true	Edwin	80	Edwin

ロード更新プロセスは、ミドルネームカラムに対して BVT 計算を開始します。ソース A は最初、信頼スコアは最も高い 90 ですが、値は NULL です。プロセスはソース A のステージングテーブルを見つけ、ミドルネームカラムで **【NULL の更新を許可する】** プロパティを確認します。プロパティは false です。プロセスは、ソース A のミドルネームカラムの信頼をゼロ未満にダウングレードします。信頼の調整後、ソース C には 80 という最も高い信頼スコアが設定されます。プロセスはソース C からミドルネームを選択して、Edwin をベースオブジェクトレコードに書き込みます。

ステージングテーブルのプロパティの変更

ステージングテーブルのプロパティは、必要なときに変更できます。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、**【ベースオブジェクト】** ノードを展開し、このステージングテーブルに関連付けられているベースオブジェクトのノードを展開します。

ステージングテーブルがベースオブジェクトに関連付けられている場合は、**【ステージングテーブル】** ノードを展開して表示します。

4. 設定するステージングテーブルを選択します。
選択したテーブルのプロパティが表示されます。
5. ステージングテーブルのプロパティを指定します。

編集するプロパティごとに、その横にある **【編集】** ボタンをクリックし、新しい値を指定します。

注: ステージングテーブルとその関連するサポートテーブル（RAW およびプライマリランディングテーブルなど）が空の場合、ソースシステムを変更できます。

ステージングテーブルまたはその関連テーブルにデータが含まれる場合は、ソースシステムを変更しないでください。

6. ベースオブジェクトのカラムのリストで、ソースシステムで提供するカラムを変更します。
 - **【すべて選択】** ボタンをクリックすると、カラムを 1 つずつクリックしなくても、すべてのカラムを選択できます。
 - **【すべてクリア】** ボタンをクリックすると、すべてのカラムのチェックを解除できます。

注: **【行 ID オブジェクト】** と **【最終更新日】** が自動的に選択されます。これらのカラムのチェックを解除したり、プロパティを変更したりすることはできません。
7. 必要に応じて、カラムのプロパティを変更します。

8. 必要に応じて、外部キーカラムのルックアップを変更します。カラムを選択し、**[編集]** ボタンをクリックしてルックアップカラムを設定します。
9. セルの更新を変更する場合は、**[セルの更新]** チェックボックスをクリックします。
10. 必要に応じて、ステージングテーブルのカラムの設定を変更します。
11. 必要に応じて、このステージングテーブルの監査証跡と差分検出を設定します。
12. **[保存]** ボタンをクリックして変更を保存します。

外部キーカラムのルックアップ

ルックアップを使用し、ロードジョブ中に親テーブルからデータを取得できます。ステージングテーブルの外部キーカラムが親テーブルのプライマリキーに関連付けられている場合は、親テーブルからデータを取得するようにルックアップを設定します。

ルックアップテーブルのターゲットカラムは、プライマリキーなどの一意のカラムでなければなりません。

ルックアップを定義した後、ベースオブジェクトに対してロードジョブを実行すると、MDM Hub によって Consumer コード相互参照テーブルのソースシステムカラムからプライマリキー内のソースシステムの Consumer コード値がルックアップされます。ルックアップの後、MDM Hub によってソースの Customer Type に対応する Customer Type である ROWID_OBJECT 値が返されます。

例

組織の MDM Hub 実装に 2 つのベースオブジェクト、Consumer 親ベースオブジェクトと Address 子ベースオブジェクトがあります。これらのベースオブジェクトには次のリレーションが設定されています。

```
Consumer.Rowid_object = Address.Consumer_Fkey
```

この場合、Address ステージングテーブルに Consumer_Fkey が含まれており、このキーによっていずれかのカラムでデータがルックアップされます。

注: Address.Consumer_Fkey は、Consumer.Rowid_object と同じでなければなりません。

この例では、次のタイプのルックアップを設定できます。

- Consumer ベースオブジェクトルックアップテーブルのプライマリキーである ROWID_OBJECT のルックアップ。
- Consumer ベースオブジェクトの相互参照テーブルのプライマリキーである PKEY_SRC_OBJECT カラムのルックアップ。
この場合は、ルックアップシステムも定義する必要があります。相互参照テーブルの PKEY_SRC_OBJECT カラムのルックアップを設定する場合は、このステージングテーブルに関連付けられたソースシステムとは異なるソースシステムに関連付けられた親テーブルを参照します。
- その他の一意のカラム（使用可能であればベースオブジェクトまたはその相互参照テーブル内のもの）のルックアップ。

ルックアップの設定

外部キーリレーションを介してルックアップを設定することができます。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. スキーマのツリーで、**[ベースオブジェクト]** ノードを展開し、このステージングテーブルに関連付けられているベースオブジェクトのノードを展開します。
4. 設定するステージングテーブルを選択します。
5. 設定する外部キーカラムの行を選択します。

[ルックアップの編集] ボタンが、外部キーカラムに対して有効になります。

6. **[ルックアップの編集]** ボタンをクリックします。

スキーママネージャにより、**[ルックアップの定義]** ダイアログボックスが表示されます。

[ルックアップの定義] ダイアログボックスには、親ベースオブジェクトとその相互参照テーブル、および一意のカラムが表示されます。

7. ルックアップのターゲットカラムを選択します。

- ベースオブジェクトに対するルックアップを定義するには、ベースオブジェクトを展開し、Rowid_Object（このベースオブジェクトのプライマリキー）を選択します。
- 相互参照テーブルに対するルックアップを定義するには、PKey Src Object（この相互参照テーブル内のソースシステムのプライマリキー）を選択します。
- その他の一意のカラムに対するルックアップを定義するには、そのカラムを選択します。

注: ルックアップはリレーションを削除するとクリアされます。

8. ルックアップカラムをリレーションテーブルの PKey Src Object にした場合は、**[ルックアップシステム]** リストからルックアップシステムを選択します。
9. **[OK]** をクリックします。
10. 必要に応じて、**[NULL の更新を許可する]** チェックボックスを設定して、すでに NULL 以外の値を含むセルに対してロードジョブで NULL 値が指定された場合の処理を指定します。
11. 各カラムについて、**[NULL の外部キーを許可する]** オプションを設定して、外部キーカラムの値が NULL の場合（使用できるルックアップ値がない場合）の処理を指定します。
12. **[保存]** ボタンをクリックして変更を保存します。

初期データロードの設定

ベースオブジェクトへの初期データロード中、バッチジョブを並行処理してパフォーマンスを向上させることができます。

C_REPOS_TABLE の以下のパラメータを設定すると、初期データロードのパフォーマンスが向上します。

PARALLEL_BATCH_JOB_THRESHOLD

バッチジョブを並行して処理します。値は、MDM Hub が使用できる利用可能な CPU コアの数よりも小さい数に設定します。例えば、マシンに搭載されている CPU が 4 個で、そのうち 2 個のみが MDM Hub 用に使用可能な場合、各 CPU のプロセッサコアが 8 個だとすると、設定できる最大値は 15 です。デフォルトは 1,000 です。

ソースシステムの信頼の設定

ここでは、Informatica MDM Hub 実装で信頼を設定する方法について説明します。

信頼について

ベースオブジェクトテーブルの同じカラムに対応する属性が複数のソースシステムに含まれている場合があります。

例えば、顧客の住所が複数のシステムに格納されている場合があります。ただし、そのデータのソースとして、いずれかのシステムの信頼度が他のシステムよりも高いとします。それらのシステムのデータが一致しない場合、Informatica MDM Hub で使用する最適な値を特定しなければなりません。

複数のソースシステムのカラムデータの相対的な信頼度を比較できるように、Informatica MDM Hub では、カラムに対して信頼を設定することができます。信頼とは、特定のデータの相対的な精度に関する信頼度を指定したものです。各ソースのそれぞれのカラムに対して、0～100 の範囲の数値で表される信頼レベルを定義することができます。信頼度は 0 が最も低く、100 が最も高くなります。この数値は、単独では意味を持ちません。他の信頼の数値と比較してどちらが高いかを判別して初めて意味を持ちます。

信頼では、データの経過時間（時間の経過とともに信頼度がどれだけ減衰したか）とデータの有効性が考慮されます。信頼は、2 つのレコードを統合する際の存続性を判断する場合や、ソースシステムの更新を信頼してマスターレコードに反映してかまわないかどうかを判断する場合に使用されます。

信頼レベル

信頼レベルは、0～100 の範囲の数値で表されます。この数値は、単独では意味を持ちません。別の信頼の数値と比較されたときにのみ意味を持ちます。

データの信頼度の時間経過による減衰

特定のソースシステムから得たデータの信頼性は、時間とともに減衰（減少）する可能性があります。この事実を信頼の計算に反映させるために、Informatica MDM Hub では信頼が有効なカラムの減衰特性を設定することができます。減衰期間は、信頼レベルが最大信頼レベルから最小信頼レベルに減衰するまでの時間です。

信頼の計算

ロードプロセスによって、ベースオブジェクト内の信頼が有効なカラムの信頼が計算されます。信頼が有効なカラムを含むレコードで、ロードプロセスによって信頼スコアがセルデータに割り当てられます。この信頼スコアは、最初はそのカラムの信頼設定に基づきます。それ以降は、信頼の計算後にロードプロセスによって検証ルール（信頼が有効なカラムに対して設定されている場合）が適用されると、場合によっては信頼スコアがダウングレードされます。

すべての信頼の計算は、システム日付に基づいて行われます。ただし、例外として、タイムラインが有効なベースオブジェクトでの履歴クエリに対する信頼の計算は、履歴上の日付に基づいて行われます。信頼が有効なベースオブジェクトの信頼の計算は、有効日に基づきません。

ロードの更新操作に対する信頼の計算

ロードプロセスの実行中に、ステージングテーブルのレコードがロードの更新操作に使用される場合、そのレコード内の信頼が有効なカラムに変更されたセル値が含まれているときは、ロードプロセスによって次の項目の信頼スコアが計算されます。

- ステージングテーブルのソースレコードのセルデータ（更新された情報が含まれる）
- ベースオブジェクトのターゲットレコードのセルデータ（既存の情報が含まれる）

ソースレコードのセルデータの信頼スコアがターゲットレコードのセルデータの信頼スコアよりも高い場合は、Informatica MDM Hub によって、ベースオブジェクトレコードのセルがステージングテーブルレコードのセルデータで更新されます。

2 つのベースオブジェクトレコードを統合するときの信頼の計算

ベースオブジェクトの 2 つのレコードが統合されるとき、Informatica MDM Hub によって、マージされる 2 つのレコードの信頼できるカラムごとに信頼スコアが計算されます。信頼スコアが最も高いセルが最終的な統

合されたレコードに残ります。信頼スコアが同じ場合は、Informatica MDM Hub によってレコードが比較されます。

信頼が有効なカラムの制御テーブル

Informatica MDM Hub では、ベースオブジェクト内の信頼が有効なそれぞれのカラムについて、最終更新日とソースシステムの識別子が格納された対応する制御テーブルでレコードを管理します。それらの設定に基づいて、Informatica MDM Hub では、カラム値の現在の信頼をいつでも計算することができます。

ベースオブジェクトの履歴が有効になっている場合、Informatica MDM Hub では、ベースオブジェクトとその相互参照テーブル用の履歴テーブルとは別に、制御テーブル用の履歴テーブルも保持します。

ベースオブジェクトレコードと相互参照レコードのセルの値

ベースオブジェクトの相互参照テーブルには、各ソースシステムの最新の値が格納されます。ベースオブジェクトには、デフォルトでは（信頼を設定していなければ）、どのソースシステムのものかに関係なく最新の値が格納されます。

信頼が有効なカラムの場合は、ベースオブジェクトレコード内のセルの値が相互参照テーブル内の対応するレコードの値と同じにならないことがあります。ロードプロセス中、信頼計算の後に実行する検証ルールではセルの信頼をダウングレードすることができるため、前回セルの値を提供したソースによってセルが更新されない場合があります。

信頼スコアのオーバーライド

特定の値が正しい場合、データスチュワードは、その値で計算された信頼設定を上書きすることができます。また、データスチュワードは、ベースオブジェクトのレコードに値を直接入力することもできます。詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

状態が有効なベースオブジェクトの信頼

状態が有効なベースオブジェクトの信頼は、アクティブ、保留、および削除状態のレコードに対して計算されます。削除状態のレコードでは、削除状態のレコードがアクティブおよび保留状態のレコードに勝たないように信頼がダウングレードされます。

信頼が有効なカラムの数に関するバッチジョブの制約

同期バッチジョブは、ベースオブジェクトに信頼が有効なカラムが多数あると失敗することがあります。

同様に、自動マージジョブは、信頼が有効なカラムまたは検証が有効なカラムが多数あると失敗することがあります。ジョブが失敗する原因となるカラム数が正確に決まっているのではなく、ジョブが失敗するかどうかはカラム名の長さ（自動マージジョブの場合は検証が有効なカラムも含む）の数に基づきます。カラム名で許容される最大の長さは約 26 文字です。この問題を回避するために、信頼が有効なカラムの数は 100 個未満にし、カラム名は短くするようにしてください。次善策として、ベースオブジェクトの保存前にすべての信頼/検証カラムを有効にして、同期ジョブが実行されないようにする方法もあります。

信頼のプロパティ

この節では、信頼が有効なカラムに対して設定できる信頼プロパティについて説明します。

信頼プロパティは、ベースオブジェクト内の信頼が有効なカラムに対してレコードを提供できるソースシステムごとに個別に設定します。

Maximum Trust: 最大信頼度

最大信頼度（最初の信頼度）は、データ値が変更された場合の信頼レベルです。例えば、ソースシステム X で電話番号フィールドが 555-1234 から 555-4321 に変更された場合、新しい値では、電話番号フィールドに対し

てシステム X の最大信頼度レベルが与えられます。最大信頼度レベルを高く設定することによって、ソースシステムにおける変更が常にベースオブジェクトに適用されるようにすることができます。

Minimum Trust: 最小信頼度

最小信頼度は、データ値が古くなったとき（減衰期間が経過した後）の信頼レベルです。この値は最大信頼度以下である必要があります。

注: 最大信頼度と最小信頼度が同じ場合、減衰曲線は平らになり、減衰期間と減衰タイプは影響しなくなります。

Units: 単位

減衰期間の計算に使用する単位（日、週、月、四半期、または年）を指定します。




Decay: 減衰

減衰期間の計算に使用する（日、週、月、四半期、または年の）数を指定します。

注: グラフの表示を最適化するには、指定する減衰期間を 1~100 の範囲に制限します。

Graph Type: グラフタイプ

減衰は、減衰期間中に信頼レベルが低下するパターンをたどります。グラフタイプはこのような減衰パターンを示し、次のいずれかの設定になります。

アイコン	Graph Type: グラフタイプ	説明
	Linear: 線形	最も単純な減衰。減衰は、最大信頼度から最小信頼度への直線をたどります。
	RISL (Rapid Initial Slow Later)	低下のほとんどが減衰期間の最初に発生します。減衰は凹曲線をたどります。ソースシステムがこのグラフタイプである場合、システムからの新しい値はおそらく信頼されますが、その後にその値は上書きされる可能性が高くなります。
	SIRL (Slow Initial Rapid Later)	低下のほとんどが減衰期間の最後に発生します。減衰は凸曲線をたどります。ソースシステムがこのグラフタイプである場合、値が減衰期間の最後に近づくまで、他のシステムがこの値を上書きする可能性は比較的低くなります。

Test Offset Date: テストオフセット日付

デフォルトでは、信頼減衰グラフに表示される信頼減衰の開始日は現在のシステム日付です。異なる開始日に基づいて特定のソースシステムの信頼減衰の影響を確認するには、異なるテストオフセット日付を指定します。

信頼の値の設定に関する考慮事項

正しい信頼の値を選択するプロセスは複雑になることがあります。

各システムを別々に検討するだけでは十分とは言えません。特定のカラムにデータを渡すすべてのソースシステムについて、それらの信頼の設定の組み合わせが期待どおりに動作することを確認する必要があります。ソースシステムの信頼レベルは絶対値ではありません。信頼が有効なカラムにデータを渡す他のソースシステムの信頼レベルとの相対値です。

信頼を決定するときは、以下の点について検討します。

- このデータ値がソースシステムで検証されているかどうか。その検証の信頼度。

- ソースシステムのユーザーにとってのこのデータ値の重要度（他のデータ値との比較）。ユーザーは一般に、作業の中心となるデータの検証に最も力を入れます。
- ソースシステムの更新頻度。
- 特定の属性の予想される更新頻度。

カラムの信頼

ベースオブジェクトカラムの信頼を有効にして設定します。オペレーショナル参照ストア内のその他のテーブルに対して信頼を有効にすることはできません。

Informatica MDM Hub はカラムの信頼をデフォルトで無効にします。信頼が無効になっている場合、Informatica MDM Hub では最後に実行されたロードプロセスのデータが、その取得元のソースシステムに関係なく使用されます。ベースオブジェクトのカラムデータが1つのシステムだけから取得されている場合は、そのカラムに対して信頼を無効にしておきます。

データが複数のソースシステムから取得される可能性があるカラムについては、信頼を有効にする必要があります。カラムに対して信頼を有効にする場合は、カラムにデータを提供する可能性があるソースシステムの相対的な信頼性を指定する信頼レベルも割り当てます。

信頼カラムと検証ルールを追加すると、バッチロードとバッチマージのパフォーマンスは低下します。信頼カラムと検証ルールを追加することで、制御テーブルの更新文が長くなります。40個を超える数の信頼カラムを有効にすると、Informatica MDM Hub はチャンクにある制御テーブルで一度に40個までカラムを更新します。カラムに最後にロードしたデータが最善データであると考えられる場合は、そのカラムの信頼を有効にしないでください。

過度の数のカラムで信頼と検証を有効にしないでください。Microsoft SQL Server の場合、ページサイズの限度は8 KB です。Unicode 文字をサポートするために、Informatica MDM Hub では NCHAR および NVARCHAR データタイプが使用されます。ダブルバイトサポートにより、最大レコードサイズは4000文字になっています。レコードサイズが4000文字を超過する場合、バッチプロセスが突然失敗する可能性があります。

カラムに対する信頼の有効化

ベースオブジェクトカラムに信頼を有効にするには、ベースオブジェクトカラムのプロパティを編集します。

1. モデルワークベンチにある Hub コンソールで、**【スキーマ】** を選択します。
2. 書き込みロックを取得します。
3. スキーママネージャの左側のペインで、信頼を適用するカラムを持つベースオブジェクトを展開します。**【カラム】** を選択します。
4. そのカラムの **【信頼】** チェックボックスを選択します。 **【保存】** ボタンをクリックします。

信頼が有効なカラムの信頼を設定する前に

信頼が有効なカラムの信頼を設定する前に、以下が完了している必要があります。

- ベースオブジェクトのカラムに対して信頼を有効にする
- スキーママネージャでステージングテーブルを設定する（関連するソースシステム、およびベースオブジェクトのカラムに対応するステージングテーブルのカラムを含む）

管理ソースシステムの信頼の指定

少なくとも、管理ソースシステムの信頼が有効なカラムには信頼設定を指定する必要があります（デフォルトで *Admin* と呼ばれます）。このソースシステムは、Informatica MDM Hub で実行する手動更新を表します。

このソースシステムから、信頼が有効な任意のカラムにデータを渡すことができます。このソースシステムの信頼設定を（他のソースシステムに比較して）高い値に設定し、手動更新が他のソースシステムからの既存の値を上書きするようにします。

ベースオブジェクト内の信頼が有効なカラムへの信頼レベルの割り当て

ベースオブジェクト内の信頼が有効なカラムに信頼レベルを割り当てる手順

1. システムと信頼ツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、[信頼] ノードを展開します。
信頼が有効なカラムを持つすべてのベースオブジェクトが表示されます。
4. ベースオブジェクトを選択します。
選択したベースオブジェクト内の信頼が有効なカラムが読み取り専用ビューに表示され、そのカラムにデータを提供するソースシステムがチェックマークで示されます。
注: 信頼が有効なカラムとソースシステムの間に関連付けは、このベースオブジェクトのステージングテーブルで指定されています。
5. ベースオブジェクトを展開して信頼が有効なカラムを表示します。
6. 設定する信頼が有効なカラムを選択します。
選択した信頼が有効なカラムについて、そのカラムに関連付けられているソースシステムのリストと信頼減衰グラフが表示されます。リストには、ソースシステムごとに設定できる編集可能な信頼の設定が表示されます。
7. 各カラムの信頼のプロパティを指定します。
8. 必要に応じて、オフセット日付を変更できます。
9. **【保存】** ボタンをクリックして変更を保存します。
信頼減衰グラフが、信頼が有効なカラムの各ソースシステムに対して指定した信頼の設定に基づいて更新されます。
X 軸が信頼スコアで、Y 軸が時間を表します。

信頼が有効なカラムのオフセット日付の変更

デフォルトでは、信頼減衰グラフには、すべてのソースシステムについての現在のシステム日付以降の信頼の減衰が示されます。別の日付（将来の日付など）を指定して、現在の信頼の設定をテストし、その日以降の信頼の減衰を確認することができます。オフセット日付は保存されません。

信頼が有効なカラムのオフセット日付を変更する手順

1. システムと信頼ツールで、信頼が有効なカラムを選択します。
2. 別のオフセット日付を指定するソースシステムの横にある **【カレンダー】** ボタンをクリックします。
日付を指定するように求められます。
3. 別の日付を選択します。
4. **【OK】** を選択します。
信頼減衰グラフが、現在の信頼の設定と指定したオフセット日付に基づいて更新されます。

オフセット日付を削除する手順

- オフセット日付を削除するソースシステムの横にある **【削除】** ボタンをクリックします。
信頼減衰グラフが、現在の信頼の設定と現在のシステム日付に基づいて更新されます。

信頼の設定を変更した後の同期バッチジョブの実行

レコードをベースオブジェクトにロードした後、カラムの信頼を有効にした場合、またはそのベースオブジェクト内の信頼が有効なカラムの信頼設定を変更した場合は、統合プロセスを実行する前に同期バッチジョブを実行する必要があります。このバッチジョブを実行しないと、統合プロセスの実行中にエラーが発生します。

検証ルールの設定

ここでは、Informatica MDM Hub 実装で検証ルールを設定する方法について説明します。

検証ルールの概要

検証ルールでは、セルの値が特定の条件に一致する場合にセルの値の信頼をダウングレードします。

注: MDM Hub では、IBM DB2 または Microsoft SQL Server 用のカスタム検証ルールはサポートされません。

各検証ルールで以下を指定します。

- セルの値が有効かどうかを判別する条件
- 条件を満たす場合に実行するアクション（特定の比率分信頼をダウングレード）

例えば、次のような検証ルールがあるとします。

Downgrade trust on First_Name by 50% if Length < 3'

各要素は次のとおりです。

条件

Length < 3

アクション

Downgrade trust on First_Name by 50%

カラムに対して [最小信頼度の保持] フラグが設定されている場合、信頼をカラムの最小信頼度未満にダウングレードすることはできません。ベースオブジェクトの検証ルールを設定するには、スキーママネージャを使用します。

検証ルールは、ロードプロセス中、ベースオブジェクト内の信頼が有効なカラムの信頼が計算された後に実行されます。検証ルールが定義されている場合、ロードプロセスでは、まずそれらを適用して最終的な信頼スコアを特定します。その後、その最終的な信頼の値を使用して、更新されたレコードのセルデータをベースオブジェクトのレコードに反映するかどうかを判断します。

検証チェック

検証チェックは、ベースオブジェクト内の任意のカラムに対して実行できます。検証チェックによるダウングレードは、同じカラムおよび検証可能なその他のカラムに適用できます。したがって、1つのカラムに無効なデータがあると、多数のカラムで信頼のダウングレードが発生する可能性があります。

例えば、住所が完全な場合はフラグが OK で、住所が不完全な場合はフラグが BAD である住所検証フラグを使用したとします。ここで、検証フラグが OK でない場合にすべての住所フィールドの信頼をダウングレードする検証ルールを設定できます。この場合は、検証フラグもダウングレードされることに注意してください。

必須のカラム

検証ルールは入力データのソースにかかわらず適用されます。ただし、検証ルールは、ステージングテーブルまたは入力（サービス統合フレームワーク（SIF）要求）が必須カラムをすべて含んでいる場合にのみ適用されます。必須カラムがない場合、検証ルールは適用されません。

検証ルールを変更した後の信頼スコアの再計算

ベースオブジェクトに既存のデータがある場合に検証ルールを変更したときは、再検証ジョブを実行して新しいデータと既存のデータの信頼スコアを再計算する必要があります。

検証ルールと状態が有効なベースオブジェクト

状態が有効なベースオブジェクトの検証ルールは、アクティブ、保留、および削除状態のレコードに対して適用されます。BVT を計算するときに、削除状態のレコードでは信頼がダウングレードされ、削除状態のレコードがアクティブおよび保留状態のレコードに勝たないようにされます。

検証カラムの数に関する自動マージジョブの制約

検証が有効なカラムが多数あると、自動マージジョブは失敗することがあります。

ジョブが失敗する原因となるカラム数が正確に決まっているわけではなく、ジョブが失敗するかどうかはカラム名の長さや検証が有効なカラムの数に基づきます。カラム名で許容される最大の長さは約 26 文字です。この問題を回避するために、検証が有効なカラムの数は 60 個未満にし、カラム名は短くするようにしてください。次善策として、ベースオブジェクトの保存前にすべての信頼/検証カラムを有効にして、同期ジョブが実行されないようにする方法もあります。

カラムに対する検証ルールの有効化

検証ルールは、スキーママネージャでベースオブジェクトに対してカラムベースで有効にされ、設定されます。

検証ルールは、オペレーショナル参照ストア内の他のテーブルのカラムには適用されません。

検証ルールは、デフォルトでは無効になっています。ただし、信頼が有効なカラムは信頼のダウングレードに検証ルールを使用するため、このカラムに対しては検証ルールを有効にする必要があります。

ベースオブジェクトを更新するときに、信頼が有効なカラムに値を指定しない場合、信頼が有効なカラムに依存する検証ルールは関連付けられた相互参照レコードに適用されません。

ダウングレード率が適用される仕組み

検証ルールでは、次のアルゴリズムに従って信頼スコアがダウングレードされます。

$$\text{Final trust} = \text{Trust} - (\text{Trust} * \text{Validation_Downgrade} / 100)$$

例えば、検証ダウングレードの割合が 50% であり、信頼レベルが 60 で計算されているとします。

$$\text{Final Trust Score} = 60 - (60 * 50 / 100)$$

最終的な信頼スコアは次のようになります。

$$\text{Final Trust Score} = 60 - 30 = 30$$

検証ルールのシーケンス

MDM Hub は指定したシーケンスに従って検証ルールを実行します。

1 つのカラムに複数の検証ルールを設定すると、MDM Hub は設定したシーケンスに従って検証ルールを実行します。また、MDM Hub は各検証ルールのダウングレードの割合を考慮します。ダウングレードの割合は累積されません。ダウングレードの割合が最も高い検証ルールが他の変更を上書きします。

最も高いダウングレードの割合を持つ検証ルールが複数存在する場合、MDM Hub は最小信頼度設定の保持を確認します。最小信頼度設定の保持がすべての検証ルールに対して有効になっている場合、MDM Hub はダウングレードの割合の最も高い最初の検証ルールを適用します。

[検証ルール] ノードへの移動

検証ルールを設定するには、スキーママネージャで、ベースオブジェクトの [検証ルール] に移動します。

1. スキーママネージャを起動します。
2. 書き込みロックを取得します。
3. 設定するベースオブジェクトのツリーを展開し、[検証ルールのセットアップ] ノードをクリックします。

検証ルールエディタが表示されます。

検証ルールエディタは、次のセクションに分かれています。

ペイン	説明
ルール数	選択したベースオブジェクトに対して設定されている検証ルールの数。
検証ルール	選択したベースオブジェクトに対して設定されている検証ルールの一覧。
プロパティペイン	選択した検証ルールのプロパティ。

検証ルールのプロパティ

検証ルールには次のプロパティがあります。

ルール名

検証ルールの一意でわかりやすい名前。

注: ルール名にはカンマを挿入しないでください。カンマを挿入すると、検証ルールの順序が乱れる場合があります。

ルールタイプ

検証ルールのタイプ。ルールタイプは次のいずれかの値を取ります。

ルールタイプ	説明
存在確認	セルの値が NULL、つまりセルの値が存在しない場合、信頼はダウングレードされます。
ドメイン確認	セルの値が許可された値の範囲内でない場合、信頼はダウングレードされます。

ルールタイプ	説明
参照整合性	セルの値が別のテーブルのカラムに含まれる一連の値に存在しない場合、信頼はダウングレードされます。このルールは、明示的な外部キーが定義されていない場合に使用されます。正しくないセル値は、高い信頼を持つ正しいセル値がない場合に許容されます。
パターンの検証	セルの値が適合状態（LIKE）にある場合、または指定されたパターンに適合しない（NOT LIKE）場合、信頼はダウングレードされます。
カスタム	<p>複合検証ルールを入力するためのカスタムルールを使用します。カスタムルールは、例えば LENGTH や ABS といった SQL 関数が必要な場合、または静的テーブルへの complex join が必要な場合にのみ使用します。</p> <p>カスタム SQL コードは、データベースプラットフォームの SQL 構文に準拠する必要があります。入力する SQL は設計時には検証されません。有効でない SQL 構文エラーは、ロードプロセスの実行時に問題を引き起こします。</p>

ルールカラム

各カラムに対してダウングレード率を指定し、最小信頼度を保持するかどうかを指定します。

ダウングレード率

この検証ルール条件に適合した場合に、指定したカラムの信頼レベルが低下する率。この率が高くなるほど、ダウングレードも大きくなります。例えば、0%では信頼がまったく変化しませんが、100%ダウングレードでは、最小信頼度の保持が指定されていない限り信頼が完全に低下します（指定済みの場合は、その値まで低下します）。

信頼が 100%ダウングレードし、そのカラムに対して最小信頼度を有効にしていない場合は、そのカラムの値はベースオブジェクトに設定されません。

最小信頼度の保持

ダウングレードによって信頼レベルがカラムの最小信頼度レベルを下回った場合の挙動を指定します。最小信頼度を保持すると、信頼レベルは最小信頼度までは低下しても、それ以下にはなりません。このボックスをクリア（チェック解除）すると、信頼レベルは指定された率だけ低下し、最小信頼度を下回ることがあります。

ルール SQL

検証ルールの条件を SQL WHERE 句として指定します。ロードプロセスは検証ルールを実行します。[ルール SQL] フィールドで指定した条件にレコードが一致する場合、この検証ルールに設定したダウングレードパーセンテージだけ、信頼値がダウングレードされます。

検証ルールエディタによって、この検証ルール用に選択したルールタイプに基づいて SQL WHERE 句を設定するように求められます。ロードプロセスの実行中、このクエリを使用してステージングテーブルのデータの有効性がチェックされます。

次の表に、ルールタイプと各ルールタイプの SQL WHERE 句の例を示します。

ルールタイプ	WHERE 句	例	結果
存在確認	WHERE S.ColumnName IS NULL	WHERE S.MIDDLE_NAME IS NULL	ミドルネームが NULL のレコードに関して、影響を受けるカラムがダウングレードされます。条件を満たさないレコードは影響を受けません。
ドメイン確認	WHERE S.ColumnName IN ('?', '?', '?')	WHERE S.Gender NOT IN ('M', 'F', 'U')	Gender が M、F、または U 以外の値の場合、影響を受けるカラムがダウングレードされます。
参照整合性	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	アカウントタイプテーブルにないアカウントタイプ値を持つレコードに関して、影響を受けるカラムがダウングレードされます。
パターンの検証	WHERE S.ColumnName LIKE 'Pattern'	WHERE S.eMail_Address NOT LIKE '%@%'	メールアドレスが@文字を含まない場合にダウングレードが適用されます。
カスタム	WHERE	WHERE LENGTH(S.ZIP_CODE) > 4	郵便番号カラムの長さが 4 未満の場合にダウングレードが適用されます。

テーブルのエイリアスとワイルドカード

ワイルドカード文字 (*) を使用して、エイリアスでテーブルを参照できます。

- s.* はステージングテーブルのエイリアスです。
- i.* は一時テーブルのエイリアスで、更新されるレコードの ROWID_OBJECT、PKEY_SRC_OBJECT、および ROWID_SYSTEM 情報を提供します。i エイリアスを使用できるのは、検証ルール内の ROWID_OBJECT、PKEY_SRC_OBJECT、および ROWID_SYSTEM カラムだけです。

カスタムのルールタイプと SQL WHERE 構文

カスタム検証ルールについては、整形されて適切にチューニングされた SQL 文を作成します。SQL 文は、受信データの WHERE 条件の一部として実行されます。

注: カスタム検証ルールを実装すると、ロードプロセスのパフォーマンスが低下します。そのため慎重に使用します。

データベースプラットフォームで規定されている SQL 構文を使用します。SQL WHERE 句の構文およびワイルドカードパターンの詳細については、Informatica MDM Hub の実装で使用しているデータベースプラットフォームの製品マニュアルを参照してください。

括弧を使用して優先順位を指定します。括弧が間違っていたり省略されたりすると、予期しない結果が生じたり、クエリの実行に時間がかかったりする場合があります。例えば、次の文はあいまいで、優先順位がデータベースサーバーによって決定されます。

```
WHERE conditionA AND conditionB OR conditionC
```

次の文では、括弧を使用して明示的に優先順位を指定しています。

```
WHERE (conditionA AND conditionB) OR conditionC  
WHERE conditionA AND (conditionB OR conditionC)
```

この 2 つの文でレコードを評価すると、まったく異なる結果になります。

テーブル間結合のカスタム検証規則の例

2 つのテーブルからのデータを結合するカスタム検証規則の条件を作成できます。一方のテーブルには、受信データの親レコードが含まれています。もう一方のテーブルは、値の静的リストがあるルックアップテーブルです。

注: 条件には親レコードの値を使用します。子レコードの値を必要とする条件を作成すると、ルールが結果を返しません。カスタム検証規則で親テーブルから子テーブルに結合することはできません。

受信データはエイリアス S でテーブルとして参照し、結合されたテーブルはエイリアス I で参照します。

次のコードは、Put (BO - C_AAA_BO) で表現された結合の検証規則 SQL 文を示しています。

```
WHERE NOT EXISTS (SELECT 1 FROM C_B_LU_ADDR_TP T WHERE T.ADDR_TP = S.COL1)
```

次のコードは、生成された SQL 文を示しています。

```
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
```

検証規則の追加

検証規則を追加する手順

1. 検証規則エディタに移動します。
2. **【検証規則の追加】** ボタンをクリックします。
[検証規則の追加] ダイアログが表示されます。
3. 検証規則のプロパティを指定します。
4. この検証規則のルールカラムを選択する場合は、**【編集】** ボタンをクリックします。
[ルールカラムの選択] ダイアログが表示されます。
選択できるカラムは、[検証] フラグが有効になっているカラムです。

この検証ルール WHERE 句で指定した条件を満たす場合に信頼レベルをダウングレードするカラムを選択し、**【OK】** をクリックします。

注: 検証ルールで日付を使用する必要がある場合は、to_date 関数を使用して日付の実際の形式を指定するか、またはデータベースで要求される形式で日付が指定されるようにします。

5. **【OK】** をクリックします。

検証ルールのリストに新しいルールが追加されます。

注: ベースオブジェクトに既存のデータがある場合に検証ルールを変更したときは、再検証ジョブを実行して新しいデータと既存のデータの信頼スコアを再計算する必要があります。

検証ルールのプロパティの編集

検証ルールを編集する手順

1. スキーママネージャで検証ルールエディタに移動します。
2. [検証ルール] リストで、設定する検証ルールを選択します。
検証ルールエディタに、選択した検証ルールのプロパティが表示されます。
3. 検証ルールの編集可能なプロパティを指定します。ルールタイプは変更できません。
4. この検証ルールのルールカラムを選択する場合は、**【編集】** ボタンをクリックします。

[ルールカラムの選択] ダイアログが表示されます。

選択できるカラムは、[検証] フラグが有効になっているカラムです。

この検証ルール WHERE 句で指定した条件を満たす場合に信頼レベルをダウングレードするカラムを選択し、**【OK】** をクリックします。



5. **【保存】** ボタンをクリックして変更を保存します。

注: ベースオブジェクトに既存のデータがある場合に検証ルールを変更したときは、再検証ジョブを実行して新しいデータと既存のデータの信頼スコアを再計算する必要があります。

検証ルールの順序の変更

検証ルールの実行順序は非常に重要です。

リスト内の検証ルールの順序を変更するには、以下のボタンを使用します。

アイコン	処理....
	選択した検証ルールをシーケンス内で上に移動します。
	選択した検証ルールをシーケンス内で下に移動します。

検証ルールの削除

検証ルールを削除する手順

1. スキーママネージャで検証ルールエディタに移動します。
2. 検証ルールリストで、削除する検証ルールを選択します。
3. **【削除】** ボタンをクリックします。

削除を確認するように求められます。

4. **【はい】** をクリックします。

注: ベースオブジェクトに既存のデータがある場合に検証ルールを変更したときは、再検証ジョブを実行して新しいデータと既存のデータの信頼スコアを再計算する必要があります。

第 21 章

一致プロセスの設定

この章では、以下の項目について説明します。

- [作業を開始する前に, 384 ページ](#)
- [一致プロセスの設定タスク, 384 ページ](#)
- [\[一致/マージ設定の詳細\] ダイアログへの移動, 386 ページ](#)
- [ベースオブジェクトの一致プロパティの設定, 387 ページ](#)
- [関連レコードの一致パスの設定, 391 ページ](#)
- [バスコンポーネントの設定, 398 ページ](#)
- [一致カラムの設定, 401 ページ](#)
- [一致ルールセットの一致カラムルールの設定, 414 ページ](#)
- [プライマリキーの一致ルールの設定, 435 ページ](#)
- [一致キーの分布の調査, 437 ページ](#)
- [マッチプロセスからのレコードの除外, 440 ページ](#)
- [近接検索, 440 ページ](#)
- [軽量一致, 442 ページ](#)

作業を開始する前に

作業を開始する前に、Informatica MDM Hub をインストールし、『*Multidomain MDM のインストールガイド*』の手順に従って Hub ストアを作成し、スキーマを構築しておく必要があります。

一致プロセスの設定タスク

ここでは、マッチプロセスに関連する設定タスクの概要を示します。

データの把握

一致ルールを定義する前に、データについて深く理解し、次のことを把握する必要があります。

- 重複レコードの特定に使用するカラム内の値の分布
- 重複しているレコードの合計数の大まかな割合

一致プロセスに関連するベースオブジェクトのプロパティ

一致プロセスの動作に影響するベースオブジェクトのプロパティを以下に示します。

プロパティ	説明
重複一致しきい値	初期データロードに対する重複データの致ジョブでのみ使用されます。
最大一致経過時間 (分)	一致ルールを実行する際のタイムアウト (分)。この時間が経過すると一致プロセスが終了します。
一致フラグ監査テーブル	有効にした場合、監査テーブル (<i>BusinessObjectName_FMHA</i>) が作成され、マージマネージャで自動マージ用のキューに手動一致レコードを追加したユーザーのユーザー ID が取り込まれます。

一致ルールを定義するための設定手順

一致ルールを定義する手順

1. ベースオブジェクトの一致プロパティを設定します。
2. 一致カラムを定義します。
3. 一致ルールの一致ルールセットを定義します。
4. ルールセットの一致ルールを定義します。
5. 手順 3 と 4 を繰り返して一致ルールを完成させます。
6. 把握できているデータの内容に基づいて、プライマリキーに基づく一致処理が必要かどうかを判断します。
7. プライマリキーの一致処理に適したデータの場合は、プライマリキーの一致ルールを作成します。
8. ルールを調整します。このプロセスは、代表的なデータセットに一致ルールを適用したり、結果を分析したり、一致のパフォーマンスを最適化するために設定を調整したりするたびに繰り返し行います。

インターナショナルデータを含むベースオブジェクトの設定

Informatica MDM Hub では、米国以外のポピュレーションのデータを含むベースオブジェクトや、異なるポピュレーション（米国と中国など）のデータを含むベースオブジェクトのマッチングをサポートしています。

分散一致設定

オペレーショナルリファレンスストアに複数のプロセスサーバーを設定すると、分散一致を設定できます。複数のプロセスサーバーを同時に実行することで、一致プロセスのスループットを向上させることができます。

分散一致は、`cmxcleanse.properties` ファイルで設定する必要があります。有効にするには、`cmx.server.match.distributed_match` プロパティを 1 に設定する必要があります。デフォルトでは無効になっています。

データロードの設定

データロードプロセスを設定して、中間ファイルを使用したり、データをデータベースに直接ロードしてトークン化およびマッチプロセスを行うことができます。`cmxcleanse.properties` ファイルでプロパティを設定して、データロードの方法やバッチのサイズを指定することができます。デフォルトはダイレクトロードです。

デフォルトの動作を変更するには、データロードプロパティを `cmxcleanse.properties` ファイルに追加します。`cmxcleanse.properties` ファイルは次のディレクトリにあります。

Windows の場合:<MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\cleanse\resources

UNIX の場合:<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/resources

次の表に、トークン化およびマッチングに関するデータロードプロパティを示します。

プロパティ	説明
cmx.server.tokenize.file_load	中間ファイルを使用して、データをトークン化するためにデータベースにロードするかどうかを指定します。true に設定すると、中間ファイルを使用してデータをロードします。直接データロードの場合は、false に設定します。Oracle 環境と IBM DB2 環境の場合、デフォルトは true です。Microsoft SQL Server 環境の場合、デフォルトは false です。 注: Microsoft SQL Server に適用できないデータをロードする場合は、中間ファイルを使用します。
cmx.server.tokenize.loader_batch_size	ダイレクトロード中にデータベースに送る INSERT 文の最大数。デフォルトは 1000 です。
cmx.server.match.file_load	中間ファイルを使用して、マッチングを行うためにデータをデータベースにロードするかどうかを指定します。true に設定すると、中間ファイルを使用してデータをロードします。直接データロードの場合は、false に設定します。Oracle 環境と IBM DB2 環境の場合、デフォルトは true です。外部マッチングに設定されている Microsoft SQL Server 環境と IBM DB2 環境の場合、デフォルトは false です。 注: 中間ファイルを使用したデータのロードは Microsoft SQL Server には適用できません。
cmx.server.match.loader_batch_size	ダイレクトロード中にデータベースに送る INSERT 文の最大数。デフォルトは 1000 です。

[一致/マージ設定の詳細] ダイアログへの移動

ベースオブジェクトの一致およびマージプロセスを設定するには、最初に以下の手順を実行します。

1. スキーママネージャを起動します。
2. スキーマナビゲーションツリーで、一致プロパティを定義するベースオブジェクトを展開します。
3. スキーマナビゲーションツリーで、**[一致/マージ設定]** を選択します。

[一致/マージ設定の詳細] ダイアログボックスが表示されます。

設定を変更する場合は書き込みロックを獲得する必要があります。

[一致/マージ設定の詳細] ダイアログには以下のタブがあります。

タブ名	説明
プロパティ	一致/マージ設定の要約と、設定可能な各種の一致/マージ設定が表示されます。
パス	異なるベースオブジェクト間または同じベースオブジェクト内のレコードの親/子リレーションのマッチパスを設定できます。
一致カラム	一致カラムルールのマッチカラムを設定できます。
一致ルールセット	一致ルールセットを使用して検索ストラテジを定義できます。
プライマリキーの一致ルール	プライマリキーの一致ルールを定義できます。
一致キーの分布	一致キーの分布を表示します。
マージ設定	設定をマージおよびリンクできます。

ベースオブジェクトの一致プロパティの設定

一致カラムや一致ルールなどの他の一致機能を設定する前に、ベースオブジェクトの一致プロパティを設定する必要があります。

これらの一致プロパティは、ベースオブジェクトのすべてのルールに適用されます。

一致プロパティの設定

各ベースオブジェクトに対して一致プロパティを設定します。これらの設定は、その一致ルールおよびルールセットすべてに適用されます。

ベースオブジェクトの一致プロパティを設定する手順

1. スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ペインを表示します。
2. [一致/マージ設定の詳細] ペインで、**[プロパティ]** タブをクリックします。
[プロパティ] タブが表示されます。
3. 書き込みロックを取得します。
4. 変更するプロパティ設定を編集するには、該当するフィールドの横の **[編集]** ボタンをクリックします。
5. **[保存]** ボタンをクリックして変更を保存します。

一致プロパティ

この節では、[プロパティ] タブでの構成設定について説明します。

計算済みの読み取り専用フィールド

「プロパティ」タブには、以下の読み取り専用フィールドが表示されます。

表 1. 読み取り専用の一致プロパティ

プロパティ	説明
一致カラム	このベースオブジェクトに対して設定されている一致カラムの数。読み取り専用。
一致ルールセット	このベースオブジェクトに対して設定されている一致ルールセットの数。読み取り専用。
アクティブなセットの一致ルール	現在選択されているアクティブなルールセットでこのベースオブジェクトに対して設定されている一致ルールの数。読み取り専用。
プライマリキーの一致ルール	このベースオブジェクトに対して設定されているプライマリキーの一致ルールの数。読み取り専用。

手動統合の最大一致数

この設定によって、データスチュワードが多数の一致を手動で統合する手間を軽減することができます。

この設定は、データスチュワードが決定すべき一致候補の数を制限します（デフォルトは 1,000）。この制限値に到達すると、手動統合用のレコード数が減少するまで、Informatica MDM Hub は一致プロセスの実行を停止します。

この値は、consolidation_ind=2 のレコード数をチェックすることで算出されます。自動一致およびマージの各サイクルの最後にこのカウントがチェックされ、手動統合用の一致の最大数を超えている場合は、自動一致およびマージプロセスが終了します。

一致ジョブバッチサイクルあたりの行数

この設定では、一致プロセス（一致、自動一致、またはマージジョブ）の実行中に Informatica MDM Hub が処理するレコード数の上限を指定します。一致プロセスの実行を開始する際に、一致ジョブバッチに含まれるレコードにフラグを設定することによって開始されます。一致用に準備された新しい/未統合のレコードのプールが（CONSOLIDATION_IND=4）、一致プロセスによって CONSOLIDATION_IND が 3 に変更されます。フラグ付けされるレコードの数は、「一致ジョブバッチサイクルあたりの行数」で指定します。一致プロセスは、バッチジョブのこれらのレコードと、ベースオブジェクトのすべてのレコードを一致させます。

一致ジョブバッチのレコード数は、一致処理の実行の所要時間に影響します。指定する値は、データセットのサイズ、一致ルールの複雑さ、一致プロセスの実行に利用できるタイムウィンドウの長さによって決まります。デフォルトの一致バッチサイズは低（10）です。この値は、ベースオブジェクトのレコード数、および一致ルールを基にこれらのレコードに対して生成される一致レコード数に基づいて増加できます。

- 一致バッチサイズが小さくなるほど、一致および統合プロセスの実行に必要な時間は増加します。
- 一致バッチサイズが大きくなるほど、時間あたりに処理される各一致および統合プロセスは増加します。

各ベースオブジェクトに対して、一致バッチサイズについて処理時間と処理数のバランスが最適となるポイントがあります。この最適バッチサイズの特定は、使用環境内におけるパフォーマンス調整の一環として行う必要があります。一致バッチサイズは、一致および統合するレコード量の 10%から始めます。一致ジョブだけを実行し、その一致ルールに対して生成される一致の数を確認して、その結果を考慮して値を増減していきます。

一致しないすべての行を一意とする

この機能は、一致プロセスで一致が見つからなかったレコードを Informatica MDM Hub で一意 (CONSOLIDATION_IND=1) とマークする場合に有効にします (【はい】 に設定します)。

有効にすると、それらのレコードの状態が、Informatica MDM Hub によって自動的に統合済みに変更されます (統合インジケータが 2 から 1 に変更されます)。統合済みのレコードは、自動マージバッチジョブでデータスチュワードのキューから削除されます。

デフォルトでは、このオプションは無効になっています。開発環境では、このオプションを無効にする場合があります。例えば、特定の一致ルールセットに対して一意と見なすレコードを決定するために、一致ルールのテストと調整を繰り返す場合などです。

このオプションは、プロダクション環境では常に有効にします。そうしないと、統合インジケータが 2 のレコードが多数ある場合に終了してしまうことがあります。この未処理のレコードの数が [手動統合の最大一致数] の設定値を超えた場合は、それらのレコードを先に処理しないと、他のレコードの一致や統合の処理を続行できなくなります。

一致/検索ストラテジ

一致/検索ストラテジを選択して、必要とされる一致の信頼性とパフォーマンスを指定します。

次のいずれかのオプションを選択します。

ストラテジオプション	説明
あいまい	スペルの差異、スペルミスの可能性、および一致するレコードが不一致と見なされる可能性があるその他の差異を考慮する確率的な一致。これは、ベースオブジェクトのデータを一致させる主要な方法です。このドキュメントでは、 あいまい一致ベースオブジェクト と呼んでいます。 注: あいまい一致/検索ストラテジを指定する際、あいまい一致キーを指定する必要があります。
完全	一致カラムに同一の値を持つレコードのみの一致。完全一致を指定する場合は、このベースオブジェクトに対する完全一致カラムのみを定義できます (完全一致ベースオブジェクトにあいまい一致カラムを含めることはできません)。このドキュメントでは、 完全一致ベースオブジェクト と呼んでいます。

完全一致ストラテジの方が処理は高速ですが、完全一致では、データに誤りがあると一致が見落とされる可能性があります。どちらの方法が適しているかは、データの特性、データについて把握できている情報、一致や統合に関する具体的な要件などによって異なります。

あいまいポピュレーション

一致/検索ストラテジが「あいまい」である場合は、一致させるレコードの一定の特性を定義する**ポピュレーション**を選択する必要があります。

データの特性は、国によって変わる場合があります。デフォルトでは、Informatica MDM Hub にデモポピュレーションが設定されていますが、Informatica には国ごとの標準ポピュレーションが用意されています。別のポピュレーションが必要である場合は、Informatica サポートにお問い合わせください。完全一致/検索ストラテジを選択した場合は、この値が無視されます。

ポピュレーションは、一致処理で以下の機能を果たします。

- 氏名や住所などの識別データによく見られる避けられない相違や間違いに対応します。
例えば、US ポピュレーションには、米国のデータで使用される代表的な識別番号（社会保障番号など）の情報が含まれています。ポピュレーションには、一般的な名前の分布に関する情報も含まれています。例えば、US ポピュレーションでは、Smith 姓が比較的高い割合を占めています。一方、非英語圏のポピュレーションでは、Smith が一般的な名前に含まれることはありません。
- Informatica MDM Hub で一致トークンがどのように構築されるかを指定します。
- 一致させるデータのポピュレーションに、検索ストラテジと一致目的がどのように作用するかを指定します。

以前の行 ID オブジェクトのみ一致させる

このプロパティは、現在のレコードを ROWID_OBJECT 値がより低いレコードと照合する場合に有効にします。

例えば、現在のレコードの ROWID_OBJECT 値が 100 であれば、このレコードはベースオブジェクト内の ROWID_OBJECT 値が 100 未満であるその他のレコードとのみ照合されます。ROWID_OBJECT 値が 100 を超えるレコードは、マッチプロセスで無視されます。

「以前の行 ID オブジェクトのみ一致させる」プロパティは、必要な照合数を減らし、パフォーマンスを高めるために使用します。ただし、PUT API 呼び出しを実行するか、レコードが行 ID の順序で挿入されていない場合は、レコードの一致が十分に行われないことがあります。データの特性と特定の一致要件に基づいて、パフォーマンスと一致件数のトレードオフを評価する必要があります。

「以前の行 ID オブジェクトのみ一致させる」プロパティを有効にすると、初期データロードのパフォーマンスを向上させることができます。増分データロードについてこのプロパティを有効にすると、一致候補を失う可能性があります。デフォルトでは、このオプションは無効になっています。

1 回だけ一致させる

「[以前の行 ID オブジェクトのみ一致させる](#)」(ページ 390)がオンになっている（選択されている）状態であいまいなキー一致を行う場合にのみ使用できます。

「1 回だけ一致させる」が有効（オン）になっている場合は、レコードの一致が検出された時点で、Informatica MDM Hub ではこの検索範囲（一連の類似する一致キー値）でそのレコードがそれ以上一致されません。この機能を使用すると、重複の件数が減り、パフォーマンスが向上する可能性があります。Informatica MDM Hub では、あるレコードに対するすべての一致を検索範囲内で検出する代わりに、レコードごとに 1 つずつ一致を検出できます。後続の一致サイクルでは、マージプロセスによって、これらの一致がベースオブジェクトに関連付けられた XREF レコードの大きなグループに入れられます。

デフォルトでは、このオプションはオフ（無効）になっています。ただしこの機能を有効にすると、一致を見逃す可能性があります。例えば、レコード A がレコード B およびレコード C と一致しても、レコード B とレコード C が一致しない場合などです。データの特性および特定の一致要件に基づいて、パフォーマンスと一致件数との間のトレードオフを評価する必要があります。

動的一致分析しきい値

一致プロセスでは、動的一致分析によって、一致プロセスに容認できないほど長い時間がかかるかどうかが判別されます。

このしきい値は、許容される最大比較件数を示します。

動的一致分析しきい値を有効にするには、ゼロ以外の値を指定します。きわめて似たデータがある（一致が集中して発生している）場合は、この機能を有効にして、データのホットスポットに消費される作業量を削減します。ホットスポットとは、一致過多のデータ（一致が非常に多く発生している部分）を表すレコードのグループです。「動的一致分析しきい値」が有効になっている場合は、指定された数よりも多くの潜在的な一致候補

を生成するレコードが、一致プロセスでスキップされます。デフォルトでは、このオプションはゼロ（無効）に設定されています。

Informatica MDM Hub では、所定の検索範囲で一致を実施する前に、検索レコード（一致が検索されるレコード）の数が計算され、その数値にファイルレコードの数（一致キーテーブルから返された比較が必要なレコードの数）が掛けられます。その結果が指定された動的な一致分析しきい値よりも大きい場合は、その範囲のデータで比較が実行されず、その範囲がさらなる調査の対象としてアプリケーションサーバーログに記録されます。

保留中のレコードに対して一致を有効にする

デフォルトでは、一致プロセスに組み込まれるのはアクティブレコードだけで、保留レコードは無視されます。状態管理が有効なオブジェクトの場合は、このチェックボックスを選択して保留レコードを一致プロセスに加えます。削除済みレコードは、この設定に関係なく一致プロセスで無視されます。

長い ROWID_OBJECT 値のサポート

ベースオブジェクトに、ROWID_OBJECT 値が 12 桁を超える多数のレコードなどがある場合、プロセスサーバーで長い値の明示的なサポートを有効にする必要があります。

プロセスサーバーで長い Rowid Object 値の使用を有効にするには、cmxcleanse.properties ファイルを編集し、cmx.server.bmg.use_longs 設定を以下のように設定する必要があります。

```
cmx.server.bmg.use_longs=1
```

デフォルトでは、このオプションは無効になっています。

関連レコードの一致パスの設定

ここでは、Informatica MDM Hub 実装で一致処理に使用される、関連レコードの一致パスを設定する方法について説明します。

一致パス

一致パスによって、レコード間の階層をトラバースすることができます。この階層は、ベースオブジェクト間に存在する場合もあれば（テーブル間パス）、単一のベースオブジェクト内に存在する場合もあります（テーブル内パス）。一致パスは、別々のテーブルまたは同じテーブルにある関連レコードが関係する一致カラムルールを設定するために使用されます。

外部キーのリレーションとフィルタ

他のレコードを指す一致パスの設定には、2 つの主要なコンポーネントが関係します。

コンポーネント	説明
外部キーリレーション	他のレコードとのリレーションをトラバースするために使用されます。親から子へのリレーションおよび子から親へのリレーションを指定できます。
フィルタ（オプション）	指定したカラム（ADDRESS_TYPE や PARTY_TYPE など）の値に基づいて、レコードを選択的に含めるか、または除外することができます。

リレーションベースオブジェクト

これらの種類のリレーション、特に多対多のリレーションの一致ルールを設定するには、Informatica MDM Hub にレコード間のリレーションを示すためにリレーションベースオブジェクトとして提供される個別のベースオブジェクトを作成する必要があります。このリレーションベースオブジェクトに、Informatica MDM Hub プロセス（ランド、ステージ、ロード）を使用する代わりにデータ管理ツール（Informatica MDM Hub の外部）を使用してリレーションに関する情報を入力できます。

リレーションのタイプ別に、個別のリレーションベースオブジェクトを設定します。開始日、終了日、およびその他のリレーションの詳細など、リレーションタイプの追加の属性を含めることができます。リレーションベースオブジェクトは、一致カラムルールの設定を可能にする一致パスを定義します。

重要: テーブル間パスまたはテーブル間一致パスでレコードのリレーション定義に使用するベースオブジェクトに対して、一致プロセスおよび統合プロセスを実行しないでください。実行した場合、リレーションのデータが変わり、レコード間の関連付けが失われます。

テーブル間パス

テーブル間パスは、2つの異なるベースオブジェクトのレコード間にリレーションを定義します。多くの場合、このリレーションは外部キーリレーションを設定するだけで定義できます。つまり、子ベースオブジェクト内のキーカラムが、親ベースオブジェクトのプライマリキーを指すようにします。

ただし場合によっては、レコード間のリレーションがより複雑で、2つのテーブルのレコード間にリレーションを定義する中間のベースオブジェクトが必要となることがあります。

テーブル間パスのベースオブジェクトの例

Informatica MDM Hub の実装に2つのベースオブジェクトがある以下の例について考えてみます。

ベースオブジェクト	説明
Person	組織の従業員、他の組織（見込み客、顧客、ベンダ、またはパートナ）の従業員、契約者などのあらゆるタイプの個人が含まれています。
Address	郵送先、納入先、自宅、仕事先などのあらゆるタイプの住所が含まれています。

この例では、多対多のリレーションが発生する可能性があります。

- 個人が複数の住所（自宅や仕事先など）を持っている可能性があります。
- 1つの住所に複数の個人が存在する可能性があります（仕事場や住居など）。

異なるベースオブジェクトのレコード間に発生するこの種のリレーションに対して一致ルールを設定するには、Informatica MDM Hub に対して2つのベースオブジェクトのレコード間のリレーションを記述するベースオブジェクト（PersAddrRel など）を別途作成します。

ベースオブジェクトの例のカラム

Person ベースオブジェクトには以下のカラムが含まれているとします。

カラム	タイプ	説明
ROWID_OBJECT	CHAR(14)	プライマリキー。ベースオブジェクト内のこの人物を一意に識別します。
TYPE	CHAR(14)	人物のタイプ（従業員や顧客担当者など）。
NAME	VARCHAR(50)	人物の名前（この例では簡略化）。
EMPLOYER	VARCHAR(50)	人物の雇用主。
...

Address ベースオブジェクトには以下のカラムが含まれているとします。

カラム	タイプ	説明
ROWID_OBJECT	CHAR(14)	プライマリキー。この従業員を一意に識別します。
TYPE	CHAR(14)	住所のタイプ（自宅、職場、郵送先住所、出荷先住所など）。
NAME	VARCHAR(50)	この住所の個人または組織の名前。
ADDRESS_1	VARCHAR(50)	住所の 1 行目。
ADDRESS_2	VARCHAR(50)	住所の 2 行目。
CITY	VARCHAR(50)	市区町村
STATE_PROV	VARCHAR(50)	都道府県
POSTAL_CODE	VARCHAR(50)	郵便番号
...

2つのベースオブジェクト内のレコード間のリレーションを定義するために、PersonAddrRel ベースオブジェクトに以下のカラムが含まれているとします。

カラム	タイプ	説明
ROWID_OBJECT	CHAR(14)	プライマリキー。ベースオブジェクト内のこの人物を一意に識別します。
PERS_FK	CHAR(14)	Person ベースオブジェクトの ROWID_OBJECT カラムに対する外部キー。
ADDR_FK	CHAR(14)	Address ベースオブジェクトの ROWID_OBJECT カラムに対する外部キー。

外部キーカラムのカラムのタイプ（CHAR(14)）は参照先のプライマリキーと一致します。

設定手順の例

リレーションベースオブジェクト（PersonAddrRel）を設定した後に、以下のタスクを実行します。

1. このベースオブジェクトから Person および Address ベースオブジェクトの ROWID_OBJECT への外部キーを設定します。
2. PersAddrRel ベースオブジェクトに、レコード間のリレーションを表すデータをロードします。

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
...

この例では、Person #786 に 2 つの住所があり、Address #1028 に 2 人の人物がいます。

3. 関連するレコードの一致カラムルールを設定する場合は、PersonAddrRel ベースオブジェクトを使用します。

テーブル内パス

ベースオブジェクト内では、個々のレコード間に親/子リレーションが存在する可能性があります。

Informatica MDM Hub では、同じベースオブジェクト内のレコード間のリレーションを明確化し、カラム一致ルールを設定する際にそれらのリレーションを使用することができます。

テーブル内パスのベースオブジェクトの例

従業員の間に直属リレーションが存在する次の従業員ベースオブジェクトの図について考えてみます。



従業員間のリレーションは階層的です。階層の最上位には CEO が位置し、最上位親レコードとなります。

ベースオブジェクトの例のカラム

Employee ベースオブジェクトには以下のカラムが含まれているとします。

カラム	タイプ	説明
ROWID_OBJECT	CHAR(14)	プライマリキー。ベースオブジェクト内のこの従業員を一意に識別します。
NAME	VARCHAR(50)	従業員名。
TITLE	VARCHAR(50)	従業員の役職。
...

リレーションベースオブジェクトの作成

このようなオブジェクトの一致ルールを設定するには、Informatica MDM Hub に対してレコード間のリレーションを示す独立したベースオブジェクトを作成します。

例えば、以下のカラムを含む EmplRepRel ベースオブジェクトを作成して設定します。

カラム	タイプ	説明
ROWID_OBJECT	CHAR(14)	プライマリキー。このリレーションレコードを一意に識別します。
EMPLOYEE_FK	CHAR(14)	従業員レコードの ROWID_OBJECT に対する外部キー。
REPORTS_TO_FK	CHAR(14)	マネージャレコードの ROWID_OBJECT に対する外部キー。

注: 外部キーカラムのカラムのタイプ (CHAR(14)) は参照先のプライマリキーと一致します。

設定手順の例

このベースオブジェクトを設定した後に、以下のタスクを実行する必要があります。

1. このベースオブジェクトから従業員ベースオブジェクトの ROWID_OBJECT への外部キーを設定します。
2. このベースオブジェクトに、レコード間のリレーションを表すデータをロードします。

ROWID_OBJECT	EMPLOYEE	REPORTS_TO
1	7	93
2	19	71
3	24	82
4	29	82
5	31	82
6	31	71

ROWID_OBJECT	EMPLOYEE	REPORTS_TO
7	48	16
8	53	12

レコード間には、多対多のリレーションを定義できます。例えば、ROWID_OBJECT が 31 の従業員は、2 人の上司（ROWID_OBJECT=82 および ROWID_OBJECT=71）の直属部下であり、この上司（ROWID_OBJECT=82）には 3 人（ROWID_OBJECT=24、29、および 31）の直属部下がいます。

3. 関連するレコードの一致カラムルールを設定する場合は、EmplRepRel ベースオブジェクトを使用します。

例えば、従業員の上司を考慮に入れた一致ルールを作成して、より正確な一致を得ることができます。

注: この例では、REPORTS_TO フィールドを使用してリレーションを定義していますが、RELATIONSHIP_TYPE のようなより汎用的で柔軟な情報を使用してレコードを関連付けることも可能です。

[パス] タブへの移動

ベースオブジェクトの [パス] タブに移動する手順

1. スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログに移動します。
2. **[パス]** タブをクリックします。
[パスコンポーネント] ウィンドウが表示されます。

[パス] タブのセクション

[パス] タブには 2 つのセクションがあります。

セクション	説明
パスコンポーネント	リレーションのトラバースに使用する外部キーを設定します。
フィルタ	レコードを一致に含めるまたは除外するために使用するフィルタを設定します。

ルートベースオブジェクト

ルートベースオブジェクトは画面の [パスコンポーネント] セクションに自動的に表示され、常に使用できます。

ルートベースオブジェクトは、子リレーションまたは親リレーションのないエンティティを表します。親レコードまたは子レコードに関する一致ルールを設定する場合、ルートベースオブジェクトにパスコンポーネントを明示的に追加する必要があります。またこれらのリレーションは事前に設定しておく必要があります。

一致パスのフィルタの設定

一致パスのフィルタには、指定するカラムの値に基づいて、照合するレコードを含めることも除外することもできます。カラムに対するフィルタを定義するには、フィルタ条件で、一致処理の対象に含めるレコードを決定する値を 1 つ以上指定します。例えば、出荷先住所と請求先住所を含む Address ベースオブジェクトの場合、請求先住所を含めて出荷先住所を除外したフィルタを設定できます。照合プロセスを実行すると、MDM Hub によって一致バッチ内のレコードと請求先住所レコードが照合されます。

注: 日付カラムにフィルタ条件を指定する場合は、データベースロケールと互換性のある日付の値の正しい文字列表現を使用します。

Informatica MDM Hub では、フィルタに以下のプロパティがあります。

設定	説明
カラム	現在選択されているベースオブジェクトで設定するカラム。
演算子	このフィルタに使用する演算子。以下のいずれかの値を指定します。 <ul style="list-style-type: none">- IN—指定された値が入っているカラムを含めます。- NOT IN—指定された値が入っているカラムを除外します。
値	このフィルタに使用する 1 つ以上の値。

例えば、Address ベースオブジェクト内の郵送先住所のみを一致させたい場合は、以下のように指定します。

設定	値の例
カラム	ADDR_TYPE
演算子	IN
値	MAILING

この例では、郵送先住所、つまり COLUMN フィールドに「MAILING」が含まれているレコードのみが一致の対象となります。それ以外のレコードはすべて無視されます。

フィルタの追加

複数のフィルタを追加する場合、Informatica MDM Hub では、論理 AND 演算子を使用して式全体が評価されます。以下に例を示します。

xExpr AND yExpr AND zExpr

フィルタを追加する手順

1. スキーママネージャで、**[パス]** タブに移動します。
2. 書き込みロックを取得します。
3. **[フィルタ]** セクションで、**[追加]** ボタンをクリックします。
[フィルタの追加] ダイアログが表示されます。
4. このパスコンポーネントのプロパティを指定します。
5. このフィルタの値を指定します。
6. **[保存]** ボタンをクリックして変更を保存します。

フィルタの値の編集

フィルタの値を編集する手順

1. 次のいずれかを実行します。
 - フィルタを追加する。

- フィルタのプロパティを編集する。
2. [フィルタの追加] または [フィルタの編集] ダイアログで、[値] フィールドの横の **【編集】** ボタンをクリックします。
スキーママネージャは [値の編集] ダイアログを表示します。
 3. このフィルタの値を設定します。
 - 値を追加する場合は、**【追加】** ボタンをクリックします。プロンプトが表示されたら、値を指定して **【OK】** をクリックします。
 - 値を削除する場合は、[値の編集] ダイアログで値を選択し、**【削除】** ボタンをクリックします。値を削除するためのプロンプトが表示されたら **【はい】** をクリックします。
 4. **【OK】** をクリックします。
 5. **【保存】** ボタンをクリックして変更を保存します。

フィルタのプロパティの編集

フィルタのプロパティを編集する手順

1. スキーママネージャで、**【パス】** タブに移動します。
2. 書き込みロックを取得します。
3. [フィルタ] セクションで、**【編集】** ボタンをクリックします。
[フィルタの追加] ダイアログが表示されます。
4. このパスコンポーネントのプロパティを指定します。
5. このフィルタの値を指定します。
6. **【保存】** ボタンをクリックして変更を保存します。

フィルタの削除

フィルタを削除する手順

1. スキーママネージャで、**【パス】** タブに移動します。
2. 書き込みロックを取得します。
3. [フィルタ] セクションで、削除するフィルタを選択し、**【削除】** ボタンをクリックします。
削除を確認するように求められます。
4. **【はい】** をクリックします。

パスコンポーネントの設定

ここでは、スキーママネージャでパスコンポーネントを設定する方法について説明します。パスコンポーネントは、一致カラムでそのテーブルのカラムを使用するために、外部キーを使用して親テーブルと子テーブルの間の接続を定義する手段です。

表示名

このパスコンポーネントが Hub コンソールに表示されるときの名前。

物理名

データベース内のパスコンポーネントの実際の名前。Informatica MDM Hub では、パスコンポーネントの物理名の候補として、入力した表示名に基づく名前が示されます。

子レコードの欠如を許可する

【子レコードの欠如を許可する】 オプションは、一致パスの子ベースオブジェクトにレコードが存在するかのチェックに基づいた照合のときに、親レコードを考慮する必要があるかを示します。

オプションを有効すると、一致パスコンポーネントのレベルでの子レコードの欠如を許可することができます。一致パスコンポーネントのレベルは、メインの親ベースオブジェクトの一致パスにある多くの子ベースオブジェクトレベルで構成されている可能性があります。

このオプションを有効にして一致パスコンポーネント上の子レコードの欠如を許可する場合、親のベースオブジェクトレコードとそれに関連する子ベースオブジェクトレコードの間に一致が起きます。このオプションを有効にした子ベースオブジェクトの中に親のベースオブジェクトレコードが子レコードを持たない場合でも、この一致は起きます。デフォルトで、子レコードの欠如を許可するこのオプションが、親ベースオブジェクトの一致パスの子ベースオブジェクトすべてに対して有効になります。このオプションが有効になっているとき、これは両立的であるため、親とオプションを有効にした子ベーステーブルとの間でデータベースの外部結合を実行します。Informatica Data Director の基本検索は、子レコードを持たないレコードの結果を返します。

すべての一致パスコンポーネントで子レコードの欠如を許可するこのオプションを無効にすると、すべての子ベースオブジェクトの中にレコードを持つ親ベースオブジェクトレコードで照合が行われます。親のレコードが、このオプションを無効にした子ベースオブジェクトの中に子レコードを持たない場合でも、その親レコードには照合が行われません。このオプションが無効になっているとき、それは排他的であるため、正規のデータベース等価結合と同様です（親レコードに子レコードがない場合は親レコードも子コードも返さない）。このオプションを無効にすると、外部結合に関連するパフォーマンスへの影響が避けられます。Informatica Data Director の基本検索では、子レコードを持たないレコードについて結果を返しませんが、

ベースオブジェクトに対しあいまい一致を実行する必要がある場合は、親ベースオブジェクトレコードをトークン化する必要があります。子レコードの欠如を許可するオプションを無効にした子ベースオブジェクトのすべてが関連の子ベースオブジェクトレコードを持つ場合、親ベースオブジェクトレコードはトークン化されます。親ベースオブジェクトレコードに子ベースオブジェクトがあり、子ベースオブジェクトに子レコードの欠如を許可するオプションが無効に設定され、レコードが含まれていない場合、親レコードはトークン化されません。

データが完全であれば、期待通りに親同士の一一致が起こり、MDM Hub が一致判定基準に従って照合処理にすべての親レコードを含めます。親レコードが親の一致パス内の子ベースオブジェクトそれぞれに子レコードを持っていて、親に対する一致ルールに子カラムが含まれている場合、そのデータは完全です。しかし、データが不完全である場合、親レコードが一致の対象になれるレコードを持たない子ベースオブジェクトのパスコンポーネントに対し、子の有無をチェックするオプションを有効にしてください。データの中に、親レコードで子ベースオブジェクトからのレコードが欠けているものがあり、さらに他の親レコードで別の子ベースオブジェクトからのレコードが欠けているものがあれば、そのデータは不完全です。また、親の一致ルールに子カラムを含めていない場合、そのデータは不完全です。これにより、親レコードとそれに関連する子ベースオブジェクトの（レコードのカラムを一致パスのカラムが基準にしている）レコードが、一致から除外されないようになります（親がそのような子ベースオブジェクトの中にレコードを持たない場合）。

注: Informatica MDM Hub は、子レコードの欠如を許可するオプションが有効になっているときに、親テーブルと子テーブルの間で外部結合を実行します。これは、オプションを有効にした各一致パスコンポーネントでパフォーマンスに影響します。そのため、必要がない場合は、このオプションは無効にした方が効率的です。

制約

プロパティ	説明
テーブル	スキーマのテーブルのリスト。
方向	外部キーの方向。 親から子 子から親 該当なし
外部キーの参照先	外部キーで参照するカラム。このカラムは、別のベースオブジェクトのものであっても同じベースオブジェクトのものであってもかまいません。

パスコンポーネントの追加

パスコンポーネントを追加する手順

1. スキーママネージャで、**[パス]** タブに移動します。
2. 書き込みロックを取得します。
3. **[パスコンポーネント]** セクションで、**[追加]** ボタンをクリックします。
[パスコンポーネントの追加] ダイアログが表示されます。
4. このパスコンポーネントのプロパティを指定します。
5. **[OK]** をクリックします。
6. **[保存]** ボタンをクリックして変更を保存します。

パスコンポーネントの編集

パスコンポーネントを編集する手順

1. スキーママネージャで、**[パス]** タブに移動します。
2. 書き込みロックを取得します。
3. パスコンポーネントツリーで、削除するパスコンポーネントを選択します。
4. **[パスコンポーネント]** セクションで、**[編集]** ボタンをクリックします。
[パスコンポーネントの編集] ダイアログが表示されます。
5. このパスコンポーネントのプロパティを指定します。以下の値を変更できます。
 - 表示名
 - 子レコードの欠如を許可する
6. **[OK]** をクリックします。
7. **[保存]** ボタンをクリックして変更を保存します。

パスコンポーネントの削除

パスコンポーネントは削除できますが、ルートベースオブジェクトは削除できません。パスコンポーネントを削除する手順

1. スキーママネージャで、**[パス]** タブに移動します。
2. 書き込みロックを取得します。

3. パスコンポーネントツリーで、削除するパスコンポーネントを選択します。
4. [パスコンポーネント] セクションで、**[削除]** ボタンをクリックします。
削除を確認するように求められます。
5. **[はい]** をクリックします。
6. **[保存]** ボタンをクリックして変更を保存します。

一致カラムの設定

ここでは、一致カラムルールで使用できるように一致カラムを設定する方法について説明します。プライマリキーの一致ルールを設定する場合は、[「プライマリキーの一致ルールの設定」 \(ページ 435\)](#)の手順を参照してください。

一致カラムについて

一致カラムとは、一致ルールで使用するカラム（名前カラムや住所カラムなど）です。ルールの定義でカラムを使用する前に、そのカラムを一致ルールで使用するカラムとして指定し、含まれているデータに関する情報を指定する必要があります。

注: マッチルールを定義するためにカラムを選択するときに、数値タイプまたは 10 進データタイプのカラムはマッチカラムとして使用できません。

一致カラムのタイプ

照合ルールには、あいまいタイプまたは完全タイプのカラムを設定できます。

以下の表に、照合ルールのカラムタイプを示します。

カラムのタイプ	説明
あいまい	<i>確率的な一致。</i> スペル、略称、語順、完全性、信頼性、およびその他の不整合性の点で変動するデータを含むカラムに使用します。例えば、あいまいカラムには、番地、地理的座標（緯度、経度、標高など）、人または組織の名前などがあります。
完全	<i>確認的な一致。</i> 一貫性のある予想可能なパターンが含まれるカラムに使用します。完全一致カラムは、データが同一である場合のみ一致します。例えば、ID、郵便番号、業種コード、その他の明確に定義された情報などがあります。

検索ストラテジに依存する一致カラム

設定できる一致カラムのタイプは、設定中のベースオブジェクトのタイプによって異なります。

ベースオブジェクトのタイプは、選択した一致/検索ストラテジで定義されています。

一致ストラテジ	説明
あいまい一致ベースオブジェクト	あいまい一致カラムおよび完全一致カラムを設定することができます。
完全一致ベースオブジェクト	完全一致カラムを設定できますが、あいまい一致カラムは設定できません。

パスコンポーネント

パスコンポーネントは、一致カラム定義に使用するソーステーブル、またはレコードの階層を移動するための一致パスです。一致パスは、別々のテーブルまたは同じテーブルにある関連レコードが関係する一致カラムルールを設定するために使用されます。パスコンポーネントを指定する前に、一致パスを設定する必要があります。

一致パスのパスコンポーネントを設定する手順

1. [パスコンポーネント] フィールドの横にある **【編集】** ボタンをクリックします。
[一致パスコンポーネントの選択] ダイアログが表示されます。
2. 一致パスコンポーネントを選択します。
3. **【OK】** をクリックします。

フィールド名

あいまい一致カラムを照合ルールに追加するときには、リストからフィールド名を選択できます。

以下の表は、照合ルールのあいまい一致カラムを追加するときを選択できるフィールド名を示しています。

フィールド名	説明
Address_Part1	地域を記述する最後の行の直前までの部分的な住所を含んでいます。住所の構成要素の位置は、データビジュレーションと同じく、標準の語順であることが必要です。このデータは1つのフィールドで渡します。ベースオブジェクトによっては、マッチングの前にこれらの属性を1つのフィールドに連結できます。例えば米国では、Address_Part1 の文字列に次のフィールドが含まれています。気付+建物名+通りの番号+通りの名前+通りの種類+アパートの詳細。Address_Part1 では、住所専用に設計されたメソッドとオプションが使用されます。 一致過多を防ぐために、照合プロセスではフィールドに10文字以上含まれる場合にのみ転置エラーを考慮します。例えば、「PO Box 38」という照合条件では「PO Box 83」は一致語句とみなされませんが、「13 Capital Street」という照合条件では「31 Capital Street」は一致語句とみなされます。
Address_Part2	住所内の地域を記述する行。例えば米国では、標準的な Address_Part2 に以下の要素が含まれています。都市+州+郵便番号 (+国)。Address_Part2 での一致には、住所専用に設計されたメソッドとオプションが使用されます。
Attribute1、Attribute2	2つの汎用フィールド。MDM Hub では、属性フィールドの照合は、入れ替えや欠落が発生している文字や数字を補う汎用の文字列一致アルゴリズムを使用して行われます。

フィールド名	説明
Date	あらゆるタイプの日付（誕生日、有効期限、契約日、変更日、作成日など）を照合します。日付を日+月+年の形式で渡します。SSA_Date フィールドの名称には、日付の構成要素間の区切り文字を使用しても省略してもかまいません。日付での照合では、日付専用設計されたメソッドとオプションが使用されます。日付での照合では、このデータ型での典型的なエラーや差異が解消されます。
Geocode	<p>地理的な座標（緯度、経度、および標高）を照合します。地理的な座標を次の順に指定します。</p> <ol style="list-style-type: none"> 1. 緯度 2. 経度 3. 標高 <p>このデータは、1つのフィールドまたは複数のフィールドで渡すことができます。1つのフィールドでジオコードデータを連結する場合は、カンマまたはスペースで値を区切ります。</p> <p>ジオコードでは、入れ替えや欠落が発生している文字や数字を補う文字列一致アルゴリズムが使用されます。</p>
ID	あらゆるタイプの ID（口座番号、顧客番号、クレジットカード番号、運転免許証番号、パスポート番号、ポリシー番号、SSN などの ID コード、VIN など）を照合します。[ID] フィールドでは、入れ替えや欠落が発生している文字や数字を補う文字列一致アルゴリズムが使用されます。
Organization_Name	組織の名称（組織名、事業所名、団体名、部署名、機関名、通称など）を照合します。このフィールドでは、単一の名前または複合名（正式名と通称など）での照合をサポートします。照合の条件として、1つの Organization_Name カラムで複数の名前（正式名と通称など）を使用することもできます。
Person_Name	人物の名前と一致します。個人のフルネームを使用してください。名前（ファーストネーム）、ミドルネーム、および姓（ファミリーネーム）の位置は、ポピュレーションにおける標準の語順であることが必要です。例えば、英語圏での標準の語順は、ファーストネーム+ミドルネーム+ファミリーネームです。ベースオブジェクトの設計によっては、照合処理の前にこれらのフィールドを連結して1つのフィールドにすることができます。このフィールドは、単一の名前、つまりアカウント名（JOHN & MARY SMITH など）での照合をサポートします。複数の名前を使用することもできます（結婚後の姓と旧姓など）。
Postal_Area	Address_Part2 フィールドを使用しないことで郵便番号を重視するために使用します。ZIP コードを含むあらゆるタイプの郵便番号に使用できます。Postal_Area フィールド名では、入れ替えや欠落が発生している文字や数字を補う文字列一致アルゴリズムが使用されます。
Telephone_Number	電話番号を照合するために使用します。Telephone_Number フィールド名では、入れ替えや欠落が発生している数字または地域コードを補う文字列一致アルゴリズムが使用されます。

一致させる複数のカラムの選択

複数のカラムを一致用に指定する場合:

- 各値が一致用のフィールドに連結されます。その際、各値の間にスペースが挿入されます。例えば、ベースオブジェクトの名のカラム、ミドルネームのカラム、姓のカラム、および末尾の敬称のカラムを選択します。この場合、連結されたフィールドは以下のようになります（文字列の最後の語の後にスペースが続く）。

first middle last suffix

以下に例を示します。

Anna Maria Gonzales MD

- スペースまたは NULL データを含むデータの場合:
 - データにスペースがある場合、そのスペースはそのまま残ります。フィールドは NULL になりません。
 - すべてのフィールドが NULL の場合、連結された値は NULL になります。
 - 連結されたフィールドのいずれかの部分が NULL の場合、NULL に置き換わる余分なスペースは追加されません。

注: 完全一致カラムについては、カラムを連結しないことをお勧めします。

あいまい一致ベースオブジェクトの一致カラムの設定

あいまい一致ベースオブジェクトには、あいまい一致カラムと完全一致カラムの両方を含めることができます。完全一致ベースオブジェクトを使用する場合は、[「完全一致ベースオブジェクトの一致カラムの設定」 \(ページ 407\)](#)を参照してください。

あいまい一致ベースオブジェクトの [一致カラム] タブへの移動

あいまい一致ベースオブジェクトの一致カラムを定義する手順

1. スキーママネージャで、設定するあいまい一致ベースオブジェクトを選択します。
2. **[一致/マージ設定]** ノードをクリックします。
3. **[一致カラム]** タブをクリックします。

あいまい一致ベースオブジェクトに関する [一致カラム] タブが表示されます。

あいまい一致ベースオブジェクトの [一致カラム] タブには以下のセクションがあります。

プロパティ	説明
あいまい一致キー	あいまい一致キーのプロパティ。
一致カラム	一致カラムとそのプロパティ: <ul style="list-style-type: none">- フィールド名- カラムのタイプ- パスコンポーネント- ソーステーブルパスコンポーネントまたは（パスコンポーネントがルートの場合）ベースオブジェクトで参照されるテーブル
一致カラムのコンテンツ	ベースオブジェクトの利用可能なカラムのリスト。また、一致用に選択したカラム也表示されます。

あいまい一致キーのプロパティの設定

ここでは、あいまい一致ベースオブジェクトの一致カラムのプロパティを設定する方法について説明します。
あいまい一致キーは、ベースオブジェクトの特殊なカラムであり、一致カラムであいまい一致/検索ストラテジを使用する場合にスキーママネージャによって追加されます。このカラムは、検索や一致の実行時にこのベースオブジェクトの一致候補を生成するために使用されるプライマリフィールドです。どのあいまい一致ベースオブジェクトにも、あいまい一致キーが必ず 1 つだけ含まれます。

キーのタイプ

一致キータイプは、カラムの重要な特性を Informatica MDM Hub に伝えます。Informatica MDM Hub には名前と住所に関する情報がある程度存在するため、この情報は Informatica MDM Hub が正しくキーを生成して検索効率を高めるのに役立ちます。これは、一致候補の初期リストを作成する検索の主要な基準となります。このキータイプは、あいまい一致キーを構成する物理カラム内の主要なデータタイプに基づいている必要があります。

あいまい一致ベースオブジェクトの場合は、次のいずれかのキータイプを選択できます。

キータイプ	説明
Person_Name	あいまい一致キーに個人のデータのみが含まれている場合に使用されます。
Organization_Name	あいまい一致キーに組織のデータのみが含まれている場合、または組織と個人の両方のデータが含まれている場合に使用されます。
Address_Part1	あいまい一致キーに、連結される住所データが含まれている場合に使用されます。

注: キータイプは、選択するポピュレーションに基づきます。上記の一連のキータイプは、デフォルトのポピュレーション (US) に適用されます。それ以外のポピュレーションでは、キータイプが異なる可能性があります。別のポピュレーションが必要である場合は、Informatica サポートにお問い合わせください。

キー幅

一致キー幅によって、あいまい一致キーの分析の完全性、予想される一致候補の数、およびキーが消費するディスク容量が決まります。キー幅はあいまい一致オブジェクトにのみ適用されます。

キー幅	説明
標準	信頼性とスペース使用量のバランスがとれているため、ほとんどのあいまい一致キーに適しています。
拡張	一致候補の数が増える代わりに、キーを生成するための処理時間が長くなる可能性があります。このオプションでは、カラムの連結により一致機能がある程度強化されます。このキー幅は、以下の場合に最も適しています。 <ul style="list-style-type: none">- データセットが極端に大きくない。- データセットが完全ではない。- 処理時間とディスク容量の要件に対応できるだけの十分なリソースがある。
限定	一致の信頼性をある程度犠牲にしてディスク容量を節約します。このオプションでは、一致候補が少なくなる可能性があります。検索は速くなります。このオプションは、検索を高速化してキーに使用されるディスク容量を減らすために一致不足を容認する場合に適しています。標準キーに比べて、限定キーでは語順に違いがあるレコードの一致件数が少なくなります。これを選択すると標準キーセットのサブセットが提供されますが、ディスク容量が限られている場合や、データボリュームが極端に大きい場合は最適なオプションになり得ます。
優先	ベースオブジェクトレコードごとにキーが 1 つずつ生成されます。このオプションは、一致の信頼性をある程度犠牲にしてパフォーマンスを高め（実行される一致の回数を削減）、ディスク容量を節約します（一致キーテーブルのサイズを縮小）。データの特性によっては、優先キー幅を使用すると一致候補の数がより少なくなる場合があります。

あいまい一致キーのプロパティの設定手順

あいまい一致ベースオブジェクトにあいまい一致キーのプロパティを設定する手順

1. スキーママネージャで、[一致カラム] タブに移動します。
2. 書き込みロックを取得します。
3. あいまい一致ベースオブジェクトに以下の設定を行います。

プロパティ	説明
Key Type	主に一致に使用されるフィールドのタイプ。これは、一致候補の初期リストを作成する検索の主要な基準となります。このキータイプは、ベースオブジェクトに格納されたデータの主なタイプに基づいて設定されます。
キー幅	キーが生成される検索範囲のサイズ。
パスコンポーネント	このあいまい一致キーに対するパスコンポーネント。キータイプとして指定するカラムを含むテーブルです（ベースオブジェクト、子ベースオブジェクトテーブル、または相互参照テーブル）。

4. **【保存】** ボタンをクリックして変更を保存します。

あいまい一致ベースオブジェクトのあいまい一致カラムの追加

あいまい一致ベースオブジェクトのあいまい一致カラムを定義する手順

1. スキーママネージャで、[一致カラム] タブに移動します。
2. 書き込みロックを取得します。
3. あいまい一致カラムを追加するには、**【追加】** ボタンをクリックします。
[あいまい一致カラムの追加] ダイアログが表示されます。
4. 以下の設定を指定します。

プロパティ	説明
一致パスコンポーネント	このあいまい一致カラムの一致パスコンポーネント。あいまい一致カラムの場合、ソーステーブルは、親テーブル、親相互参照テーブル、または任意の子ベースオブジェクトテーブルのいずれかにすることができます。
フィールド名	フィールドの名前。一致カラムのデータタイプを選択します。

5. あいまい一致のベースオブジェクトのカラムを指定します。
【選択したカラム】 リストにカラムを追加するには、カラム名を選択し、右矢印ボタンをクリックします。
注: 複数のカラムを追加した場合、スペース区切り文字を使用して値が連結されます。
6. **【OK】** をクリックします。
[一致カラム] リストに一致カラムが追加されます。
7. **【保存】** ボタンをクリックして変更を保存します。

あいまい一致ベースオブジェクトの完全一致カラムの追加

あいまい一致ベースオブジェクトの完全一致カラムを定義する手順

1. スキーママネージャで、[一致カラム] タブに移動します。

- 書き込みロックを取得します。
- 完全一致カラムを追加するには、**【追加】** ボタンをクリックします。
[完全一致カラムの追加] ダイアログが表示されます。
- 以下の設定を指定します。

プロパティ	説明
一致パスコンポーネント	この完全一致カラムのマッチパスコンポーネント。完全一致カラムの場合、ソーステーブルは、親テーブルまたは子物理カラム、あるいはその両方にすることができます。
フィールド名	Hub コンソールで表示するフィールドの名前。

- 完全一致のベースオブジェクトカラムを指定します。
- 【選択したカラム】** リストにカラムを追加するには、カラム名を選択し、右矢印ボタンをクリックします。
注: 複数のカラムを追加した場合、スペース区切り文字を使用して値が連結されます。完全一致カラムのカラムは連結しないでください。
- 【OK】** をクリックします。
[一致カラム] リストに一致カラムが追加されます。
- 【保存】** ボタンをクリックして変更を保存します。

あいまい一致ベースオブジェクトの一致カラムのプロパティの編集

一致カラムのプロパティを編集する代わりに、以下を実行する必要があります。

- 一致カラムの削除
- 新しい一致カラムの追加

あいまい一致ベースオブジェクトの一致カラムの削除

あいまい一致ベースオブジェクトの一致カラムを削除する手順

- スキーママネージャで、**【一致カラム】** タブに移動します。
- 書き込みロックを取得します。
- [一致カラム] リストで、削除する一致カラムを選択します。
- 【削除】** ボタンをクリックします。
削除を確認するように求められます。
- 【はい】** をクリックします。
- 【保存】** ボタンをクリックして変更を保存します。

完全一致ベースオブジェクトの一致カラムの設定

一致カラムルールを定義する前に、基になる一致カラムを定義する必要があります。完全一致ベースオブジェクトには、完全一致カラムだけを含めることができます。あいまい一致ベースオブジェクトの一致カラムの設定の詳細については、[「あいまい一致ベースオブジェクトの一致カラムの設定」 \(ページ 404\)](#)を参照してください。

完全一致ベースオブジェクトの「一致カラム」タブへの移動

完全一致ベースオブジェクトの一致カラムを定義する手順

- スキーママネージャで、設定する完全一致ベースオブジェクトの「一致/マージ設定の詳細」ダイアログを表示します。
- 「一致カラム」タブをクリックします。
「一致カラム」タブは完全一致ベースオブジェクトに対して表示されます。
完全一致ベースオブジェクトの「一致カラム」タブには以下のセクションがあります。

プロパティ	説明
一致カラム	一致カラムとそのプロパティ: <ul style="list-style-type: none">- フィールド名- カラムのタイプ- パスコンポーネント- ソーステーブルパスコンポーネントまたは（パスコンポーネントがルートの場合）ベースオブジェクトで参照されるテーブル
一致カラムのコンテンツ	利用可能なカラムと一致用に選択したカラムのリスト。

完全一致ベースオブジェクトの一致カラムの追加

完全一致ベースオブジェクトには、完全一致カラムだけを追加できます。あいまい一致カラムは使用できません。

完全一致ベースオブジェクトの完全一致カラムを追加する手順

- スキーママネージャで、「一致カラム」タブに移動します。
- 書き込みロックを取得します。
- 完全一致カラムを追加するには、「追加」ボタンをクリックします。
「完全一致カラムの追加」ダイアログが表示されます。
- 以下の設定を指定します。

プロパティ	説明
一致パスコンポーネント	この完全一致カラムの一致パスコンポーネント。完全一致カラムの場合、ソーステーブルは、親テーブルまたは子物理カラム、あるいはその両方にすることができます。
フィールド名	Hub コンソールに表示されるこのフィールドの名前。

- 完全一致のベースオブジェクトのカラムを指定します。
- 「選択したカラム」リストにカラムを追加するには、カラム名を選択し、右矢印をクリックします。
注:
 - 複数のカラムを追加した場合、スペース区切り文字を使用して値が連結されます。
 - 完全一致カラムについては、カラムを連結しないことをお勧めします。
- 「OK」をクリックします。
選択した一致カラムが「一致カラム」リストに追加されます。
- 「保存」ボタンをクリックして変更を保存します。

完全一致ベースオブジェクトの一致カラムのプロパティの編集

完全一致カラムにルートパスではないパスが含まれる場合は、一致カラムを【**デフォルトの一致サブタイプ**】に設定できます。【一致ルールセット】で、この完全一致カラムを含むあいまい一致ルールを追加すると、【**一致サブタイプ**】オプションが選択されます。

注: 他のプロパティを編集する場合は、一致カラムを削除して保存し、プロパティが更新された一致カラムを追加します。

1. 【**一致カラム**】タブで、次の属性を持つ一致カラムを選択します。
 - **カラムタイプ** = 完全一致
 - **パスコンポーネント** = <ルート以外のパス>
2. 【**編集**】をクリックします。
3. 【**デフォルトの一致サブタイプ**】を選択し、【**OK**】をクリックします。
4. 【**保存**】をクリックします。

関連項目：

- [「一致サブタイプ」 \(ページ 424\)](#)

完全一致ベースオブジェクトの一致カラムの削除

完全一致ベースオブジェクトの一致カラムを削除する手順

1. スキーママネージャで、【**一致カラム**】タブに移動します。
2. 書き込みロックを取得します。
3. 【一致カラム】リストで、削除する一致カラムを選択します。
4. 【**削除**】ボタンをクリックします。

削除を確認するように求められます。
5. 【**はい**】をクリックします。
6. 【**保存**】ボタンをクリックして変更を保存します。

一致ルールセット

一致ルールセットは、プロパティの一部が共通している一致カラムルールの論理的な集まりです。

一致ルールセットは一致カラムルールにのみ関連付けられ、プライマリキーの一致ルールには関連付けられません。

一致ルールセットを使用すると、状況によってさまざまな一致カラムルールのセットを実行することができます。一致プロセスが実行されるたびに、1つの一致ルールセットのみが使用されます。別の一致ルールセットを使用して一致させるには、その一致ルールセットを選択して再度一致プロセスを実行します。

設定した一致ルールセットは、Data Director で拡張クエリを作成するのに使用されます。一致ルールセットを変更または削除した場合、その一致ルールセットを使用するクエリは削除されます。

注: 一致ルールセットの一致カラムルールを 1 つでも満たせば、レコード間の一致が宣言されます。

一致ルールセットで指定する内容

一致ルールセットには以下のものが含まれます。

- 検索ストラテジを決定する検索レベル
- 任意の数の自動および手動一致カラムルール

- 必要に応じて、一致プロセス中にレコードを選択的に一致バッチに含めたり一致バッチから除外したりすることができるフィルタ

複数の一致ルールセットとデフォルトの指定

設定できるルールセットの数に制限はありません。

一致バッチジョブを実行する場合、ユーザーは、ベースオブジェクトに定義されているルールセットのリストから、ルールセットを1つ選択します。

管理者はスキーママネージャで、1つの一致ルールセットをデフォルトとして指定できます。

一致ルールセットを使用する状況

一致ルールセットを使用すると、異なる一致カラムルールの要件に対して異なるタイミングで対応できます。

例えば、ある一致ルールセットを初期データロードに対して使用し、別の一致ルールセットを後続の増分ロードに対して使用できます。同様に、ある一致ルールセットを使用してすべてのレコードを処理し、別の一致ルールセットとフィルタを併用して一部のレコードのみを処理することもできます。

ルールセットの評価

(一致ルールセットの一致ルールへの変更を含む) 一致ルールセットへの変更を保存する前に、スキーママネージャは一致ルールセットを分析し、一致ルールセットに問題がある場合は警告メッセージを表示します。

注: これは警告メッセージにすぎません。メッセージを無視して変更を保存することができます。

例には、以下の内容の一致ルールセットが含まれます。

- 既存の一致ルールセットと同一である
- 空である (一致カラムルールが追加されていない)
- あいまい一致ベースオブジェクトに対するあいまい一致カラムを含む
- 1つ以上のあいまい一致カラムを含むが、完全一致カラムを含まない (一致のパフォーマンスに影響する可能性がある)
- 同じソースカラムを持つあいまい一致カラムおよび完全一致カラムを含む

一致ルールセットのプロパティ

この節では、一致ルールセットのプロパティについて説明します。

名前

ルールセットの名前。一意のわかりやすい名前を指定します。

検索レベル

あいまい一致ベースオブジェクトでのみ使用されます。一致ルールセットを設定する場合、**検索レベル**を定義して、どの程度厳密かつ徹底的に候補の一致を検索するかを Informatica MDM Hub に指示します。

一致プロセスの目標は、データの一致を適切な数だけ見つけることです。

- 関連する一致が少なすぎてもいけません (一致不足)
- 無関係の一致を含み、生成される一致が多すぎてもいけません (一致過多)

あいまい一致キーの名前またはアドレスに対して、Informatica MDM Hub では、一致候補の可能性があるレコードを決定するため、また、一致カラムルールを適用するレコードを決定するために、定義済みの検索レベルを使用してさまざまなキー範囲を生成します。

以下の検索レベルのいずれか 1 つを選択できます。

検索レベル	説明
低	最も厳密な一致候補の検索レベルです。この検索レベルは高速に処理されますが、他の検索レベルよりも一致結果が少数になり、一致不足となる可能性があります。低レベルは、データセットが比較的正確かつ完全な場合、または一致データが多い大規模なデータセットに対する場合に適切です。
標準	ほとんどの一致ルールセットに適しています。
高	一致候補を通常レベルよりも多く生成します。他のレベルよりも多くの一致が生成されるため、一致過多となる可能性があります。処理に時間もかかります。このレベルは、小規模の不完全なデータセットに適切です。
最高	高レベルよりも多くの一致候補を生成します。一致過多となる可能性がより高く、処理に要する時間もさらに長くなります。このレベルは、小規模の不完全なデータセットに対して、または一致レコード候補数を最大限必要とする場合に適切です。

選択すべき検索レベルは、データセットのサイズ、時間的制約、一致の重要度によって決定します。環境および要件によっては、一致不足がより適切な場合もあれば、一致過多がより適切な場合もあります。比較的信頼性が高く完全なデータを処理する実装では低レベルに設定し、より信頼性が低いデータや重要な問題に対しては、高レベルや最高レベルを使用します。

適切な検索レベルは、プロジェクトのフェーズによっても変化します。初期の一致に対しては、制約の緩いレベル（最高または高）を使用し、データの重複が増えてきたら制約を厳しくしていきます。

ルールによる検索を有効にする

[ルールによる検索を有効にする] 一致ルールセットを有効にする場合、この特定の一致ルールセットを **SearchMatch** API で使用するために予約します。[ルールによる検索を有効にする] 一致ルールセットを使用するには、**SearchMatch** MatchType が NONE になっていることと、**SearchMatch** 要求で一致ルールセットを指定する必要があります。[ルールによる検索を有効にする] を使用すると、MDM Hub はあいまい一致だけを使用します。

複数の空フィールドがあっても検索を実行することができます。[ルールによる検索を有効にする] 一致ルールセットを使用して検索を実行すると、SearchMatch は検索要求で提供された空フィールドを無視します。これにより検索結果から関連レコードが除外されるのを防止します。

SearchMatch の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

注: オプションの Hub サーバープロパティを有効にしてあいまい一致ベースオブジェクトで完全一致を実行すると、一致ルールに影響を及ぼします。完全一致とあいまい一致の両方に同じソースカラムを使用すると、MDM Hub が一致を返しません。この問題を回避するには、cmxserver.properties ファイルで cmx.server.match.exact_match_fuzzy_bo_api プロパティを無効にする必要があります。Hub Server プロパティの詳細については、[「Hub サーバーのプロパティ」](#) (ページ 621)を参照してください。

フィルタリングを有効にする

この一致ルールセットに対してフィルタリングを有効にするかどうかを指定します。

- オンに（選択）すると、この一致ルールセットにフィルタを定義できるようになります。ユーザーは、一致ジョブ実行時にフィルタが定義された一致ルールセットを選択して、フィルタ条件を満たすレコードのサブセットのみが一致ジョブで処理されるようにすることができます。
- オフにする（選択しない）と、一致バッチジョブ実行時にすべてのレコードが一致ルールセットによって処理されます。

例えば、複数タイプの組織（顧客、ベンダ、見込み客、パートナなど）を含んでいる組織ベースオブジェクトがある場合は、一致させたいタイプのレコードのみを選択的に処理する各種の一致ルールセット（MatchAll（フィルタなし）、MatchCustomersOnly、MatchVendorsOnly など）を定義できます。

注: 一致ルールセットでフィルタを有効にすることはできますが、子ベースオブジェクトおよびグループ関数ではフィルタは許可されません。

フィルタリング SQL

デフォルトでは、一致バッチジョブが実行されると、一致ルールセットによってすべてのレコードが処理されます。

〔フィルタリングを有効にする〕チェックボックスが選択されている（オンになっている）場合は、フィルタ条件を指定して、その条件を満たすルールのみを処理するように制限できます。フィルタは、SQL 文の WHERE 句に似ています。フィルタ式には、使用するデータベースプラットフォームでの WHERE 句の構文で有効な任意の式を使用できます。

注: 一致ルールセットフィルタは、一致バッチの対象として選択されたベースオブジェクトレコード（一致元レコード）にのみ適用され、一致プールのレコード（一致先レコード）には適用されません。

例えば、実装で使用している Organization ベースオブジェクトに、複数のタイプの組織（顧客、ベンダ、見込み客、パートナなど）が含まれているとします。フィルタを使用して、顧客データのみを処理する一致ルールセット（MatchCustomersOnly）を定義できます。

```
org_type='C'
```

顧客レコード以外のレコードはすべて一致ジョブで無視され、処理されません。

注: 一致ジョブでレコードを正しくフィルタリングする適切な SQL 式を指定するのは、管理者の役割です。スキーママネージャでは、使用するデータベースプラットフォームに従って SQL 構文が検証されますが、フィルタ条件のロジックまたは妥当性は検査されません。

一致ルール

ウィンドウのこの領域には、選択した一致ルールセット用に設定された一致カラムルールのリストが表示されます。

〔一致ルールセット〕 タブへの移動

〔一致ルールセット〕 タブに移動する手順

1. スキーママネージャで、設定するベースオブジェクトの〔一致/マージ設定の詳細〕ダイアログを表示します。
2. 〔一致ルールセット〕タブをクリックします。
選択したベースオブジェクトの〔一致ルールセット〕タブが表示されます。

3. [一致ルールセット] タブは、以下のセクションで構成されています。

セクション	説明
一致ルールセット	設定済みの一致ルールセットの一覧。
プロパティ	選択した一致ルールセットのプロパティ。

一致ルールセットの追加

新しい一致ルールセットを追加する手順

- スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログの [一致ルールセット] タブを表示します。
- 書き込みロックを取得します。
- [追加] ボタンをクリックします。
[一致ルールセットの追加] ダイアログが表示されます。
- この新しい一致ルールセットの一意のわかりやすい名前を入力します。
- [OK] をクリックします。
リストに新しい一致ルールセットが追加されます。
- 一致ルールセットを設定します。

一致ルールセットのプロパティの編集

一致ルールセットのプロパティを編集する手順

- スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログの [一致ルールセット] タブを表示します。
- 書き込みロックを取得します。
- 設定する一致ルールセットを選択します。
スキーママネージャのプロパティパネルにそのプロパティが表示されます。
- この一致ルールセットのプロパティを設定します。
- この一致ルールセットの一致カラムを設定します。
- [保存] ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
- 変更の保存を確認するように求められた場合は、[OK] ボタンをクリックして変更を保存します。

一致ルールセットの名前の変更

一致ルールセットの名前を変更する手順

- スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログの [一致ルールセット] タブを表示します。
- 書き込みロックを取得します。
- 名前を変更する一致ルールセットを選択します。
- [編集] ボタンをクリックします。

「ルールセット名の編集」ダイアログが表示されます。

5. この一致ルールセットの一意のわかりやすい名前を指定します。
6. **[OK]** をクリックします。
リスト内の一致ルールセットの名前が更新されます。

一致ルールセットの削除

一致ルールセットを削除する手順

1. スキーママネージャで、設定するベースオブジェクトの「一致/マージ設定の詳細」ダイアログの「**一致ルールセット**」タブを表示します。
2. 書き込みロックを取得します。
3. 削除する一致ルールセットの名前を選択します。
4. **[削除]** ボタンをクリックします。
削除を確認するように求められます。
5. **[はい]** をクリックします。

スキーママネージャによって、削除した一致ルールセットがそのセットに含まれるすべての一致カラムルールとともにリストから削除されます。

一致ルールセットの一致カラムルールの設定

一致カラムルールにより、一致プロセスで一致と見なす条件が決まります。

一致カラムルールでは、2つのレコードが統合可能な類似するレコードであるかどうかを特定します。各一致ルールは、類似点の有無を調べる必要がある1つ以上の一致カラムのセットとして定義されます。一致ルールを設定するには、ソースシステム内およびソースシステム間で一致するレコードを特定する条件を設定します。

一致カラムルールを設定するための前提条件

一致カラムルールを設定するためには、まず次の作業を完了する必要があります。

- 一致ルールで使用するカラムの設定
- 少なくとも1つの一致ルールセットの作成

完全一致ベースオブジェクトとあいまい一致ベースオブジェクトの一致カラムルールの違い

一致カラムルールのプロパティは、完全一致ベースオブジェクトとあいまい一致ベースオブジェクトで異なります。

- 完全一致ベースオブジェクトの場合は、正確なカラムタイプのみを設定できます。
- あいまい一致ベースオブジェクトの場合は、あいまいタイプまたは完全タイプのカラムを設定できます。

一致カラムルールに対して、一致レコードを自動または手動で統合するかどうかを指定します。

一致レコードの統合オプションの指定

一致カラムルールに対して、一致レコードを自動または手動で統合するかどうかを指定します。

あいまい一致ベースオブジェクトのみに適用される一致ルールのプロパティ

この節では、あいまい一致ベースオブジェクトの一致ルールプロパティについて説明します。これらのプロパティは、完全一致ベースオブジェクトには適用されません。

一致/検索ストラテジ

あいまい一致のベースオブジェクトの場合、一致/検索ストラテジでは、一致プロセスがレコードの検索と一致に使用する戦略を定義します。一致/検索ストラテジによって、候補 A と候補 B をあいまい一致または完全一致オプションを使用して一致させる方法が決まります。

注: あいまい一致のベースオブジェクトの詳細については、「[一致/検索ストラテジ](#)」 (ページ 389) を参照してください。

一致/検索ストラテジは、一致候補の量と質に影響する可能性があります。完全一致ストラテジにはクリーンで完全なデータが必要です。データがクレンジングされていない場合、またはデータが不完全な場合は、一致プロセスでいくつかの一致が見逃されてしまう可能性があります。あいまい一致ストラテジでは多くの一致が見つかりますが、その多くは重複していません。一致ルールプロパティを定義する場合は、利用可能なすべての候補を検索しながら、関連性のない候補を回避する最適なバランスを見つける必要があります。

次のいずれかのストラテジオプションを選択します。

一致/検索ストラテジオプション	説明
あいまい一致ストラテジ	スペルの変化、考えられるスペルミス、その他の違いを考慮した確率的な一致です。
完全一致ストラテジ	同一のレコードと一致する完全一致です。

すべてのあいまい一致ベースオブジェクトには、一致パスコンポーネントで指定したカラムに基づいて生成されるあいまい一致キーが含まれます。あいまいな一致ストラテジを使用する場合、一致プロセスは一致キーのインデックスを検索して一致候補を識別します。一致プロセスが候補を識別できるようにするには、一致ルールのあいまい一致キーから 1 つ以上のカラムを指定する必要があります。例えば、一致キーが [名前] カラムに基づいて生成される場合、一致ルールに [名前] カラムが含まれている必要があります。

次の表に、一致ルールタイプ、一致/検索ストラテジ、および各組み合わせの説明を示します。

一致ルールタイプ	使用する一致/検索ストラテジ	説明
あいまい一致ルール	あいまい一致ストラテジ	あいまい一致カラムを含み、完全一致カラムを含む可能性があります。あいまい一致ルールは、名前のスペル間違いなどのデータのバリエーションを含むカラムに推奨されます。
完全一致ルール	完全一致ストラテジ	完全一致カラムのみが含まれます。一致プロセスは、データベース内の行で SQL 文を実行することで一致を識別します。 完全一致ルールは、ID や誕生日などの、あいまい一致機能が必要なカラムに適しています。
フィルター一致ルール	あいまい一致ストラテジ	完全一致カラムのみが含まれます。ルールは、関連する一致キーに基づいてフィルターされた一致候補で実行されます。一致プロセスでは、一致キーインデックスを検索して一致候補を識別します。 完全一致カラムのいずれかにあいまい一致キーと同じソースカラムがある場合は、フィルター一致ルールを使用することをお勧めします。 フィルター一致ルールは、データベースではなくプロセスサーバーで完全一致ルールを実行することにより、パフォーマンスを向上させます。

ヒント: データベースサーバーに関連するパフォーマンスの問題に制約される場合、完全一致ルールの代わりにフィルター一致ルールを使用することを検討します。フィルター一致ルールでは、完全一致ルールで実行するよりも大きいバッチを実行できます。また、フィルター一致でバッチサイズを大幅に増やしても、一致ジョブの時間はわずかに増加しません。

一致目的

あいまい一致ベースオブジェクトの場合は、**一致目的**によって一致ルールの背後にある主要な目的が定義されます。例えば、人物の一致を特定する場合に、2つのレコードが同じ人物のものかどうかを判別するための重要な要素が住所であるときは、住居という一致目的を選択します。

定義するすべての一致ルールに対し、Informatica が提供する一連の定義済み一致目的からそのルールの目的を選択する必要があります。各一致目的には、2つのレコードの比較方法について、その目的に合った最適な方法に関する情報が含まれています。Informatica MDM Hub では、選択された一致目的を基に一致ルールを適用して、一致するレコードを判別します。ルールの動作は、選択した目的によって異なります。選択可能な一致目的のリストは、使用するポピュレーションによって変わります。

一致目的で決まる内容

一致目的では以下の内容が決まります。

- 一致ルールの動作
- 必要なカラム
- 一致プロセスで 사용되는各カラムに対する Informatica MDM Hub の注目度

すべての属性（目的を除く）が同じである2つのルールでは、目的が異なるため、異なる一致セットが返されます。

必須のフィールドとオプションのフィールド

各一致目的では、必須フィールドとオプションフィールドの組み合わせがサポートされます。各フィールドには、そのフィールドが一致の決定に与える影響に応じて重みが付けられます。一部の目的の一部のフィールドはグループ化することができます。グループ化には2つのタイプがあります。

- [必須] -フィールドメンバの少なくとも1つがNULLでないことを必要とします。
- [最高スコア] -グループ内のフィールドから最も高いスコアだけを全体の一致スコアに提供します。

例えば、[個人] 一致目的は以下のように設定します。

- Person_Name が必須フィールドです。
- ID Number または Date of Birth のいずれかが必要です。
- その他の属性はオプションです。

各目的から返される全体スコアは、関与するフィールドのスコアを加算して計算されます。フィールドのスコアは、事前にそれぞれのフィールドの重みを掛けた後に、全フィールドの重みの合計で割って算出されます。フィールドがオプションで、かつ提供されていない場合、そのフィールドは重み計算に算入されません。

名前の形式

Informatica MDM Hub の照合にはデフォルトの名前形式があり、これによって姓の位置を指定します。次のオプションがあります。

- [左] ーフルネームの始めが姓（例 Smith Jim）。
- [右] ーフルネームの終わりが姓（例 Jim Smith）。

Informatica MDM Hub が使用する名前形式は、ユーザーが使用する目的によって異なります。ユーザーが Organization を使用する場合、デフォルトの形式は姓、名、ミドルネームです。ユーザーが Person または Resident を使用する場合、デフォルトの形式は名、ミドルネーム、姓です。

注: マッチングのためにデータをフォーマットする場合は、Informatica MDM Hub で使用される名前形式はユーザーが使用する目的によって異なるということに注意します。選択されたポピュレーション内に名前が存在しないという特別な場合には、選択する名前形式と照合目的によって違いが生じます。

一致目的のリスト

以下の表に、Informatica が提供している照合目的を示します。

一致目的	説明
Person_Name	<p>個人を名前で特定します。名前のみのルックアップが必要で、かつ人間が選択を行うことができる場合に、この目的をオンライン検索に使用します。マッチングには、一致を判断するために名前に加えて他の属性が必要となります。この目的を使用する場合は、ルールに住所フィールドを含めることはできません。この目的で、住所情報がある人物と住所情報がない人物を照合します。ルールに住所フィールドが含まれている場合は、代わりに Resident 一致目的を使用します。</p> <p>この目的で使用されるフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Address_Part1 - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールド内で Postal_Area を繰り返し値として使用します。</p>
Individual	<p>名前属性、ID 番号属性、および誕生日属性で個人を特定します。</p> <p>Individual 一致目的には、Person_Name 一致目的で提供される追加情報が必要です。このため、Individual 一致目的は Person_Name による検索の後で使用します。</p> <p>この照合目的で使用されるフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - ID (必須) - Date (必須) - Attribute1 - Attribute2
Resident	<p>ある住所にいる個人を特定します。この照合目的は、Person_Name または Address_Part1 による検索の後で使用します。オプションの入力フィールドは、一致の限定やランク付けに役立ちます (詳細情報が得られる場合)。</p> <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p> <p>この照合目的で使用されるフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Address_Part1 (必須) - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode

一致目的	説明
Household	<p>姓が同じかほぼ同じである個人が同じ住所を使用している一致を特定します。</p> <p>この照合目的は、Address_Part1 による検索の後で使用します。</p> <p>注: Person_Name による検索は実用的ではありません。これは、最終的に Person_Name 内の単語が 1 つでも一致していれば一致と見なされても、多くの場合単語 1 つによる検索はうまくいかないためです。</p> <p>Person_Name フィールドの主要な単語である姓が重視されるため、これはレコードをマッチング用に提示する際に単語の順序が重要となる数少ないケースの 1 つです。</p> <p>ただし、一方の名前の主要な単語ともう一方の名前のその他の単語が一致した場合は、妥当なスコアが生成されます。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Address_Part1 (必須) - Address_Part2 - Postal_Area - Telephone_Number - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>
Family	<p>姓が同じかほぼ同じである個人が同じ住所または同じ電話番号を使用している一致を特定します。</p> <p>この照合目的は、Address_Part1 と Telephone_Number による段階的検索（マルチ検索）の後で使用します。</p> <p>注: Person_Name による検索は実用的ではありません。これは、最終的に Person_Name 内の単語が 1 つでも一致していれば一致と見なされても、多くの場合単語 1 つによる検索はうまくいかないためです。</p> <p>Person_Name フィールドの主要な単語である姓が重視されるため、これはレコードをマッチング用に提示する際に単語の順序が重要となる数少ないケースの 1 つです。</p> <p>ただし、一方の名前の主要な単語ともう一方の名前のその他の単語が一致した場合は、妥当なスコアが生成されます。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Address_Part1 (必須) - Telephone_Number (必須) (スコアは Address_Part_1 と Telephone_Number の最高スコアに基づきます) - Address_Part2 - Postal_Area - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>

一致目的	説明
Wide_Household	<p>姓または電話番号が同じである複数の個人が同じ住所を共有している一致を特定します。</p> <p>この照合目的は、Address_Part1 による検索の後で使用します。</p> <p>注: Person_Name による検索は実用的ではありません。これは、最終的に Person_Name 内の単語が 1 つでも一致していれば一致と見なされても、多くの場合単語 1 つによる検索はうまくいかないためです。</p> <p>Person_Name フィールドの主要な単語である姓が重視されるため、これはレコードをマッチング用に提示する際に単語の順序が重要となる数少ないケースの 1 つです。</p> <p>ただし、一方の名前の主要な単語ともう一方の名前のその他の単語が一致した場合は、妥当なスコアが生成されます。</p> <p>この照合目的で使用されるフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Address_Part1 (必須) - Person_Name (必須) - Telephone_Number (必須) (スコアはと Person_Name と Telephone_Number の最高スコアに基づきます) - Address_Part2 - Postal_Area - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>
Address	<p>住所の一致を特定します。住所は、郵送先、居住地、配送先、説明用、公式、非公式などの可能性があります。</p> <p>必須フィールドは Address_Part1 です。Address_Part2、Postal_Area、Telephone_Number、ID、Date、Attribute1、Attribute2 の各フィールドは、住所をさらに区別するためのオプションの入力フィールドです。例えば、市区町村や都道府県の名前が Address_Part2 として提供されていれば、さまざまな場所にある一般的な番地「100 Main Street」を区別するのに役立ちます。</p> <p>この照合目的で使用されるフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Address_Part1 (必須) - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 で Postal_Area を繰り返し値として渡します。その場合、使用される Address_Part2 スコアは、スコア付けされた 2 つのフィールドのうちの高い方になります。</p>

一致目的	説明
Organization	<p>組織を主に名前で照合します。この目的は、名前だけのルックアップが必要で、かつ人間が選択を行うことができるオンライン検索に適しています。バッチでのマッチングには、一致を判断するために名前に加えて他の属性が必要となります。この照合目的は、ルールに住所フィールドが含まれていない場合に使用します。この照合目的を使用すると、住所情報がある組織と住所情報がない組織との間の照合を行うことができます。ルールに住所フィールドが含まれている場合は、Division 一致目的を使用します。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Organization_Name (必須) - Address_Part1 - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>
Division	<p>ある住所を所在地とする組織を特定します。この照合目的は、Organization_Name または Address_Part1 (あるいはこの両方) による検索の後で使用します。</p> <p>この照合目的は、Address_Part1 が必須フィールドである点を除いて、基本的に Organization と同じです。この照合目的は、例えば住所 Y を所在地とする会社 X (複数の住所が提供されている場合は住所 W を所在地とする会社 Z) を照合するために使用できます。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Organization_Name (必須) - Address_Part1 (必須) - Address_Part2 - Postal_Area - Telephone_Number - ID - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>

一致目的	説明
Contact	<p>特定の場所にいる組織内の担当者を特定します。</p> <p>この照合目的は、Person_Name による検索の後で使用します。ただし、Organization_Name または Address_Part1 が検索条件として使用される場合があります。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Organization_Name (必須) - Address_Part1 (必須) - Address_Part2 - Postal_Area - Telephone_Number - ID - Date - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>
Corporate_Entity	<p>組織を正式な社名（INC や LTD などの正式な末尾を含む）で特定します。この照合目的は、例えば ABC TRADING INC という名称と ABC TRADING LTD という名称を区別する必要があるアプリケーションなどで使用されます。</p> <p>この照合目的は、Organization_Name による検索の後で使用します。この照合目的は、より厳密なマッチングが行われる点と、正式な末尾が不要な情報とされない点を除いて、基本的に Organization と同じです。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Organization_Name (必須) - Address_Part1 - Address_Part2 - Postal_Area - Telephone_Number - ID - Attribute1 - Attribute2 - Geocode <p>Address_Part2 と Postal_Area の間で最高スコアを得るには、Address_Part2 フィールドで Postal_Area を繰り返し値として渡します。</p>
Wide_Contact	<p>所在地に関係なく組織内の担当者を大まかに特定します。</p> <p>この照合目的は、Person_Name による検索の後で使用します。</p> <p>必須フィールドに加えて ID、Attribute1、および Attribute2 をオプションで指定して、担当者をさらに限定することができます。</p> <p>この照合目的で使用するフィールドは以下のとおりです。</p> <ul style="list-style-type: none"> - Person_Name (必須) - Organization_name (必須) - ID - Attribute1 - Attribute2
Fields	<p>特定のでない一般的な用途に使用します。この照合目的には必須フィールドはありません。すべてのフィールドタイプを、オプションの入力フィールドとして使用できます。</p>

一致レベル

あいまい一致ベースオブジェクトの場合は、一致レベルによって一致の精度が決まります。あいまい一致ベースオブジェクトには、以下の一致レベルのいずれかを指定できます。

表 2. 一致レベル

レベル	説明
標準	ほとんどの一致に適しています。
保守的	標準レベルよりも生成される一致の数が少なくなります。実際は一致するデータが、一致フラグを設定されずに一致プロセスを通過する場合があります。この状況は、一致不足と呼ばれます。
ルーズ	標準レベルよりも生成される一致の数が多くなります。ルーズな一致では、実際は一致でない一致候補が多数生成される可能性があります。この状況は、一致過多と呼ばれます。このレベルを手動マージ用の一致ルールで使用すると、より厳格な他の一致ルールで一致候補が見逃されていないかを確認できます。

一致させるデータに関する知識に基づいて、標準、保守的（一致数減少）、ルーズ（一致数増加）のいずれかのレベルを選択します。わからない場合は、標準を使用してください。

ジオコード半径

一定の半径内に存在するレコードを識別する場合は、あいまいなベースオブジェクトの照合ルールプロパティとしてジオコード半径を使用します。[ジオコード半径] フィールドは、照合カラムの [ジオコード] フィールド名を選択すると有効になります。

ジオコード半径をメートル単位で指定します。ジオコード半径は、C_REPOS_MATCH_RULE テーブルの GEOCODE_RADIUS カラムに保存されます。指定するジオコード半径にレコードが近い場合、マッチ率は高くなります。指定するジオコード半径から遠い位置にレコードが存在する場合、マッチ率は下がります。

許容限度の調整

許容限度は、あいまい一致ベースオブジェクトの一致の許容度を決定する数値です。この設定動作は一致レベルとまったく同じですが、一致レベルよりも細かな制御を提供します。許容限度は、Informatica によって、ポピュレーション内にその一致目的に応じて定義されています。[許容限度の調整] では、この一致ルールで一致と見なす基準の大まかな調整を行うことができます。

- 調整の値が正の場合、一致基準が保守的になります。
- 調整の値が負の場合、一致基準がルーズになります。

例えば、特定のフィールドおよびポピュレーションについて、標準的な一致レベルの許容限度が 80、ルーズな一致レベルの許容限度が 70、保守的な一致レベルの許容限度が 90 であるとします。この場合、調整で正の数値（3 など）を指定すると、許容レベルが少し保守的になります。負の数値（-2 など）を指定すると、許容レベルはルーズになります。

この設定により、必要に応じて一致設定を微調整することができ、特定の状況で役立つ場合があります。許容限度を調整すると、ほんの少しの調整だけでも一致処理に大きく影響し、一致過多や一致不足になることがあります。そのため、値を少しずつ大きくしながら、さまざまな設定で繰り返しテストして、データに最適な設定を特定することをお勧めします。

一致ルールの一致カラムのプロパティ

この節では、一致ルールに対して設定できる一致カラムのプロパティについて説明します。

一致サブタイプ

さまざまなタイプのデータを含んでいるベースオブジェクトの場合は、一致サブタイプオプションを使用して、同じベースオブジェクト内の特定のタイプのデータに一致ルールを適用できます。親/子パスコンポーネントを持つ完全一致カラムに対して、一致サブタイプを有効または無効にすることができます。一致サブタイプは、以下にのみ使用できます。

- ルートでないパスコンポーネントに基づく完全一致カラムタイプ
- あいまい一致/検索ストラテジを使用する一致ルール

一致サブタイプを使用するには、使用する「サブタイプ」カラムとして機能する完全一致カラムを、一致ルールごとに1つ以上指定します。セグメント一致に使用されるかどうかにかかわらず、すべての完全一致カラムにサブタイプインジケータを設定できます。一致プロセスでは、サブタイプカラムが評価されてから、他の一致カラムが評価されます。一致サブタイプは、一致プロセスのパフォーマンスに影響する可能性があるため、適切に使用してください。

一致サブタイプは標準の親/子一致シナリオと同じように動作しますが、一致サブタイプとマークされた一致カラムが一致対象のすべてのレコードで同じでなければならないという要件が追加されます。以下の例では、一致サブタイプカラムが Address Type であり、一致ルールが Address Line1、City、および State で構成されています。

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

一致サブタイプを使用しない場合、Parent ID 3 は 5 および 7 と一致します。一方、一致サブタイプを使用した場合は、Parent ID 3 が 5 と 7 と一致しません。これは一致させる行が異なる住所タイプに区分されているためです。Parent ID 5 および 7 は、一致させる行がいずれも「Billing」住所タイプであるため、互いに一致します。

関連項目：

- [「完全一致ベースオブジェクトの一致カラムのプロパティの編集」 \(ページ 409\)](#)

NULL 一致

NULL 一致を使用して、他の NULL 値に一致する NULL 値を一致プロセスがどのように処理するかを指定します。

注: NULL 一致とセグメント一致は相互に排他的です。NULL 一致は完全一致カラムに適用されます。

デフォルトでは NULL 一致は無効であり、MDM Hub は一致を検索するときに NULL を等しくない値として扱います。NULL 一致を無効にすると、NULL 値はどの値とも一致しなくなります。

NULL 一致を有効にするには、一致カラムに対して次のいずれかの NULL 一致オプションを選択します。

- NULL は NULL に一致
- NULL は非 NULL に一致

NULL は NULL に一致

一致プロセスが 2 つの NULL 値を一致と見なすように指定するには、[NULL は NULL に一致] オプションを有効にします。

NULL 一致のシナリオに基づいて、次の値のペアのいずれかが一致と見なされます。

- 両方のセル値は NULL です。
- 両方のセル値が同一である。

NULL は非 NULL に一致

一致プロセスが NULL 値を任意の非 NULL 値の一致と見なすように指定するには、[NULL は非 NULL に一致] オプションを有効にします。

NULL 一致のシナリオに基づいて、次の値のペアはすべて一致と見なされます。

- 一方のセル値が NULL で、もう一方のセル値が NULL でない。
- 両方のセル値が同一である。

重要: [NULL は非 NULL に一致] オプションを有効にすると、[一致グループの構築] では、1 つの NULL と非 NULL の一致しかグループに含めることが許可されないため、不要な遷移一致が生じる可能性が低減されます。同じ値が削除されないようにするには、同一の完全一致ルールを 2 つ作成します。一方の完全一致ルールでは、[NULL は非 NULL に一致] オプションを有効にします。もう一方の完全一致ルールでは、[NULL は非 NULL に一致] オプションを無効にします。一致ルールの実行シーケンスで、[NULL は非 NULL に一致] オプションが無効にされた完全一致ルールを、[NULL は非 NULL に一致] オプションが有効にされた完全一致ルールよりも必ず前に配置します。

1 つのセル値が非 NULL である場合の [NULL は NULL に一致] の例

マージジョブの実行対象である CUSTOMER ベースオブジェクトがあるとします。完全一致ルールで、NULL 値を非 NULL 値に一致させるオプションを有効にします。CUSTOMER ベースオブジェクトには、John Smith のレコードが 3 つあり、そのうちの 2 つは内線番号が NULL 値になっています。

John Smith のベースオブジェクトレコードは、3 分の 2 のレコードの内線番号が NULL であることを示しています。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	-
2	John Smith	6053128215	-
3	John Smith	6053128215	236

一致とマージジョブの後に、ベースオブジェクトは、NULL 値を持つレコードがマージされることを示します。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	-
3	John Smith	6053128215	236

一致とマージジョブの後に、相互参照テーブルは、NULL 値を持ついずれか 1 つのレコードの Rowid_Object と、非 NULL 値を持つレコードが存続していることを示します。

Rowid_XREF	Rowid_Object	個人名	電話番号	内線番号
1	1	John Smith	6053128215	-

Rowid_XREF	Rowid_Object	個人名	電話番号	内線番号
2	1	John Smith	6053128215	-
3	3	John Smith	6053128215	236

注: このシナリオでは、NULL 値が別の NULL 値と一致するため、2 つのレコードが一致しているかどうかかわからない可能性があります。NULL 値を持つレコードが一致していることを確認するために、MDM Hub を設定して、手動マージ用にそれらのレコードにフラグを付けることができます。

一方のセル値が NULL である場合に NULL が非 NULL に一致する例

マージジョブの実行対象である CUSTOMER ベースオブジェクトがあるとします。完全一致ルールに対して、NULL 値を非 NULL 値に一致させるオプションを有効にします。CUSTOMER ベースオブジェクトには John Smith のレコードが含まれ、レコードの 1 つには内線番号の NULL 値があります。

John Smith のベースオブジェクトレコードは、内線番号が 236 または NULL のいずれかであることを示しています。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236
2	John Smith	6053128215	-

一致とマージジョブの後、NULL 値を含むレコードが非 NULL 値を含むレコードにマージされることがベースオブジェクトに示されます。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236

一致とマージジョブの後、NULL 値を含むレコードのマージ先であるレコードの Rowid_Object が存続することが相互参照テーブルに示されます。

Rowid_XREF	Rowid_Object	個人名	電話番号	内線番号
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	-

注: このシナリオでは、NULL 値が別の NULL 値と一致するため、2 つのレコードが一致しているかどうかかわからない可能性があります。NULL および非 NULL の値を持つレコードが一致していることを確認するために、MDM Hub を設定して、手動マージ用にそれらのレコードにフラグを付けることができます。

両方のセル値が同一である場合の [NULL は非 NULL に一致] の例

一致とマージジョブの実行対象である CUSTOMER ベースオブジェクトが存在します。完全一致ルールに対して、NULL 値を非 NULL 値に一致させるオプションを有効にします。CUSTOMER ベースオブジェクトには、John Smith 氏のレコードが含まれます。両方のレコードで、内線番号の値が同じであるとします。

John Smith のベースオブジェクトレコードは、内線番号が 236 であることを示しています。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236
2	John Smith	6053128215	236

一致とマージジョブの後、ベースオブジェクトには 1 つのレコードが示されます。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236

一致とマージジョブの後、Rowid_Object 1 があるレコードと同じ値を含む Rowid_Object 2 があるレコードが、Rowid_Object 1 があるレコードにマージされます。

一致とマージジョブの後、同じ値を含むレコードのマージ先であるレコードの Rowid_Object が存続することが、相互参照テーブルに示されます。

Rowid_XREF	Rowid_Object	個人名	電話番号	内線番号
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	236

同じ値を持つレコードが、同じ値を持つ別のレコードにマージされる場合に、そのレコードにはマージ先のレコードと同じ Rowid_Object があります。

1つのセル値が NULL でありもう 1 つが同じ値であるときの [NULL は非 NULL に一致] の例

一致とマージジョブの実行対象である CUSTOMER ベースオブジェクトが存在します。完全一致ルールに対して、NULL 値を非 NULL 値に一致させるオプションを有効にします。CUSTOMER ベースオブジェクトには、John Smith 氏のレコードが含まれます。レコードの 1 つに内線番号の NULL 値があり、もう一方のレコードに同じ値があるとしします。

John Smith のベースオブジェクトレコードは、内線番号が 236 または NULL のいずれかであることを示しています。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236
2	John Smith	6053128215	-
3	John Smith	6053128215	236

一致とマージジョブの後、ベースオブジェクトには 2 つのレコードが示されます。

Rowid_Object	個人名	電話番号	内線番号
1	John Smith	6053128215	236
3	John Smith	6053128215	236

[一致グループの構築] では 1 つの NULL と非 NULL の一致しかグループに含めることが許可されないため、一致とマージジョブの後、Rowid_Object 2 を持つレコードは、Rowid_Object 1 を持つレコードにマージされます。Rowid_Object 1 のレコード内の値と同じ値が含まれる Rowid_Object 3 のレコードは、マージされません。

一致とマージジョブの後、NULL 値を含むレコードのマージ先であるレコードの Rowid_Object が存続することが相互参照テーブルに示されます。

Rowid_XREF	Rowid_Object	個人名	電話番号	内線番号
1	1	John Smith	6053128215	236
2	1	John Smith	6053128215	-
3	3	John Smith	6053128215	236

非 NULL 値を含むレコードにマージされる NULL 値を含むレコードには、マージ先レコードの Rowid_Object が入ります。マージされないレコードの Rowid_Object 値は同じままです。

注: このシナリオでは、NULL 値が別の非 NULL 値と一致するため、2 つのレコードが一致しているかわからない可能性があります。NULL および非 NULL の値を持つレコードが一致していることを確認するために、MDM Hub を設定して、手動マージ用にそれらのレコードにフラグを付けることができます。また、Rowid_Object 1 と Rowid_Object 3 がある同じレコードが一致していることを確認するために、同じである 2 つの完全一致ルールを作成します。一方の完全一致ルールでは、[NULL は非 NULL に一致] オプションを有効にします。もう一方の完全一致ルールでは、[NULL は非 NULL に一致] オプションを無効にします。一致ルールの実行シーケンスで、[NULL は非 NULL に一致] オプションが無効にされた完全一致ルールを、[NULL は非 NULL に一致] オプションが有効にされた完全一致ルールよりも必ず前に配置します。

非同等一致

カラム内の同じ値が相互に一致することを回避するには、一致ルールで非同等一致オプションを使用します。非同等一致は完全一致カラムにのみ適用されます。

注: 非同等一致とセグメント一致は相互に排他的です。一方を選択すると、もう一方は選択できなくなります。

非同等一致オプションは、不一致オプションとして考慮できます。オプションが有効な場合の一致結果は、オプションが無効な場合の一致結果の反対になります。

非同等一致を使用して、あいまい一致の組織名カラムおよびまったく一致しない組織タイプカラムを持つレコードを照合できます。組織名が同じであっても、組織タイプが同じでないと一致レコードとはみなされません。

NULL 一致なしの非同等一致

まず、NULL 一致が無効な場合の、非同等一致オプションの効果について考えます。次の表からわかるように、NULL 値は一致せずに同等値は一致します。非同等一致オプションが有効な場合は、反対が真になり、すべての NULL 値は一致し、同等値は一致しません。

次の表は、非同等一致が有効化される前と後の単純一致の結果を示します。

値	非同等一致=False	非同等一致=True
- レコード 1 = NULL - レコード 2 = "Fred"	一致なし	一致
- レコード 1 = NULL - レコード 2 = NULL	一致なし	一致
- レコード 1 = "Bill" - レコード 2 = "Fred"	一致なし	一致
- レコード 1 = "Fred" - レコード 2 = "Fred"	一致	一致なし

NULL は NULL に一致する非同等一致

同様に、非同等一致オプションを有効にして、NULL は NULL に一致オプションを設定した場合、一致の結果は反対の結果に切り替わります。

次の表は、NULL は NULL に一致し、非同等一致が有効化される前と後の一致の結果を示します。

値	非同等一致=False NULL は NULL に一致=True	非同等一致=True NULL は NULL に一致=True
- レコード 1 = NULL - レコード 2 = "Fred"	一致なし	一致
- レコード 1 = NULL - レコード 2 = NULL	一致	一致なし
- レコード 1 = "Bill" - レコード 2 = "Fred"	一致なし	一致
- レコード 1 = "Fred" - レコード 2 = "Fred"	一致	一致なし

NULL は非 NULL に一致する非同等一致

非同等一致オプションを有効にして、NULL は非 NULL に一致オプションを設定した場合、一致の結果は反対の結果に切り替わります。ある環境で、NULL 値を持つレコードは一致しません。次の表の説明を参照します。

次の表は、NULL は非 NULL に一致し、非同等一致が有効化される前と後の一致の結果を示します。

値	非同等一致=False NULL は非 NULL に一致=True	非同等一致=True NULL は非 NULL に一致=True
- レコード 1 = NULL - レコード 2 = "Fred"	一致	一致なし
- レコード 1 = NULL - レコード 2 = NULL	一致なし	一致*
- レコード 1 = "Bill" - レコード 2 = "Fred"	一致なし	一致
- レコード 1 = "Fred" - レコード 2 = "Fred"	一致	一致なし

* 一致ルールで [完全一致でフィルタ済み] を使用した場合、結果は一致します。しかし、一致ルールで [完全] を使用した場合、結果は一致しません。

セグメント一致

注: セグメント一致と非同等一致は相互に排他的です。一方を選択すると、もう一方は選択できなくなります。セグメント一致と NULL 一致も相互に排他的です。一方を選択すると、もう一方は選択できなくなります。

完全一致カラムに対してのみ、セグメント一致を使用して、一致ルールを特定のデータのサブセットに限定することができます。例えば、セグメント一致を使用して別個の国の顧客に別々の一致ルールを定義して、特定の国コードに特定のルールを限定的に使用することができます。セグメント一致は、完全一致とあいまい一致のベースオブジェクト両方に適用されます。

[セグメント一致] チェックボックスをチェック（選択）する際、他の 2 つのオプションを設定できます。[すべてのデータのセグメント一致] と [セグメント一致の値] です。

すべてのデータでセグメント一致を行う

このチェックボックスをチェック解除すると（デフォルト）、Informatica MDM Hub は、[セグメント一致の値] で定義した値のセット内でのみレコードを一致させます。

例えば、見込み客、パートナ、顧客、およびサプライヤを含むベースオブジェクトがあるとします。[セグメント一致の値] に値「見込み客」と「パートナ」が含まれており、[すべてのデータのセグメント一致] がチェック解除されている場合、Informatica MDM Hub では「見込み客」または「パートナー」を含むレコード内でのみ一致が実行されます。「顧客」および「サプライヤ」のレコードはすべて無視されます。

[すべてのデータのセグメント一致] がチェック（選択）されている場合、「見込み客」と「パートナ」は「顧客」および「サプライヤ」に一致しますが、「顧客」と「サプライヤ」は相互に一致しません。

複数のカラムの値の連結

連結されたカラムに対するセグメント一致を完全一致で有効にしている場合、連結されたフィールドにあるデータのそれぞれにスペース文字を追加する必要があります。

注: 完全一致カラムについては、カラムを連結しないことをお勧めします。

セグメント一致の値

セグメント一致の場合、セグメント一致に使用するセグメント値のリストを指定します。

セグメント一致を定義する（一致カラムの）値を 1 つまたは複数指定する必要があります。例えば、特定の一致ルールで性別によってセグメント一致を定義するとします。セグメント一致値 M（男性）を指定した場合、この一致ルールに対して、Informatica MDM Hub は、男性のレコードに対してのみ（他の一致カラムに基づいて）一致を検索します。[すべてのデータのセグメント一致] も有効にしない限り、他の男性レコードに対する一致のみを実行できます。

注: セグメント一致値では、大文字と小文字が区別されます。あいまい一致ベースオブジェクトおよび完全一致ベースオブジェクトに対してセグメント一致を使用する場合、設定した値は、一致バッチジョブの実行時に大文字と小文字が区別されます。






一致カラムルールの完全一致カラムに対する要件



完全一致カラムは以下のルールに従います。

- 完全一致カラムの名前は 27 文字以上にできません。
- 完全一致カラムのタイプは VARCHAR または CHAR です。
- 一致カラムは、テキストカラムの一致、またはベースオブジェクトからのテキストカラムの組み合わせの一致に使用されます。
- 数値または日付を使用する場合、ベースオブジェクトにロードする前に、クレンジング関数を使用して VARCHAR に変換する必要があります。
- 一致カラムは、子ベースオブジェクトからのカラムの一致にも使用できます。この場合、テキストカラムまたは子ベースオブジェクトのテキストカラムの組み合わせが基になります。子ベースオブジェクトの一致カラムに対する一致は、テーブル間一致と呼ばれます。
- テーブル間一致を使用して子テーブルに対する一致ルールを作成する場合は（外部キーを使用）、子に対する各一致ルールに親テーブルからの外部キーを含める必要があります。外部キーを含めない場合、子をマージする際、親テーブルに属していた子レコードは失われます。

カラムの一致ルールを設定するためのコマンドボタン

[一致ルールセット] タブのリストで一致ルールセットを選択すると、スキーママネージャに以下のコマンドボタンが表示されます。

ボタン	説明
	一致ルールを追加します。
	選択した一致ルールのプロパティを編集します。
	選択した一致ルールを削除します。
	選択した一致ルールをシーケンス内で上に移動します。
	選択した一致ルールをシーケンス内で下に移動します。

ボタン	説明
	手動統合ルールから自動統合ルールに変更します。このボタンは、手動統合レコードを選択してからクリックします。
	自動統合ルールから手動統合ルールに変更します。このボタンは、自動統合レコードを選択してからクリックします。

注: 一致処理の後に一致ルールを変更した場合、一致をリセットするように求められます。一致をリセットすると、一致テーブルの内容と統合インジケータが2のレコードの内容がすべて削除され、統合インジケータが4にリセットされます。

一致カラムルールの追加

あいまい一致、完全一致、またはフィルター一致のルールを追加できます。

一致カラムを使用して新しい完全一致またはあいまい一致ルールを追加する手順

- スキーママネージャで、設定するベースオブジェクトの「一致/マージ設定の詳細」ダイアログを表示します。
- 書き込みロックを取得します。
- 「一致ルールセット」タブをクリックします。
- リストから一致ルールセットを選択します。
選択した一致ルールセットのプロパティが表示されます。
- 画面の「一致ルール」セクションで、「追加」ボタンをクリックします。
「一致ルールの編集」ダイアログが表示されます。このダイアログは、完全一致ベースオブジェクトとあいまい一致ベースオブジェクトで若干異なります。
- あいまい一致ベースオブジェクトの場合、ダイアログボックスの上部で一致ルールのプロパティを設定します。
- この一致ルール的一致カラムを設定します。
 - 「一致カラム」リストの横にある「編集」ボタンをクリックします。
「一致カラムの追加/削除」ダイアログボックスが表示され、定義した一致カラムのみが表示されます。完全一致ベースオブジェクト、または完全一致/検索ストラテジが設定された一致ルールの場合は、完全タイプのカラムだけを選択できます。あいまい一致ベースオブジェクトの場合は、あいまいタイプまたは完全タイプのカラムを選択できます。
 - 一致カラムルールに追加するフィールドを選択します。
 - 「OK」をクリックします。
選択したフィールドが「一致カラム」リストに表示されます。
- 「一致カラム」リストの各一致カラムについて一致プロパティを設定します。
- 「OK」をクリックします。
- 完全一致の場合は、この一致ルール的一致プロパティを指定します。「OK」をクリックします。
- 「保存」ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
- 変更の保存を確認するように求められた場合は、「OK」ボタンをクリックして変更を保存します。

フィルター一致ルールの設定

フィルター一致ルールは、完全一致カラムを含むあいまい一致ベースオブジェクトに対して設定できます。

フィルター一致ルールを設定する必要があるあいまい一致ベースオブジェクトに対して、完全一致ルールをすでに追加してある必要があります。

完全一致カラムを使用してフィルター一致ルールを追加する手順

1. スキーママネージャで、設定するあいまい一致ベースオブジェクトの [一致/マージ設定の詳細] ダイアログを表示します。
2. 書き込みロックを取得します。
3. **[一致ルールセット]** タブをクリックします。
4. 完全一致ルールタイプの一致ルールを選択し、**[編集]** ボタンをクリックします。
[一致ルールの編集] ダイアログが表示されます。
注: あいまい一致ルールをフィルター一致ルールに直接変更することはできません。
5. [一致/検索ストラテジ] ドロップダウンリストから、**[あいまい]** を選択し、**[OK]** をクリックします。
一致ルールタイプが [フィルタ] に変更されます。
6. **[保存]** ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
7. 変更の保存を確認するように求められた場合は、**[OK]** ボタンをクリックして変更を保存します。

一致カラムルールの編集

既存の一致ルールのプロパティを編集する手順

1. スキーママネージャで、設定する完全一致ベースオブジェクトの [一致/マージ設定の詳細] ダイアログを表示します。
2. 書き込みロックを取得します。
3. **[一致ルールセット]** タブをクリックします。
4. リストから一致ルールセットを選択します。
選択した一致ルールセットのプロパティが表示されます。
5. 画面の [一致ルール] セクションで、**[編集]** ボタンをクリックします。
[一致ルールの編集] ダイアログが表示されます。このダイアログは、完全一致ベースオブジェクトとあいまい一致ベースオブジェクトで若干異なります。
6. あいまい一致ベースオブジェクトの場合は、必要に応じてダイアログボックスの上部で一致ルールプロパティを変更します。
7. この一致ルールの一致カラムを設定します。
事前に一致カラムとして定義したカラムだけが表示されます。
 - 完全一致ベースオブジェクト、または完全一致/検索ストラテジが設定された一致ルールの場合は、完全タイプのカラムだけを選択できます。
 - あいまい一致ベースオブジェクトの場合は、あいまいタイプまたは完全タイプのカラムを選択できます。
 - a. **[一致カラム]** リストの横にある **[編集]** ボタンをクリックします。
スキーママネージャは [一致カラムの追加/削除] ダイアログを表示します。

- b. 対象に含めるカラムの横にあるチェックボックスをチェック（選択）します。
 - c. 対象から除外するカラムの横にあるチェックボックスをチェック解除（選択解除）します。
 - d. **【OK】** をクリックします。
選択したカラムが「一致カラム」リストに表示されます。
8. 編集する一致カラムの一致プロパティを変更します。
 9. **【OK】** をクリックします。
 10. 完全一致の場合は、この一致ルールの一一致プロパティを指定します。 **【OK】** をクリックします。
 11. **【保存】** ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
 12. 変更の保存を確認するように求められた場合は、**【OK】** ボタンをクリックして変更を保存します。

一致カラムルールの削除

一致カラムルールの削除する手順

1. スキーママネージャで、設定する完全一致ベースオブジェクトの「一致/マージ設定の詳細」ダイアログを表示します。
2. 書き込みロックを取得します。
3. **【一致ルールセット】** タブをクリックします。
4. リストから一致ルールセットを選択します。
5. 「一致ルール」セクションで、削除する一致ルールを選択します。
6. **【削除】** ボタンをクリックします。
削除を確認するように求められます。
7. **【はい】** をクリックします。

一致カラムルールの実行シーケンスの変更

一致カラムルールの実行シーケンスを変更する手順

1. スキーママネージャで、設定する完全一致ベースオブジェクトの「一致/マージ設定の詳細」ダイアログを表示します。
2. 書き込みロックを取得します。
3. **【一致ルールセット】** タブをクリックします。
4. リストから一致ルールセットを選択します。
5. 「一致ルール」セクションで、上または下に移動する一致ルールを選択します。
6. 次のいずれかを実行します。
 - 選択した一致ルールを実行シーケンス内で上に移動するには、**【上へ】** ボタンをクリックします。
 - 選択した一致ルールを実行シーケンス内で下に移動するには、**【下へ】** ボタンをクリックします。
7. **【保存】** ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
8. 変更の保存を確認するように求められた場合は、**【OK】** ボタンをクリックして変更を保存します。

一致カラムルールの統合オプションの指定

一致プロセスで一致レコードを手動または自動統合用にキューに追加するかどうかは、一致カラムルールによって決定されます。

注: 自動統合用に設定されている一致ルールがある場合、ベースオブジェクトには 200 を超えるユーザー定義カラムを設定できません。

一致ルールに対して手動または自動統合を切り替える手順

1. スキーママネージャで、設定する完全一致ベースオブジェクトの [一致/マージ設定の詳細] ダイアログを表示します。
2. 書き込みロックを取得します。
3. **[一致ルールセット]** タブをクリックします。
4. リストから一致ルールセットを選択します。
5. [一致ルール] セクションで、設定する一致ルールを選択します。
6. 次のいずれかを実行します。
 - **[上へ]** ボタンをクリックして、手動統合ルールを自動統合ルールに変更します。
 - **[下へ]** ボタンをクリックして、自動統合ルールを手動統合ルールに変更します。
7. **[保存]** ボタンをクリックして変更を保存します。


変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
8. 変更の保存を確認するように求められた場合は、**[OK]** ボタンをクリックして変更を保存します。

カラムの一致ウェイトの設定

あいまい一致カラムの場合、[一致ルールの編集] ダイアログボックスで一致ウェイトを変更することができません。Informatica MDM Hub では、各カラムに内部的な一致ウェイトが割り当てられます。これは、一致処理におけるこのカラムの重要度を示す数値（テーブル内の他のカラムとの相対値）です。一致ウェイトは、選択した一致目的やポピュレーションによって異なります。例えば、一致目的が個人名の場合、Informatica MDM Hub で一致を評価する際に、他のカラム（住所など）のデータの一致よりも重要度が高い名前カラムのデータの一致が表示されます。

カラムの一致ウェイトを調整してウェイトを高くすると、Informatica MDM Hub で一致の値が分析される際、そのカラムの重要度（他のカラムとの相対値）が高くなります。

カラムの一致ウェイトを設定する手順

1. [一致ルールの編集] ダイアログボックスで、リストからカラムを選択します。
2. ボタン  **[一致ウェイトの調整]** をクリックします。

調整を行った場合、選択したカラムの名前が太字フォントで示されます。
3. **[保存]** ボタンをクリックして変更を保存します。

変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
4. 変更の保存を確認するように求められた場合は、**[OK]** ボタンをクリックして変更を保存します。

カラムに対するセグメント一致の設定

セグメント一致は、一致ルールを特定のデータのサブセットに限定するために完全一致カラムで使用されます。

完全一致カラムに対してセグメント一致を設定する手順

1. [一致ルールの編集] ダイアログボックスで、[一致カラム] リストから完全一致カラムを選択します。
2. [セグメント一致] チェックボックスをチェック（選択）してこの機能を有効にします。
3. 必要に応じて、[すべてのデータでセグメント一致を行う] チェックボックスをチェック（選択）します。
4. セグメント一致を行うセグメント一致の値を指定します。
 - a. [編集] ボタンをクリックします。
スキーママネージャは[値の編集] ダイアログを表示します。
 - b. 次のいずれかを実行します。
 - 値を追加するには、[追加] ボタンをクリックし、追加する値を入力して [OK] をクリックします。
 - 値を削除するには、リストから値を選択し、[削除] ボタンをクリックします。削除を確認するように求められたら、[はい] をクリックします。
5. [OK] をクリックします。
6. [保存] ボタンをクリックして変更を保存します。
変更が保存される前に、スキーママネージャで一致ルールセットが分析され、一致ルールセットに矛盾がある場合にはメッセージが表示されます。
7. 変更の保存を確認するように求められた場合は、[OK] ボタンをクリックして変更を保存します。

プライマリキーの一致ルールの設定

ここでは、Informatica MDM Hub 実装のプライマリキーの一致ルールを設定する方法について説明します。
一致カラムの一致ルールを設定する場合は、[「一致カラムの設定」 \(ページ 401\)](#)の手順を参照してください。

プライマリキーの一致ルールについて

プライマリキーの一致は、ベースオブジェクトに対する複数の異なるソースシステムに同一のプライマリキー値が含まれている場合に使用できます。

このような状況はソースシステムで頻繁に発生するわけではありませんが、発生した場合でも、Informatica MDM Hub のプライマリキー一致オプションを利用すれば、一致するプライマリキーを含むソースシステムのレコードを迅速に照合し、自動的に統合することができます。

例えば、2つのシステムで同じ顧客 ID のセットを使用しているとします。同一のプライマリキー値を使用して両方のシステムから顧客 XYZ123 に関する情報が提供された場合、2つのシステムは確実に同じ顧客を参照しており、レコードは自動的に統合されます。

プライマリキーの一致を指定するときは、同じプライマリキー値を含むソースシステムを指定するだけです。Informatica MDM Hub のマージバッチジョブまたはリンクバッチジョブの実行時に一致するレコードが自動的に統合されるようにする場合は、さらに [一致するレコードの自動マージ] チェックボックスをチェックします。

注: プライマリキー一致ルールを指定した後、キー一致バッチジョブを実行する必要があります。

プライマリキーの一致ルールの追加

新しいプライマリキーの一致ルールを追加する手順

1. スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログを表示します。
2. 書き込みロックを取得します。
3. **[プライマリキーの一致ルール]** タブをクリックします。
[プライマリキーの一致ルール] タブが表示されます。
[プライマリキーの一致ルール] タブには以下のカラムがあります。

カラム	説明
キーの組み合わせ	このプライマリキーの一致ルールを使用して一致させる 2 つのソースシステム。これらのソースシステムは Informatica MDM Hub で事前に定義されている必要があります。また、このベースオブジェクトのステージングテーブルがこれらのソースシステムに関連付けられている必要があります。
自動マージ	このプライマリキーの一致ルールで自動統合と手動統合のどちらを行うかを指定します。

4. **[追加]** ボタンをクリックしてプライマリキーの一致ルールを追加します。
[プライマリキーの一致ルールの追加] ダイアログが表示されます。
5. プライマリキーに基づいてレコードを一致させる 2 つのソースシステムの横にあるチェックボックスをチェック (選択) します。
6. プライマリキーが同じレコードが確実に一致すると考えられる場合は、**[一致するレコードの自動マージ]** チェックボックスをチェック (選択) します。
[一致するレコードの自動マージ] の設定は、必要に応じて後で変更できます。
7. **[OK]** をクリックします。
[プライマリキーの一致ルール] タブに新しいルールが表示されます。
8. **[保存]** ボタンをクリックして変更を保存します。
既存の一致をリセットするかどうかを確認するメッセージが表示されます。
9. 必要に応じて、**[はい]** をクリックして、一致テーブルに現在格納されている一致をすべて削除します。

プライマリキーの一致ルールの編集

プライマリキーの一致ルールを定義した後に、**[一致するレコードの自動マージ]** チェックボックスの値を変更できます。

既存のプライマリキーの一致ルールを編集する手順

1. スキーママネージャで、設定するベースオブジェクトの [一致/マージ設定の詳細] ダイアログを表示します。
2. 書き込みロックを取得します。
3. **[プライマリキーの一致ルール]** タブをクリックします。
[プライマリキーの一致ルール] タブが表示されます。
4. 編集するプライマリキーの一致ルールまでスクロールします。
5. 自動マージを有効にする場合は **[一致するレコードの自動マージ]** チェックボックスをオンに、無効にする場合はオフにします。

6. **【保存】** ボタンをクリックして変更を保存します。
既存の一致をリセットするかどうかを確認するメッセージが表示されます。
7. 必要に応じて、**【はい】** をクリックして、一致テーブルに現在格納されている一致をすべて削除します。

プライマリキーの一致ルールの削除

既存のプライマリキーの一致ルールを削除する手順

1. スキーママネージャで、設定するベースオブジェクトの **【一致/マージ設定の詳細】** ダイアログを表示します。
2. 書き込みロックを取得します。
3. **【プライマリキーの一致ルール】** タブをクリックします。
【プライマリキーの一致ルール】 タブが表示されます。
4. 削除するプライマリキー一致ルールを選択します。
5. **【削除】** ボタンをクリックします。
削除を確認するように求められます。
6. **【はい】** を選択します。
スキーママネージャにより、削除したルールが **【プライマリキーの一致ルール】** タブから削除されます。
7. **【保存】** ボタンをクリックして変更を保存します。
既存の一致をリセットするかどうかを確認するメッセージが表示されます。
8. 必要に応じて、**【はい】** をクリックして、一致テーブルに現在格納されている一致をすべて削除します。

一致キーの分布の調査

この節では、一致キーテーブル内の一致キーの分布を調べる方法について説明します。

一致キーの分布について

一致キーとは、一致する候補を特定するために使用されるあいまい一致キーカラムのデータをエンコードする文字列です。

トークン化プロセスにより、ベースオブジェクトのすべてのレコードに対する一致キーが生成され、一致キーテーブルに格納されます。トークン化プロセスでは、ベースオブジェクトレコードのデータの内容に応じて、ベースオブジェクトレコードごとに少なくとも 1 つ（複数の場合もあり）の一致キーが生成されます。一致キーは、以降の一致プロセスで、ベースオブジェクトレコード間の一致候補を特定するために使用されます。

スキーママネージャの **【一致/マージのセットアップの詳細】** ペインにある **【一致キーの分布】** タブで、一致キーテーブル内の一致キーの分布を調べることができます。このツールは、データ内の潜在的なホットスポット（一致キーが集中しているために一致過多になる可能性がある箇所）を特定するのに役立ちます。ホットスポットでは、関係がない一致も含め、必要以上に多くの一致が一致プロセスで生成されます。データ内のホットスポットの発生箇所がわかれば、データクレンジングおよび一致ルールを調整してホットスポットを減らし、一致プロセスで使用する一致キーの最適な分布を生成することができます。すべてのキーが比較的均等に分布した状態が理想的です。

【一致キーの分布】 タブへの移動

【一致キーの分布】 タブに移動する手順

1. スキーママネージャで、設定するベースオブジェクトの【一致/マージ設定の詳細】ダイアログを表示します。
2. 【一致キーの分布】 タブをクリックします。
【一致キーの分布】 タブが表示されます。

【一致キーの分布】 タブのコンポーネント

【一致キーの分布】 タブには、ヒストグラム、一致キー、および一致カラムが表示されます。

ヒストグラム

ヒストグラムには、一致キーテーブル内の一致キーの統計的分布が表示されます。

軸	説明
キー (X 軸)	一致キーの開始文字 (1 文字以上)。フィルタが適用されない場合 (デフォルト) は、一致キーの先頭の文字です。フィルタが適用される場合は、一致キーの左端の文字から始まる一連の文字です。
カウント (Y 軸)	一致キーテーブルでの、開始文字 (1 文字以上) で始まる一致キーの数。ヒストグラム内の他の文字と比べて極端に突出している (一致キーの数が多い) 部分が、一致キーテーブルにおけるホットスポットを表します。

一致キーのリスト

【一致キーの分布】 タブの一致キーリストには、一致キーテーブル内のレコードが表示されます。




レコードごとに、以下のカラムのセルデータが表示されます。

カラム名	説明
ROWID	この一致キーに関連付けられているベースオブジェクト内のレコードを一意に識別する ROWID_OBJECT。
KEY	生成された一致キー。一致キーテーブル内の SSA_KEY カラム。

設定されている一致ルールおよびレコード内のデータの性質によっては、ベースオブジェクトテーブル内の単一レコードに複数の一致キーを生成することができます。

一致キーテーブルのレコードのページング

一致キーテーブルのレコード上を移動するには以下のコマンドボタンを使用します。

ボタン	説明
	一致キーテーブルのレコードの最初のページを表示します。
	一致キーテーブルのレコードの前のページを表示します。
	一致キーテーブルのレコードの次のページを表示します。
<input type="text" value="1"/>	入力したページ番号にジャンプします。

一致カラム

[一致キーの分布] タブの [一致カラム] 領域には、一致キーリスト内の選択したレコードの一致カラムデータが表示されます。

これは一致キーテーブル内の SSA_DATA カラムです。このベースオブジェクトに対して設定されている一致カラムごとに、カラム名とセルデータが表示されます。

一致キーのフィルタリング

一致キーフィルタを使用して、ホットスポットまたはその他の一致キー分布パターンを重点的に調査することができます。

一致キーフィルタにより、ヒストグラムおよび一致キーリスト内のデータが、フィルタ条件を満たす一致キーのサブセットに制限されます。デフォルトではフィルタが定義されていません。つまり、一致キーテーブル内のすべてのレコードが表示されます。

フィルタ条件では、適格となる一致キーの開始文字列シーケンス（左から右に評価）を指定します。例えば、文字 M で始まる一致キーのみを表示するには、フィルタに M を選択します。一致キーをさらに制限し、文字 MD で始まる一致キーのデータのみを表示するには、フィルタに文字 D を追加します。フィルタ式が長いほど、表示内容は限定的になります。

フィルタの設定



フィルタを設定する手順

1. ヒストグラムで、フィルタに追加する文字に対応する縦棒をクリックします。
2. X 軸の横の文字の上の縦棒をクリックすると、ヒストグラムが更新されて、この文字で始まるすべての一致キーが表示されます。

一致キーリストには、フィルタの設定に一致する一致キーだけが表示されます。

フィルタの切り替え

以下のコマンドボタンを使用してフィルタを操作します。

ボタン	説明
	フィルタをクリアします。デフォルト値を表示します（フィルタなし）。
	前に選択したフィルタを表示します（フィルタから右端の文字を削除）。

マッチプロセスからのレコードの除外

Informatica MDM Hub には、マッチプロセスからレコードを選択的に除外するためのメカニズムがあります。例えば、マッチプロセスで無視したいレコードがデータに含まれている場合は、この操作が必要となります。

この機能を設定するには、スキーママネージャで EXCLUDE_FROM_MATCH という名前のコラムをベースオブジェクトに追加します。このコラムは、デフォルト値をゼロ（0）とする整数型にする必要があります。

レコードを一致処理から除外するには、テーブルにデータが入力された後で、かつ一致ジョブを実行する前に、データマネージャでそのレコードの EXCLUDE_FROM_MATCH コラムの値を 1 に変更します。一致ジョブが実行されると、EXCLUDE_FROM_MATCH の値がゼロ（0）であるレコードだけがトークン化されて処理され、それ以外のレコードはすべて無視されます。

マッチプロセスからのレコードの除外は、以下に対して実行可能です。

- 完全一致ベースオブジェクトとあいまい一致ベースオブジェクトの両方
- 重複データの一致処理を行わない一致カラムルールのみ（プライマリキー一致ルールではなく）

近接検索

ジオコード半径を指定して、おおよそその半径内に収まっているレコードを検索できます。近接検索は、カラムに緯度、経度、標高（標高は省略可能）が表示されるベースオブジェクトに対して実行できます。

選択するベースオブジェクトに緯度、経度、標高などの地理的座標のためのカラムが 1 つまたは複数含まれていることを確認します。地理的座標は符号付きの度数形式で表す必要があり、カンマまたはスペースで区切って複数または 1 つのカラムに入力できます。例えば、Geocode という名前の 1 つのカラムを作成すると、このカラムにすべての地理的座標 -23.399437, -52.090904, 203 を表示できます。あるいは、3 つのカラムを作成し、カラム latitude に値 -23.399437、カラム longitude に値 -52.090904、カラム elevation に値 203 を表示することもできます。

ジオコード半径（1000 メートルなど）を指定すると、1000 メートル以内のレコードを検索できます。近接検索を使用するには、MDM Hub が一致キーの生成に使用する別のあいまい一致カラム（Person_Name など）が必要です。MDM Hub は、これらの一致キーに基づいてレコードをグループ化します。次に MDM Hub は、ジオコード半径が含まれる照合ルールを使用し、照合される各グループで他方から 1000 メートル以内であるレコードを照合します。

近接検索の設定

近接検索を設定するには、ジオコードカラムと、その他のあいまい一致カラム（1つ以上）を設定する必要があります。MDM Hub は、設定するジオコードカラム以外のあいまい一致カラムに基づいて近接検索の一致キーを生成します。

1. スキーマツールを選択し、書き込みロックを取得します。
スキーマナビゲータが表示されます。
2. 近接検索を設定するベースオブジェクトを展開し、[一致/マージ設定] を選択します。
[一致/マージ設定の詳細] ペインが表示されます。
3. [プロパティ] タブをクリックし、ベースオブジェクトの一致プロパティを設定します。
[一致/検索ストラテジ] プロパティの値は、必ず [あいまい] に設定します。
4. 関連レコードが存在する場合は、[パス] タブをクリックして一致パスを設定します。
5. [照合カラム] タブをクリックし、照合ルールの照合カラムを設定します。
 - a. [あいまい一致キー] セクションで、あいまい一致キープロパティを設定します。
この際、[キータイプ] リストから、一致キー生成のベースとなるキータイプを必ず選択します。
あいまい一致カラムが、[照合カラム] セクションに追加されます。
 - b. あいまい一致カラムのソースカラムを選択し、これを [選択したカラム] リストに移動します。
 - c. [あいまい一致カラムの追加] をクリックします。
[あいまい一致カラムの追加] ダイアログボックスが表示されます。
 - d. [フィールド名] リストから [ジオコード] を選択します。
 - e. 近接検索の照合カラムを作成するため、地理的座標データが含まれるベースオブジェクトのカラムを [選択したカラム] リストに移動します。
地理的座標データのすべてが単一のカラムに入っている場合は、そのカラムを [選択したカラム] リストに移動します。
 - f. [OK] をクリックします。
6. [一致ルールセット] タブをクリックし、一致ルールセットを設定します。
 - a. [一致ルールセット] セクションの [追加] をクリックします。
[一致ルールセットの追加] ダイアログボックスが表示されます。
 - b. 一致ルールセットの名前を入力し、[OK] をクリックします。
一致ルールセットが [一致ルールセット] セクションに表示されます。
 - c. [一致ルール] セクションの [追加] をクリックします。
[一致ルールの編集] ダイアログボックスが表示されます。
 - d. [編集] をクリックします。
[一致カラムの追加/削除] ダイアログボックスが表示されます。
 - e. [ジオコード] と、一致キー生成のベースとなるあいまい一致カラムを選択します。
 - f. [OK] をクリックします。
[一致ルールの編集] ダイアログボックスが表示されます。
 - g. 照合ルールプロパティ ([ジオコード半径] など) を設定し、[OK] をクリックします。
照合ルールが [照合ルール] セクションに表示されます。

これで、バッチジョブまたはサービス統合フレームワーク API を利用して近接検索を実行できるようになります。

軽量一致

一致のパフォーマンスを向上させるには、軽量一致を設定します。軽量一致は非常に高速なスコア推定を生成します。アルゴリズムにより、明確な不一致を含む候補は拒否され、完全スコアリングに渡されません。一般的なデータセットでは、アルゴリズムにより候補の 99%以上が拒否され、パフォーマンスが向上します。

ヒント: ID 番号や誕生日などの値を含むカラムで厳密な完全一致ルールをすでに使用している場合、軽量一致によりパフォーマンスが大幅に向上することはない可能性があります。

SSA-NAME3 は軽量一致スコアに基づき、候補を拒否します。SSA-NAME3 では、SSA-NAME3 スコアと一致率が高い可能性のある候補を拒否することもあり得ます。軽量一致を適用するフィールドを注意深く選択し、しきい値のチューニングを使用することで、このリスクを軽減できます。

LWM_FIELDS コントロールを使用して軽量一致をフィールドに適用します。LWM_LIMIT コントロールを使用して軽量一致スコアの拒否および許容限度を設定します。

軽量一致の設定

cmxcleanse.properties ファイルのプロパティを設定することで、軽量一致を設定します。

例えば、次の例では、指定したフィールドで軽量一致を有効にします。

```
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
```

次のプロパティ定義では、軽量一致プロパティと一緒に使用方法について説明します。

cmx.server.match.lwm

オプション。手動で追加する必要があります。軽量一致機能を制御します。軽量一致機能と一致レコードでの完全スコアリングを有効にするには、Y に設定します。一致レコードでの完全スコアリングは行わずに軽量一致機能を有効にするには、ONLY に設定します。デフォルトは N です。

このプロパティは cmx.server.match.lwm_param および cmx.server.match.stats プロパティとともに使用します。

cmx.server.match.lwm_param

オプション。手動で追加する必要があります。cmx.server.match.lwm プロパティは Y または ONLY に設定する必要があります。このプロパティ値は次の形式で SSA-NAME3 コントロールに設定します。

LWM=Y LWM_FIELDS=<field1>,<weight1>[,...,<fieldn>,<weightn>] LWM_LIMIT=<Reject>[,<Accept>]

cmx.server.match.stats

オプション。手動で追加する必要があります。cmx.server.match.lwm プロパティは Y または ONLY に設定する必要があります。

SSA-NAME3 コントロール

次の SSA-NAME3 コントロールを cmx.server.match.lwm_param プロパティに追加できます。コントロールはスペースで区切ります。

LWM=Y/N/ONLY

軽量一致を有効または無効にします。軽量一致を有効にするには、値 Y を使用します。軽量一致は、高速スコア推定を使用して明確な不一致を拒否します。軽量一致をパスしたレコードは、堅牢なスコアリングおよびランキングのための完全スコアリングに移動します。SSA-NAME3 は完全なスコアおよびディシジョンを呼び出し元に戻します。

注: SDF ウィザードを使用してシステム定義ファイルを作成する場合、軽量一致はデフォルトで有効です。

軽量一致を無効にするには、値 N を使用します。SSA-NAME3 一致では、すべての一致レコードに対して完全スコアリングを実行します。

軽量一致を有効にし、完全スコアリングを無効にするには、値 ONLY を使用します。軽量一致は推定を最終スコアとして呼び出し元に戻します。

LWM_FIELDS

軽量一致およびそのウェイトを適用するフィールドを指定します。これらの値は実行時に一致目的で定義された値を上書きします。軽量一致スコアに基づき、SSA-NAME3 は明確な不一致を拒否します。いずれの値も設定しない場合、SSA-NAME3 は一致目的からフィールドを取得し、それらに同じウェイトを割り当てます。

LWM_FIELDS コントロールの構文は次のとおりです。

LWM_FIELDS=<field1>,<weight1>[,...,<fieldn>,<weightn>]

ここで、field は目的コントロールで定義した有効なフィールド名で、weight は指定したフィールド (0-100) の他のフィールドと比較したときの相対的な重要性です。

例: LWM_FIELDS=Person_Name,5,Address_Part1,1

軽量一致は、住所などバリエーションの多くないフィールドに適用する際に役立ちます。軽量一致は、SSA-NAME3 が編集リストを介してバリエーションを処理する、バリエーションの多いフィールドでは効率的ではなく、レコードを正しく拒否しない可能性があります。

LWM_LIMIT

軽量一致スコアの、許容および拒否の限度を指定します。この限度に基づき、SSA-NAME3 は検索結果を許容または拒否します。

LWM_LIMIT コントロールの構文は次のとおりです。

LWM_LIMIT=<Reject>[,<Accept>]

ここで、Reject および Accept は 0-100 の整数値です。

例: LWM_LIMIT=50,90

LWM=N の場合、LWM_LIMIT コントロールは無効です。

LWM=Y の場合、SSA-NAME3 は拒否限度よりも低いスコアの軽量一致を拒否します。許容限度は無効なので省略できます。

LWM=ONLY の場合、SSA-NAME3 は拒否限度よりも低いスコアの軽量一致を拒否します。許容限度よりも高いスコアを許容します。拒否限度以上のレコードと許容限度よりも低いレコードのスコアは未決定とマークされます。

デフォルトの拒否限度は 65 で、デフォルトの許容限度は 90 です。許容限度を設定せず、拒否限度が 90 よりも高い場合、許容限度は拒否限度と同じになります。

第 22 章

一致ルールの設定例

この章では、以下の項目について説明します。

- [一致ルールの設定例の概要, 444 ページ](#)
- [一致ルールの設定シナリオ, 445 ページ](#)
- [一致ルールの設定, 446 ページ](#)
- [手順 1. データの確認, 446 ページ](#)
- [手順 2. 一致ジョブのベースオブジェクトの特定, 447 ページ](#)
- [手順 3. 一致プロパティの設定, 447 ページ](#)
- [手順 4. 一致パスの定義, 448 ページ](#)
- [手順 5. 一致カラムの定義, 453 ページ](#)
- [手順 6. 一致ルールセットの定義, 458 ページ](#)
- [手順 7. 一致ルールの追加, 460 ページ](#)
- [手順 8. 一致ルールのマージオプションの設定, 463 ページ](#)
- [手順 9. 一致プロパティの確認, 464 ページ](#)
- [手順 10. 一致ルールのテスト, 466 ページ](#)

一致ルールの設定例の概要

一致ルールの設定例では、MDM Hub 内の重複レコードの一致とマージを行うための一致ルールを設定する方法を示します。一致ルールの設定例は、MDM Hub Resource Kit に含まれている、MDM Hub のサンプルオペレーショナル参照ストア（ORS）で使用可能なデータに基づいています。

レコードが重複している、つまり互いに一致していることを判断するため、MDM Hub は一致ルールを使用します。ベースオブジェクト内のレコードに対して一致ルールを設定します。完全一致またはあいまい一致の一致ルールを定義できます。完全一致ルールは、一致カラムに同一の値を持つレコードを照合します。あいまい一致ルールは、スペルミス、転置、省略、音標の違いといった、データパターンの差異の可能性を考慮した一致確立の規定に基づき、同様のレコードを照合します。設定する一致ルールは、データの特性と、特定の一致とマージの要件に依存します。

一致ルールの設定例では、段階的に設定プロセスを紹介します。この例は、MDM Hub のサンプル ORS で使用可能な Party ベースオブジェクトデータに基づいています。この例は、Party ベースオブジェクト内の個人のデータに対して一致ルールを設定して、名前と住所に基づいてデータの一致とマージを行う方法を示しています。この例で記述されている一致ルールと同様の事前設定済みの一致ルールを表示するには、MDM Hub のサンプル ORS をセットアップします。サンプルオペレーショナル参照ストアの詳細については、『*Multidomain MDM のサンプル ORS ガイド*』を参照してください。

一致ルールの設定シナリオ

組織では顧客情報を、Party ベースオブジェクトとその他の関連付けられたベースオブジェクトに保存しています。Party ベースオブジェクトには多くの顧客の重複レコードが含まれています。重複レコードを識別して、マージのキューに入れられるように、一致ルールを設定します。

Party ベースオブジェクトには、姓名が欠落している可能性がある、または不正なエントリや矛盾するエントリを持つ重複する顧客レコードが含まれています。[表示名] カラムには各レコードの値があります。レコードは、グループタイプ Person または Organization として分類されています。顧客の住所は関連付けられた Address ベースオブジェクトに保存されています。Party Address Rel リレーションベースオブジェクトは、Party ベースオブジェクトと Address ベースオブジェクトのレコード間のリレーションを定義します。

一致ルールを作成して、Party ベースオブジェクト内でグループタイプが Person の重複する顧客レコードを照合します。グループタイプが Person のレコードは個人に属します。Party ベースオブジェクトおよび Address ベースオブジェクトに保存されている名前と住所に基づいて照合を実行できます。

重複レコードを照合するには、ほぼ同様に自動マージのキューに入れることができるレコードに対して、自動マージ一致ルールを作成する必要があります。また、重複の可能性はあるものの、マージプロセスの前にデータスチュワードが確認する必要があるレコードに対して、手動マージ一致ルールを作成する必要があります。

Party ベースオブジェクトの重要なカラムとサンプルレコードを示します。

行 ID オブジェクト	名	姓	Organization Name	表示名	グループタイプ
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

Address ベースオブジェクトの重要なカラムとサンプルレコードを示します。

行 ID オブジェクト	アドレス行 1	アドレス行 2	都市名	州コード	郵便番号
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

LU Address Type ルックアップベースオブジェクトの重要なカラムとサンプルの住所タイプルックアップを示します。

行 ID オブジェクト	アドレスタイプ	アドレスタイプ表示	アドレスタイプ説明
1	BILL	BILLING	BILLING
4	LGL	LEGAL	LEGAL
10	RSID	RESIDENCE	RESIDENCE
11	SHIP	SHIPPING	SHIPPING

Party Address Rel リレーションベースオブジェクトの重要なカラムとサンプルレコードを示します。

行 ID オブジェクト	グループ ID	アドレス ID	アドレスタイプ	Status CD
976	1019	920	BILL	NULL

行 ID オブジェクト	グループ ID	アドレス ID	アドレスタイプ	Status CD
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

一致ルールの設定

一致ルールを設定するには、以下のタスクを実行します。

1. データを確認します。
2. 一致ジョブのベースオブジェクトを特定します。
3. 一致プロパティを設定します。
4. 一致パスを定義します。
5. 一致カラムを定義します。
6. 一致ルールセットを定義します。
7. 一致ルールを追加します。
8. 一致ルールのマージオプションを設定します。
9. 一致プロパティを確認します。
10. 一致ルールをテストします。

手順 1. データの確認

一致ルールを作成する前に、データを確認して理解します。ベースオブジェクトには不正な値、矛盾する値、欠落値がある場合があります。

顧客データのデータ値の品質、一貫性、一意性、論理を確認します。一致ルールを作成して個人を照合するには、個人に関連付けられている属性を理解する必要があります。

サンプルデータセットには、Party ベースオブジェクトと Address ベースオブジェクトが含まれています。Party ベースオブジェクトには、欠落値のある姓カラムと名カラムがあります。姓カラムと名カラムには欠落値があるため、これらのカラムに基づく一致ルールは理想的とは言えません。表示名カラムにはすべてのレコードに値があり、一致ルールで一致カラムとして使用するのに適しています。

Party ベースオブジェクトのグループタイプカラムは、顧客レコードを個人または組織として識別します。組織に属する顧客レコードの一致は検索しないので、グループタイプカラムを使用して、組織に属する顧客レコードを除外することができます。一致プロセスに関係のないデータをフィルタすると、一致のパフォーマンスを向上できます。

Address ベースオブジェクトには、アドレス行 1、都市名、郵便番号など、一致ルールで一致カラムとして使用することができるカラムがあります。Address ベースオブジェクト内のカラムは、Party ベースオブジェクト内の重複レコードを識別するのに役立ちます。名前と住所の属性に基づいて個人を照合する一致ルールを作成できます。

手順 2. 一致ジョブのベースオブジェクトの特定

名前と住所の属性に基づいて個人のデータを照合します。個人のデータの一致ルールを作成する前に、一致ルールのベースとなる特定のデータを含むベースオブジェクトを特定する必要があります。

例では、個人のデータは Party ベースオブジェクトと Address ベースオブジェクトに保存されています。個人の名前は Party ベースオブジェクトに、個人の住所は Address ベースオブジェクトにあります。Party ベースオブジェクトの名前と Address ベースオブジェクトの住所に基づいて、個人の一致を判断し、重複レコードを除外することができます。重複を見つけるため、個人の名前が含まれている Party ベースオブジェクトで一致ジョブを実行できます。

手順 3. 一致プロパティの設定

一致ルールの一致カラムを設定する前に、照合/検索ストラテジなどの Party ベースオブジェクトの一致プロパティを設定します。Party ベースオブジェクトには個人の重複レコードがあるため、一致プロパティを設定します。

一致プロパティは、データの特性、データに関する知識、一致要件とマージ要件に基づいて決定する必要があります。Party ベースオブジェクトには、スペルの違いや転置といった欠陥があります。Party ベースオブジェクトのデータには、確率的な一致に基づくあいまい一致と検索ストラテジが適しています。あいまい一致と検索ストラテジを設定する際に、一致ルールのポピュレーションを設定する必要があります。ポピュレーションは、照合に使用するデータの特性を定義します。照合に選択したデータは米国中心のため、US ポピュレーションを一致プロパティとして設定する必要があります。

一致プロパティの設定







一致プロパティを設定するには、Hub コンソールでスキーマツールを使用します。

1. Hub コンソールで、スキーマツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、一致プロパティを設定する Party ベースオブジェクトを展開します。
4. **[一致/マージ設定]** をクリックします。
[一致/マージ設定の詳細] ページが表示されます。
5. **[プロパティ]** タブをクリックします。^o
[グループの一致/マージ設定の詳細] セクションが表示されます。
6. 一致ルールを設定するには、以下のプロパティを設定する必要があります。

プロパティ	値と説明
一致/マージストラテジ	あいまい。個人名データ内に存在するスペルの差異、誤り、および入れ替えを考慮した確率的な一致ストラテジです。
あいまいポピュレーション	US。一致させるレコードに関して任意の特性を定義するポピュレーションです。

以下の図に、【一致/マージ設定の詳細】 ページの【プロパティ】 タブを示します。

The screenshot shows the 'Match/Merge Setup Details' dialog box with the 'Properties' tab selected. The dialog has a title bar and four tabs: 'Match Rule Sets', 'Primary key match rules', 'Match Key Distribution', and 'Merge Settings'. The 'Properties' tab is active, showing a table of settings for 'Party Match/Merge Setup Details'.

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 

手順 4.一致パスの定義

関連レコードに一致ルールを設定するには、一致パスコンポーネントを設定します。関連レコードは、1 つまたは複数のベースオブジェクトに存在する場合があります。

子レコードを含む一致ルールを設定するには、ルートベースオブジェクトにパスコンポーネントを追加する必要があります。一致パスにコンポーネントを追加する際に、関連する親と子のベースオブジェクト間の接続を定義します。

次のコンポーネントを一致パスに追加します。

- グループ。個人と組織のデータを含むベースオブジェクト。
- 住所。個人または組織の顧客の住所を含むベースオブジェクト。
- LU アドレスタイプ。アドレスタイプのルックアップを含むルックアップベースオブジェクト。
- グループアドレスリレーション。Party ベースオブジェクトのレコードと Address ベースオブジェクトのレコード間のリレーションを定義するリレーションベースオブジェクト。

一致パスコンポーネントの追加

一致パスは、親と子のベースオブジェクト間の接続を定義します。親と子のレコードを含む一致ルールを設定するには、ルートベースオブジェクトに一致パスコンポーネントを追加します。また、一致プロセス時にデータを追加または除外するためのフィルタも追加します。

1. Party ベースオブジェクトの【一致/マージ設定の詳細】 ページで、【パス】 タブをクリックします。

2. 一致パスコンポーネントとして Party Address Rel リレーションベースオブジェクトを追加します。
 - a. [パスコンポーネント] セクションから、[C_PARTY のルート] 一致パスコンポーネントを選択します。

次の図に、[一致/マージ設定の詳細] ページの [パス] タブを示します。

The screenshot shows the 'Match/Merge Setup Details' dialog box with the 'Path Components' 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. [パスコンポーネント] セクションで [追加] をクリックします。
- 次の図に、[パスコンポーネントの追加] ダイアログボックスを示します。

[パスコンポーネントの追加] ダイアログボックスが表示されます。

The screenshot shows the 'Add Path Component' dialog box. It has an 'Identity' section with the following fields:

- Display name: (empty)
- Physical name: C_MT_
- Check For Missing Children: ☒

Below the 'Identity' section is a 'Constraints' table with the following data:

Table	Direction	Foreign Key On
Party Cross-Reference	Parent-to-Child	Rowid Object

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

- c. [表示名] フィールドで、Hub コンソールに表示する一致パスコンポーネントの名前として「グループアドレスリレーション」と入力します。

【物理名】フィールドには、入力した一致パスコンポーネントの表示名に基づく名前が入力されます。物理名とは、データベース内にあるパスコンポーネントの名前です。

次の図に、【パスコンポーネントの追加】ダイアログボックスを示します。

【パスコンポーネントの追加】ダイアログボックスは、名前フィールドが入力された状態で表示されます。

Identity		
Display name	Party Address Rel	
Physical name	C_MT_PARTY_ADDRESS_REL	
Check For Missing Children	<input checked="" type="checkbox"/>	

Constraints:		
Table	Direction	Foreign Key On
Party Address Rel	Parent-to-Child	Rowid Object

- d. 【子の有無を確認する】オプションが有効になっていることを確認します。

このオプションはデフォルトで有効になっています。有効になっている場合は、親ベースオブジェクトレコードとそれに関連する子ベースオブジェクトレコード間で一致が発生します。このオプションを有効にした子ベースオブジェクトの中に親のベースオブジェクトレコードが子レコードを持たない場合でも、この一致は起きます。

注: 一致キーの生成は、【子の有無を確認する】オプションの設定によって異なります。【子の有無を確認する】オプションが無効で、親レコードに子レコードがない場合は、親レコードの一致キーが生成されません。

- e. [制約] セクションから、制約を選択して [OK] をクリックします。
- グループアドレスリレーションのパスコンポーネントが [パスコンポーネント] セクションに表示されます。
- 次の図に、[一致/マージ設定の詳細] ページの [パス] タブを示します。

The screenshot shows the 'Match/Merge Setup Details' dialog box with the 'Path Components' tab selected. The dialog has several tabs: 'Primary key match rules', 'Match Key Distribution', 'Merge Settings', 'Properties', 'Paths', 'Match Columns', and 'Match Rule Sets'. 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
Party Address Rel	C_MT_PARTY_ADDRESS_REL	Party Address Rel	Parent-to-Child	Yes

Below the table is a 'Filters' section with columns for 'Column', 'Operator', and 'Values'.

3. [グループアドレスリレーション] 一致パスコンポーネントの子として、Address ベースオブジェクトを一致パスに追加します。
- [パス] タブの [パスコンポーネント] セクションから、[グループアドレスリレーション] 一致パスコンポーネントを選択します。
 - [追加] をクリックします。
- [パスコンポーネントの追加] ダイアログボックスが表示されます。
- [表示名] フィールドで、Address ベースオブジェクトの Hub コンソールに表示する名前を入力します。

- d. [制約] セクションから、制約を選択して [OK] をクリックします。

[アドレス] パスコンポーネントは、[パスコンポーネント] セクションに [グループアドレスリレーション] 一致パスコンポーネントの子として表示されます。リレーションの方向は [子から親] として表示されます。

次の図に、[一致/マージ設定の詳細] ページの [パス] タブを、パスコンポーネントを設定した状態で示します。

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
Address	C_MT_ADDRESS	Address	Child-to-Parent	Yes

Filters		
Column	Operator	Values

mdm_sample admin 63M of 494M

4. 一致用に個人を追加して組織を除外するには、[グループタイプ] カラムのある Party ベースオブジェクトにフィルタを設定します。
 - a. [パスコンポーネント] セクションから、[C_PARTY のルート] 一致パスコンポーネントを選択します。
 - b. [フィルタ] セクションで [追加] をクリックします。
[フィルタの追加] ダイアログボックスが表示されます。
 - c. [カラム] リストから [グループタイプ] を選択します。
 - d. [演算子] リストから [IN] を選択します。
 - e. [値] フィールドの横にある [編集] をクリックします。
[値の編集] ダイアログボックスが表示されます。
 - f. [追加] をクリックします。
[値の追加] ダイアログボックスが表示されます。
 - g. [値] フィールドに [個人名] を入力し、[OK] をクリックします。
 - h. [OK] をクリックします。

[グループタイプ] カラムのフィルタが [フィルタ] セクションに表示されます。[グループタイプ] カラムのフィルタにより、[組織] のグループタイプに属する「Ryerson Area Med Ctr」などのレコードが、一致キー生成から除外されるようになります。一致キー生成に含まれるのは、[個人] グループタイプに属する「Rachel Arsen」や「Ahmed Rauf」などのレコードのみです。

手順 5.一致カラムの定義

一致パスコンポーネントを設定したら、一致ルールで使用する一致カラムを定義します。Party ベースオブジェクトにあいまい一致と検索ストラテジを選択したので、あいまい一致と完全一致の両方のカラムを設定できます。

個人の重複レコードを照合する一致ルールを作成します。重複レコードを照合するには、一致ルールに個人の名前と住所データのカラムを含める必要があります。

個人の重複レコードを照合するため、次のカラムを一致カラムとして定義します。

- 表示名
- アドレス行 1
- アドレス行 2
- 都市名
- 州コード
- 郵便番号

一致カラムの定義

Party ベースオブジェクトの一致カラムを定義します。

1. Party ベースオブジェクトの **【一致/マージ設定の詳細】** ページで、**【一致カラム】** タブをクリックします。
2. Party ベースオブジェクトにあいまい一致キーを設定します。
 - a. **【キータイプ】** リストから **【個人名】** を選択します。

【パスコンポーネント】 が **【ルート (グループ)】** に設定されます。これが **【一致カラム】** セクションに追加される **【Person_Name】** あいまい一致キーのパスコンポーネントです。

- b. **【キー幅】** リストから **【標準】** を選択します。

MDM Hub が一致キーを生成する検索範囲が標準サイズに設定されます。

次の画像は、個人名あいまい一致キーが設定された **【一致カラム】** タブを示しています。

The screenshot shows the 'Match/Merge Setup Details' window with the following sections:

- Primary key match rules**:
 - Properties**:
 - Fuzzy Match Key**:
 - Key Type**: Person Name
 - Key Width**: Standard
 - Path Component**: Root (Customer)
 - Paths**: (Empty)
- Match Key Distribution**:
 - Match Columns**:

	Field Name	Column Type	Path Component	Source Table
	Person_Name	Fuzzy Match Key	Root	Party
- Merge Settings**:
 - Match Rule Sets**: (Empty)

Match Column Contents - Source Table: Customer

Available columns:		Selected columns:
Display Name	➡	
First Name	➡	
Last Name	➡	
Middle Name	➡	
Organization Name	➡	
Party Type	➡	

MDM Hub は、設定した個人名あいまい一致キーに基づいて一致キーを生成します。

3. **【表示名】** カラムを Person_Name あいまい一致キーカラムのソースカラムとして定義します。

- a. **【一致カラムのコンテンツ】** セクションで、**【使用可能なカラム】** リストから **【表示名】** カラムを選択します。
- b. 右矢印をクリックします。

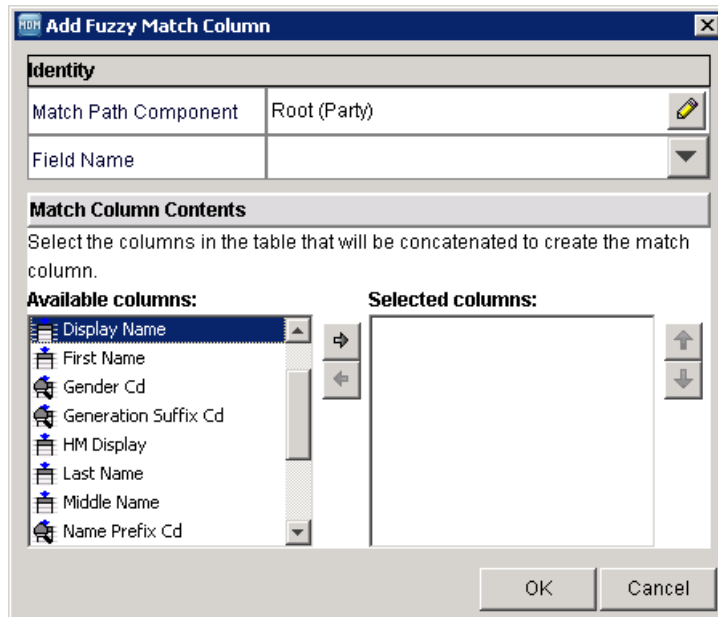
【表示名】 カラムが **【選択したカラム】** リストに移動し、Person_Name 一致カラムのソースカラムが定義されます。

4. あいまい一致カラム (Address_Part1) を追加して番地を含めます。

a. **【あいまい一致カラムの追加】** をクリックします。

【あいまい一致カラムの追加】 ダイアログボックスが表示されます。

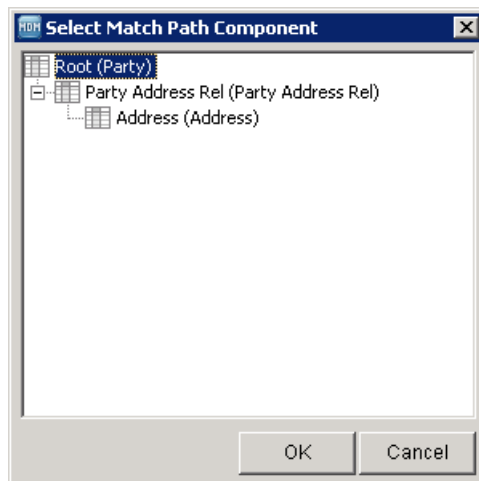
次の画像は、**【あいまい一致カラムの追加】** ダイアログボックスを示しています。



b. **【一致パスコンポーネント】** フィールドの横の**【編集】** をクリックします。

【一致パスコンポーネント】 ダイアログボックスが表示されます。

次の画像は、**【一致パスコンポーネントの選択】** ダイアログボックスを示しています。



c. 顧客住所データを含む Address ベースオブジェクトを選択し、**【OK】** をクリックします。

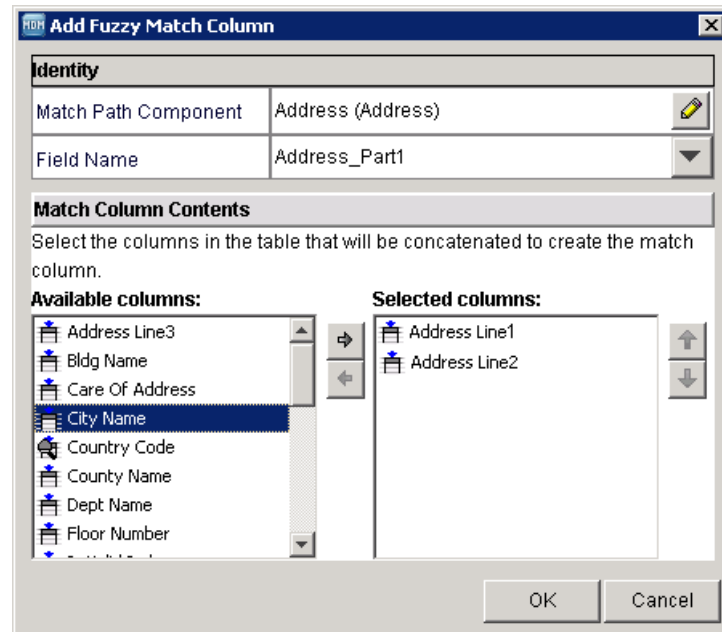
d. **【フィールド名】** リストから **【Address_Part1】** を選択します。

e. **【使用可能なカラム】** リストから、連結して Address_Part1 一致カラムを作成する **【アドレス行 1】** と **【アドレス行 2】** のカラムを選択します。

f. カラムを **【選択したカラム】** リストに移動するには、右矢印をクリックします。

[アドレス行 1] と [アドレス行 2] のカラムが【選択したカラム】リストに移動します。これらのカラムが連結され、Address_Part1 一致カラムが作成されます。【あいまい一致カラムの追加】ダイアログボックスが表示されます。

次の画像は、【あいまい一致カラムの追加】ダイアログボックスを示しています。



- g. **[OK]** をクリックします。

Address_Part1 一致カラムが **[一致カラム]** セクションに追加されます。

次の画像は、個人名およびアドレスデータに一致カラムが設定されている **[一致カラム]** タブを示しています。

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. あいまい一致カラム (Address_Part2) を追加して、顧客住所データに都市名と州コードを含めます。
- a. **[あいまい一致カラムの追加]** をクリックします。
[あいまい一致カラムの追加] ダイアログボックスが表示されます。
- b. **[一致パスコンポーネント]** フィールドの横の **[編集]** をクリックします。
[一致パスコンポーネント] ダイアログボックスが表示されます。
- c. 顧客住所データを含む Address ベースオブジェクトを選択し、**[OK]** をクリックします。
- d. **[フィールド名]** リストから **[Address_Part2]** を選択します。
- e. **[使用可能なカラム]** リストから、連結して Address_Part2 一致カラムを作成する **[都市名]** と **[州コード]** のカラムを選択します。
- f. カラムを **[選択したカラム]** リストに移動するには、右矢印をクリックします。
[都市名] と **[州コード]** のカラムが **[選択したカラム]** リストに移動します。これらのカラムが連結され、Address_Part2 一致カラムが作成されます。
- g. **[OK]** をクリックします。
Address_Part2 一致カラムが **[一致カラム]** セクションに追加されます。

6. 完全一致カラム (Postal_Area) を追加して、顧客住所に郵便番号を含めます。
 - a. **【完全一致カラムの追加】** をクリックします。
【完全一致カラムの追加】 ダイアログボックスが表示されます。
 - b. **【一致パスコンポーネント】** フィールドの横の **【編集】** をクリックします。
【一致パスコンポーネントの選択】 ダイアログボックスが表示されます。
 - c. 顧客住所データを含む Address ベースオブジェクトを選択し、**【OK】** をクリックします。
 - d. **【フィールド名】** リストから **【Postal_Area】** を選択します。
 - e. **【使用可能なカラム】** フィールド名で、Postal_Area 一致カラムを作成するために使用する **【郵便番号】** カラムを選択します。
 - f. カラムを **【選択したカラム】** リストに移動するには、右矢印をクリックします。
【郵便番号】 カラムが **【選択したカラム】** リストに移動します。
 - g. **【OK】** をクリックします。
 Postal_Area 一致カラムが **【一致カラム】** セクションに追加されます。

次の画像は、個人名およびアドレスデータに一致カラムが設定されている **【一致カラム】** タブを示しています。

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

手順 6。一致ルールセットの定義

検索レベルが **【標準】** に設定されている一致ルールセットを定義します。一致ルールは、定義する一致ルールセット内に作成できます。一致ルールセットは少なくとも 1 つ作成する必要があります。一致ルールセットには、いくつかの共通プロパティを持つ一致ルールの論理セットを含めることができます。

一致ルールセットの定義

一致ルールを追加できるルールセットを定義します。

1. Party ベースオブジェクトの **【一致/マージ設定の詳細】** ページで、**【一致ルールセット】** タブをクリックします。
2. 一致ルールセットを追加するには、**【追加】** をクリックします。
【一致ルールセットの追加】 ダイアログボックスが表示されます。
3. WS など、一意の名前をルールセットに入力し、**【OK】** をクリックします。
4. **【検索レベル】** リストから **【標準】** を選択します。
一致候補を検索する検索レベルが **【標準】** に設定されます。
5. SearchMatch API で排他的に使用するために一致ルールセットを予約しないようにするには、**【ルールによる検索を有効にする】** オプションを無効にします。
6. **【フィルタリングを有効にする】** オプションが無効になっていることを確認します。

注: **【フィルタリングを有効にする】** オプションが無効の場合、一致バッチジョブが実行されると、すべてのレコードが一致ルールセットによって処理されます。**【フィルタリングを有効にする】** オプションが有効の場合、この一致ルールセットのフィルタを定義できるため、フィルタリング済みレコードのみが一致ルールセットによって処理されます。

以下の図に、WS 一致ルールセットが設定された状態の **【一致/マージ設定の詳細】** ページを示します。

The screenshot shows the 'Match/Merge Setup Details' dialog box. The 'Match Rule Set' tab is selected, and the 'WS (*)' rule set is highlighted in the left pane. The right pane shows the configuration for this rule set:

Match Rule Set	
Name	WS
Search Level	Typical
Enable Search by Rules	<input type="checkbox"/>
Enable Filtering	<input type="checkbox"/>
Filtering SQL	

Below the configuration table is a section for 'Match Rules' with a table header:

Rule #	Auto	Type	Accept Limit
--------	------	------	--------------

There are buttons for adding (+), deleting (-), and moving up/down arrows on the right side of the Match Rules table.

手順 7.一致ルールを追加

一致ルールを追加する前に、自動マージまたは手動マージに一致ルールを設定するかどうかを決める必要があります。住所で個人を照合するには、名前のコンポーネントと住所のコンポーネントを持つ一致目的が必要です。

自動マージに一致ルールを設定すると、MDM Hub は一致レコードをデータスチュワードの介入なしでマージします。手動マージに一致ルールを設定すると、MDM Hub は一致レコードを自動的にマージしません。データスチュワードが一致レコードを確認してからマージジョブを開始できます。

また、あいまい一致、完全一致、またはフィルター一致のルールを設定するかどうかを決める必要もあります。あいまい一致ルールは、不整合がある可能性を持つデータに必要です。データ品質が高い場合は、完全一致ルールを設定できます。大規模なバッチジョブを実行し、同時に最適なパフォーマンスを確保するため、フィルター一致ルールを設定することができます。フィルター一致ルールは、完全一致カラムに加えてあいまい一致キーを使用する完全一致ルールです。

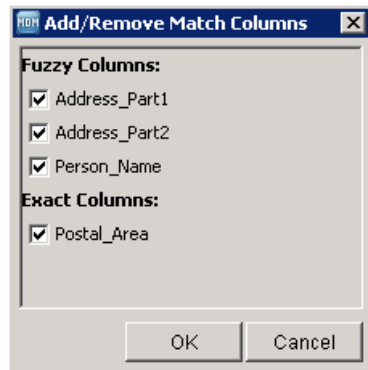
住所で個人を照合するには、名前のコンポーネントと住所のコンポーネントを持つ Resident 一致目的を使用します。

一致ルールの追加

名前とアドレスに基づいて重複する個人名を一致させるには、[Resident] 一致目的のあいまい一致ルールを WS 一致ルールセットに追加します。WS 一致ルールセットは以前の手順で作成しています。

1. Party ベースオブジェクトの **【一致/マージ設定の詳細】** ページで、**【一致ルールセット】** タブをクリックします。
2. 一致レベルが **【標準】** の手動マージ一致ルールを追加します。
 - a. **【一致ルール】** セクションで **【追加】** をクリックします。
【一致ルールの編集】 ダイアログボックスが表示されます。
 - b. **【一致/検索ストラテジ】** リストから **【あいまい】** を選択します。
あいまい一致/検索ストラテジとは、スペルの差異、誤り、入れ替えを考慮したレコードを含む確率的な一致ストラテジです。
 - c. **【一致カラム】** フィールドの横にある **【編集】** をクリックします。
【一致カラムの追加/削除】 ダイアログボックスが表示されます。
 - d. 以下のカラムを選択します。
 - Address_Part1
 - Address_Part2
 - Person_Name
 - Postal_Area

以下の図に、一致カラムを選択した状態の【一致カラムの追加/削除】ダイアログボックスを示します。



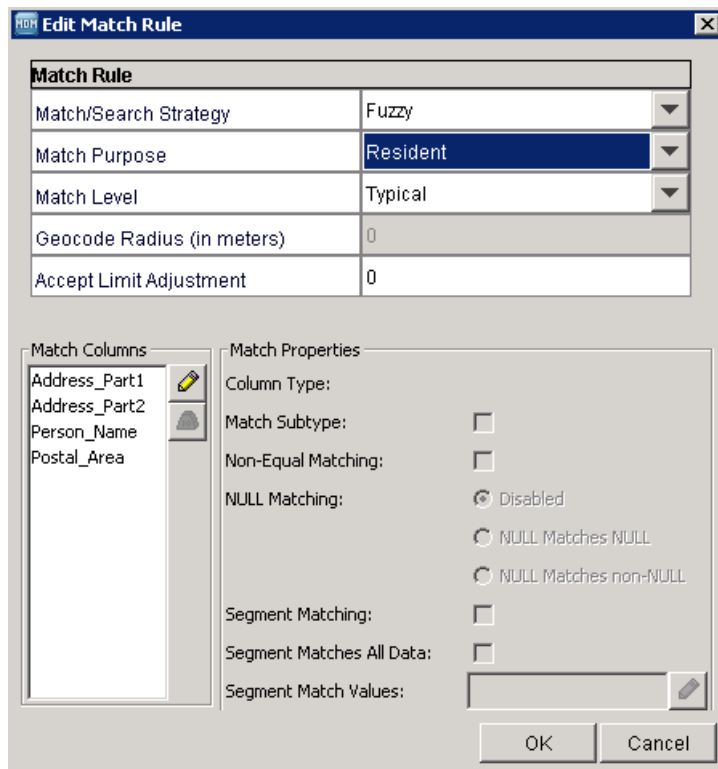
- e. **[OK]** をクリックします。

一致カラム名が【一致カラム】フィールドに表示されます。

- f. アドレスに基づいて個人名の一致を特定するには、【一致目的】リストから【住居】を選択します。
g. レコード間の一致精度を定義するには、【一致レベル】フィールドから【標準】を選択します。

【標準】一致レベルは、個人名を一致させるほとんどの一致ジョブに最適です。【保守的】一致レベルを使用すると、【標準】一致レベルよりも結果は少なくなり、一致フラグが表示されることなく、一部の一致候補が一致プロセスを通り抜ける可能性があります。【ルーズ】一致レベルを使用すると、【標準】一致レベルよりも結果は大幅に増え、正しくない一致も含まれる可能性があります。

以下の図に、あいまい一致ルールを設定した状態の【一致ルールの編集】ダイアログボックスを示します。



- h. **【OK】** をクリックします。
- 一致目的を **【住居】** に設定した手動マージ一致ルールが、**【一致ルール】** セクションに表示されます。
3. 一致レベルを **【ルールズ】** に設定した手動マージ一致ルールを追加します。
- a. **【一致ルール】** セクションで **【追加】** をクリックします。
- 【一致ルールの編集】** ダイアログボックスが表示されます。
- b. **【一致/検索ストラテジ】** リストから **【あいまい】** を選択します。
- あいまい一致/検索ストラテジとは、スペルの差異、誤り、入れ替えを考慮したレコードを含む確率的な一致です。
- c. **【一致カラム】** フィールドの横にある **【編集】** をクリックします。
- 【一致カラムの追加/削除】** ダイアログボックスが表示されます。
- d. **【あいまいカラム】** オプションから以下のあいまいカラムを選択します。
- Address_Part1
 - Person_Name
- e. **【OK】** をクリックします。
- あいまい一致カラムが **【一致カラム】** フィールドに表示されます。
- f. アドレスに基づいて個人名の一致を特定するには、**【一致目的】** リストから **【住居】** を選択します。
- g. レコード間の一致精度を定義するには、**【一致レベル】** フィールドから **【ルールズ】** を選択します。
- h. **【OK】** をクリックします。
- 一致目的を **【住居】** に設定した手動マージ一致ルールが、**【一致ルール】** セクションに表示されます。

以下の図では、[一致ルールセット] タブに 2 つのあいまい一致が表示され、WS 一致ルールセットに手動マージ一致ルールが設定されています。

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	<input type="checkbox"/>
Enable Filtering	<input type="checkbox"/>
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)

手順 8.一致ルールのマージオプションの設定

一致プロセス時に、一致レコードを自動マージまたは手動マージのキューに入れるかを一致ルールによって決定する必要があります。一致ルールの手動マージと自動マージは切り替えることができます。一致プロセス後にレコードを自動マージするには、一致ルールの一致レベルを [標準] に設定します。

Resident 一致目的と [標準] 一致レベルによって一致ルールの精度を高めることで、自動マージに役立てます。

一致ルールを自動マージ一致ルールとして設定

一致プロセス中に一致したレコードを自動マージのキューに入れるには、一致ルールを自動マージ一致ルールとして設定します。

1. 自動マージ一致ルールに変更する、一致レベルが標準の手動マージ一致ルールを選択します。
2. [上に移動] をクリックします。

手動マージ一致ルールが自動マージ一致ルールに変わります。

次の画像は、WS 一致ルールセットに設定されている自動マージ一致ルールと手動マージ一致ルールが表示された【一致ルールセット】タブを示しています。

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. 【保存】をクリックします。
【ルールセットの評価】ダイアログボックスが表示されます。
4. 【OK】をクリックします。
作成した一致ルールが保存されます。

手順 9. 一致プロパティの確認

一致ルールを設定を保存したら、一致プロパティを確認します。一致プロパティは、一致とマージ設定をまとめたものです。

一致とマージプロパティには、次の詳細が示されている必要があります。

- 設定済みの一致カラム数
- 設定済みの一致ルールセット数
- アクティブな一致ルールセット内の一致ルールの数
- 手動統合の最大一致数
- 一致バッチジョブサイクルごとの行数
- 一致/検索ストラテジのタイプ

- あいまいポピュレーションの名前

一致プロパティの確認

一致ルール設定の概要のため、一致プロパティを確認します。

- ▶ Party ベースオブジェクトの【一致/マージ設定の詳細】ページで、【プロパティ】タブをクリックします。
【グループの一致/マージ設定の詳細】ビューが表示されます。
以下の表に一致とマージ設定の概要を示します。

プロパティ	値
一致カラム	4
一致ルールセット	1
アクティブなセットの一致ルール	2
プライマリキーの一致ルール	0
手動統合の最大一致数	1000
一致ジョブバッチサイクルあたりの行数	10
一致しないすべての行を一意とする	いいえ
一致/検索ストラテジ	あいまい
あいまいポピュレーション	US
以前の行 ID オブジェクトのみ一致させる	無効
1 回だけ一致させる	無効
動的一致分析しきい値 (0=無効)	0

以下の図に、【一致/マージ設定の詳細】 ページの【プロパティ】 タブを示します。

Match/Merge Setup Details	
Match Rule Sets	Primary key match rules
Properties	Paths
Match Key Distribution	Merge Settings
Match Columns	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 checked="" type="checkbox"/>
Dynamic Match Analysis Threshold (0=disa...	0

手順 10.一致ルールへのテスト

一致ルールをテストするには、一致ジョブを実行し、結果を確認します。

より大きなデータセットを代表するサンプルデータで一致ルールをテストできます。一致ルールをテストするには、データセットを代表するサンプルで一致ジョブを実行します。一致ジョブが完了したら、照合結果を確認します。

照合結果は、Party ベースオブジェクトに関連付けられている C_PARTY_MTCB 照合テーブルで確認できます。また、Hub コンソールのマージマネージャおよび Informatica Data Director の一致候補オプションでも一致結果を確認できます。一致したレコードを確認してその正確性を検証します。

C_PARTY_MTCB 照合テーブルには、サンプルの一致結果を含む重要なカラムが示されます。

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

ROWID_OBJECT カラムには、ROWID_OBJECT_MATCHED カラムの行 ID を持つレコードと一致したレコードの行 ID が含まれています。

ROWID_MATCH_RULE カラムには、作成する一致ルールの行 ID が含まれています。一致ルールは、リポトリ内の C_REPOS_MATCH_RULE テーブルにあります。SVR1.JJ4J は、住居照合目的で一致レベルがルーズの一致ルールの行 ID です。SVR1.JJ4E は、住居照合目的で一致レベルが標準の一致ルールの行 ID です。

例は、行 ID 1191 の William De Haan のレコードに 2 つの一致があったことを示しています。このレコードは、行 ID 1019 を持つ個人 Will R De Haan と、行 ID 1419 を持つ個人 Bill Roger De Haan と一致しています。一致は名前カラムと住所カラムに基づいており、似ているもののマージの前にデータスチュワードによる確認を必要としています。William De Haan のレコードの 2 つの一致はどちらも手動マージのキューに入れられています。

照合テーブルには、行 ID 1154 の Rachel Arsen のレコードが行 ID 1106 の Rachel Arsen の別のレコードと一致していることが示されています。Rachel Arsen のレコードは、どちらも Address ベースオブジェクト内にある住所が似ているため、問題なく自動マージのキューに入れることができます。

次のテーブルは、個人のレコードが他の同様のレコードと一致した個人の名前をいくつか示しています。

表示名	一致した表示名
WILLIAM DE HAAN	WILL R DE HAAN
WILLIAM DE HAAN	BILL ROGER DE HAAN
RACHEL ARSEN	RACHEL ARSEN
AHMED RAUF	AHMED RAUF

この例の一致結果は正しいようです。より大きな一致ジョブのために設定した一致ルールを使用できます。

注: 一致ジョブの最中に発生する可能性のある問題を確認するには、次のディレクトリにある cmxserver.log ファイルを確認します。

UNIX の場合:<infamdm_install_directory>/hub/server/logs

Windows の場合:<infamdm_install_directory>\hub\server\logs

[一致ジョブバッチサイクルあたりの行数] などの一致プロパティにより、タイムアウトが発生する場合があります。プロパティは、**[一致/マージ設定]** ページの **[プロパティ]** タブで変更できます。

一致ジョブの進捗を確認し、サマリ統計を取得するには、プロセスサーバーログを確認します。

プロセスサーバーログ cmxserver.log は、次のディレクトリにあります。

UNIX の場合:<プロセスサーバーのインストールディレクトリ>/hub/cleanse/logs

Windows の場合:<プロセスサーバーのインストールディレクトリ>\hub\cleanse\logs

第 23 章

Elasticsearch による検索

この章では、以下の項目について説明します。

- [Elasticsearch による検索の概要, 468 ページ](#)
- [Elasticsearch アーキテクチャによる検索, 468 ページ](#)
- [検索のインストールおよび設定, 469 ページ](#)
- [手順 1. Elasticsearch のインストールとセットアップ, 470 ページ](#)
- [手順 2。検索のための MDM Hub プロパティの設定, 475 ページ](#)
- [手順 3。プロビジョニングツールを使用した検索の設定, 479 ページ](#)
- [手順 4. オペレーショナル参照ストアの検証, 490 ページ](#)
- [手順 5。検索データのインデックス処理, 490 ページ](#)
- [キーストア、トラストストア、および証明書の作成（オプション）, 491 ページ](#)

Elasticsearch による検索の概要

Data Director アプリケーションまたはカスタムアプリケーションを使用して、特定のビジネスエンティティ内でデータを検索することができます。MDM はオープンソースの全文検索エンジンである Elasticsearch を使用します。

注: Elasticsearch による検索は、サポートされなくなった Solr による検索に代わるものです。

Elasticsearch をインストールしてセットアップする必要があります。分散インデックス処理および検索を実行するために、Elasticsearch を単一ノードクラスタまたはマルチノードクラスタとしてセットアップしてください。

データ検索が実行される前に、ビジネスユーザーおよびデータスチュワードは Hub サーバーおよびプロセスサーバーを設定してからデータのインデックス処理を行う必要があります。データをインデックス処理する際、MDM Hub ではレコードを処理し、インデックス付きレコードを Elasticsearch サーバーに追加します。1 つのノードに対してデータが大きい場合は、複数のノードを使用し、インデックスを複数のシャードに分割します。

Elasticsearch アーキテクチャによる検索

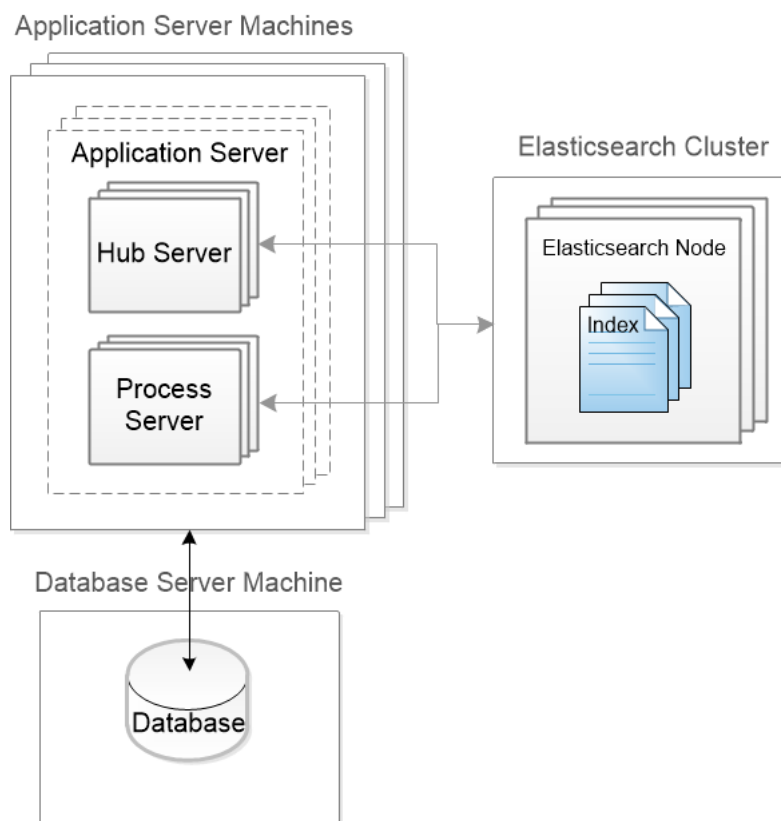
MDM Hub では検索にサーバーおよびプロセスサーバーを使用します。Elasticsearch クライアントは、インデックス処理をリアルタイムで実行するために、Hub サーバーに埋め込まれています。プロセスサーバーは、初期データロード後は、初期インデックス処理のみを実行します。すべての MDM Hub コンポーネントのインス

ツール先が単一のマシンか複数のマシンかどうかにかかわらず、検索用のすべての Hub サーバーインスタンスおよびプロセスサーバーインスタンスを設定する必要があります。

Elasticsearch は、MDM Hub コンポーネントがインストールされているマシンにインストールすることも、別のマシンにインストールすることもできます。Elasticsearch には、Java 仮想マシン (JVM) の排他的にサポートされている (MDM Hub コンポーネントで共有しない) バージョンが必要です。

MDM Hub コンポーネントを Elasticsearch で検索できるように設定するには、Elasticsearch クラスターをセットアップします。クラスターとは、1 つ以上のノードのコレクションです。ノードはそれぞれ単一のサーバーで、データを格納してインデックス処理や検索の操作に加わります。MDM Hub トポロジと、インデックスを作成するデータの量に基づき、クラスターに設定するノードは 1 つになることも複数になることもあります。各ノードに複数のインデックスを設定できます。データが失われないようにし、クラスターの安定性を維持するには、マルチノードクラスター向けに Zen Discovery を設定できます。

次の図に、インストールトポロジを検索用に設定した例を示します。



検索のインストールおよび設定

検索を使用するには、Elasticsearch で MDM Hub を設定します。

1. Elasticsearch をインストールしてセットアップします。
2. 検索用の MDM Hub のプロパティを設定します。
3. プロビジョニングツールを使用して検索を設定します。
4. オペレーショナルリファレンスストア (ORS) を検証します。
5. 検索データをインデックス処理します。

手順 1.Elasticsearch のインストールとセットアップ

検索を設定するには、Elasticsearch をインストールしてセットアップする必要があります。

Elasticsearch をセットアップするには、次のタスクを実行します。

1. インストール前作業を完了します。
2. Elasticsearch をインストールします。
3. Elasticsearch Java 仮想マシン (JVM) を設定します。
4. Elasticsearch プロパティファイルを設定します。
5. Elasticsearch を保護します。
6. 分析プラグインをインストールします。
7. ストップワード、シノニム、文字マッピングの設定
8. Elasticsearch を起動します。

インストール前のタスクの完了

Elasticsearch クラスタをインストールおよび設定する前に、環境を準備し、高可用性を設定するかどうかを決定します。

すべての環境でのタスク

次のタスクを実行して、インストール環境を準備します。

- 各マシンが Elasticsearch のサポートしているバージョンのハードウェア要件を満たしていることを確認します。ハードウェアの詳細については、Elasticsearch のマニュアルを参照してください。
- 各マシンが Elasticsearch のサポートしているバージョンのソフトウェア要件（サポートしているオペレーティングシステムおよび Java バージョンなど）を満たしていることを確認します。ソフトウェア要件の詳細については *Elasticsearch のサポートマトリックス* を参照してください。
- スワップ、ファイル記述子および仮想メモリなどの重要なシステム構成を完了します。重要なシステム構成の詳細については、Elasticsearch のマニュアルを参照してください。

UNIX 環境でのタスク

UNIX 環境では、次のタスクを実行します。

- ファイル記述子の数の不足によるデータ損失を避けるために、ファイル記述子の数は 65536 以上に設定します。
- メモリのスワップを防止するには、スワップを防止するようにシステムを設定します。mlockall を使用して、メモリ内のヒープをロックするように Java 仮想マシン (JVM) を設定できます。

高可用性の要件

インデックス処理および検索するデータが大量にある場合、可用性の高い Elasticsearch クラスタを実装することをお勧めします。高可用性クラスタは複数のノードを持ち、ノード間で負荷を分散できます。プロダクション環境で 1 つのノードに障害が発生した場合、クラスタは別のノードに負荷を分散します。

インストール前のタスクとして、可用性の高い Elasticsearch クラスタを実装するかどうかを決定します。実装する場合、Elasticsearch クラスタを通常どおりに構成しますが、次の追加の要件を満たすことを確認します。

- Elasticsearch クラスタ内に 3 つ以上のノードがある。

ヒント: 最初は小さなクラスタを設定して、必要に応じて拡大縮小することができます。負荷を分析し、ノードの障害を処理するための十分な容量があることを確認します。

- 各ノードは個別の専用マシンに構成される。
- 安定性とパフォーマンスを確保するために、少なくとも3つのノードをマスタノードにする。
Elasticsearch では、奇数のマスタノードが推奨されることに注意してください。
 - クラスタ内のノードが3つのみである場合、すべてのノードをマスタノードとして構成します。
 - クラスタ内に3つよりも多くのノードがある場合、3ノードをマスタノードとして構成し、残りのノードをデータノードとして構成します。
- Elasticsearch クラスタサイズに基づき、レプリカの数を決する。プロビジョニングツールを使用して Elasticsearch インデックスを構成する場合、使用するレプリカ数を指定できます。
- ノードごとに、`elasticsearch.yml` 構成ファイルに次の追加のプロパティを設定します。
 - `discovery.zen.minimum_master_nodes`
 - `discovery.zen.ping.unicast.hosts`

ハードウェア要件、システム構成、およびプロパティ値を含む、高可用性クラスタの詳細については、Elasticsearch のマニュアルを参照してください。

Elasticsearch のインストール

Hub サーバーとプロセスサーバーのインストール後、検索を設定するには、Elasticsearch クラスタをインストールしてセットアップします。

Elasticsearch インストールにサポートされているオペレーティングシステムと Java バージョンを使用していることを確認します。詳細については、Elasticsearch のサポートマトリックスを参照してください。

Elasticsearch をインストールし、クラスタをセットアップする方法の詳細については、Elasticsearch のドキュメントを参照してください。

1. Elastic Web サイトから、サポートされているバージョンの Elasticsearch アーカイブファイルをダウンロードします。
サポートされているバージョンの詳細については、Product Availability Matrix (PAM) を参照してください。PAM には <https://network.informatica.com/community/informatica-network/product-availability-matrices> からアクセスできます。
2. Elasticsearch アーカイブファイルを抽出します。

Elasticsearch Java 仮想マシン (JVM) の設定

マシン上で使用可能な RAM の量に基づいてヒープサイズを使用するように、Elasticsearch Java 仮想マシン (JVM) を設定します。JVM を設定するには、`jvm.options` ファイルを編集します。

1. 次のディレクトリにある `jvm.options` ファイルを検索します。
`<elasticsearch installation directory>/config`

2. テキストエディタを使用してファイルを開き、以下のプロパティを編集します。

プロパティ	説明
-Xms	最小ヒープサイズ。デフォルトは 1 GB です。
-Xmx	最大ヒープサイズ。デフォルトは 1 GB です。
-XX:HeapDumpPath	ヒープダンプパス。デフォルトは/var/lib/elasticsearch です。マルチクラスタ環境では、このプロパティを別のパスに設定する必要があります。

注: 最小ヒープサイズ (Xms) と最大ヒープサイズ (Xmx) を同じ値に設定します。その他のプロパティについては、デフォルト設定を使用します。

Elasticsearch プロパティファイルの設定

Informatica は、サンプルの Elasticsearch プロパティファイルを提供しています。Elasticsearch を設定するには、そのプロパティファイルを編集します。

1. 次のディレクトリにある elasticsearch.yml ファイルを検索します。
`<elasticsearch installation directory>/config`
2. テキストエディタを使用してファイルを開き、以下のプロパティを編集します。

プロパティ	説明
bootstrap.memory_lock	メモリのロックを設定します。Elasticsearch メモリがスワップされて消されることがないようにするには、true を設定します。デフォルトは true です。
cluster.name	Elasticsearch クラスタに一意の名前を指定します。複数のクラスタがある場合、各クラスタの名前が一意であることを確認します。クラスタに複数のノードがある場合、そのクラスタ内の各ノードで、同じクラスタ名が指定されていることを確認します。
discovery.zen.minimum_master_nodes	データの損失を防ぎ、クラスタの安定性を維持するために、複数ノードクラスタに必要です。値を(マスタに適格なノード数 / 2) + 1 に設定します。 例えば、クラスタに 3 つのノードがあり、それらすべてがマスタに適格なノードであり、データを格納できる場合、このプロパティを $(3 / 2) + 1$ に設定します。この値は、2 に丸められます。
discovery.zen.ping.unicast.hosts	複数ノードのクラスタに必要です。このプロパティは、検出設定の指定に使用されます。この設定は、クラスタ内のノードの IP アドレスとポートポートのリストです。このプロパティを設定するには、 <code>["host1:port1","host2:port2","host3:port3"]</code> の形式を使用します。
http.port	HTTP 要求のポートです。デフォルトは 9200 です。

プロパティ	説明
network.host	ホストの IP アドレスは、バインドアドレスとして使用されます。
node.data	CRUD や検索などのデータ関連操作を実行するためのデータノードとして、ノードを有効にします。デフォルトは true です。
node.ingest	インデックス処理の前に、データを変換してエンリッチ化する取り込みデータノードとして、ノードを有効にします。デフォルトは true です。
node.master	クラスタを制御するマスタノードとして、ノードを有効にします。クラスタに複数のノードがある場合、1 つ以上のノードをマスタノードとして有効にします。高可用性のためには、複数のノードをマスタノードとして設定します。デフォルトは true です。
node.name	ノードの一意の名前を指定します。
path.data	データを保存するディレクトリのパスを指定します。複数のデータディレクトリを設定できます。複数のデータディレクトリの設定に関する詳細については、Elasticsearch のマニュアルを参照してください。
path.logs	ログファイルのパスを指定します。
transport.tcp.port	TCP バインドポートを指定します。デフォルトは 9300 です。

3. 同じ名前 (elasticsearch.yml) でプロパティファイルを保存します。

Elasticsearch の保護 (オプション)

Elasticsearch のインストール後、MDM Hub と Elasticsearch 間の通信を保護します。また、Elasticsearch クラスタも保護します。

Elasticsearch の保護の詳細については、Elasticsearch のセキュリティに関するドキュメントを参照してください。

分析プラグインのインストール

音標および日本語 (kuromoji) 分析プラグインをインストールします。これは、新しいアナライザ、トークナイザ、トークンフィルタおよび文字フィルタを追加することで Elasticsearch を拡張します。音標分析プラグインはトークンを分析し、それに相当する音標に変換します。日本語 (kuromoji) 分析プラグインは Kuromoji アナライザを使用することで日本語を分析します。

1. 音標および日本語 (kuromoji) 分析プラグインを Elastic Web サイトからダウンロードします。
2. 次のコマンドを実行することで、各クラスタに音標分析プラグインをインストールします。

```
sudo bin/elasticsearch-plugin install analysis-phonetic
```
3. 次のコマンドを実行することで、各クラスタに日本語 (kuromoji) 分析プラグインをインストールします。

```
sudo bin/elasticsearch-plugin install analysis-kuromoji
```
4. 各クラスタノードを再起動します。

ストップワード、シノニム、文字マッピングの設定

検索の実行時に、MDM では"and"、"an"、"is"などの一般的な語を無視できます。また、検索文字列のシノニムを検索することもできます。例えば、"William"を検索するときに、検索結果に"Will"や"Willy"などのシノニムを含めることができます。

一般的な語を無視したり、検索結果にシノニムを含めたりするために、Informatica ではストップワードやシノニムを含むテキストファイルを用意しています。また、ユーザーが独自にファイルを設定することもできます。

中国語、日本語、韓国語などの言語でデフォルトの Elasticsearch アナライザを使用するために、mapping-FoldToASCII.txt というマッピングファイルも提供しています。デフォルト アナライザの文字フィルタは、マッピングファイルを使用して Unicode の基本ラテン文字ブロックに含まれていない英字、数字、シンボリック文字を対応する ASCII 文字に変換します。

stopwords.txt、synonyms.txt、stopwords_ja.txt、および mapping-FoldToASCII.txt ファイルを必要とされる場合は、Informatica グローバルカスタマサポートにお問い合わせください。

ストップワード、シノニム、文字マッピングを設定するには、次の手順を実行します。

1. 次の場所に analysis ディレクトリを作成します。
`<elasticsearch installation directory>/config`
2. stopwords.txt ファイルと synonyms.txt ファイルを analysis ディレクトリにコピーします。
3. 日本語など他言語のストップワードを設定するには、次の場所に lang ディレクトリを作成します。
`<elasticsearch installation directory>/config/analysis`
4. stopwords_ja.txt などの他言語のストップワードファイル、および mapping-FoldToASCII.txt を lang ディレクトリにコピーします。

検索用ストップワードリストのカスタマイズ

検索で無視される単語や句のリストをカスタマイズするには、stopwords.txt ファイルを編集します。

1. テキストエディタを使用して、以下の場所にある stopwords.txt ファイルを開きます。
`<elasticsearch installation directory>/config/analysis`
2. stopwords.txt ファイルを編集して保存します。
3. stopwords.txt ファイルの編集前にデータがインデックス処理されていた場合、インデックスを手動で削除し、Elasticsearch を再起動し、データを再度インデックス処理します。

stopwords.txt ファイルの更新の詳細については、Elasticsearch のマニュアルを参照してください。

検索に含めるシノニムのリストのカスタマイズ

検索に使用するシノニムをカスタマイズするには、synonyms.txt ファイルを編集します。

1. テキストエディタを使用して、以下の場所にある synonyms.txt ファイルを開きます。
`<elasticsearch installation directory>/config/analysis`
2. synonyms.txt ファイルを編集して保存します。
3. synonyms.txt ファイルの編集前にデータがインデックス処理されていた場合、インデックスを手動で削除し、Elasticsearch を再起動し、データを再度インデックス処理します。

synonyms.txt ファイルの更新の詳細については、Elasticsearch のマニュアルを参照してください。

Elasticsearch の起動

Elasticsearch のセットアップ後、変更内容が有効になるように、Elasticsearch クラスタ内の各ノードを起動します。

ヒント: Elasticsearch を起動するときに、メモリのロックの問題が発生する場合、soft memlock unlimited と hard memlock unlimited を設定する必要がある可能性があります。

1. コマンドプロンプトを開き、次のディレクトリに移動します。

<elasticsearch installation directory>/bin

2. 次のコマンドを実行します。

UNIX の場合: elasticsearch.sh

Windows の場合: elasticsearch.bat

アップグレード Elasticsearch

アップグレードされた Elasticsearch ノードを追加した後、インデックスの詳細で新しいノードを更新するかどうかを指定します。次に、スマート検索データの初期インデックス処理バッチジョブを手動で実行して、アップグレードされたノードにインデックスを追加します。既存のノードによって引き続き検索要求が管理されます。バッチジョブが正常に実行され、ノードが更新されると、新しいノードが検索要求を管理できるようになります。

注: エラスティック検索バージョン 6 からバージョン 7 にアップグレードする場合に適用されます。エラスティック検索認証サポートは、バージョン 7.x でのみ有効です。

Elasticsearch ノードの設定の詳細については、[「Elasticsearch クラスタの設定」 \(ページ 479\)](#)を参照してください。

手順 2。検索のための MDM Hub プロパティの設定

MDM Hub プロパティを設定するには、Hub コンソール、プロセスサーバーのプロパティファイル、Hub サーバーのプロパティファイルを使用します。

1. プロセスサーバーのプロパティを設定します。
2. Hub サーバーのプロパティを設定します。

Hub サーバーのプロパティの設定

すべての Hub サーバーインスタンスで検索を有効化するように設定する必要があります。Hub コンソールの Hub サーバーツールおよび cmxserver.properties ファイルを使用して、検索用の Hub サーバーのプロパティを設定します。

1. テキストエディタを使用して、<MDM Hub インストールディレクトリ>\hub\server\resources\cmxserver.properties にある cmxserver.properties ファイルを開きます。
2. 検索用の次のプロパティを設定します。

cmx.ss.enabled

検索を有効にするかどうかを示します。新規インストールでは、デフォルトは true です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは false です。

ex.max.conn.per.host

ホストに接続する Elasticsearch ノードの最大数を設定します。ホスト上の Elasticsearch クラスタノードの数に設定します。

ex.max.threads

Apache Elasticsearch クラスタでノードごとに非同期ノンブロッキングレシーバを使用するスレッドの最大数を設定します。デフォルトは 1 です。
この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

es.index.refresh.interval

「スマート検索データの初期インデックス処理」バッチジョブが実行された後に Elasticsearch がデータへの変更をコミットする間隔（秒）を設定します。そのデータを検索できるようになるのは、この時間間隔の後です。デフォルトは 30 です。
このプロパティは、初回のインデックス処理でインデックス処理量が大きくなった場合に影響します。この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

cmx.e360.view.enabled

MDM 管理者がエンティティ 360 フレームワークを実装した場合、IDD ユーザーは **【検索】** ボックスを使用して、レコードを編集および管理するためのレコードやエンティティタブを探します。新規インストールでは、デフォルトは true です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは false です。

search.provisioning.numshards

オプション。ご使用の Elasticsearch 環境で作成するシャードの数。値はシャードの最大数およびノードの合計数により異なります。例えば、シャードの最大数が 1 で、ノードの数が 3 の場合は、3 つのシャードを作成することができます。デフォルトは Hub サーバーの合計数です。

search.provisioning.numreplicas

オプション。別々のノードで作成する Elasticsearch エンジンドキュメントの部数。異なるノードのシャードにドキュメントの複数のコピーを作成するには、レプリケーション要素を使用します。1 つ以上のノードが予期せずシャットダウンした場合の高可用性を実現するには、ドキュメントの複数のコピーが必要です。例えば、レプリケーション要素が 2 の場合、2 つのノードにあるドキュメントの 2 つのコピーを取得します。Elasticsearch のデフォルトは 0 です。

cmx.task.search.records.return

ユーザーがビジネスエンティティを使用して Data Director のタスクマネージャでタスクを検索するときの Elasticsearch ページネーションを制御します。デフォルトは 1000 です。
この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

cmx.server.batch.smartsearch.initial.block_size

スマート検索データの初期インデックス処理のバッチジョブが各ブロックで処理できる最大レコード数。デフォルトは 250 です。大きなデータセットをインデックス処理する場合は、レコード数を増やします。推奨最大値は 1000 です。

cmx.server.enrichcopager.thread_pool

プロパティを手動で追加します。ReadCO 操作を同時に実行できるように、スレッドプールから EnrichCoPager プロパティが使用するスレッドの数を設定します。デフォルトは 30 です。スレッドの数を 1 に設定すると、プロパティは無効になります。

cmx.server.enrichcopager.min_rec_for_multithreading

EnrichCoPager プロパティがマルチスレッドを使用するまでに返すレコードの最小数を設定します。デフォルトは 2 です。

ssl.keyStore

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルの絶対パスおよびファイル名。

ssl.keyStore.password

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルのプレーンテキストパスワード。

ssl.trustStore

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルの絶対パスおよびファイル名。

ssl.trustStore.password

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルのプレーンテキストパスワード。

3. 次のプロパティを設定して、ビジネスエンティティの検索結果を CSV ファイルにエクスポートするときのデフォルトの区切り記号をカスタマイズします。

cmx.server.exportSeparator=

オプション。プロパティを追加して、エクスポートされた検索結果が含まれる CSV ファイル内のデフォルトの区切り文字を置き換える区切り文字を指定します。デフォルトはカンマです。サポートされている区切り文字は、`, \ ; | : " ' ~ ^ & #` です。

Hub サーバーのプロパティを更新したら、オペレーショナルリファレンスストア（ORS）を検証して、Hub コンソールを再起動する必要があります。

プロセスサーバーのプロパティの設定

すべてのプロセスサーバーインスタンスで検索を有効化するように設定します。Hub コンソールのプロセスサーバーツールおよび `cmxcleanse.properties` ファイルを使用して、検索用のプロセスサーバーのプロパティを設定します。

1. ノードの Hub コンソールでプロセスサーバーツールを起動します。
2. **書き込みロック** > **ロックの取得** の順にクリックします。
3. プロセスサーバーツールの右側のペインで、**プロセスサーバーの追加** ボタンをクリックします。
プロセスサーバーの追加/編集 ダイアログボックスが表示されます。

4. 検索用のプロセスサーバーの次のプロパティを設定します。

プロパティ	説明
サーバー	この Process サーバーをデプロイしたアプリケーションサーバーの IP アドレスまたは完全修飾ホスト名。 注: localhost をホスト名として使用しないでください。
ポート	この Process サーバーをデプロイしたアプリケーションサーバーの HTTP または HTTPS ポート。
Elasticsearch 処理	この Process サーバーがバッチジョブ「スマート検索データの初期インデックス処理」を実行するかどうかを示します。このバッチジョブは、あるビジネスエンティティについて、検索可能なフィールドのすべての値のインデックスを作成します。
保護された接続 (HTTPS)	この Process サーバーが HTTPS プロトコルを使用するかどうかを示します。選択した場合、[ポート] オプションが HTTPS ポート番号に設定されていることを確認します。

5. **[OK]** をクリックしてから **[保存]** をクリックします。
 6. テキストエディタを使用して、次の場所にある `cmxcleanse.properties` ファイルを開きます。

<MDM Hub インストールディレクトリ>\hub\cleanse\resources

7. 検索用の次のプロパティを設定します。

`cmx.ss.enabled`

検索の実行にプロセスサーバーを使用するかどうかを示します。デフォルトは `false` です。検索の実行にプロセスサーバーを使用する場合は、`true` に設定します。

`ex.max.conn.per.host`

ホストに接続する Elasticsearch ノードの最大数を設定します。ホスト上の Elasticsearch クラスタノードの数に設定します。

`ex.max.threads`

Apache Elasticsearch クラスタでノードごとに非同期ノンブロッキングレシーバを使用するスレッドの最大数を設定します。デフォルトは 1 です。
この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

`search.provisioning.numreplicas`

オプション。別々のノードで作成する Elasticsearch エンジンドキュメントの部数。異なるノードのシャードにドキュメントの複数のコピーを作成するには、レプリケーション要素を使用します。1 つ以上のノードが予期せずシャットダウンした場合の高可用性を実現するには、ドキュメントの複数のコピーが必要です。例えば、レプリケーション要素が 2 の場合、2 つのノードにあるドキュメントの 2 つのコピーを取得します。Elasticsearch のデフォルトは 0 です。

`MAX_INITIAL_RESULT_SIZE_TO_CONSIDER`

オプション。プロパティを手動で追加します。Data Director アプリケーションで表示する検索結果の合計数。推奨最大値は 250 です。デフォルトは 130 です。130 を超える値に設定すると、Data Director アプリケーションのパフォーマンスに影響を与えます。

ssl.keyStore

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルの絶対パスおよびファイル名。

ssl.keyStore.password

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルのプレーンテキストパスワード。

ssl.trustStore

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルの絶対パスおよびファイル名。

ssl.trustStore.password

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルのプレーンテキストパスワード。

cmx.websphere.security.ssl.config.url

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。WebSphere 環境の場合のみ。プロパティを手動で追加します。ssl.client.props ファイルの絶対パス (ファイル名を含む)。

8. cmxcleanse.properties ファイルを保存します。
9. アプリケーションサーバーを再起動します。

手順 3。プロビジョニングツールを使用した検索の設定

Elasticsearch をセットアップして MDM Hub のプロパティを設定したら、プロビジョニングツールを使用して検索環境を設定します。

1. Elasticsearch クラスタを設定します。
2. 必要に応じて、Elasticsearch のカスタムインデックス設定を作成します。
3. 検索可能なフィールドを設定します。
4. 検索およびクエリ結果の表示を設定します。
5. 必要に応じて、類似レコードが表示されるようにレイアウトを設定します。

Elasticsearch クラスタの設定

プロビジョニングツールを使用して、MDM アプリケーション向けに Elasticsearch クラスタを設定します。検索 API でこの設定が使用されます。Data Director アプリケーションと任意のカスタムアプリケーションが、この検索 API を使用します。

注: Elasticsearch クラスタを設定するときは、クラスタのマスタノードのみを指定する必要があります。

1. サポートされているブラウザを開いて、次の URL を入力します。
`https://<MDM Hub Server host name>:<MDM Hub Server port number>/provisioning/`
[ログイン] ページが表示されます。
2. ユーザー名とパスワードを入力し、[ログイン] をクリックします。

3. **【データベース】** リストから、Elasticsearch クラスタを設定するデータベースを選択します。
4. **【設定】 > 【インフラストラクチャの設定】** をクリックします。
【インフラストラクチャの設定】 ページが表示されます。
5. リストから **【Elasticsearch クラスタ】** を選択してから **【ESCluster】** をクリックします。
ツリービューのパネルに **【ESCluster】** が表示されます。
6. Elasticsearch クラスタノードを設定するには、ツリービューのパネルで **【esNode】** を選択してから **【作成】** をクリックします。
7. 設定した Elasticsearch クラスタの次のプロパティを指定します。

プロパティ	説明
名前	Elasticsearch クラスタ内のマスタノードの名前を指定します。
URL	Elasticsearch クラスタ内のマスタノードの URL を指定します。URL の形式は <code>https://<host name>:<port></code> です。
エラスティック検索ノードの更新	<p>既存のノードが引き続き検索要求を管理する場合に、インデックスの詳細でノードを更新するかどうかを示します。スマート検索データの初期インデックスバッチジョブを手動で実行し、インデックスが新しいノードに追加された後に既存のノードを削除できます。次に、チェックボックスをオフにして、新しいノードが検索要求を管理できるようにします。</p> <p>注: エラスティック検索バージョン 6 からバージョン 7 にアップグレードする場合に適用されます。エラスティック検索認証サポートは、バージョン 7.x でのみ有効です。</p> <p>エラスティック検索のインストールと設定の詳細については、『<i>Multidomain MDM 設定ガイド</i>』を参照してください。</p>

8. **【適用】** をクリックします。
9. 追加のマスタノードを作成する場合、手順 [6](#) から [8](#) を繰り返します。
10. 変更内容を MDM Hub にパブリッシュします。
 - a. **【パブリッシュ】** をクリックします。
確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。
 - b. 変更内容を確認するか、確認せずにパブリッシュします。
 - 確認せずにパブリッシュする場合は、**【パブリッシュ】** をクリックします。
 - 確認後にパブリッシュするには、**【変更の確認】** をクリックし、画面に表示される指示に従います。

エラスティック検索のカスタムインデックス設定の作成（オプション）

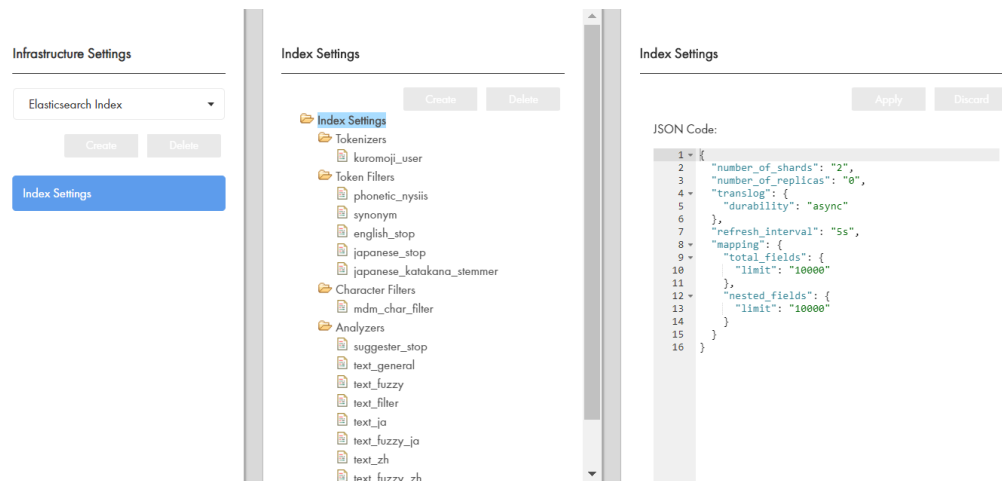
Informatica が提供するエラスティック検索インデックスの設定が要件を満たしていない場合は、カスタムインデックス設定を作成できます。カスタムインデックス設定には、アナライザを含める必要があります。アナライザは、テキストをトークンまたは用語に変換して、検索用に転置インデックスに追加します。

アナライザはトークナイザを 1 つだけ持つ必要があり、かつ 0 個以上の文字フィルタとトークンフィルタを持つことができます。トークナイザは、トークンに変換される文字のストリームを受け取ります。トークンフィルタは、トークナイザによって生成されたトークンのストリームを受け取り、トークンを追加、削除、または変更する場合があります。文字フィルタは文字のストリームを受け取り、ストリーム内の文字を追加、削除、または変更できます。

カスタムアナライザで使用するトークナイザ、トークンフィルタ、および文字フィルタは、Informatica のデフォルト、カスタム、またはエラスティック検索の組み込みコンポーネントにすることができます。デフォルト設定は編集できません。アナライザを設定すると、エラスティック検索の組み込みトークナイザとトークンフィルタを選択できます。

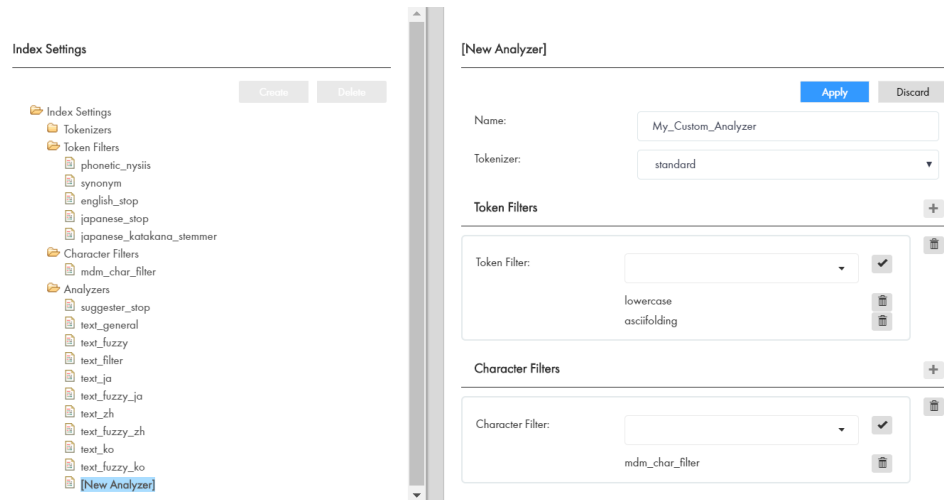
エラスティック検索インデックスの設定の詳細については、エラスティック検索のドキュメントを参照してください。

1. プロビジョニングツールにログインします。
2. **[データベース]** リストから、エラスティック検索インデックス設定を設定するデータベースを選択します。
3. **[設定]** > **[インフラストラクチャの設定]** をクリックします。
[インフラストラクチャの設定] ページが表示されます。
4. インフラストラクチャの設定リストから、**[エラスティック検索インデックス]** を選択し、**[インデックスの設定]** をクリックします。
[インデックスの設定] が **[ツリービュー]** パネルに表示され、インデックス設定の **[JSON コード]** ボックスが **[プロパティ]** パネルに表示されます。インデックス設定が変更されていない場合、ページにはデフォルト設定が表示されます。



5. **[JSON コード]** ボックスに、[分析] モジュール以外のモジュールのインデックス設定を入力します。また、シャードの数、レプリカの数、更新間隔など、特定のインデックスモジュールに関連付けられていないインデックス設定を入力します。
6. トークナイザ、トークンフィルタ、文字フィルタなどのアナライザコンポーネントを設定します。
 - a. **[ツリービュー]** パネルで、設定するコンポーネントを選択し、**[作成]** をクリックします。

- b. [プロパティ] パネルで、コンポーネントの名前と JSON コードを入力します。
 - c. [適用] をクリックします。
7. アナライザを設定します。
 - a. [ツリービュー] パネルで [アナライザ] を選択し、[作成] をクリックします。
 - b. [プロパティ] パネルで、アナライザの名前、トークナイザ、トークンフィルタ、文字フィルタを指定します。
 トークンフィルタは、アナライザで使用する順序で指定します。
 次の図は、カスタムアナライザの設定例です。



- c. [適用] をクリックします。
8. 変更内容を MDM Hub にパブリッシュします。
 - a. [パブリッシュ] をクリックします。
 確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。
 - b. 変更内容を確認するか、確認せずにパブリッシュします。
 - 確認せずにパブリッシュする場合は、[パブリッシュ] をクリックします。
 - 確認後にパブリッシュするには、[変更の確認] をクリックし、画面に表示される指示に従います。
9. アプリケーションサーバーのログで、インデックス設定に関連する検証エラーを確認し、変更を加えます。

エラスティック検索組み込みトークナイザとトークンフィルタ

カスタムアナライザで使用可能なエラスティック検索組み込みトークナイザとトークンフィルタを選択できます。

次のエラスティック検索組み込みトークナイザは、カスタムアナライザで使用できます。

- standard
- letter
- lowercase
- whitespace
- uax_url_email

- classic
- thai
- keyword

次のエラスティック検索組み込みトークンフィルタは、カスタムアナライザで使用できます。

- asciifolding
- standard
- lowercase
- uppercase
- porter_stem
- trim
- cjk_width
- cjk_bigram
- classic
- apostrophe
- kuromoji_baseform

カスタムおよび組み込みのエラスティック検索アナライザのコンポーネントの詳細については、エラスティック検索のドキュメントを参照してください。

検索可能なフィールドの設定

プロビジョニングツールを使用して、フィールドを検索可能なフィールドとして設定し、フィールドのプロパティを設定できます。検索要求では、検索可能なフィールドとして設定したフィールドのみを検索します。

複数の検索可能なフィールドを設定すると、検索要求のパフォーマンスに影響する場合があります。重要なフィールドのみを検索可能なフィールドに設定します。例えば、国番号、性別コード、または住所のタイプを含むフィールドよりも、氏名、組織名、または電子メールアドレスを含むフィールドを検索可能なフィールドに設定します。

1. プロビジョニングツールにログインします。
2. **【データベース】** リストから、フィールドを設定するデータベースを選択します。
3. **【ビジネスエンティティ】** > **【モデリング】** をクリックします。
【モデリング】 ページが表示されます。
4. リストから **【ビジネスエンティティ】** を選択し、検索可能なフィールドを設定するビジネスエンティティを選択します。
5. **【ツリービュー】** パネルで **【フィールド】** を選択し、**【作成】** をクリックします。
6. 要件に基づいて、次のプロパティを設定します。

名前

【ツリービュー】 パネルに表示するフィールドの名前。

ラベル

Data Director のビュー内に表示するフィールドのラベル。

読み取り専用

オプション。エンティティビューでフィールドを編集できるかどうかを示します。フィールドを編集不可フィールドとして設定するには、プロパティを選択します。

必須

オプション。フィールドが必須かどうかを示します。フィールドを必須フィールドとして設定するには、**【必須】**を選択します。デフォルトは、必須フィールドではありません。

URI

データ型を宣言する名前空間を定義します。設定するデータ型は、選択した URI によって異なります。

文字列、整数、ブールなどの基本的なデータ型を設定するには、`commonj.sdo` を選択します。イメージ URL、ハイパーリンク、ファイル添付などの Informatica のデータ型を設定するには、`urn:co-types.informatica.mdm` を選択します。

タイプ

フィールドのデータ型。設定できるデータ型は、選択した URI によって異なります。次のデータ型を設定できます。文字列、整数、10 進数、日付、ブール、イメージ URL、ハイパーリンク、ファイル添付。

デフォルトでは、ビジネスエンティティフィールドのデータ型は、フィールドが関連付けられているベースオブジェクトカラムのデータ型にできるだけ近いものになります。例えば、ベースオブジェクトの文字列カラムにイメージの情報が含まれている場合、ビジネスエンティティフィールドにイメージ URL データ型を設定します。フィールドに Web URL、ファイル URI、FTP リンク、および電子メールアドレスをハイパーリンクとして表示する場合は、フィールドにハイパーリンクデータ型を設定します。

注: セキュリティ上の理由により、Google Chrome および Mozilla Firefox でファイル URI を開くことはできません。

表示形式

日付フィールドの表示形式。日付フィールドの表示形式を指定する前に、URI を指定して **【タイプ】** プロパティを **【日付】** に設定します。

日付を表示するビジネスエンティティフィールドには、**【日付】** または **【日時】** 形式を選択できます。設定する表示形式は、すべての Data Director ビューの日付フィールドに適用されます。

日付フィールドに表示形式が設定されていない場合、そのフィールドの日付形式は、関連するベースオブジェクトカラムの形式と同じです。

フィルタ

静的フィルタを定義して、ユーザーがフィールドに入力可能または入力不可のデータを規定します。

【フィルタ】プロパティを有効にした場合、以下のフィールドを指定します。

- **演算子**: 指定した値がフィールドで許可されるか拒否されるかを制御します。

オプション	説明
次に含まれる	値はユーザーに表示されます。カンマ区切りの値リストを許可された値として追加すると、Data Director ユーザーはルックアップリストから値を選択できます。同じフィールドにフィールドフィルタも設定している場合、静的フィルタとフィールドフィルタの論理積に従って値が表示されます。
次に含まれない	値はユーザーに表示されません。

- **値**: フィールドのデータ型に一致する値を追加します。複数の値を追加するには、値のカンマ区切りリストを入力します。ユーザーがフィールドを空白にすることを許可するには、空白の引用符("") を値として追加します。

注: プロビジョニングツールでは、値は検証されません。誤ったデータ型の値を追加すると、その値は Data Director に表示されません。

カラム

フィールドに関連付けるベースオブジェクトのカラムの名前。

7. **【検索可能】** を選択します。

追加のフィールドプロパティが表示されます。

8. 要件に基づいて、次のプロパティを 1 つ以上選択します。

- 検索アナライザ
- 提案元
- ソート可能
- フィルタ可能
- ファセット範囲
- ファセット
- 表示可能

9. **【ファセット】** を選択した場合、必要に応じて、**【ファセット範囲】** フィールドで、ファセットとして設定する数値または日付フィールドの範囲を次の形式で指定します。

<Start Value>,<End Value>,<Frequency>

例: 1000,2000,50

注: Data Director アプリケーションにはファセットの範囲が表示されません。REST Web サービスを使用して検索を実行すると、応答でファセットの範囲が返されます。

10. **【適用】** をクリックします。

11. 変更内容を MDM Hub にパブリッシュします。

- a. **【パブリッシュ】** をクリックします。

確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。

- b. 変更内容を確認するか、確認せずにパブリッシュします。

- 確認せずにパブリッシュする場合は、**【パブリッシュ】** をクリックします。

- 確認後にパブリッシュするには、**【変更の確認】** をクリックし、画面に表示される指示に従います。

検索可能なフィールドのプロパティ

検索可能なフィールドのプロパティを設定するには、プロビジョニングツールを使用するか、変更リストをリボジトリに適用します。

検索可能な参照エンティティフィールドをフィルタ可能なファセットとして有効にすると、Data Director レコードビューに次の形式でフィルタフィールドラベルが表示されます。

<ビジネスエンティティのルックアップフィールドラベル> - <参照エンティティのルックアップフィールドラベル>

フィルタ可能なプロパティを参照エンティティに設定した場合、フィルタが機能するために、すべての依存リファレンスエンティティにフィルタ可能のプロパティを確実に設定します。

次の表に、検索可能なフィールドのプロパティを示します。

プロパティ	説明
検索可能	<p>検索要求により、検索文字列をフィールドで検索できるかどうかを示します。フィールドを検索要求に含めるには、このプロパティを有効にします。フィールドを検索要求に含めない場合は、このプロパティを無効にします。</p> <p>【検索可能】 プロパティを有効にする場合は、検索用の追加のプロパティを設定できます。</p> <p>注: システムカラムは、consolidationInd カラムと hubStateInd カラムを除いて、検索可能として設定することはできません。</p> <p>設定できる追加のプロパティは、次のとおりです。</p> <ul style="list-style-type: none"> - 検索アナライザ - 提案元 - ソート可能 - フィルタ可能 - ファセット範囲 - ファセット - 表示可能
検索アナライザ	<p>フィールドに使用するカスタム検索アナライザを指定します。フィールドに含まれるデータのタイプに基づいて、適切な検索アナライザを決定します。</p>
提案元	<p>Data Director アプリケーションにフィールドの値を検索文字列として提案するかどうかを示します。フィールドの値を検索文字列として提案するには、このプロパティを有効にします。フィールドの値を検索文字列として提案しない場合は、このプロパティを無効にします。</p> <p>重要: データセキュリティを確保するために、機密データを含むフィールドの【提案元】プロパティは有効にしないでください。</p>
ソート可能	<p>このプロパティは使用しないでください。</p>
フィルタ可能	<p>フィールドでフィルタを有効にするかどうかを示します。Data Director アプリケーションは、【検索】 ワークスペースでフィルタ可能なフィールドをフィルタとして表示します。フィールドをフィルタとして設定するには、このプロパティを有効にします。フィールドをフィルタとして設定しない場合は、このプロパティを無効にします。</p>

プロパティ	説明
ファセット範囲	<p>ファセットとして設定する数値または日付フィールドの範囲を示します。範囲は次の形式で指定します。</p> <p><Start Value>,<End Value>,<Frequency></p> <p>範囲に開始値は含まれますが、終了値は含まれません。例えば、整数フィールドのファセット範囲を 1000,2000,500 と設定すると、検索要求は次の範囲を返します。</p> <p>[1000 to 1500] [1500 to 2000]</p> <p>1000 to 1500 の範囲には 1000～1499 の値が含まれ、1500 to 2000 の範囲には 1500～1999 の値が含まれます。</p> <p>範囲の有効な最小値と最大値、および範囲の数を 10 に制限するオフセットが設定されていることを確認します。</p> <p>ファセットは負の数値で設定できませんが、検索要求には引き続き負の値が表示されます。</p> <p>日付フィールドの場合は、頻度に Y M D のサフィックスを追加します。この Y は年、M は月、D は日を示します。例えば、2M は 2 か月を示します。</p>
ファセット	<p>フィールドをファセットとして設定するかどうかを示します。ファセットフィールドは検索結果の値をグループ化し、各グループの数を表示します。</p> <p>Data Director アプリケーションは、ファセットフィールド、検索結果に基づいてグループ化されたフィールド値、および各グループの数を [検索] ワークスペースに表示します。</p> <p>子レコードフィールドがファセットフィールドとして設定されている場合、Data Director アプリケーションはファセットフィールドのツールチップを表示します。ツールチップテキストの形式は<子レコード名>/<子レコードフィールド名>です。</p> <p>[ファセット] プロパティは [フィルタ可能] プロパティと連携して機能するため、フィールドをファセットとして設定する場合は、[フィルタ可能] プロパティを有効にします。フィールドをファセットとして設定しない場合は、[ファセット] プロパティを無効にします。</p> <p>注: データセキュリティフィルタがファセットフィールドに設定されている場合、機密データを保護するために、フィールドのファセットは無効になります。</p> <p>検索ページのファセット値に使用する [ファセットラベル形式] メニューオプションを設定する場合は、[ファセット] プロパティを有効にします。</p> <p>選択可能なオプションは、以下のとおりです。</p> <ul style="list-style-type: none"> - 小文字 - ファセット値を小文字で表示します。例: john autumn。 - 大文字 - ファセット値を大文字で表示します。例: JOHN AUTUMN。 - タイトルケース - ファセット値の先頭の文字を大文字で表示します。例: John Autumn。
表示可能	このプロパティは使用しないでください。

検索またはクエリ結果の表示の設定

プロビジョニングツールを使用して、検索に使用するビジネスエンティティビューを設定します。検索結果には、検索結果用に設定するビジネスエンティティビューに属するフィールドのみが含まれます。また、検索フィルタの表示順序を変えることもできます。

検索可能なビューを設定する前に、検索結果に使用するビジネスエンティティビューを作成します。

注: 検索結果にビジネスエンティティの子レコードフィールドを表示するには、ビジネスエンティティから変換したビジネスエンティティビューを使用します。ビューのルートレコードレベルに、子レコードのフィールドが含まれていることを確認します。

1. プロビジョニングツールにログインします。
2. [データベース] リストから、アプリケーションが関連付けられているデータベースを選択します。

3. **【設定】** > **【アプリケーションエディタ】** をクリックします。
【アプリケーション】 ページが表示されます。
4. **【アプリケーション】** リストから、検索を設定するアプリケーションを選択します。
アプリケーションがない場合、検索を設定する前に、アプリケーションを作成します。
5. **【ツリービュー】** パネルで **【検索設定】** を選択し、**【作成】** をクリックします。
6. **【プロパティ】** パネルでビジネスエンティティと、検索またはクエリ結果を表示するのに使用するビジネスエンティティビューを選択します。
ビジネスエンティティビューを選択しないと、検索およびクエリ結果にはすべてのビジネスエンティティフィールドが含まれます。
7. 検索を設定した場合、必要に応じて、フィルタを選択し、検索フィルタの表示順序を設定します。
 - a. **【フィルタ表示順序】** の横の **【編集】** アイコンをクリックします。
【フィルタ表示順序の編集】 ダイアログボックスが表示されます。このダイアログボックスには、ビジネスエンティティモデルでフィルタ可能として設定されているフィールドであるフィルタが含まれます。
 - b. **【使用可能なフィルタ】** セクションから **【選択したフィルタ】** セクションへフィルタをドラッグします。
 - c. 順序を設定するには、フィルタをドラッグして上下に移動します。
 - d. **【OK】** をクリックします。
8. **【適用】** をクリックします。
検索設定は、一時的なワークスペースに保存されます。
9. 変更内容を MDM Hub にパブリッシュします。
 - a. **【パブリッシュ】** をクリックします。
確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。
 - b. 変更内容を確認するか、確認せずにパブリッシュします。
 - 確認せずにパブリッシュする場合は、**【パブリッシュ】** をクリックします。
 - 確認後にパブリッシュするには、**【変更の確認】** をクリックし、画面に表示される指示に従います。

類似レコードを表示するレイアウトの設定（オプション）

Data Director アプリケーションにデータを入力してレコードを作成する場合、入力したデータに基づいて取得された類似レコードを表示できます。類似レコードを表示するには、類似レコードの検索の基準となるフィールドを定義するようにレイアウトを設定する必要があります。

1. プロビジョニングツールにログインします。
2. **【データベース】** リストから、アプリケーションを設定するデータベースを選択します。
3. **【設定】** > **【コンポーネントエディタ】** をクリックします。
コンポーネントエディタが表示されます。
4. **【コンポーネント】** タイプのリストから **【類似するレコード】** を選択し、**【作成】** をクリックします。
5. **【プロパティ】** パネルに、類似するレコードのコンポーネントの名前を入力します。
6. **【XML】** フィールドに、類似するレコードを検索するフィールドのリストが含まれる XML 設定を入力します。

次の表で、類似するレコードコンポーネントの設定に使用できる XML 要素について説明します。

要素	説明
searchableFields	検索の基準となるフィールドを 1 つ以上指定します。searchableFields 要素は、field name 要素の一部です。
field name	類似するレコードの検索の基準となるフィールドの名前を指定します。field name 要素は searchableFields 要素の子です。複数の field name 要素を設定できます。
searchType	実行する検索のタイプを指定します。 searchType 要素には、次の子要素を含めることができます。 - smartSearch - searchMatch
smartSearch	検索を使用して類似するレコードを検索することを指定します。
searchMatch	クエリを使用して類似するレコードを検索することを指定します。 searchMatch 要素には、次の子要素を含めることができます。 - fuzzy - matchRuleSet
fuzzy	あいまい検索を実行するかどうかを指定します。あいまい検索を実行するには、true に設定します。 fuzzy 要素は searchMatch 要素の子です。この要素を追加しないと、完全一致検索が実行されます。
matchRuleSet	類似するレコードの検索に使用する一致ルールセットの名前を指定します。 matchRuleSet 要素は searchMatch 要素の子です。
label	検索フィールドの値のラベル形式を指定します。ラベル形式を設定するには、existsFormat 属性を使用します。
column	ラベル形式で使用する単一のカラムを指定します。ラベルのカラムを設定するには、columnUId 属性を使用します。これは、カラムの一意の ID です。column 要素は label 要素の子です。ラベルには、複数のカラムを指定できます。

構成サンプルについては、*Multidomain MDM のプロビジョニングツールガイド*を参照してください。

7. **【適用】** をクリックします。

作成した類似するレコードのコンポーネントが **【コンポーネントエディタ】** パネルと **【ツリービュー】** パネルに表示されます。

8. 変更内容を MDM Hub にパブリッシュします。

a. **【パブリッシュ】** をクリックします。

確認ダイアログボックスが表示され、変更内容をパブリッシュまたは確認するように求められます。

b. 変更内容を確認するか、確認せずにパブリッシュします。

- 確認せずにパブリッシュする場合は、**【パブリッシュ】** をクリックします。
- 確認後にパブリッシュするには、**【変更の確認】** をクリックし、画面に表示される指示に従います。

手順 4. オペレーショナル参照ストアの検証

Elasticsearch 構成の影響を受けるオペレーショナル参照ストア（ORS）のメタデータを検証するには、Hub コンソールで Repository Manager ツールを使用します。

1. Hub コンソールを起動し、MDM Hub マスタデータベースに接続します。
2. **【設定】** ワークベンチを展開し、**【Repository Manager】** をクリックします。
Repository Manager が表示されます。
3. **【検証】** タブをクリックし、検証するリポジトリを選択します。
4. **【検証】** をクリックします。
【検証チェックの選択】 ダイアログボックスが表示されます。
5. 実行する検証チェックを選択します。
6. **【OK】** をクリックします。
Repository Manager により、リポジトリが検証され、すべての問題が **【検出された問題】** ペインに表示されます。
7. 問題を修復するには、**【修復】** をクリックします。

手順 5. 検索データのインデックス処理

環境内にデータが含まれる場合、「スマート検索データの初期インデックス処理」バッチジョブを手動で実行し、データをインデックス処理します。環境内にデータがない場合、「スマート検索データの初期インデックス処理」ジョブを実行する必要はありません。ロードバッチジョブを実行してデータをロードすると、ロードバッチジョブで「スマート検索データの初期インデックス処理」バッチジョブが自動的に実行され、データがインデックス処理されます。検索要求では、レコードの検索のためにインデックスが使用されます。

「スマート検索データの初期インデックス処理」バッチジョブは、ビジネスエンティティに関係するすべてのベースオブジェクトに対して実行します。「スマート検索データの初期インデックス処理」バッチジョブをベースオブジェクトに対して実行すると、Elasticsearch サーバーによって検索可能なフィールド内のデータがインデックス処理されます。次に、このジョブによって、検索可能なフィールドが属するビジネスエンティティを表すすべてのコレクションに対して、インデックス処理されたデータが追加されます。コレクションが大きすぎる場合、コレクションを 1 つまたは複数のシャードに分割できます。シャードは、複数のノードにわたって分割されるコレクションの論理的な断片です。検索を実行すると、Elasticsearch サーバーがコレクションを読み取って一致するフィールドを返します。

「スマート検索データの初期インデックス処理」バッチジョブは、すべてのレコードのインデックス処理要求をキューに入れた後、レコードを非同期的にインデックス処理して、正常に完了したことを報告します。検索要求は、インデックス要求が正常に完了した後にのみ、インデックス処理されたレコードを表示できます。この処理には数分かかる場合があります。

重要: データのインデックス処理後にフィールドの検索可能なプロパティを更新した場合、インデックスは削除されます。データをインデックス処理するには、「スマート検索データの初期インデックス処理」バッチジョブを実行する必要があります。さらに、インデックス処理はリソースを大量に消費する処理であるため、「スマート検索データの初期インデックス処理」バッチジョブは同時に複数実行しないでください。

キーストア、トラストストア、および証明書の作成 (オプション)

Elasticsearch のインストール後、MDM Hub と Elasticsearch との間の通信を保護するために必要なキーストア、トラストストア、セキュリティ証明書を作成できます。キーストア、トラストストア、証明書を作成するには、Hub サーバーがインストールされたマシンのいずれか 1 台のみで、sip_ant スクリプトを実行します。次に、そのキーストア、トラストストア、証明書を、Hub サーバーがインストールされているその他すべてのマシンにコピーします。

注: キーストア、トラストストア、証明書は、sip_ant スクリプトを使用しなくても作成できます。

次の表に、必要なキーストアとトラストストアを示します。

キーストア/トラストストア名	説明
MDM_ESCLIENT_FILE_JKS.keystore	クライアント証明書とそのキーが格納された Elasticsearch キーストア。
MDM_ESKEYSTORE_FILE_JKS.keystore	クライアント証明書とノード証明書が格納された Elasticsearch キーストア。Elasticsearch クラスタに複数のノードがある場合、すべてのノードがこの証明書を使用します。
MDM ESTRUSTSTORE_FILE_JKS.keystore	クライアントノードと Elasticsearch ノードの署名済み証明書が格納された Elasticsearch トラストストア。

1. コマンドプロンプトを開き、Hub サーバーがインストールされたいずれかのマシンの次のディレクトリに移動します。
<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/bin
2. キーストア、トラストストア、証明書を作成するには、次のコマンドを実行します。
UNIX の場合: sip_ant.sh generate_mdm_es_store
Windows の場合: sip_ant.bat generate_mdm_es_store
3. キーストアとトラストストアのパスワードを入力するように求められたら、パスワードを指定します。
キーストア、トラストストア、証明書が次のディレクトリに作成されます。
<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/resources/certificates
4. 次のキーストアとトラストストアを Elasticsearch の各インストールの<Elasticsearch installation directory>/config ディレクトリにコピーします。
 - MDM_ESCLIENT_FILE_JKS.keystore
 - MDM_ESKEYSTORE_FILE_JKS.keystore
 - MDM ESTRUSTSTORE_FILE_JKS.keystore
5. 次のキーストアとトラストストアを、Elasticsearch クラスタに属する各 Hub サーバーノードの <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/resources/certificates ディレクトリにコピーします。
 - MDM_ESCLIENT_FILE_JKS.keystore
 - MDM ESTRUSTSTORE_FILE_JKS.keystore

第 24 章

統合プロセスの設定

この章では、以下の項目について説明します。

- [統合プロセスの設定の概要, 492 ページ](#)
- [統合設定, 492 ページ](#)
- [統合設定の変更, 496 ページ](#)

統合プロセスの設定の概要

統合プロセスは、一致ペアを 1 つのマスタレコードへとマージします。一致プロセスを設定した後で、MDM Hub 実装の統合プロセスを設定します。

統合プロセスを設定するには、[一致/マージ設定の詳細] ページの [マージ設定] タブを使用します。ソースシステムの特性と、レコードのアンマージ方法を指定できます。

ビジネスエンティティサービスまたはデータディレクタがレコードのマージに使用されている場合、ユーザーはマージされるレコードの値を上書きできます。ユーザーがマージされるレコードの値を上書きし、ワークフローが構成されている場合、タスクアクションを待機している保留中のレコードの詳細は保留中の制御テーブルに保存されます。

統合設定

統合設定は、Informatica MDM Hub での統合プロセスの動作に影響します。ここでは、[一致/マージ設定の詳細] ページの [マージ設定] タブで設定可能な設定について説明します。

不変行 ID オブジェクト

特定のベースオブジェクトに対して、ソースシステムを不変ソースとして指定することができます。これは、マージが行われる場合でも、そのソースシステムからのレコードが一意として受け入れられる (CONSOLIDATION_IND = 1) を意味します。そのソースからのレコードが完全に統合された後に、そのレコードが変更されることはありません。また、他のレコードとの一致の対象にもなりません (他のレコードをそのレコードと一致させることは可能です)。不変ソースとして設定できるのは、1 つのソースシステムだけです。

注: 子ベースオブジェクトで [親がマージされたときにキューに再追加する] 設定が UNCONSOLIDATED ONLY に設定されている場合、親のマージイベントでは、統合インジケータが 1 に設定されているレコードを除き、子レコードの統合インジケータは 4 に設定されます。統合インジケータが 1 に設定されている子レコー

ドをキューに再追加するには、[親がマージされたときにキューに再追加する] 設定を手動で 2 に設定する必要があります。

不変ソースは個別システムでもあります。すべてのレコードが、Informatica MDM Hub にマスターレコードとして格納されます。不変ソースシステムのすべてのソースレコードでは、ロードおよび PUT の統合インジケータが常に 1 (統合されたレコード) になります。

ベースオブジェクトに不変ソースを指定するには、[不変行 ID オブジェクト] の横にあるドロップダウンリストをクリックし、ソースシステムを選択します。

このリストには、このベースオブジェクトに関連付けられているソースシステムが表示されます。不変ソースシステムに指定できるのは、1 つのソースシステムだけです。

不変ソースシステムは、例えば Informatica MDM Hub がソースデータの唯一の永続ストアである場合などに適用できます。不変ソースシステムを指定すると、ソース内の一致が行われず、不変ソースからのレコードが自動的に一意として受け入れられるため、ロード、一致、マージの各プロセスが効率化されます。2 つの不変レコードをマージする必要がある場合は、その変更を許可するためにデータスチュワードが手動で検証を行う必要があります。その時点で、Informatica MDM Hub ではデータスチュワードが残っているキーを選択することができます。

個別システム

個別システムは、統合されずにベースオブジェクトに挿入されるデータを提供します。個別システムからのレコードを *同じ* システムの他のレコードと一致させることはありませんが、他のシステムの他のレコード (ロード時に CONSOLIDATION_IND が 4 に設定されたもの) との間で一致させることはできます。個別ソースシステムを指定し、ソースシステムごとにレコードを自動で統合するか手動で統合するかを設定できます。

個別ソースシステム

ソースシステムを *個別* ソース (ゴールデンソースとも呼ばれます) として指定することができます。これは、そのソースからのレコードがマージされないことを意味します。例えば、ABC ソースが個別ソースとして指定されている場合は、一致ルールでこのソースから取得した 2 つのレコードの一致 (またはマージ) が行われることはありません。個別ソースからのレコードは、自動一致とマージプロセスでの一時的な一致でも一致の対象となりません。このようなレコードは、一致のフラグを設定して手動でマージする必要があります。

個別ソースシステムを指定する手順

1. [マージ設定] タブのソースシステムのリストで、レコードをマージしないためにシステム内マージを禁止する必要があるソースシステムを選択 (オン) します。
2. 個別ソースシステムごとに、自動ルールのみを使用するかどうかを指定します。

自動ルールのみ

このオプションは、個別システムに対してのみ有効にできます。関連付けられた個別ソースシステムに対してどのタイプのルールを実行するかを設定できます。このチェックボックスは、Informatica MDM Hub でこの個別システムに対して自動統合ルールのみを適用する場合 (手動統合ルールを適用しない場合) にチェック (選択) します。デフォルトでは、このオプションは無効 (チェックなし) になっています。

親のマージ解除時に子をマージ解除 (カスケードマージ解除)

重要: この機能は、一致ルールと外部キーが設定されている子ベースオブジェクトにのみ適用されます。

子ベースオブジェクトでは、Informatica MDM Hub のカスケードマージ解除機能を使用できます。この機能を使用すると、親ベースオブジェクト内のレコードがマージ解除された場合の動作を指定できます。デフォルトでは、この機能は無効になっており、親レコードをマージ解除しても関連付けられた子レコードはマージ解除されません。[マージ設定] タブの下部にある [親のマージ解除時に子をマージ解除] で、子ベースオブジェ

クトの「カスケードマージ解除」チェックボックスをオンにした（選択した）場合、親オブジェクト内のレコードがマージ解除されると、Informatica MDM Hub では子ベースオブジェクト内の影響を受けるレコードもマージ解除されます。

カスケードマージ解除の前提条件

カスケードマージ解除を有効にする前提条件

- 親子リレーションを子ベースオブジェクトで設定しておく必要があります。
- 子ベースオブジェクトの外部キーカラムが一致有効カラムである必要があります。

「マージ設定」タブの下部にある「親のマージ解除時に子をマージ解除」部分に、外部キーが設定されている子ベースオブジェクトの一致有効カラムのみが表示されます。

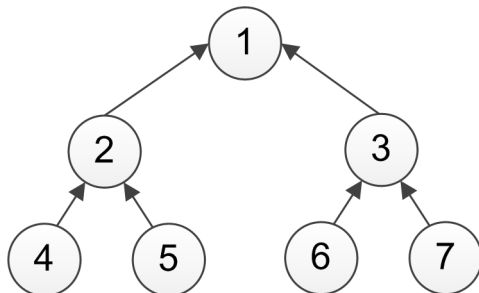
子ベースオブジェクトカスケードアンマージの動作

子ベースオブジェクトに対するカスケードアンマージプロセスの影響は、親がツリーアンマージプロセスを実行中であるか線形マージ解除プロセスを実行中であるかによって異なります。

カスケードアンマージプロセス前後の親ベースオブジェクトレコードと子ベースオブジェクトレコードの設定を次に示します。

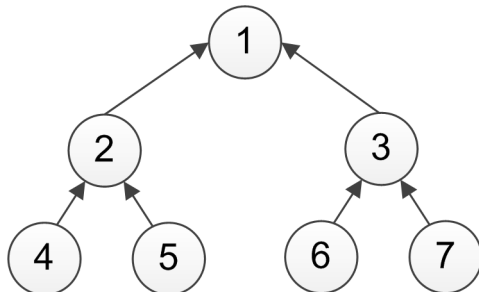
マージされた親ベースオブジェクトレコード

下図に、レコード「2」から「7」のデータで構成されるベストバージョンオブトゥールースである、親ベースオブジェクトレコード「1」を示します。



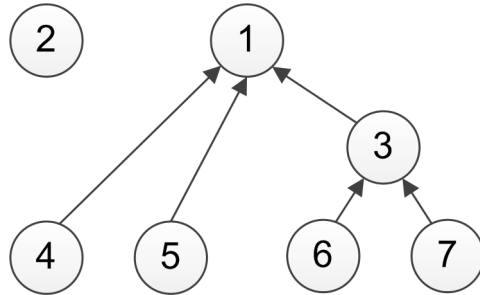
マージされた子ベースオブジェクトレコード

下図に、親ベースオブジェクトレコードと同じデータ構造を持つ子ベースオブジェクトレコード「1」を示します。通常、子ベースオブジェクトレコードは、親ベースオブジェクトレコードとは異なる構造を持ちます。



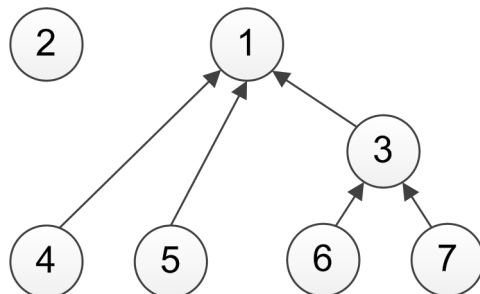
線形マージ解除後の親ベースオブジェクト

下図に、線形マージ解除プロセス後のセパレートベースオブジェクトとして、レコード「2」を示します。



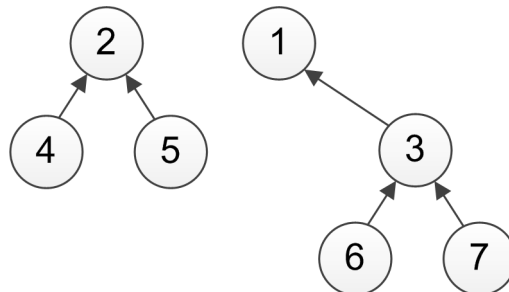
線形マージ解除後の子ベースオブジェクト

親ベースオブジェクトレコードで線形マージ解除プロセスが実行されると、子ベースオブジェクトレコードでも同じプロセスが実行されます。下図に、親ベースオブジェクトレコードの線形マージ解除プロセスが子ベースオブジェクトレコードに与える影響を示します。



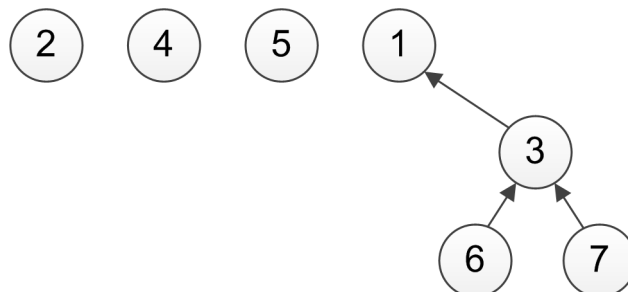
ツリーアンマージ後の親ベースオブジェクト

下図に、ツリーアンマージプロセス実行後にそのツリー構造を維持するセパレートベースオブジェクトである、レコード「2」を示します。



ツリーアンマージ後の子ベースオブジェクト

親ベースオブジェクトレコードでツリーアンマージプロセスが実行されると、子ベースオブジェクトのアンマージされたツリーで線形マージ解除プロセスが実行されます。下図に、親ベースオブジェクトレコードのツリーアンマージプロセスが子ベースオブジェクトレコードに与える影響を示します。



複数の子を持つ親

1つの親ベースオブジェクトに複数の子ベースオブジェクトがある場合、各子ベースオブジェクトに対して明示的にカスケードマージ解除を有効にできます。いったん設定すると、親ベースオブジェクトがマージ解除された場合に、すべての関連する子ベースオブジェクトにある、影響を受けるレコードもすべて同様にマージ解除されます。

カスケードマージ解除の使用に関する考慮事項

影響を受けるレコードを、どの実装でも完全にマージ解除しなければならないわけではありません。完全なマージ解除を実行すると、影響を受ける子レコードが多くなり、マージ解除にかかるパフォーマンスのオーバーヘッドが増加する場合があります。また、このプロパティを有効にした方がよいとは限りません。例えば、Customer が Customer Type の子である場合がこれに該当します。この場合、Customer Type をマージ解除する際に、Customer のマージ解除が必要とは限りません。ただし、通常、Customer をマージ解除する場合は、顧客にリンクされたアドレスもマージ解除することをお勧めします。

注: カスケードマージ解除を有効にしても、前回の手動マージ解除が子ベースオブジェクトに対して行われた場合は、子レコードがマージ解除されないことがあります。

マージ解除機能を有効にすると、この機能は子テーブルと子相互参照テーブルに適用されます。有効にした後に親相互参照をマージ解除すると、元の子相互参照もマージ解除されます。この機能は親には影響しません。柔軟な対応ができるよう、子テーブルにのみ動作します。

統合設定の変更

「マージ設定」タブで統合設定を変更する手順

1. スキーママネージャで、設定するベースオブジェクトの「一致/マージ設定の詳細」ダイアログを表示します。
2. 書き込みロックを取得します。
3. 「マージ設定」タブをクリックします。
選択したベースオブジェクトの「マージ設定」タブが表示されます。
4. 以下の設定を変更します。
 - 不変行 ID オブジェクト
 - 個別システム
 - 親のマージ解除時に子をマージ解除（カスケードマージ解除）
5. 「保存」ボタンをクリックして変更を保存します。

第 25 章

保留中の制御テーブル

保留中の制御テーブルは、MDM Hub がベースオブジェクトに対して自動的に作成するシステムテーブルです。このテーブルは、マージタスクアクションを待機している、信頼された値へのオーバーライドを含むレコードに関する情報を追跡します。テーブル名の形式は C_<base object name>_PCTL です。保留中の制御テーブルには、制御テーブルのすべてのカラムと相互作用 ID を保存する追加のカラムが含まれます。

保留中の制御テーブルは、タスクワークフローがマージ操作に設定されるときに使用されます。このテーブルはビジネスエンティティサービス API によってのみ使用され、サービス統合フレームワーク（SIF）API およびバッチジョブには使用されません。

ユーザーがレコード内の信頼された値を手動で上書きするときに、マージプロセスはレコードを保留中の制御テーブルに挿入します。マージアクションを保留しているレコードは相互作用 ID により保護されます。レコードの信頼が有効な各カラムは、保留中の制御テーブルに 4 つのカラムを持ちます。ユーザーが手動で正しい値を入力するときに、相互参照レコードが保留状態で作成されます。マージタスクが承認されると、保留中の相互参照レコードおよび保留中のオーバーライドが昇格され、レコードがマージされます。

次のテーブルで、保留中の制御テーブル内のカラムについて説明します。

カラム名	説明
INTERACTION_ID	マージタスクの一部である関連する信頼されたオーバーライドをグループ化する識別子。マージタスクアクションの完了に必要です。
<trust-enabled column name>_LRS	最新の更新をベースオブジェクトレコードに提供するソースシステムの最後の行 ID。
<trust-enabled column name>_LUD	レコードの最終値を提供した相互参照レコードの最終更新日。
<trust-enabled column name>_SRX	レコードの最終値を提供した相互参照レコードの識別子。
<trust-enabled column name>_OTS	信頼のオーバーライドを促進するエンコードされた信頼設定。

ベースオブジェクトに対して履歴が有効な場合、MDM Hub は保留中の制御テーブルに対して別個の履歴テーブルを保持します。保留中の制御テーブルに関連する履歴テーブルの名前の形式は C_<base object name>_HPCT です。

第 26 章

パブリッシュプロセスの設定

この章では、以下の項目について説明します。

- [パブリッシュプロセスの概要, 498 ページ](#)
- [パブリッシュプロセスの設定手順, 499 ページ](#)
- [メッセージキューツールの起動, 499 ページ](#)
- [グローバルメッセージキュー設定の設定, 499 ページ](#)
- [メッセージキューサーバーの設定, 500 ページ](#)
- [送信メッセージキューの設定, 502 ページ](#)
- [JMS メッセージの並行処理の設定, 504 ページ](#)
- [JMS セキュリティの設定, 504 ページ](#)
- [メッセージトリガの設定, 504 ページ](#)
- [メッセージキューのポーリングの無効化, 509 ページ](#)
- [JMS メッセージの XML リファレンス, 509 ページ](#)
- [従来の JMS メッセージの XML リファレンス, 523 ページ](#)

パブリッシュプロセスの概要

MDM Hub パブリッシュプロセスを設定して、Hub ストアのデータ変更について XML メッセージを生成し、アウトバウンド Java Messaging System (JMS) メッセージキューにメッセージをパブリッシュできます。外部アプリケーションは、MDM Hub が JMS メッセージキューにパブリッシュした XML メッセージを取得できます。

パブリッシュプロセスを設定する前に、Hub サーバーとプロセスサーバーがデプロイされているアプリケーションサーバーに対して JMS メッセージキューと接続ファクトリをセットアップします。パブリッシュプロセスのパフォーマンスを強化する場合は、JMS メッセージの並行処理を設定します。

重要: サービス統合フレームワーク (SIF) は、JMS メッセージキュー (siperian.sif.jms.queue) に対してメッセージ駆動型 Bean を使用し、受信する非同期 SIF 要求を処理します。siperian.sif.jms.queue JMS メッセージキューは、MDM Hub のインストールプロセス中にセットアップされ、すべての MDM Hub インストールに必須の要件です。MDM Hub パブリッシュプロセスで設定する JMS メッセージキューは、外部アプリケーションを使用して JMS メッセージキューから JMS メッセージを取得する場合に必要です。

パブリッシュプロセスの設定手順

Informatica MDM Hub をインストールした後、Hub コンソールのメッセージキューツールを使用して、Informatica MDM Hub 実装のメッセージキューを設定します。送信メッセージキューでイベントをパブリッシュする場合は、必ず以下の作業を行う必要があります。

1. アプリケーションサーバーでメッセージキューを設定します。
Informatica MDM Hub インストーラは、メッセージキューと接続ファクトリの設定を自動的に設定します。詳細については、プラットフォームの『*Multidomain MDM のインストールガイド*』を参照してください。
 2. グローバルメッセージキュー設定を設定します。
 3. メッセージキューサーバーを少なくとも 1 つ追加します。
 4. メッセージキューサーバーにメッセージキューを少なくとも 1 つ追加します。
 5. パブリッシュするデータを含む ORS ごとに JMS イベントメッセージスキーマを生成します。
 6. メッセージキューのメッセージトリガを設定します。
- メッセージキューの設定が完了したら、監査マネージャを使用して実行時の動作を確認できます。

メッセージキューツールの起動

1. Hub コンソールでマスターデータベースに接続します。
メッセージキューはマスターデータベースで定義します。
2. Hub コンソールで設定ワークベンチを展開し、[メッセージキュー] をクリックします。
メッセージキューツールが表示されます。メッセージキューツールは 2 つのペインで構成されています。

ペイン	説明
ナビゲーションペイン	この Informatica MDM Hub 実装に対して定義されているメッセージキューを（ツリービューで）表示します。
プロパティペイン	選択されているメッセージキューのプロパティを表示します。

グローバルメッセージキュー設定の設定

Informatica MDM Hub 実装のグローバルメッセージキュー設定を設定する手順

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. 送信メッセージのキューを監視するデータ変更監視の設定を指定します。
4. データ変更監視を有効または無効にするには、[データ変更監視ステータスの切り替え] ボタンをクリックします。

以下の監視設定を指定します。

監視設定	説明
受信タイムアウト（ミリ秒）	デフォルトは 0 です。メッセージをキューから受信できる時間です。
受信バッチサイズ	デフォルトは 100 です。1 回のパスで処理してメッセージキューに入れることができるイベントの最大数です。
メッセージチェック間隔（ミリ秒）	デフォルトは 300000 です。受信メッセージのポーリングまたは送信メッセージの処理を開始する前に一時停止する時間です。受信メッセージキューと送信メッセージキューの両方に同じ値が適用されます。
非同期チェック間隔（ミリ秒）	設定した場合、定期的に ORS メタデータをポーリングし、ORS 内のデザインオブジェクトに前回のポーリング以降に変更が行われていた場合、XML メッセージスキーマを再生成します。 デフォルトでは、この機能は無効になっています（0 に設定されています）。この機能は、以下に該当する場合にのみ使用できます。 データ変更監視が有効になっている。 ORS 固有の XML メッセージスキーマが JMS イベントスキーママネージャを使用して生成されている。 注: この値は、[メッセージチェック間隔] の値以上にしてください。

5. 変更するプロパティの横にある【編集】ボタンをクリックします。
6. 【保存】ボタンをクリックして変更を保存します。

メッセージキューサーバーの設定

MDM Hub 実装用のメッセージキューサーバーを設定します。

MDM Hub でメッセージキューを定義する前に、MDM Hub でメッセージキューの処理に必要なメッセージキューサーバーを設定します。メッセージキューサーバーを MDM Hub に追加する前に、Hub サーバーとプロセスサーバーがデプロイされているアプリケーションサーバーインスタンス上にメッセージキューサーバーを設定します。メッセージキューサーバーを MDM Hub で定義するには、JBoss および WebLogic 用のメッセージキューサーバーの接続ファクトリ名と、WebSphere 用のメッセージキューサーバーのサーバー名とポート番号が必要です。

メッセージキューサーバーのプロパティ

この説では、メッセージキューサーバーに対して構成できる設定について説明します。

WebLogic および JBoss のプロパティ

以下のメッセージキューサーバーのプロパティを設定できます。

プロパティ	説明
接続ファクトリ名	このメッセージキューサーバーの接続ファクトリの名前。
表示名	Hub コンソールに表示される、このメッセージキューサーバーの名前。
説明	このメッセージキューサーバーに関する情報。

WebSphere のプロパティ

IBM WebSphere 実装には以下のプロパティがあります。

プロパティ	説明
サーバー名	メッセージキューが定義されているサーバーの名前。
チャンネル	メッセージキューが定義されているサーバーのチャンネル。
ポート	メッセージキューが定義されているサーバーのポート。

メッセージキューサーバーの追加

メッセージキューサーバーを追加して、メッセージキューに接続します。

メッセージキューサーバーを追加する手順

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインの任意の場所を右クリックし、**【メッセージキューサーバーの追加】**を選択します。
[メッセージキューサーバーの追加] ダイアログボックスが表示されます。
4. メッセージキューサーバーのプロパティを指定します。

メッセージキューサーバーのプロパティの編集

既存のメッセージキューサーバーのプロパティを編集する手順

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、設定するメッセージキューサーバーの名前を選択します。
4. このメッセージキューサーバーの編集可能なプロパティを変更します。
変更するプロパティの横にある**【編集】** ボタンをクリックします。
5. **【保存】** ボタンをクリックして変更を保存します。

メッセージキューサーバーの削除

既存のメッセージキューサーバーを削除する手順

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、削除するメッセージキューサーバーの名前を右クリックし、ポップアップメニューから **削除** を選択します。
メッセージキューツールから、削除を確認するように求められます。
4. **はい** をクリックします。

送信メッセージキューの設定

ここでは、Informatica MDM Hub 実装の送信 JMS メッセージキューを設定する方法について説明します。

メッセージキューについて

Informatica MDM Hub で送信 JMS メッセージキューを定義する前に、メッセージキューを処理するメッセージキューサーバーを定義する必要があります。JMS では、メッセージキューは XML メッセージのステージング領域として機能します。Informatica MDM Hub は、メッセージキューに XML メッセージをパブリッシュします。外部アプリケーションは、パブリッシュされたそれらの XML メッセージをメッセージキューから取得します。

メッセージキューのプロパティ

以下のメッセージキューのプロパティを設定できます。

プロパティ	説明
Queue Name	メッセージキューの名前。アプリケーションサーバーに設定されている JNDI キュー名と一致指せる必要があります。
表示名	Hub コンソールに表示される、このメッセージキューの名前。
説明	このメッセージキューに関する情報。

メッセージキューサーバーへのメッセージキューの追加

接続先のメッセージキューサーバーにメッセージキューを追加します。

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、メッセージキューを追加する先のメッセージキューサーバーの名前を右クリックし、**メッセージキューの追加** を選択します。
[メッセージキューの追加] ダイアログボックスが表示されます。
4. メッセージキューのプロパティを指定します。

5. **[OK]** をクリックします。
キューの割り当てを選択するように求められます。
6. 次のキュー割り当てオプションのいずれかを選択します。

オプション	説明
未割り当てのままにする	キューは現在割り当てられておらず、使用されていません。オプションは、キューを Informatica MDM Hub の API 応答用のアウトバウンドキューとして使用する場合、またはキューが現在割り当てられておらず使用されていないことを示す場合に選択します。
メッセージキュートリガで使用する	キューは現在割り当てられており、スキーママネージャで定義されているメッセージトリガで使用できます。
従来の XML を使用する	Informatica MDM Hub の実装では、現在のバージョンの XML メッセージ形式ではなく従来の XML メッセージ形式を使用する必要があります。

7. **[保存]** をクリックします。

メッセージキューのプロパティの編集

既存のメッセージキューのプロパティを編集する手順

1. Hub コンソールで、メッセージキューツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、設定するメッセージキューの名前を選択します。
4. このメッセージキューの編集可能なプロパティを変更します。
5. 変更するプロパティの横にある **[編集]** ボタンをクリックします。
6. 必要に応じて、キューの割り当てを変更します。
7. **[保存]** ボタンをクリックして変更を保存します。

メッセージキューの削除

メッセージキューを削除するには、メッセージキューツールを使用します。

削除するメッセージキューにメッセージトリガが関連付けられていないことを確認します。メッセージキューにメッセージトリガが関連付けられている場合は、メッセージキューを削除する前にそのメッセージトリガを削除します。

1. 設定ワークベンチで、**メッセージキューツール**を起動します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで、削除するメッセージキューの名前を右クリックし、ポップアップメニューの **[削除]** をクリックします。
メッセージキューツールから、削除を確認するように求められます。
4. **[はい]** をクリックします。

JMS メッセージの並行処理の設定

JMS メッセージの並行処理を設定する場合、Hub Server はバッチでのメッセージ処理の負荷を複数のプロセスサーバーに分散します。JMS メッセージの並行処理のバッチサイズを設定する必要があります。Hub Server プロパティファイルで、JMS メッセージの並行処理を設定します。

1. 次のディレクトリにある `cmxserver.properties` ファイルを開きます。
<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/resources
2. JMS メッセージの並行処理に必要なプロパティを追加し、ファイルを保存します。
次の表に、JMS メッセージの並行処理用のプロパティを示します。

プロパティ	説明
<code>mq.data.change.threads</code>	パブリッシュプロセス中に JMS メッセージを処理するために使用するスレッドの数。デフォルトは 1。
<code>mq.data.change.batch.size</code>	パブリッシュプロセスの各バッチで処理する JMS メッセージの数。デフォルトは 500。
<code>mq.data.change.timeout</code>	JMS メッセージを処理できる時間（秒）。デフォルトは 120。

JMS セキュリティの設定

メッセージキューを保護するには、JMS セキュリティを設定する必要があります。

まず、アプリケーションサーバーで JMS セキュリティを設定します。`cmxserver.properties` ファイルに以下のプロパティを設定して、アプリケーションサーバーに設定したユーザー資格情報が使用されるように MDM Hub を設定します。

```
<connection factory name>.qcf.username=<user name>
<connection factory name>.qcf.password=<password>
```

メッセージトリガの設定

MDM Hub 実装のメッセージトリガを設定するには、スキーママネージャツールを使用します。

メッセージトリガは、MDM Hub が外部アプリケーションに伝達するアクションを特定します。ルールが定義されているアクションが発生すると、MDM Hub によって XML メッセージが JMS メッセージキューに送られます。このメッセージを入れる JMS メッセージキューをメッセージトリガで指定します。

メッセージトリガの動作例を次のシナリオに示します。

1. ベースオブジェクトにレコードを挿入します。
2. この挿入アクションによりメッセージトリガが開始されます。
3. MDM Hub でこのメッセージトリガが評価され、適切なメッセージキューにメッセージが送られます。
4. 外部のアプリケーションでメッセージキューがポーリングされ、メッセージが取得されて処理されます。

すべてのトリガに対して同じメッセージキューを使用することも、トリガごとに異なるメッセージキューを使用することもできます。アクションでメッセージトリガを起動するには、メッセージキューを設定し、そのベースオブジェクトおよびアクションのメッセージトリガを定義します。

メッセージトリガのイベントのタイプ

次のタイプのイベントでメッセージトリガが起動し、メッセージがキューに追加されることがあります。

次の表に、メッセージキュールールを定義できるイベントを示します。

イベント	説明	メッセージタイプコード
新しいデータの追加	次の方法を使用してデータを追加するために使用されます。 <ul style="list-style-type: none"> - ロードプロセスによる方法 - データマネージャを使用する方法 - HTTP、SOAP、EJB などのプロトコルを介して PUT または CLEANSE_PUT を使用する API 動詞による方法 	1
新しい保留データの追加	「保留」状態でレコードが作成されます。状態が有効なベースオブジェクトに適用されます。	10
既存のデータの更新	次の方法を使用してデータを更新するために使用されます。 <ul style="list-style-type: none"> - ロードプロセスによる方法 - データマネージャを使用する方法 - HTTP、SOAP、EJB などのプロトコルを介して PUT または CLEANSE_PUT を使用する API 動詞による方法 信頼ルールによってベースオブジェクトのカラムの更新が妨げられた場合、メッセージは生成されません。 1 つ以上の指定されたカラムが更新された場合、単一のメッセージが生成されます。このメッセージには、すべての出力システムのすべての相互参照レコードからのデータが含まれます。	2
既存の保留データの更新	「保留」状態の既存のレコードが更新されます。状態が有効なベースオブジェクトに適用されます。	11
更新 (XREF のみ変更)	次のシナリオでデータを更新するために使用されます。 <ul style="list-style-type: none"> - ロードプロセスによって、相互参照のみが変更された場合。 - HTTP、SOAP、EJB などのプロトコルを介して PUT または CLEANSE_PUT を使用する API によって相互参照のみが変更された場合。 	6
保留データの更新 (XREF のみ変更)	「保留」状態の相互参照レコードが更新されます。また、レコードの昇格も含まれます。状態が有効なベースオブジェクトに適用されます。	12
データのマージ	次の方法を使用してデータをマージするために使用されます。 <ul style="list-style-type: none"> - マージマネージャによる方法 - HTTP、SOAP、EJB などのプロトコルを介する API 動詞による方法 - 自動一致とマージプロセスによる方法 	4

イベント	説明	メッセージタイプコード
データのマージ (ベースオブジェクトを更新)	次のシナリオでデータをマージするために使用されます。 <ul style="list-style-type: none"> - ベースオブジェクトが更新された場合 - 新しい相互参照を挿入するためにレコードが行 ID によってロードされ、ベースオブジェクトが更新された場合 信頼ルールのためにベースオブジェクトカラムがマージできない場合は、メッセージは生成されません。 1 つ以上の指定されたカラムが更新された場合、単一のメッセージが生成されます。このメッセージには、すべての出力システムのすべての相互参照レコードからのデータが含まれます。	7
データのマージ解除	次の方法を使用してデータをマージ解除するために使用されます。 <ul style="list-style-type: none"> - データマネージャによる方法 - HTTP、SOAP、EJB などのプロトコルを介して UNMERGE を使用する API 動詞による方法 	8
一意のデータとしての受け入れ	次の方法を使用してレコードを一意として受け入れるために使用されます。 <ul style="list-style-type: none"> - マージマネージャによる方法 - 一致ルールを設定したときにオプションで有効にするスキーマツールによる方法 レコードが一致ルールによって自動的に、またはデータスチュワードによって手動で一意のレコードとして受け入れられると、MDM Hub で、レコード情報（すべての出力システムの相互参照情報など）を示すメッセージが生成されます。このメッセージはキューに追加されます。	3
ベースオブジェクトデータの削除	ベースオブジェクトレコードが論理削除され、状態が「削除済み」に変更されます。状態が有効なベースオブジェクトに適用されます。	14
XREF データの削除	相互参照レコードが論理削除され、状態が「削除済み」に変更されます。状態が有効なベースオブジェクトに適用されます。	15
保留中の BO データの削除	「保留」状態のベースオブジェクトレコードが物理削除されます。状態が有効なベースオブジェクトに適用されます。	16
保留中の XREF データの削除	「保留」状態の相互参照レコードが物理削除されます。状態が有効なベースオブジェクトに適用されます。	17
リストア	削除した相互参照レコードがリストアされ、状態が「アクティブ」に変更されます。状態が有効なベースオブジェクトに適用されます。	19

メッセージトリガのベストプラクティス

ここで紹介しているベストプラクティスを参考に、実装に対してメッセージトリガを効率よく作成します。

- ベースオブジェクトのメッセージトリガ定義にメッセージキューが使用されている場合は、MDM Hub では「このメッセージキューはメッセージトリガで現在使用されています。」というメッセージが表示されます。この場合、メッセージキューのプロパティを編集することはできません。代わりに、別のメッセージキューを作成して、必要な変更を加える必要があります。
- MDM Hub インストーラは、siperian.sif.jms.queue というデフォルトの JMS キューを作成します。メッセージトリガの設定時にこのキューを使用すると、MDM Hub によってエラーが生成されます。

- メッセージトリガは、1つのベースオブジェクトにのみ適用され、そのベースオブジェクトに対して特定のアクションが直接行われた場合にのみ実行されます。親子の関係にある2つのテーブルが存在する場合は、各テーブルに対してメッセージキューをそれぞれ明示的に定義する必要があります。MDM Hubにより、ベースオブジェクトに対する具体的な変更（INSERTやPUTなどのロード）が検出されます。親テーブルのレコードに変更を加えた場合、親レコードについてのみメッセージトリガが実行されます。親レコードに加えた変更によって関連付けられた子レコードが影響を受ける場合には、子テーブル用の個別のメッセージトリガを明示的に設定する必要があります。

メッセージトリガシステムのプロパティ

メッセージトリガを設定するときには、少なくとも「起動元」システムを1つと「メッセージ内」システムを1つ選択する必要があります。

メッセージトリガのシステムには、次のプロパティがあります。

起動元

アクションを起動するシステムを1つ以上選択します。少なくとも1つ選択する必要があります。

メッセージ内

メッセージを表示するシステムを1つ以上選択します。少なくとも1つ選択する必要があります。

メッセージキュー内の各メッセージには、「メッセージ内」システムのいずれか1つで指定された各相互参照のpkey_src_object値が含まれます。

例えば、実装にソースシステムが3つ（A、B、およびC）あり、システムAを「起動元」システムとして選択します。ベースオブジェクトレコードには、AおよびBの相互参照レコードが含まれています。このベースオブジェクトレコードのシステムAで相互参照を更新する場合について考えてみます。次の表に、メッセージトリガの設定の組み合わせとそれぞれの結果のメッセージを示します。

「メッセージ内」のシステム	結果のメッセージ
A	システムAの相互参照を含むメッセージ。
B	システムBの相互参照を含むメッセージ。
C	メッセージなし。「メッセージ内」からの相互参照はなし。
AとB	システムAおよびBの相互参照を含むメッセージ。
AとC	システムAの相互参照を含むメッセージ。
BとC	システムBの相互参照を含むメッセージ。
A、B、およびC	システムAおよびBの相互参照を含むメッセージ。

メッセージトリガの追加

- メッセージトリガでできるようにメッセージキューを設定します。
- スキーママネージャを起動します。
- 書き込みロックを取得します。
- 監視するベースオブジェクトを展開し、「メッセージトリガのセットアップ」ノードを選択します。

メッセージトリガが1つも作成されていない場合、スキーマツールの画面には何も表示されません。

5. 次のいずれかを実行します。

- メッセージトリガが定義されていない場合は、**[メッセージトリガの追加]** をクリックします。
- メッセージトリガが定義されている場合は、**[追加]** をクリックします。

メッセージトリガの追加ウィザードが表示されます。

6. 新しいメッセージトリガの名前と説明を指定します。

7. **[Next (次へ)]** をクリックします。

メッセージのパッケージを指定するように要求されます。

8. メッセージの構築に使用するパッケージを選択します。

9. **[Next (次へ)]** をクリックします。

対象のメッセージキューを指定するように要求されます。

10. メッセージを伝達するメッセージキューを選択します。

11. **[Next (次へ)]** をクリックします。

このメッセージトリガのルールを指定するように求められます。

12. このメッセージトリガのイベントタイプを選択します。

13. このメッセージトリガのシステムプロパティを設定します。**[起動元]** のシステムと **[メッセージ内]** のシステムを少なくとも1つずつ選択する必要があります。

14. 更新オプションを選択した場合は、**[次へ]** をクリックします。それ以外の場合は、**[完了]** をクリックします。

更新アクションをクリックした場合は、更新アクションを監視するカラムを選択するように求められます。

15. 次のいずれかを実行します。

- このメッセージトリガに関連付けられたイベントを監視するカラムを選択します。
- すべてのカラムの更新を監視する場合は、**[いずれかのカラムで変更が行われた場合にメッセージを起動する]** チェックボックスをオンにします。

16. **[完了]** をクリックします。

メッセージトリガの編集

1. スキーママネージャを起動します。

2. 書き込みロックを取得します。

3. 監視するベースオブジェクトを展開し、**[メッセージトリガのセットアップ]** ノードを選択します。

4. **[メッセージトリガ]** リストで、設定するメッセージトリガをクリックします。

スキーママネージャに、選択したメッセージトリガの設定が表示されます。

5. 必要に応じて設定を変更します。

6. 編集可能なプロパティの横にある **[編集]** ボタンをクリックします。

7. **[保存]** ボタンをクリックして変更を保存します。

メッセージトリガの削除

1. スキーママネージャを起動します。

2. 書き込みロックを取得します。

3. 監視するベースオブジェクトを展開し、[メッセージトリガのセットアップ] ノードを選択します。
4. [メッセージトリガ] リストで、削除するメッセージトリガをクリックします。
5. [削除] ボタンをクリックします。
削除を確認するように求められます。
6. [はい] をクリックします。

メッセージキューのポーリングの無効化

マルチノード環境では、個々のノードに対してメッセージキューのポーリングを無効にできます。メッセージキューのポーリングを無効にするには、`cmxserver.properties` ファイルと `cmxcleanse.properties` ファイル内で、`mq.data.change.monitor.thread.start` プロパティを `false` に設定します。

DEBUG モードでは、メッセージキューのポーリングの操作を確認できます。

- `mq.data.change.monitor.thread.start` の値が `false` であるノードでは、メッセージ **Monitoring is disabled** がログに表示されます。
- `mq.data.change.monitor.thread.start` の値が `true` であるノードでは、メッセージ **Data changes monitoring started** がログに表示されます。

デフォルトでは、メッセージキューのポーリングは、MDM Hub EAR ファイルがデプロイされているすべての Java 仮想マシン上で有効になっています。データ変更の監視とメッセージのパブリッシュに複数のポーラースレッドを使用していると、メッセージが正しい順序でパブリッシュされない場合があります。メッセージのパブリッシュ順序を制御するには、単一のポーラースレッドを使用します。

JMS メッセージの XML リファレンス

この節では、Informatica MDM Hub XML メッセージの構造について説明し、メッセージの例を示します。

注: Informatica MDM Hub の実装で、現在のバージョンの XML メッセージ形式（この節で説明）ではなく従来の XML メッセージ形式（Informatica MDM Hub XU バージョン）を使用する必要がある場合は、[「従来の JMS メッセージの XML リファレンス」 \(ページ 523\)](#) を参照してください。

ORS 固有の XML メッセージスキーマの生成

パブリッシュプロセスは、XML メッセージを作成するために、Hub コンソールの JMS イベントスキーママネージャツールを使用して作成される ORS 固有のスキーマファイル（`<ors-name>-siperian-mrm-event.xsd`）に依存します。

XML メッセージの要素

以下の表に、XML メッセージの要素を示します。

フィールド	説明
ルートノード	
<siperianEvent>	XML メッセージのルートノード。
イベントのメタデータ	
<eventMetadata>	イベントメタデータのルートノード。
<messageId>	siperianEvent メッセージの一意の ID。
<eventType>	次のいずれかの値で示されるイベントのタイプ。 <ul style="list-style-type: none">- Insert- Update- Update XREF- Accept as Unique- Merge- Unmerge- Merge Update
<baseObjectId>	このアクションの影響を受けるベースオブジェクトの UID。
<packageId>	このアクションに関連付けられたパッケージの UID。
<messageDate>	このメッセージが生成された日付/時刻。
<orsId>	このイベントに関連付けられたオペレーショナル参照ストア（ORS）の ID。
<triggerId>	このメッセージを生成したイベントをトリガしたルールの UID。
イベントの詳細	
<eventTypeEvent>	イベントの詳細のルートノード。
<sourceSystemName>	このイベントに関連付けられたソースシステムの名前。
<sourceKey>	このイベントに関連付けられた PKEY_SRC_OBJECT の値。
<eventDate>	イベントが生成された日付/時刻。
<rowId>	イベントの影響を受けたベースオブジェクトレコードの行 ID。
<xrefKey>	このイベントの影響を受けた相互参照レコードのルートノード。
<systemName>	このイベントの影響を受けた相互参照レコードのシステム名。
<sourceKey>	このイベントの影響を受けた相互参照レコードの PKEY_SRC_OBJECT。
<packageName>	このイベントに関連付けられたセキュアパッケージの名前。

フィールド	説明
<columnName>	パッケージ内の各カラムは、XML ファイルの要素で表されます。例:rowidObject および consolidationInd。JMS イベントスキーママネージャツールを使用して生成される ORS 固有の XSD 内で定義されます。
<mergedRowid>	マージで消失したレコードの ROWID_OBJECT 値のリスト。このフィールドは、Merge イベントの場合のみメッセージに組み込まれます。

メッセージのフィルタリング

MessageType という名前のカスタム JMS ヘッダーを使用して、入力メッセージをメッセージタイプに基づいてフィルタリングすることができます。このメッセージヘッダーには、以下のメッセージタイプが指定されています。

メッセージタイプ	説明
siperianEvent	イベント通知メッセージ。
<serviceNameReturn>	サービス統合フレームワーク (SIF) 応答の場合は、次に示す GET 要求への応答のように、応答が SIF 要求の名前で始まります。<getReturn> <message>The GET was executed successfully - retrieved 1 records</message> <recordKey> <ROWID>2</ROWID> </recordKey> ...

XML メッセージの例

この節では、XML メッセージの例を示します。

Accept As Unique メッセージ

Accept As Unique メッセージの例を次に示します。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Accept as Unique</eventType>
    <baseObjectId>BASE_OBJECT.C_CONTACT</baseObjectId>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</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>
    </contactPkg>
  </acceptAsUniqueEvent>
</siperianEvent>
```

```

        <lastRowidSystem>SYS0          </lastRowidSystem>
        <dirtyInd>0</dirtyInd>
        <firstName>Joey</firstName>
        <lastName>Brown</lastName>
    </contactPkg>
</acceptAsUniqueEvent>
</siperianEvent>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

AMRule メッセージ

AMRule メッセージの例を次に示します。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>AM Rule Event</eventType>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_ORS</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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

BoDelete メッセージ

BoDelete メッセージの例を次に示します。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>BO 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>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>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

BoSetToDelete メッセージ

BoSetToDelete メッセージの例を次に示します。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>BO 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>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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Delete メッセージ

次のコードは Delete メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

```

        <LAST_NAME>Smith</LAST_NAME>
        <HUB_STATE_IND>-1</HUB_STATE_IND>
    </DATA>
</DATAAREA>
</SIP_EVENT>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Insert メッセージ

以下に Insert メッセージの例を示します。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Insert</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR</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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Merge メッセージ

以下に示すのは、Merge メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Merge</eventType>
    <baseObjectUid>BASE_OBJECT.C_CONTACT</baseObjectUid>
    <packageUid>PACKAGE.CONTACT_PKG</packageUid>
    <orsId>localhost-mrm-CMX_OR</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>
  <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Merge Update メッセージ

以下に示すのは、Merge Update メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Merge Update</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>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>

```

```

        </contactPkg>
    </mergeUpdateEvent>
</siperianEvent>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

No Action メッセージ

以下に示すのは、No Action メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

PendingInsert メッセージ

以下に示すのは、PendingInsert メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>
    <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

PendingUpdate メッセージ

以下に示すのは、PendingUpdate メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Pending Update</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>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>
    </contactPkg>
  </pendingUpdateEvent>
</siperianEvent>

```

```

        <consolidationInd>4</consolidationInd>
        <lastRowidSystem>CRM                </lastRowidSystem>
        <dirtyInd>1</dirtyInd>
        <firstName>John</firstName>
        <lastName>Smith</lastName>
        <hubStateInd>1</hubStateInd>
    </contactPkg>
</pendingUpdateEvent>
</siperianEvent>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

PendingUpdateXref メッセージ

以下に示すのは、PendingUpdateXref メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Pending Update XREF</eventType>
    <baseObjectId>BASE_OBJECT.C_CONTACT</baseObjectId>
    <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Unmerge メッセージ

以下に示すのは、マージ解除メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>

```

```

<eventMetadata>
  <eventType>UnMerge</eventType>
  <baseObjectId>BASE_OBJECT.C_CONTACT</baseObjectId>
  <packageId>PACKAGE.CONTACT_PKG</packageId>
  <orsId>localhost-mrm-CMX_ORS</orsId>
  <triggerId>MESSAGE_QUEUE_RULE.ContactUpdate</triggerId>
  <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Update メッセージ

以下に示すのは、更新メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<siperianEvent>
  <eventMetadata>
    <eventType>Update</eventType>
    <baseObjectId>BASE_OBJECT.C_CONTACT</baseObjectId>
    <packageId>PACKAGE.CONTACT_PKG</packageId>
    <orsId>localhost-mrm-CMX_ORS</orsId>
    <triggerId>MESSAGE_QUEUE_RULE.ContactUpdate</triggerId>
    <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>
    </contactPkg>
  </updateEvent>
</siperianEvent>

```

```

        <lastRowidSystem>CRM                </lastRowidSystem>
        <dirtyInd>1</dirtyInd>
        <firstName>Joe</firstName>
        <lastName>Black</lastName>
    </contactPkg>
</updateEvent>
</siperianEvent>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Update XREF メッセージ

以下に示すのは、XREF の更新メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

XRefDelete メッセージ

以下に示すのは、XRefDelete メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>
  </eventMetadata>
</siperianEvent>

```

```

    <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>
      <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

XRefSetToDelete メッセージ

以下に示すのは、XRefSetToDelete メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

従来の JMS メッセージの XML リファレンス

Informatica MDM Hub の実装がレガシー JMS メッセージを必要とする場合、現在のバージョンの XML メッセージ形式ではなく、従来の XML メッセージ形式を使用する必要があります。レガシー XML メッセージを使用するには、[メッセージキュー] ツールの **【従来の XML を使用する】** チェックボックスを選択します。

従来の XML のメッセージフィールド

メッセージのデータ領域の内容は、トリガで指定したパッケージによって決まります。

データ領域には以下のメッセージフィールドがあります。

フィールド	説明
ACTION	アクションの種類: Insert、Update、Update XREF、Accept as Unique、Merge、Unmerge、または Merge Update です。
MESSAGE_ID	メッセージの ID。ID の値は、C_REPOS_MQ_DATA_CHANGE テーブルの値に対応しています。
MESSAGE_DATE	イベントが生成された時刻。
TABLE_NAME	このアクションの影響を受けるベースオブジェクトテーブルまたは相互参照オブジェクトテーブルの名前です。
RULE_NAME	このメッセージを生成したイベントをトリガしたルールの名前。
RULE_ID	このメッセージを生成したイベントをトリガしたルールの ID。
ROWID_OBJECT	このアクションの影響を受けるベースオブジェクトの一意のキー。
MERGED_OBJECTS	マージで消失したレコードの ROWID_OBJECT 値のリスト。このフィールドは、MERGE イベントの場合のみメッセージに組み込まれます。
SOURCE_XREF	UPDATE イベントをトリガした相互参照の SYSTEM 値および PKEY_SRC_OBJECT 値です。このフィールドは、UPDATE イベントの場合のみメッセージに組み込まれます。
XREFS	このベースオブジェクトの出力システムにあるすべての相互参照に対する SYSTEM 値および PKEY_SRC_OBJECT 値のリストです。

従来の XML のメッセージのフィルタリング

MessageType という名前のカスタム JMS ヘッダーを使用して、入力メッセージをメッセージタイプに基づいてフィルタリングすることができます。このメッセージヘッダーには、以下のメッセージタイプが指定されています。

メッセージタイプ	説明
SIP_EVENT	イベント通知メッセージ。
<serviceNameReturn>	Services Integration Framework (SIF) 応答の場合は、次に示す GET 要求への応答のように、応答が SIF 要求の名前で始まります。 <getReturn> records</message> ... <message>The GET was executed successfully - retrieved 1 <recordKey> <ROWID>2</ROWID> </recordKey>

従来の XML のメッセージの例

次の例のメッセージは参照として使用できます。

Accept as Unique メッセージ

Accept As Unique メッセージの例を次に示します。

```
<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>SVR1.8E0</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>
```

```
</DATAAREA>
</SIP_EVENT>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

BO Delete メッセージ

次のコードは BO delete メッセージの例です。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<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>
      <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>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

BO set to Delete

次のコードは BO set to Delete メッセージの例です。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<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>
    <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Delete メッセージ

次のコードは Delete メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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_OR</DATABASE>
    <XREFS>
      <XREF>
        <SYSTEM>CRM</SYSTEM>
        <PKEY_SRC_OBJECT />
      </XREF>
    </XREFS>
  </CONTROLAREA>
</SIP_EVENT>

```

```

</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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Insert メッセージ

次のコードは、Insert メッセージの例です。

```

<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>

```

```
</DATAAREA>
</SIP_EVENT>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Merge メッセージ

次のコードは、Merge メッセージの例です。

```
<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>SVR1.8E0</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>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Merge Update メッセージ

次のコードは、Merge Update メッセージの例です。

```
<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>SVR1.8E0</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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Pending Insert メッセージ

次のコードは、Pending Insert メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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_ORS</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 />
    </DATA>
  </DATAAREA>
</SIP_EVENT>

```

```

        <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Pending Update メッセージ

次のコードは、Pending Update メッセージの例です。

```

<?xml version="1.0" encoding="UTF-16LE"?>
<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>
        <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Pending Update XREF メッセージ

次のコードは、Pending Update XREF メッセージの例です。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<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>
      <LAST_NAME>Smith</LAST_NAME>
      <HUB_STATE_IND>1</HUB_STATE_IND>
      <CITY />
      <STATE />
    </DATA>
  </DATAAREA>
</SIP_EVENT>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Update メッセージ

次に示すコードは、更新メッセージの例です。

```
<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>SVR1.8E0</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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Update XREF メッセージ

次に示すコードは、XREF の更新メッセージの例です。

```

<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>SVR1.8E0</RULE_ID>
    <ROWID_OBJECT>74          </ROWID_OBJECT>
    <SOURCE_XREF>
      <SYSTEM>Admin</SYSTEM>
      <PKEY_SRC_OBJECT>196          </PKEY_SRC_OBJECT>
    </SOURCE_XREF>
  </CONTROLAREA>
  <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

Unmerge メッセージ

次のコードは、マージ解除メッセージの例です。

```

<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-edselp2-CMX_AT</DATABASE>
    <XREFS>
      <XREF>
        <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>

```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

XREF Delete メッセージ

次のコードは、XREF delete メッセージの例です。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<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</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>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

削除する XREF のセット

次に示すコードは、メッセージを削除するように設定された XREF の例です。

```
<?xml version="1.0" encoding="UTF-16LE"?>
<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_OR</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>
```

実際に表示されるメッセージは、これとは多少異なります。データ部分に実際のデータが反映され、フィールドにもパッケージの内容が反映されます。

パート V: Informatica MDM Hub のプロセスの実行

この部には、以下の章があります。

- [バッチジョブの使用, 537](#) ページ
- [ユーザー出口, 586](#) ページ

第 27 章

バッチジョブの使用

この章では、以下の項目について説明します。

- [バッチジョブの使用の概要, 537 ページ](#)
- [バッチジョブのスレッド設定, 538 ページ](#)
- [バッチジョブの起動, 539 ページ](#)
- [バッチジョブが使用するサポートテーブル, 540 ページ](#)
- [バッチジョブの順次実行, 540 ページ](#)
- [バッチジョブの作業に関するベストプラクティス, 541 ページ](#)
- [Oracle 環境での統計収集の並列度の制限, 541 ページ](#)
- [バッチジョブの作成, 542 ページ](#)
- [情報提供用バッチジョブ \(Hub コンソールで実行されないバッチジョブ\) , 543 ページ](#)
- [プロセスサーバーの設定, 544 ページ](#)
- [バッチビューアツールを使用したバッチジョブの実行, 544 ページ](#)
- [バッチグループツールを使用したバッチジョブの実行, 552 ページ](#)
- [バッチジョブのリファレンス, 561 ページ](#)

バッチジョブの使用の概要

MDM Hub バッチジョブは、Hub コンソールのバッチビューアツールとバッチグループツールを使用して設定および実行できます。

MDM Hub では、バッチジョブは実行されるときに個々の作業単位を実行するプログラムを意味します。個々の作業単位はプロセスと呼ばれます。例えば、一致プロセスは一致ジョブによって実行されます。MDM Hub は、一致候補を検索し、一致候補に一致ルールを適用して一致を生成し、自動統合または手動統合が行われるように一致をキューに追加します。マージスタイルのベースオブジェクトの場合、自動統合は自動マージジョブで処理されます。手動統合は手動マージジョブで処理されます。

MDM Hub バッチジョブはすべてマルチスレッドです。マルチスレッドでは、単一のプロセスのコンテキスト内に存在する複数のスレッドを許可します。また、MDM Hub バッチジョブは、親ベースオブジェクトの一致パスにあるすべての子ベースオブジェクト上で並列で実行できます。

バッチジョブを開始する前に、次の作業を完了しておく必要があります。

- Informatica MDM Hub をインストールし、Hub ストアを作成する。
- スキーマを構築する。

バッチジョブのスレッド設定

マルチスレッドは一般的なプログラミングモデルで、単一プロセスのコンテキスト内で複数スレッドの処理が可能です。ロードジョブや再検証ジョブをはじめ、MDM Hub のバッチジョブはいずれもマルチスレッドです。

バッチジョブを実行するとき、MDM Hub はバッチ処理のためにキューに追加されているレコードを、MDM Hub で平行処理できる複数のブロックに分割します。各バッチジョブに使用するスレッドの数は `cmxserver.properties` ファイル内で設定できます。

MDM Hub が親ベースオブジェクトに対してロードジョブを実行すると、MDM Hub は対応する子ベースオブジェクトにバッチロックを適用します。子ベースオブジェクトにバッチロックが適用されている場合、別の親が並行してロードまたはマージジョブを実行することはできません。MDM Hub が子ベースオブジェクトにバッチロックを適用するのは、子と親ベースオブジェクトの間に一意のキーリレーションがある場合のみです。

マルチスレッドバッチジョブのプロセス

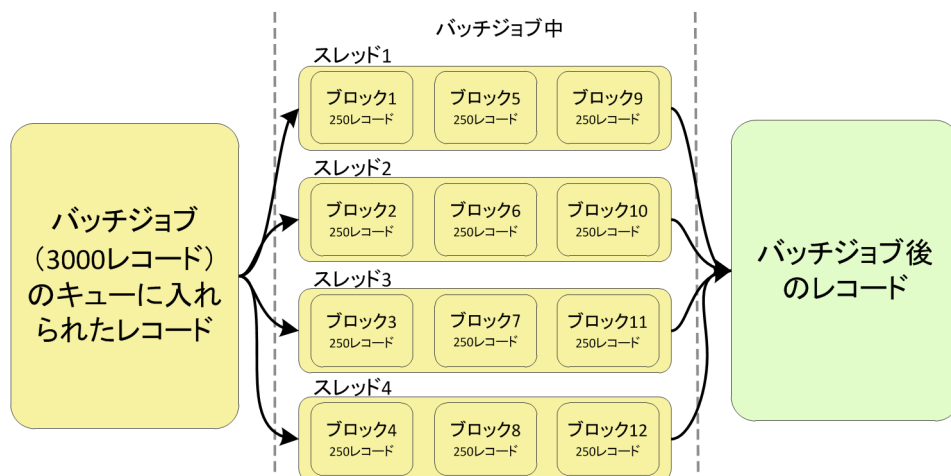
MDM Hub がバッチジョブを処理する際、レコードを処理するスレッドとブロックを並行して作成します。バッチジョブを設定する際、1つのバッチジョブでレコードを処理するスレッドの数とブロックのサイズを設定します。

1. バッチジョブの実行時に、MDM Hub はレコードの総数を設定済みのブロックサイズで割ります。
2. MDM Hub は設定されたスレッドの数に基づいてスレッドを作成します。
3. MDM Hub は1つのブロックを1つのスレッドに割り当てて処理します。
4. スレッドがブロックの処理を終了した後で、MDM Hub はキューにある未処理のブロックをそのスレッドに割り当てます。すべてのブロックが処理されるまで、この手順が繰り返されます。

マルチスレッドバッチジョブの例

MDM Hub は設定されたブロックサイズを使用して、各ブロックで処理する必要があるレコードの数を決定します。MDM Hub は処理するレコードの総数をブロックサイズで割ります。

例えば、バッチに含まれる処理の必要なレコードの数が3000で、設定したスレッドの数が4、ブロックサイズが250であるとして、MDM Hub はレコードの総数をブロックサイズで割ります。その結果、12個のブロックにそれぞれ250個のレコードが含まれます。次にMDM Hub はブロックを各スレッドに割り当てて処理します。スレッドがブロックの処理を終了すると、キューにある次のブロックがそのスレッドに割り当てられます。MDM Hub は、すべてのブロックの処理が終わるまでブロックをスレッドに割り当て続けます。



マルチスレッドバッチのパフォーマンス

スレッド数がお使いの環境で効率的に処理できるスレッド数よりも多い場合は、バッチパフォーマンスが低下する可能性があります。また、パフォーマンスとブロックサイズは、データベースの環境に依存しています。

MDM Hub では、複数の子ベースオブジェクトに対するマルチスレッドロードジョブまたはマージジョブは、順番に実行されるように設定する必要があります。

マルチスレッドバッチジョブのプロパティ

使用するスレッドの数と、マルチスレッドバッチジョブを処理するブロックサイズを設定する必要があります。

各バッチジョブに使用するスレッドの数は `cmxserver.properties` ファイル内で設定します。次の表に、マルチスレッドバッチジョブを設定するプロパティを示します。

プロパティ	説明
<code>cmx.server.automerge.threads_per_job</code>	MDM Hub が自動マージバッチジョブの処理に使用するスレッドの数。デフォルトは 1 です。
<code>cmx.server.automerge.block_size</code>	自動マージジョブに対して各ブロックで処理するレコードの数。デフォルトは 250 です。
<code>cmx.server.batch.threads_per_job</code>	MDM Hub がロード、BVT の再計算、バッチジョブの再検証の処理に使用するスレッドの数。デフォルトは 10 です。 <code>cmx.server.batch.threads_per_job</code> の値は、すべてのプロセッサで使用する可能なバッチ処理のスレッドの総数以下にする必要があります。
<code>cmx.server.batch.load.block_size</code>	ロードジョブに対して各ブロックで処理するレコードの数。デフォルトは 250 です。
<code>cmx.server.batch.recalculate.block_size</code>	BVT の再計算とバッチジョブの再検証に対して各ブロックで処理するレコードの数。デフォルトは 250 です。

バッチジョブの起動

バッチジョブは個別に、または MDM Hub コンソールやサービス統合フレームワーク API を使用してグループとして起動することができます。

次のツールを使用して、バッチジョブを起動することができます。

バッチビューアツール

MDM Hub コンソールでバッチビューアツールを使用して、バッチジョブを個別に起動します。

バッチグループツール

MDM Hub コンソールでバッチグループツールを使用して、複数のバッチジョブを 1 つのグループとして起動します。バッチグループツールを使用すると、バッチジョブの起動のシーケンスを設定したり、複数のバッチジョブを並行して実行することができます。

個別のサービス統合フレームワーク API

MDM Hub コンソールで使用する可能な各バッチジョブには、対応するサービス統合フレームワーク API があります。API を使用して個々のバッチジョブを起動します。

サービス統合フレームワーク API の ExecuteBatchGroup

サービス統合フレームワーク API の ExecuteBatchGroup を使用して、バッチグループを起動します。

サービス統合フレームワーク API の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

バッチジョブが使用するサポートテーブル

以下の表に、Informatica MDM Hub のバッチジョブが使用するさまざまなサポートテーブルを示します。

- ランディングテーブル。
- ステージングテーブル。
- RAW テーブル。
- 却下テーブル。
- 一致キーテーブル。
- 一致テーブル。
- システム制御テーブルと履歴テーブル。
- XREF テーブル。

バッチジョブの順次実行

特定のバッチジョブでは、他のジョブが最初に完了している必要があります。例えば、ベースオブジェクトのランディングテーブルは、バッチジョブの実行前に作成しておく必要があります。同様に、ベースオブジェクトの一致ジョブを実行する前に、対応するステージジョブとロードジョブを実行する必要があります。また、ベースオブジェクトに依存関係がある場合（例えば親テーブルの子である場合や、他のベースオブジェクトを指す外部キーリレーションがある場合）、ベースオブジェクトが依存するテーブルに対するバッチジョブを最初に実行する必要があります。管理者や組織は、バッチジョブの実行前に完了させるバッチプロセスおよび依存関係を特定する、管理や運用プランの開発のベストプラクティスを考慮する必要があります。

バッチジョブの実行前のランディングテーブルの生成

Informatica MDM Hub バッチジョブが実行するタスクの 1 つが、ランディングテーブルから Informatica MDM Hub の適切なターゲットの場所にデータを移動することです。したがって、Informatica MDM Hub のバッチジョブを実行する前に、ソースシステムまたは ETL ツールでランディングテーブルにデータを書き込む必要があります。ランディングテーブルは Informatica MDM Hub のバッチロード用インタフェースです。ユーザーがデータをランディングテーブルに配信し、Informatica MDM Hub バッチプロセスがそのデータを操作して適切な場所にコピーします。詳細については、『Informatica MDM Hub 概要』で Informatica MDM Hub のデータ管理プロセスの説明を参照してください。

一致ジョブおよび後続の統合ジョブ

バッチジョブは、一定の順序で実行する必要があります。例えば、統合プロセスを実行する前にベースオブジェクトに対して一致ジョブを実行する必要があります。マージスタイルのベースオブジェクトの場合は、ベースオブジェクト内のすべてのレコードで一致がチェックされるまで、あるいは手動で統合できるレコード数の

上限に達するまで一致ジョブと自動マージジョブを繰り返し実行する、自動一致とマージジョブを実行することができます。

最初に親テーブルのデータをロードする

一般的なガイドラインとして、すべての親テーブル（他のテーブルから参照されるテーブル）を最初にロードする必要があります。

外部キーリレーションを持つオブジェクトへのデータのロード

2つのテーブルの間に外部キーリレーションがある場合は、参照先のテーブルを最初にロードし、参照元のテーブルを2番目にロードする必要があります。Informatica MDM Hubでは、次の外部キーリレーションが発生する可能性があります。あるベースオブジェクト（外部キーを持つ子）から別のベースオブジェクト（外部キーを持つ親）へのリレーション。

ほとんどの場合、これらのジョブは定期的に行われるようにスケジュール設定します。

バッチジョブの作業に関するベストプラクティス

バッチジョブの設計および計画にあたっては、以下の点に注意してください。

- スキーマを定義する。
スキーマは、Informatica MDM Hubのすべてのタスクの基盤となるものです。スキーマがないと、バッチジョブでは何も実行されません。
- Oracle環境では、Oracleで統計を収集する並列度を制限して、過剰な数のプロセスを使用することを避けます。統計を収集するために過剰な数のプロセスを使用すると、Hubの処理に使用されるリソースが不十分なままになることがあります。[「Oracle環境での統計収集の並列度の制限」](#)（ページ 541）を参照してください。
- ステージジョブを実行する前にマッピングを定義する。
ステージジョブで実行される変換は、マッピングで定義されます。マッピングが定義されていないと、ステージジョブのステージングプロセスで変換が行われません。
- 一致ジョブを実行する前に一致ルールを定義する。
一致ルールがないと、一致ジョブで一致が生成されません。
- プロダクション環境でジョブを実行する前に以下を実行する。
 - 小規模なデータセットでテストを行う。
 - クレンジングエンジンおよびその他のコンポーネントのテストを行って、各コンポーネントが期待どおりに動作するかどうかを確認する。
 - 各コンポーネントを個別にテストした後、統合システム全体のテストを行って、システム全体が期待どおりに動作するかどうかを確認する。

Oracle環境での統計収集の並列度の制限

Oracle環境での統計収集の並列度を制限して、Hubプロセスのパフォーマンスに悪影響を与えないようにする必要があります。

並列度を設定するには、データベース管理者特権を持つユーザーとしてログインする必要があります。

統計収集に適切な並列度を割り当てるには、次の手順を実行します。

1. 次の式を使用して、適切な並列度を計算します。
$$\text{APPROPRIATE PARALLEL DEGREE} = \text{CPU_COUNT} * \text{PARALLEL_THREADS_PER_CPU}$$

CPU_COUNT は、Oracle で使用できる CPU 数です。PARALLEL_THREADS_PER_CPU は通常 2 です。
注: サーバーに多数の CPU がある場合、CPU コア数以下の並列度の値を選択します。
注: 同一サーバー上で他のアプリケーションが Hub として実行されている場合、Hub に割り当て可能な CPU リソース数を決定し、適切な並列度を設定する際に、この数値を使用します。
2. 次の SQL*Plus コマンドを実行して、現在の並列度の設定を確認します。

```
select DBMS_STATS.GET_PREFS( 'DEGREE' ) from dual;
```
3. 必要に応じて、次のいずれかの SQL*Plus コマンドを実行して、適切な並列度を設定します。
 - Oracle 10g の場合: `DBMS_STATS.SET_PARAM ('DEGREE', <appropriate parallel degree value>) ;`
 - Oracle 11g の場合: `DBMS_STATS.SET_GLOBAL_PREFS ('DEGREE', <appropriate parallel degree value>) ;`
4. テーブルが大きい場合は、次の SQL コマンドを実行し、新しい並列度の値を使用してパフォーマンスをテストします。

```
DBMS_STATS.GATHER_TABLE_STATS
```
5. 待機イベントが解消され、許容できるパフォーマンスになるまで、毎回並列度を下げてステップ 3 と 4 を繰り返します。

バッチジョブの作成

バッチジョブは、以下の 2 つのいずれかの方法で作成されます。

- Hub Store の設定時に自動的に作成される
- Informatica MDM Hub の設定で特定の変更（ベースオブジェクトの信頼度設定の変更など）を行うと作成される

自動的に作成されるバッチジョブ

Hub ストアを設定すると、MDM Hub によって次のタイプのバッチジョブが自動的に作成されます。

- 自動一致とマージジョブ
- オートリンクジョブ
- 自動マージジョブ
- BVT スナップショットジョブ
- 外部一致ジョブ
- 一致トークンの生成ジョブ
- スマート検索データジョブの初期インデックス処理
- ロードジョブ
- 手動リンクジョブ
- 手動マージジョブ
- 手動リンク解除ジョブ
- 手動マージ解除ジョブ

- 一致ジョブ
- 一致分析ジョブ
- 昇格ジョブ
- ステージジョブ

変更の発生時に作成されるバッチジョブ

初期ロード後に一致設定およびマージ設定に変更を加えたり、プロパティを設定したり、または信頼設定を有効にすると、MDM Hub によって次のバッチジョブが自動的に作成されます。

- 一致しないレコードを一意とする
- キー一致ジョブ
- 一致テーブルのリセットジョブ
- 再検証ジョブ（カラムに対して検証を有効にしている場合）
- 同期ジョブ

情報提供用バッチジョブ（Hub コンソールで実行されないバッチジョブ）

以下のバッチジョブは情報提供用であり、Hub コンソールから手動で実行することはできません。

- 一致しないレコードを一意とする
- BVT スナップショットジョブ
- バッチマージ解除ジョブ
- 手動リンクジョブ
- 手動マージジョブ
- 手動リンク解除ジョブ
- 手動マージ解除ジョブ
- リンクスタイルからマージスタイルへの移行ジョブ
- 複数マージジョブ
- 一致テーブルのリセットジョブ

その他のバッチジョブ

- Hub 削除ジョブ

プロセスサーバーの設定

プロセスサーバーは、ロード、BVT の再計算、再検証、削除、バッチアンマージなどのバッチジョブを実行します。プロセスサーバーはアプリケーションサーバー環境にデプロイされます。バッチジョブを実行するには、プロセスサーバーの設定が必要です。

オペレーショナルリファレンスストアごとに、複数のプロセスサーバーを設定できます。複数のホストにプロセスサーバーデプロイすることで、複数の CPU 間で処理の負荷を分散させ、バッチジョブを同時に実行することができます。また、プロセスサーバーはマルチスレッド対応であるため、インスタンスごとに複数の要求を同時に処理することもできます。

関連項目：

- [「プロセスサーバーのプロパティ」 \(ページ 343\)](#)
- [「プロセスサーバーの追加」 \(ページ 345\)](#)
- [「プロセスサーバープロパティの編集」 \(ページ 346\)](#)
- [「プロセスサーバーの削除」 \(ページ 346\)](#)

バッチビューアツールを使用したバッチジョブの実行

この節では、Hub コンソールでバッチビューアツールを使用して、バッチジョブを個別に実行する方法について説明します。バッチジョブをグループで実行する手順は、[「バッチグループツールを使用したバッチジョブの実行」 \(ページ 552\)](#)を参照してください。

バッチビューアツール

バッチビューアツールでは、バッチジョブを個別に実行したり、ジョブ実行ログを表示したりできます。バッチビューアは、1 つのジョブの実行を開始する場合や、頻繁に実行する必要がないジョブ（信頼の設定の変更後に実行する同期ジョブ）を実行する場合に便利です。ジョブ実行ログには、ジョブ完了ステータスと関連メッセージ（成功、失敗、警告など）が表示されます。該当する場合は、ジョブの統計も表示されます。

注：バッチビューアでは、自動スケジュールは使用できません。

バッチビューアツールの起動

バッチビューアツールを起動する手順

- ▶ Hub コンソールでユーティリティワークベンチを展開して、**【バッチビューア】** をクリックします。バッチビューアツールが表示されます。

テーブル、データ、またはプロシージャタイプ別のグループ化

ナビゲーションツリーの下部にある **【グループ化】** コントロールを右クリックすることで、このツリーの最上位ビューを変更できます。

注：グレーで表示されたチェックマーク付きの項目が、現在選択されています。

次のいずれかのオプションを選択します。

グループ化オプション	説明
テーブル	階層に以下の各レベルで項目を表示します。 <ul style="list-style-type: none">- 1 番上のレベル: テーブル- 2 番目のレベル: プロシージャタイプ- 3 番目のレベル: batch job: バッチジョブ- 4 番目のレベル: 日付/タイムスタンプ
日付	階層に以下の各レベルで項目を表示します。 <ul style="list-style-type: none">- 1 番上のレベル: 日付/タイムスタンプ- 2 番目のレベル: 日付/タイムスタンプ別バッチジョブ
プロシージャタイプ	階層に以下の各レベルで項目を表示します。 <ul style="list-style-type: none">- 1 番上のレベル: プロシージャタイプ- 2 番目のレベル: batch job: バッチジョブ- 3 番目のレベル: 日付/タイムスタンプ

バッチジョブの手動実行

バッチジョブを手動で実行する手順

1. 実行するバッチジョブを選択します。
2. バッチジョブを実行します。

バッチジョブの選択

実行するバッチジョブを選択する手順

1. バッチビューアツールを起動します。
バッチビューアツリーに、バッチジョブのリストが表示されます。リストはプロシージャタイプ別にグループ分けされています。
2. ツリーを展開して、実行するバッチジョブを表示し、クリックして選択します。
バッチビューアに、選択したバッチジョブのプロパティとコマンドボタンが表示されます。

バッチジョブのプロパティ

バッチビューアツールのプロパティペインにバッチジョブのプロパティを表示できます。

識別情報テーブルには次のプロパティが表示されます。

名前

バッチジョブの名前。

説明

バッチジョブの説明。

ステータステーブルには次のプロパティが表示されます。

現在のステータス

バッチジョブの現在のステータス。バッチジョブのステータスは、次のうちのいずれかになります。

- 実行中
- 未完了

- 完了
- 実行しない
- <バッチジョブ> 成功
- バッチジョブの失敗についての説明

バッチジョブの実行前に設定するオプション

バッチジョブのタイプによっては、実行前に設定可能な追加のフィールドがあります。

フィールド	対象	説明
すべての一致トークンを再生成	一致トークンの生成ジョブ	一致トークンの生成範囲を制御します。ベースオブジェクト全体をトークン化するか（チェック）、ベースオブジェクト内の再トークン化が必要とフラグ付けされているレコードのみをトークン化します（チェック解除）。
更新の強制	ロードジョブ	選択した場合、ロードジョブは、レコードがロード済みかどうかにかかわらず、ステージングテーブルからベースオブジェクトにレコードをロードして更新します。
一致設定	一致ジョブ	この一致ジョブに使用する一致ルールセットを選択できます。

バッチジョブ用のコマンドボタン

バッチジョブを選択した後、以下のコマンドボタンをクリックできます。

ボタン	説明
バッチの実行	選択したバッチジョブを実行します。
履歴のクリア	バッチビューアのジョブ実行履歴をクリアします。
ステータスを未完了に設定	現在実行中のバッチジョブのステータスを未完了に設定します。
ステータスの更新	現在実行中のバッチジョブのステータスの表示を更新します。

バッチジョブの実行

注: バッチジョブが実行されている間は、アプリケーションサーバーを稼働させておく必要があります。

バッチビューアでバッチジョブを実行する手順

1. バッチビューアで、実行するバッチジョブを選択します。
2. 右側のパネルで **【バッチの実行】** をクリックします（または左側のパネルでジョブを右クリックし、ポップアップメニューから **【実行】** を選択します）。

ジョブの現在のステータスが実行中である場合は、**【バッチの実行】** ボタンが使用不能になります。バッチジョブを再度実行するには、バッチジョブが終了するまで待つ必要があります。

ステータスの更新

バッチジョブの実行中、**【ステータスの更新】** をクリックして、ステータスが変更されているかどうかを確認できます。

ジョブステータスを未完了に設定

ごくまれに、[ステータスを未完了に設定] をクリックして実行中のジョブのステータスを変更し、後でジョブを再実行することがあります。この作業は、バッチジョブの実行が停止したにもかかわらず（サーバーの再起動やクラッシュなどのエラーのため）、メタデータでのジョブアプリケーションロックが原因で Informatica MDM Hub がジョブの停止を検出していない場合にのみ行います。現在のステータスが**実行中**であるのに、データベース、アプリケーションサーバー、およびログにアクティビティが示されていない場合は、この状況が問題であることがわかります。この状況が発生した場合は、バッチジョブをもう一度実行できるように、このボタンをクリックしてジョブアプリケーションロックを解除します。解除しないと、このバッチジョブを実行することはできません。ステータスを未完了に設定すると、バッチジョブのステータスは更新されますが、ジョブは強制終了されません。ジョブステータスを未完了に設定した後、関連するデータベースプロセスも停止する必要があります。






注: このオプションは、ユーザー ID に Informatica Administrator 権限がある場合にのみ使用できます。

ジョブ実行ログの表示

Informatica MDM Hub では、バッチジョブを実行するたびにジョブ実行ログが作成されます。

ジョブ実行ステータス

ジョブ実行ログの各エントリには、次のいずれかのステータス値が設定されます。

アイコン	説明
	バッチジョブは現在実行中です。
	バッチジョブが正常に終了しました。
	バッチジョブが正常に終了しましたが、追加情報があります。例えば、ステージおよびロードの場合は、一部のレコードが却下されたことを示します。一致ジョブの場合は、ベースオブジェクトが空であるか、一致させるレコードがなくなったことを示します。
	バッチジョブが失敗しました。
	バッチジョブのステータスが、手動で「実行中」から「未完了」に変更されました。

バッチジョブのジョブ実行ログの表示

バッチジョブのジョブ実行ログを表示する手順

1. バッチビューアツールを起動します。
2. ツリーを展開し、目的のジョブ実行ログを表示してクリックします。
バッチビューアに、選択したジョブ実行ログの画面が表示されます。

ジョブ実行ログエントリのプロパティ

ジョブ実行ログのエントリごとに、バッチビューアで以下の情報が表示されます。

フィールド	説明
識別子	このバッチジョブの識別情報。 C_REPOS_TABLE_OBJECT_V テーブルに格納されています。
名前	このジョブ実行ログの名前。バッチジョブが開始された日時。
説明	次の形式で示されるこのバッチジョブの説明。 <i>JobName</i> for / from <i>BaseObjectName</i> 例: - Load from Consumer_Credit_Stg - Match for Address
Source system (ソースシステム)	以下のいずれかになります。 - 処理されたデータのソースシステム - 管理
ソーステーブル	処理されたデータのソーステーブル。
ステータス	このバッチジョブのステータス情報。
現在のステータス	このバッチジョブの現在のステータス。エラーが発生した場合は、エラーに関する情報が表示されます。
メトリック	このバッチジョブのメトリック。
[Various]	バッチジョブの実行中に収集された統計（該当する場合） - バッチジョブメトリック - 自動一致とマージのメトリック - 自動マージのメトリック - ロードジョブメトリック - 一致ジョブメトリック - 一致分析ジョブメトリック - ステージジョブメトリック - 昇格ジョブメトリック
時間	このバッチジョブのタイムスタンプ。
開始	このバッチジョブが開始された日時。
停止	このバッチジョブが終了した日時。
経過時間	このバッチジョブの実行の経過時間。

バッチジョブのメトリックについて

Informatica MDM Hub では、バッチジョブの実行時にさまざまな統計を収集します。実際に返されるメトリックはバッチジョブごとに異なります。バッチジョブが完了すると、その統計が

C_REPOS_JOB_METRIC_TYPE に登録されます。各ジョブに統計が複数ある場合もあります。ジョブのメトリックには以下のものがあります。

メトリック名	説明
総レコード数	バッチジョブによって処理されたレコードの総数。
挿入済み	バッチジョブによってターゲットオブジェクトに挿入されたレコードの数。
更新済み	バッチジョブによって更新されたターゲットオブジェクト内のレコードの数。
アクションなし	アクションが実行されなかったレコードの数（レコードがベースオブジェクトにすでに存在していたため）。
一致したレコード	バッチジョブで一致したレコードの数。
平均一致数	一致の平均数。
更新された相互参照	ベースオブジェクトの相互参照テーブルを更新したレコードの数。増分ロード中にレコードをロードした場合、レコードは事前に統合されます。レコードは、ベースオブジェクトではなく、相互参照テーブルにのみ存在します。
トークン化されたレコード	バッチジョブでトークン化されたレコードの数。スキーマツールで「ロード時に一致トークンを生成する」チェックボックスが選択されている場合にのみ適用できます。
一致フラグが設定されたレコード	一致フラグが設定されたレコードの数。
自動マージされたレコード	自動マージバッチジョブでマージされたレコードの数。
却下されたレコード	バッチジョブで却下されたレコードの数。
マージ解除されたソースレコード	バッチジョブでマージされなかったソースレコードの数。
マージに関与した XREF レコード	
一意として受け入れられたレコード	バッチジョブで一意のレコードとして受け入れられるレコードの数。 ベースオブジェクトの「一致/マージ設定」の設定で「一致しないすべての行を一意とする」オプションが有効になっている場合にのみ適用できます。
自動マージ用にキューに追加	自動一致とマージジョブで実行された一致ジョブによって自動マージ用にキューに追加されるレコードの数。
手動マージ用にキューに追加	手動マージ用にキューに追加されるレコードの数。これらのレコードを処理するには、Hub コンソールのマージマネージャを使用します。詳細については、『 <i>Multidomain MDM のデータスチュワードガイド</i> 』を参照してください。
バックフィル信頼レコード	
ルックアップなし/無効な rowid_object レコード	ルックアップ情報がないか rowid_object レコードが無効なソースレコードの数。

メトリック名	説明
保留ステータスに移行したレコード	保持ステータスになったレコードの数。
(一致用に) 分析されたレコード	一致させるレコードの数。
必要な一致比較数	一致比較の数。
クレンジングされたレコードの総数	クレンジングされたレコードの数。
ランディングしたレコードの総数	ランディングテーブルに配置されたレコードの数。
無効な rowid_object が指定されたレコード	rowid_object が無効なレコードの数。
オートリンクされたレコード	オートリンクされたレコードの数。
最善データのスナップショット	最善データ (BVT) のスナップショット。
重複一致レコード	重複する一致レコードの数。
削除されたリンク	削除されたリンクの数。
再検証されたレコード	再検証されたレコードの数。
新規ステータスにリセットされたベースオブジェクトレコード	「新規」ステータスにリセットされたベースオブジェクトレコードの数。
一致に変換されたリンク	一致に変換されたリンクの数。
自動昇格したレコード	自動昇格したレコードの数。
削除された相互参照レコード	削除された相互参照レコードの数。
削除されたレコード	削除されたレコードの数。
無効なレコード	無効なレコードの数。
昇格されていないアクティブなレコード	昇格されていないアクティブなレコードの数。
昇格されていない保護されたレコード	昇格されていない保護されたレコードの数。
削除されたベースオブジェクトレコード	削除されたベースオブジェクトレコードの数。
挿入されたリンクレコード	
リンク解除されたリンクレコード	
マージされたリンクレコード	
作成されたグループ	
マージされたグループ	
処理された一致レコード	

メトリック名	説明
処理されたリンク管理レコード	
却下されたリンク管理レコード	
処理に失敗したレコード	処理されなかったレコードの数。
楽観的ロック検証に失敗したレコード	バッチジョブの実行時に他のプロセスによって変更されたレコードの数。
再インデックスされた検索データ	再インデックスされた検索データのレコード数。
検索データから削除済み	検索データから削除されたレコードの数。
検索データに対する削除/追加に失敗しました	検索データから削除されなかったレコードの数、または検索データに追加されなかったレコードの数。
インデックスされた BO 行の合計数	インデックスされたベースオブジェクト行の総数。

却下されたレコードの表示

ステージジョブの場合にのみ、バッチジョブの結果としてレコードが却下テーブルに書き込まれると、ジョブ実行ログに「却下の表示」ボタンが表示されます。

注: HUB_STATE_IND 値が無効である場合は、レコードが却下されます。

却下されたレコードとそれぞれの却下理由を表示する手順

1. **【却下の表示】** ボタンをクリックします。
バッチビューアに、却下されたレコードのテーブルが表示されます。
2. **【閉じる】** をクリックします。


バッチジョブの失敗した実行の処理

バッチジョブの実行が失敗した場合は、以下の手順を実行します。

- このバッチジョブの実行ログエントリを表示します。
- 診断用の情報を得るために、[現在のステータス] フィールドのエラーテキストを確認します。
- 必要に応じて対処方法を実行します。

Windows のクリップボードへの現在のステータスのコピー

バッチの現在のステータスを（ドキュメントや電子メールに貼り付けるために）Windows のクリップボードにコピーする手順

- ▶  ボタンをクリックします。

ジョブ実行ログエントリの削除

選択したジョブ実行ログを削除する手順

- ▶ ジョブのプロパティページの右上隅にある **【削除】** ボタンをクリックします。

ジョブ実行履歴のクリア

長期にわたってバッチジョブを実行していると、実行済みジョブのリストのサイズが非常に大きくなる可能性があります。そのため、余分なジョブ実行ログをこのリストから定期的に削除する必要があります。

注: ジョブ履歴をクリアする実際の手順は、ビュー（[テーブル別]、[日付別]、または [プロシージャタイプ別]）ごとに若干異なります。ここでは、[テーブル別] ビューを使用している場合の手順を示します。

ジョブ履歴をクリアする手順

- 1. バッチビューアツールを起動します。
- 2. バッチビューアで、ベースオブジェクトの下のツリーを展開します。
- 3. バッチジョブのタイプの下ツリーを展開します。
- 4. 履歴をクリアするジョブを選択します。
- 5. **履歴のクリア** をクリックします。
- 6. **はい** をクリックして、このバッチジョブの実行履歴をすべて削除することを確認します。

バッチグループツールを使用したバッチジョブの実行

この節では、Hub コンソールでバッチグループツールを使用して、バッチジョブをグループで実行する方法について説明します。バッチジョブを個別に実行するには、[「バッチビューアツールを使用したバッチジョブの実行」](#)（ページ 544）を参照してください。

バッチグループについて

バッチグループとは、1つのコマンドで実行できる個々のバッチジョブ（ステージジョブ、ロードジョブ、一致ジョブなど）の集合です。バッチグループ内の各バッチジョブは、他のジョブと順番に実行することも並行して実行することも可能です。バッチグループを設定および実行するには、バッチグループツールを使用します。

バッチグループツールで使用可能にすることができるカスタムのバッチジョブおよびバッチグループの開発の詳細については、[「バッチジョブのリファレンス」](#)（ページ 561）を参照してください。

注: Hub コンソールからオブジェクトを削除する場合（マッピングを削除する場合など）、バッチグループツールでは、そのオブジェクトに依存するバッチジョブ（ステージジョブなど）があると赤色で強調表示されます。バッチグループを再実行する前に、この問題を解決する必要があります。

シーケンシャルおよびパラレル実行

以下の方法でバッチジョブを実行できます。

実行方法	説明
シーケンシャル	バッチグループで一度に 1 つのバッチジョブを実行します。
パラレル	バッチグループで複数のバッチジョブを同時に並行して実行します。

実行パス

実行パスは、バッチグループ全体が実行されるときにバッチジョブが実行されるシーケンスです。

実行パスは開始ノードで始まり、終了ノードで終わります。バッチグループツールでは実行シーケンスは検証されません。実行シーケンスは、ユーザーが正しく指定する必要があります。

例えば、ベースオブジェクトに対するロードジョブを誤ってステージジョブより前に指定していても、バッチグループツールからはエラーが通知されません。

レベル

バッチグループでは、順番に実行される 1 つ以上のレベルで実行パスが構成されます。

レベルとは、1 つ以上のバッチジョブの集合です。

- レベルに複数のバッチジョブが含まれている場合は、これらのバッチジョブが並行して実行されます。
- レベルにバッチジョブが 1 つしか含まれていない場合は、そのバッチジョブが単独で実行されます。

レベル内のすべてのバッチジョブが完了しないと、バッチグループはシーケンスの次のタスクに進むことができません。

注: レベル内のすべてのバッチジョブは並列で実行されるため、同じレベル内のバッチジョブが依存関係を持たないようにする必要があります。例えば、ベースオブジェクトのステージジョブとロードジョブは、適切な順序で実行される別々のレベルに含まれている必要があります。

バッチグループツールの開始

バッチグループツールを開始する手順

- ▶ Hub コンソールでユーティリティワークベンチを展開して、**[バッチグループ]** をクリックします。
バッチグループツールが表示されます。

バッチグループツールは以下の領域で構成されています。

領域	説明
ナビゲーションツリー	バッチグループと実行ログの階層リスト。
プロパティペイン	プロパティおよびコマンド

バッチグループの設定

ここでは、バッチグループを追加、編集、および削除する方法について説明します。

バッチグループの追加

バッチグループを追加する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. バッチグループのツリーの **[バッチグループ]** ノードを右クリックし、ポップアップメニューから **[バッチグループの追加]** を選択します。

バッチグループのツリーに **[新しいバッチグループ]** が追加されます。

この時点では実行シーケンスは空です。実行シーケンスは、新しいバッチグループの追加後に設定します。

4. 以下の情報を指定します。

フィールド	説明
名前	このバッチグループの一意のわかりやすい名前を指定します。
説明	このバッチグループの説明を入力します。

5. **【保存】** ボタンをクリックして変更を保存します。

変更が保存され、バッチグループツールのナビゲーションツリーが更新されます。

新しいバッチグループにバッチジョブを追加する方法については、[「バッチグループレベルへのバッチジョブの割り当て」 \(ページ 556\)](#)を参照してください。

バッチグループのプロパティの編集

マッピンググループのプロパティを編集する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、編集するバッチグループを表示します。
4. 必要に応じて別のバッチグループ名を指定します。
5. 必要に応じて別の説明を指定します。
6. **【保存】** ボタンをクリックして変更を保存します。

バッチグループの削除

バッチグループを削除する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、削除するバッチグループを表示します。
4. 削除するバッチグループを右クリックし、**【バッチグループの削除】** をクリックします。
削除を確認するように求められます。
5. **【はい】** をクリックします。
バッチグループツールにより、削除したバッチグループがナビゲーションツリーから削除されます。

バッチグループのレベルの設定

バッチグループには、順番に実行される 1 つ以上のレベルが含まれます。ここでは、バッチグループのレベルを設定して実行シーケンスを指定する方法について説明します。

バッチグループに対するレベルの追加

バッチグループに対するレベルを追加する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。

- ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
- バッチグループのツリーで、任意のレベルを右クリックし、以下のいずれかのオプションを選択します。

コマンド	説明
レベルを上追加	選択した項目の上にこのバッチグループに対するレベルを追加します。
レベルを下追加	選択した項目の下にこのバッチグループに対するレベルを追加します。
レベルを上移動	このバッチグループレベルを前のレベルの上に移動します。
レベルを下移動	このバッチグループレベルを次のレベルの下に移動します。
このレベルを削除	このバッチグループレベルを削除します。

[バッチグループに追加するジョブの選択] ダイアログが表示されます。

- 追加するジョブのベースオブジェクトを展開します。
- 追加するジョブを選択します。
並行して実行するジョブを選択するには、Ctrl キーを押しながら各ジョブをクリックして選択します。
- [OK] をクリックします。選択したジョブがバッチグループに追加されます。
- [保存] ボタンをクリックして変更を保存します。

バッチグループからのレベルの削除

バッチグループからレベルを削除する手順

- バッチグループツールを起動します。
- 書き込みロックを取得します。
- ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
- バッチグループで、削除するレベルを右クリックして **【このレベルを削除】** を選択します。
削除の確認ダイアログが表示されます。
- 【はい】** をクリックします。
削除したレベルが、バッチグループから削除されます。

バッチグループ内で 1 つ上のレベルに移動する手順

バッチグループ内で 1 つ上のレベルに移動する手順

- バッチグループツールを起動します。
- 書き込みロックを取得します。
- ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
- バッチグループツリーで、上に移動するレベルを右クリックし、**【レベルを上移動】** を選択します。
バッチグループ内でレベルが上に移動します。

バッチグループ内で 1 つ下のレベルに移動する手順

バッチグループ内で 1 つ下のレベルに移動する手順

- バッチグループツールを起動します。

2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
4. バッチグループツリーで、下に移動するレベルを右クリックし、[レベルを下に移動] を選択します。
バッチグループ内でレベルが下に移動します。

バッチグループレベルへのバッチジョブの割り当て

バッチグループツールで扱うジョブは、Informatica MDM Hub のバッチジョブです。各レベルに 1 つ以上のバッチジョブが含まれます。レベルに複数のバッチジョブが含まれている場合は、それらのすべてのバッチジョブが並行して実行されます。

バッチグループレベルへのバッチジョブの追加

バッチグループにバッチジョブを追加する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
4. バッチグループのツリーで、ジョブを追加するレベルを右クリックし、[このレベルにジョブを追加...] を選択します。
[バッチグループに追加するジョブの選択] ダイアログが表示されます。
5. 追加するジョブのベースオブジェクトを展開します。
6. 追加するジョブを選択します。
7. 複数のジョブを同時に選択する場合（並行して実行する場合）は、Ctrl キーを押しながらジョブをクリックします。
8. [OK] をクリックします。
9. 変更を保存します。

選択したジョブがターゲットのレベルのボックスに追加されます。Informatica MDM Hub では、同じグループレベルのバッチジョブはすべて並行して実行されます。

バッチジョブのオプションの設定

バッチグループを設定するときは、特定の種類のバッチジョブに対してジョブオプションを設定できます。これらのジョブオプションの詳細については、[「バッチジョブの実行前に設定するオプション」](#) (ページ 546) を参照してください。

レベルからのバッチジョブの削除

レベルからバッチジョブを削除する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
4. バッチグループで、削除するジョブを右クリックして [ジョブの削除] を選択します。
削除の確認ダイアログが表示されます。
5. [はい] をクリックして、選択したジョブを削除します。
削除したジョブがこのバッチグループのレベルから削除されます。

バッチジョブを1つ上のレベルに移動する手順

バッチジョブを1つ上のレベルに移動する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
4. バッチグループで、上に移動するジョブを右クリックし、**[ジョブを上に移動]** を選択します。
選択したジョブがバッチグループの1つ上のレベルに移動します。

バッチジョブを1つ下のレベルに移動する手順

バッチジョブを1つ下のレベルに移動する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、設定するバッチグループを表示します。
4. バッチグループで、下に移動するジョブを右クリックし、**[ジョブを下に移動]** を選択します。
選択したジョブがバッチグループの1つ下のレベルに移動します。

バッチグループリストの更新

バッチグループリストを更新する手順

- ▶ ナビゲーションペインの任意の場所を右クリックして、**[更新]** をクリックします。

バッチグループツールを使用したバッチグループの実行

この節では、バッチグループツールでバッチグループの実行を管理する方法について説明します。

注: バッチグループが実行されている間は、アプリケーションサーバーを稼働させておく必要があります。

注: Hub コンソールからオブジェクトを削除する場合（マッピングを削除する場合など）、バッチグループツールでは、そのオブジェクトに依存するバッチジョブ（ステージジョブなど）があると赤色で強調表示されます。バッチグループを再実行する前に、この問題を解決する必要があります。

【制御およびログ】画面への移動

【制御およびログ】画面では、バッチグループの実行を制御したり、実行ログを表示することができます。

バッチグループの【制御およびログ】画面に移動する手順

1. バッチグループツールを起動します。
2. [バッチグループ] ツリーを展開して、実行するバッチグループを表示します。
3. バッチグループを展開して **【制御およびログ】** ノードをクリックします。
このバッチグループの【制御およびログ】画面が表示されます。

[制御およびログ] 画面のコンポーネント

この画面には以下のコンポーネントがあります。

コンポーネント	説明
ツールバー	バッチグループの実行を管理するためのコマンドボタン。
バッチグループのログ	このバッチグループの実行ログ。
バッチジョブのログ	このバッチグループの個々のバッチジョブの実行ログ。

バッチグループ用のコマンドボタン

バッチグループの実行の管理には、以下のコマンドボタンを使用します。

ボタン	説明
実行	このバッチグループを実行します。
再開に設定	失敗したバッチグループの実行ステータスを再開に設定します。
未完了に設定	実行中のバッチグループの実行ステータスを未完了に設定します。
選択内容のクリア	選択したグループまたはジョブ実行ログを削除します。
すべてクリア	すべてのグループとジョブ実行ログを削除します。
更新	このバッチグループの画面を更新します。

バッチグループの実行







バッチグループを実行する手順

- バッチグループの [制御およびログ] 画面に移動します。
- ノードをクリックし、[バッチグループ] > [実行] を選択するか、[実行] ボタンをクリックします。
バッチグループツールでバッチグループが実行され、ログパネル上のバッチグループ実行のステータスが更新されます。
- [更新] ボタンをクリックして、実行結果を表示します。
バッチグループツールで進行状況が表示されます。
処理が完了すると、バッチグループツールで以下のログにエントリが追加されます。
 - このバッチグループのグループ実行ログ
 - 個々のバッチジョブのジョブ実行ログ

注: 失敗ステータスのバッチグループを実行する場合、実際には失敗したインスタンスを再実行することになり、ステータスは最終的な結果に設定されます。この場合、Hub で新しいグループログが生成されることはありません。ただし、詳細なログ（下位のログテーブル）では、失敗したインスタンスを再実行するのではなく、同じジョブを新しいインスタンスで実行することになるため、ここに表示される新しいログが Hub で生成されます。

グループ実行ステータス

各実行ログには、以下のステータス値のいずれかが設定されます。

アイコン	説明
	処理中。バッチグループは現在実行中です。
	バッチグループの実行が正常に完了しました。
	バッチグループの実行が完了しましたが、追加情報が存在します。例えば、ステージジョブおよびロードジョブの場合は、一部のレコードが却下されたことを示します。一致ジョブの場合は、ベースオブジェクトが空であるか、一致させるレコードがなくなったことを示します。
	バッチグループの実行が失敗しました。
	バッチグループの実行が不完全です。
	バッチグループの実行が、最初からやり直すためにリセットされました。

バッチグループのグループ実行ログの表示

バッチグループツールでは、バッチグループを実行するたびにグループ実行ログエントリが生成されます。

警告: バッチグループの実行中に、データベースの接続問題でジョブが失敗すると、失敗したジョブはジョブの制御テーブルに表示されません。エラーは cmxserver ログに表示されます。

次の表に、各ログエントリのプロパティを示します。

フィールド	説明
ステータス	このバッチジョブの現在のステータス。バッチグループの実行に失敗した場合に問題の説明を表示します。
開始	このバッチジョブが開始された日時。
終了	このバッチジョブが終了した日時。
メッセージ	バッチグループの実行に関するメッセージ。

バッチジョブのジョブ実行ログの表示

バッチグループツールでは、バッチグループ内のバッチジョブを実行するたびにジョブ実行ログエントリが生成されます。

各ログエントリには、以下のプロパティがあります。

フィールド	説明
ジョブ名	バッチジョブの名前。
ステータス	このバッチジョブの現在のステータス。

フィールド	説明
開始	このバッチジョブが開始された日時。
終了	このバッチジョブが終了した日時。
メッセージ	バッチグループの実行に関するメッセージ。

注: 完了したバッチジョブのメトリックを表示する場合は、バッチビューアを使用できます。

実行に失敗したバッチグループの再開

バッチグループの再開が失敗した場合は、失敗した原因を解決した後、最初からバッチグループを再開します。バッチグループを再実行する手順

1. 「実行したバッチグループのログ」のリストで、失敗したバッチグループの実行ログエントリを選択します。
2. 「再開に設定」をクリックします。

バッチグループツールにより、このバッチジョブのステータスが Restart（再開）に変更されます。

3. 失敗した原因を解決して、バッチグループをもう一度実行します。

バッチグループツールがバッチグループを実行すると、新しい実行ログエントリが作成されます。

注: バッチグループが失敗した場合、「実行したバッチグループのログ」のリストで「再開に設定」ボタンまたは「未完了に設定」ボタンのいずれかをクリックした場合以外は、前回失敗したレベルからバッチジョブが再開されます。

未完了のバッチグループ実行の処理

ごくまれに、実行中のバッチグループのステータス変更が必要になる場合があります。

- バッチグループのステータスがまだ実行中であることを示している場合は、「ステータスを未完了に設定」をクリックし、バッチグループをもう一度実行することができます。この作業は、バッチグループの実行が停止したにもかかわらず（サーバーの再起動やクラッシュなどのエラーのため）、メタデータでのジョブアプリケーションロックが原因で Informatica MDM Hub がバッチグループの停止を検出していない場合にのみ行います。

現在のステータスが実行中であるのに、データベース、アプリケーションサーバー、およびログにアクティビティが示されていない場合は、この状況が問題であることがわかります。この状況が発生した場合は、バッチグループをもう一度実行できるように、このボタンをクリックしてジョブアプリケーションロックを解除します。そうしないと、このバッチグループを実行することはできません。ステータスを未完了に設定すると、バッチグループ（およびバッチグループ内のバッチジョブ）のステータスが更新されるだけで、処理は終了しません。

ジョブステータスが未完了である場合は、ジョブステータスを再開に設定できないので注意してください。

- ジョブステータスが失敗である場合は、「再開に設定」をクリックできます。ジョブステータスが再開である場合は、ジョブステータスを未完了に設定できないので注意してください。

ステータスを変更すると、バッチグループが完了する一方で、引き続き別の作業を実行できます。

実行中のバッチグループを未完了に設定する手順

1. 「実行したバッチグループのログ」リストで、未完了とマークする実行中バッチグループの実行ログエントリを選択します。
2. 「Set to Incomplete（未完了に設定）」をクリックします。

バッチグループツールにより、このバッチジョブのステータスが Incomplete（未完了）に変更されます。

3. このバッチグループを再実行します。

注: バッチグループが失敗した場合、[実行したバッチグループのログ] のリストで [再開に設定] ボタンまたは [未完了に設定] ボタンのいずれかをクリックした場合以外は、前回失敗したレベルからバッチジョブが再開されます。

却下されたレコードの表示

（ステージジョブまたはロードジョブの実行中に）バッチグループの実行結果としてレコードが却下テーブルに書き込まれると、ジョブ実行ログで [却下の表示] ボタンが有効になります。

却下されたレコードを表示する手順

1. **[却下の表示]** ボタンをクリックします。
[却下]ウィンドウが表示されます。
2. 必要に応じて、却下されたレコード間を移動してそのレコードを確認します。
3. **[閉じる]** をクリックします。

ステータスによる実行ログのフィルタリング

すべてのバッチグループにわたる履歴ログを、バッチグループの実行ステータスに基づいて表示することができます。そのためには、**[ステータス別ログ]** ノードの下に該当するノードをクリックします。

実行ログをステータスでフィルタリングする手順

1. バッチグループツールを起動します。
2. [バッチグループ] ツリーで、[ステータス別ログ] ノードを展開します。
ログステータスリストが表示されます。
3. ログパネルの上半分に表示させる特定のバッチグループログエントリをクリックします。
Informatica MDM Hub によって、パネルの下半分にそのバッチグループの詳細なジョブ実行ログが表示されます。
注: バッチグループログを削除するには、バッチグループを選択し、**[選択内容のクリア]** ボタンをクリックします。パネルに表示されているすべてのログを削除するには、**[すべてクリア]** ボタンをクリックします。

バッチグループの削除

バッチグループを削除する手順

1. バッチグループツールを起動します。
2. 書き込みロックを取得します。
3. ナビゲーションツリーで、[バッチグループ] ノードを展開して、削除するバッチグループを表示します。
4. バッチグループで、上に移動するジョブを右クリックし、**[バッチグループの削除]** を選択します（または **[バッチグループ]** > **[バッチグループの削除]** を選択します）。

バッチジョブのリファレンス

ここでは、Informatica MDM Hub の各バッチジョブについて説明します。

バッチジョブのアルファベット順リスト

バッチジョブ	説明
一致しないレコードを一意とする	一致プロセスで一致するデータが見つからなかったレコードの統合インジケータを 1（統合済み）に設定します。これはレコードが一意で統合の必要がないことを意味するステータスです。
オートリンクジョブ	一致プロセスでオートリンクの対象と見なされ、オートリンクのフラグ (Automerge_ind=1) が設定されたレコードを自動的にリンクします。
自動一致とマージジョブ	一致ジョブの後に自動マージジョブを実行する継続的なサイクルを、一致するレコードがなくなるか、手動統合で利用できる一致の数が設定済みのしきい値を超過するまで実行します。マージスタイルのベースオブジェクトでのみ使用されます。
自動マージジョブ	一致プロセスで自動マージの対象と見なされ、自動マージのフラグ (Automerge_ind = 1) が設定されたレコードを自動的にマージします。マージスタイルのベースオブジェクトでのみ使用されます。
外部一致ジョブ	「外部で管理/準備された」レコードを既存のベースオブジェクトと一致させ、現在の一致設定に基づく結果を取得します。ベースオブジェクトのデータが実際に変更されることはありません。
一致トークンの生成ジョブ	一致処理用のデータの準備として、現在の一致設定に従って一致トークンを生成します。一致トークンとは、一致候補の特定に使用されるカラムをエンコードする文字列です。
Hub 削除ジョブ	ベースオブジェクト/相互参照レベルの入力に基づいて Hub からデータを削除します。
スマート検索データの初期インデックス処理ジョブ	あるビジネスエンティティタイプについて、検索可能なフィールドのすべての値のインデックスを作成します。検索ではインデックスを使用して、検索可能なフィールド内のデータを検索します。
キー一致ジョブ	複数のソースで同じプライマリーキーが使用されている場合に、それらのソースのレコードが一致するかを判断します。新しいレコードを他の新しいレコードおよび既存のレコードと比較し、一致ルールで定義されたソースレコードのキーの比較に基づいて一致候補を特定します。
ロードジョブ	ステージングテーブルのレコードを Hub ストア内の対応するターゲットベースオブジェクトにコピーします。ロードプロセス中に、現在の信頼および検証ルールをレコードに適用します。
手動マージジョブ	マージマネージャツールで手動でマージされたレコードのログを表示します。マージスタイルのベースオブジェクトでのみ使用されます。
手動マージ解除ジョブ	マージマネージャツールで手動でマージ解除されたレコードのログを表示します。
一致ジョブ	現在の一致ルールに基づいて、ベースオブジェクト内の重複レコードを検索します。
一致分析ジョブ	検索を実行して一致の統計を収集します。ただし、実際の一致プロセスは実行しません。大量の一致が必要になる可能性があるデータの領域が見つかったら、一致プロセスを続行する前にデータスチュワードが手動でデータを確認できるように、Informatica MDM Hub によってそのレコードが保留ステータスに移行されます。
重複データの一致ジョブ	重複レコードの割合が高いデータについて、新しいレコードを他の新しいレコードおよび既存のレコードと比較し、完全な重複を特定します。完全な重複の最大数は、このベースオブジェクトの「重複一致しきい値」の設定に基づきます。

バッチジョブ	説明
複数マージジョブ	1 つのジョブで複数のレコードをマージできるようにします。
昇格ジョブ	相互参照テーブルの PROMOTE_IND カラムを読み取り、そのカラムの値が 1 であるすべての行の状態をアクティブに変更します。
BO の再計算ジョブ	すべてのベースオブジェクト、または ROWID_OBJECT_TABLE パラメータを指定したベースオブジェクトを再計算します。
BVT の再計算ジョブ	指定された ROWID_OBJECT の BVT を再計算します。
一致テーブルのリセットジョブ	一致したすべてのレコードが、一致用のキューに追加するためにリセットされた操作のログを表示します。
再検証ジョブ	ロードプロセス中の最初の検証以降に変更されたレコードに対して検証ロジック/ルールを実行します。
ステージジョブ	ランディングテーブルのレコードをステージングテーブルにコピーします。実行中に、現在のクレンジング設定に従ってデータをクレンジングします。
同期ジョブ	ベースオブジェクトのメタデータを更新します。ベースオブジェクトがロードされた後、マージはまだ実行されていない状態で、そのベースオブジェクト内のカラムに対して信頼の設定の変更（信頼の有効化など）が行われた場合に使用されます。このベースオブジェクトのデータをマージする前に、このジョブを実行する必要があります。

一致しないレコードを一意とする

一致しないレコードを一意とするバッチジョブは、一致のないレコードのステータスを変更します。このバッチジョブは、一致のないレコードの統合 ID を「1」に設定します。値「1」は、MDM Hub でレコードを統合する必要がないことを示します。手動マージ操作でも、ターゲットレコードに一致がない場合、このレコードの統合 ID は「1」に設定されます。自動マージバッチジョブでは、統合 ID が「1」のレコードは一意のレコードとみなされます。

MDM Hub は、[一致しないすべての行を一意とする] が有効になっている場合、マージバッチジョブの後で、一致しないレコードを一意とするバッチジョブを作成します。

注: 一致しないレコードを一意とするバッチジョブは、バッチビューアからは実行できません。

自動一致とマージジョブ

自動一致とマージジョブでは、一致ジョブの後に自動マージジョブを実行するサイクルが、一致するレコードがなくなるか、手動による統合で処理できるレコード数の上限に達するまで繰り返されます。

このプロセスの一致とマージのサイクルが終了するまでの 1 サイクルあたりのレコード数は、一致バッチサイズパラメータで制御されます。

重要: 自動一致とマージジョブは、テーブル間またはテーブル内の一致パスでレコード間のリレーションを定義するために使用されているベースオブジェクトに対しては実行しないでください。実行した場合、リレーションのデータが変わり、レコード間の関連付けが失われます。

アプリケーションサーバの再起動後に表示される 2 番目のジョブ

自動一致とマージジョブを実行すると、状態にジョブが 1 つ表示されて正常に完了します。ただし、アプリケーションサーバを停止して再起動し、バッチビューアに戻ると、数秒後に警告とともに 2 番目のジョブが表示されます（一致ジョブの下にリストされます）。2 番目のジョブは、ベースオブジェクトが空であるか、または一致するレコードがそれ以上ないことを確認するためのものです。

自動一致とマージのメトリック

自動一致とマージジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリック（該当する場合）が表示されます。

メトリック	説明
一致したレコード	自動一致とマージジョブで一致したレコードの数。
トークン化されたレコード	自動一致とマージジョブの前にトークン化されたレコードの数。
自動マージされたレコード	自動一致とマージジョブでマージされたレコードの数。
一意として受け入れられたレコード	自動一致とマージジョブで一意のレコードとして受け入れられたレコードの数。 このベースオブジェクトの「一致/マージ設定」の設定で「一致しないすべての行を一意とする」が有効になっている（「はい」に設定されている）場合にのみ適用されます。
自動マージ用にキューに追加	自動一致とマージジョブで実行された一致ジョブによって自動マージ用にキューに追加されたレコードの数。
手動マージ用にキューに追加	手動マージ用にキューに追加されたレコードの数。これらのレコードを処理するには、Hub コンソールのマージマネージャを使用します。詳細については、『 <i>Multidomain MDM のデータスチュワードガイド</i> 』を参照してください。

自動マージジョブ

マージスタイルのベースオブジェクトの場合に限り、一致ジョブの実行後に自動マージジョブを実行して、一致プロセス中に自動マージ可能と判断されたレコードを自動的にマージすることができます。自動マージジョブを実行すると、自動マージのフラグ（Automerge_ind=1）が設定された一致テーブル内のすべての一致が処理されます。

注: 状態が有効なオブジェクトの場合、保留中のレコード（ソースおよびターゲットのレコード）または削除済みのレコードは自動的にマージされません。レコードが削除されると、一致テーブルから削除され、consolidation_ind が 4 にリセットされます。

自動マージジョブおよび自動一致とマージ

自動一致とマージジョブでは、一致ジョブの後に自動マージジョブを実行するサイクルが、一致するレコードがなくなるか、手動による統合で処理できるレコード数の上限に達するまで繰り返されます。

自動マージジョブと信頼が有効なカラム

信頼が有効なカラムが多数ある場合、自動マージジョブは失敗します。ジョブが失敗する原因となるカラム数が正確に決まっているわけではなく、ジョブが失敗するかどうかはカラム名の長さや信頼が有効なカラムの数

に基づきます。 カラム名で許容される最大の長さは約 26 文字です。 この問題を回避するために、信頼が有効なカラムの数は 40 個未満にし、カラム名は短くするようにしてください。

自動マージのメトリック

自動マージジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリック（該当する場合）が表示されます。

メトリック	説明
自動マージされたレコード	自動マージジョブで自動マージされたレコードの数。
一意として受け入れられたレコード	自動マージジョブで一意のレコードとして受け入れられたレコードの数。このベースオブジェクトの「一致/マージ設定」の設定で「一致しないすべての行を一意とする」が有効になっている（「はい」に設定されている）場合にのみ適用されます。

バッチマージ解除

前述の処理でマージされたレコードのマージ解除ができます。レコードのバッチマージ解除には、ExecuteBatchUnmerge SIF API を使用します。

結合されたレコードのバッチマージ解除は、以下の操作で行うことができます。

- 自動マージバッチジョブ
- 手動マージ
- レコードの手動編集
- ROWID_OBJECT によるロード
- Put SIF API による相互参照レコードの挿入または更新。

親レコードまたは関連する子テーブルに関係する、マージ解除されるオブジェクトの一致レコードが ExecuteBatchUnmerge SIF API によって削除されることはありません。これらの一致レコードを削除するには、テーブル TEST_OUTPUT_TBL 内のマージ解除レコードのリストを使用します。MDM Hub では、マージ解除されたレコードと手動で追加されたレコードは区別されます。デフォルトで、MDM Hub は手動で追加されたレコードに値「0」を割り当てます。MDM Hub は、EXPLODE_NODE_IND カラムの TEST_OUTPUT_TBL テーブルでマージ解除されたレコードに「0」以外の値を割り当てます。

また、EXPLODE_NODE_IND カラムを入力テーブルにも追加できます。EXPLODE_NODE_IND を「1」に設定すると、MDM Hub はベースオブジェクト全体をマージ解除しようとします。

ExecuteBatchUnmerge SIF API の詳細については、「*Multidomain MDM サービスの統合フレームワークガイド*」を参照してください。

BVT スナップショットジョブ

ベースオブジェクトテーブルの最善データ（BVT）とは、ソースレコードのデータのうちの最適なセルと統合されているレコードです。

注: 状態が有効なベースオブジェクトにかぎり、BVT のロジックは、HUB_STATE_IND を使用して、HUB_STATE_IND が-1 または 0（保留状態または削除済み状態）のレコードに貢献していないベースオブジェクトを無視します。オンラインの BUILD_BVT 呼び出しの場合は、INCLUDE_PENDING_IND パラメータを指定します。

考えられるシナリオを以下に示します。

1. このパラメータが 0 の場合は、アクティブなベースオブジェクトレコードだけが含まれます。
2. このパラメータが 1 の場合は、アクティブおよび保留中のベースオブジェクトレコードが含まれます。
3. このパラメータが 2 の場合は、アクティブおよび保留中の相互参照レコードに基づいて計算を行い、“what-if”機能を提供します。
4. このパラメータが 3 の場合は、アクティブな相互参照レコードに基づいて計算を行い、相互参照に基づく現在の BVT を提供します。これは、シナリオ 1 と同じになるとは限りません。

外部一致ジョブ

外部一致ジョブは、外部で準備されたレコードをベースオブジェクトと照合させて、現在の一致設定に基づいて結果を返します。外部一致ジョブでは、入力テーブルからベースオブジェクトへのデータのロードは行われず、ベースオブジェクトデータに影響を与えることはありません。標準の一致ジョブを実行する前に、外部一致を使用して、データの事前テスト、一致ルールのテスト、および結果の確認を行うことができます。

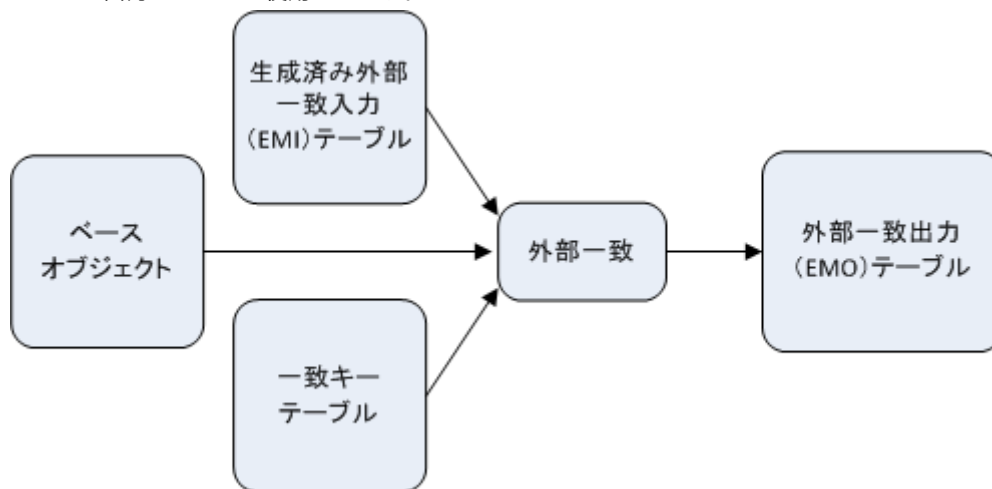
外部一致ジョブを使用して、あいまい一致ルールと完全一致ルール、およびあいまい一致ベースオブジェクトと完全一致ベースオブジェクトの両方を処理できます。

注: 連結された物理カラムで構成される完全一致カラムには、各カラムの末尾に 1 つのスペースが必要です。例えば、「John」に「Smith」が連結されている場合は、「John Smith」という一致のみが返されます。

外部一致ジョブは、バッチジョブとして実行されます。外部アプリケーションで起動できる、対応する SIF 要求はありません。

外部一致ジョブに使用される入力テーブルと出力テーブル

外部一致ジョブでは、ベースオブジェクトおよび関連する一致キーテーブルに加えて、以下の図に示す入力テーブルと出力テーブルが使用されます。



外部一致入力 (EMI) テーブル

各ベースオブジェクトには、外部一致ジョブ用の外部一致入力 (EMI) テーブルがあります。このテーブルには、以下の命名パターンが使用されます。

`C_BaseObject_EMI`

この *BaseObject* は、この外部一致ジョブに関連するベースオブジェクトの名前です。

ベースオブジェクトを作成すると、スキーママネージャによって関連する EMI テーブルが作成され、以下のシステムカラムが追加されます。

カラム名	データ型	サイズ	非 Null	説明
SOURCE_KEY	VARCHAR	50		このレコードを一意に識別し、 <i>C_BaseObject</i> _EMO テーブルのレコードにマップするために使用される 3 カラムの複合プライマリキーの一部として使用されます。
SOURCE_NAME	VARCHAR	50		このレコードを一意に識別し、 <i>C_BaseObject</i> _EMO テーブルのレコードにマップするために使用される 3 カラムの複合プライマリキーの一部として使用されます。
FILE_NAME	VARCHAR	50		このレコードを一意に識別し、 <i>C_BaseObject</i> _EMO テーブルのレコードにマップするために使用される 3 カラムの複合プライマリキーの一部として使用されます。

EMI テーブルにデータを入力するときは、これらのシステムカラムの少なくとも 1 つにデータが含まれている必要があります。カラム名は非制限的です。つまり、3 カラムの複合プライマリキーが一意である場合、どのような識別データでも入れることができます。

また、Person_Name や Exact_Cust_ID など、特定のカラムに対して一致ルールを設定すると、スキーママネージャによってそのカラムが *C_BaseObject*_EMI テーブルに追加されます。

注: 外部一致テーブルのカラムをスキーママネージャで表示するには、**[外部一致テーブル]** ノードを展開します。

EMI テーブル内のレコードは、一致ジョブで使用される一致バッチに似ています。一致バッチには、ベースオブジェクト内の残りのレコードと一致する一連のレコードが含まれます。異なる点は、一致ジョブでは一致バッチレコードがベースオブジェクト内に存在し、外部一致ではそれらのレコードが別の入力テーブルに存在している点です。

外部一致出力テーブル

各ベースオブジェクトには、外部マッチングバッチジョブの出力データを含む外部マッチング出力テーブルがあります。外部一致バッチジョブの実行前に、MDM Hub は外部一致出力テーブルを削除して再作成します。

MDM Hub は外部一致出力テーブルに *C_<base_object_name>_EMO* という名前をつけます。この *base_object_name* は、外部一致バッチジョブに関連したベースオブジェクトの名前です。

外部マッチングバッチジョブは外部マッチング出力テーブルに一致したペアを入力します。外部マッチングジョブはマッチテーブルを入力しません。外部マッチング出力テーブルの各行は、マッチしたレコードのペアを表します。レコードのペアのうち、一方は外部マッチング入力テーブルのレコード、もう一方はベースオブジェクトのレコードです。プライマリキー (SOURCE_NAME および FILE_NAME と結合した SOURCE_KEY) は外部マッチング入力テーブルからのレコードを特定します。ROWID_OBJECT_MATCHED の値はベースオブジェクトからのレコードを特定します。

外部マッチング出力テーブルには次のカラムが含まれています。

カラム名	MDM Hub データ型 (サイズ)	Null ができる	説明
SOURCE_KEY	VARCHAR (50)	はい	C_<base_object_name>_EMI テーブルのソースレコードに再びマッピングされます。
SOURCE_NAME	VARCHAR (50)	はい	C_<base_object_name>_EMI テーブルのソースレコードに再びマッピングされます。
FILE_NAME	VARCHAR (50)	はい	C_base_object_name_EMI テーブルのソースレコードに再びマッピングされます。
ROWID_OBJECT_MATCHED	CHAR (14)	いいえ	外部一致入力テーブルのレコードと一致した、ベースオブジェクト内のレコードの ROWID_OBJECT。
ROWID_MATCH_RULE	CHAR (14)	はい	外部一致バッチジョブで一致の判定に使われたマッチルールを特定します。
AUTOMERGE_IND	NUMBER (38)	いいえ	レコードが一致プロセスでの自動統合の対象として適格かどうかを示します。AUTOMERGE_IND は、次の値のいずれかになります。 - 0: レコードは自動統合の対象になりません。 - 1: レコードは自動統合の対象になります。 - 2: レコードは自動一致の対象になりますが、1 つ以上のレコードが保留中です。状態管理が有効になっていて、[保留中のレコードに対して一致を有効にする] オプションを選択している場合に、この値が 2 になります。保留中のレコードでグループを構築しないでください。保留中のレコードは個々の一致として残します。 自動マージバッチジョブでは、AUTOMERGE_IND 値が 1 のレコードが処理されます。
CREATOR	VARCHAR2 (50)	はい	レコードの作成を行ったユーザーまたはプロセス。
CREATE_DATE	TIMESTAMP	はい	レコードが作成された日付。

入力テーブルのポピュレーション

外部一致ジョブを実行する前に、EMI テーブルにベースオブジェクト内のレコードと照合するレコードを入力する必要があります。データを EMI テーブルにロードするプロセスは、MDM Hub に対する外部プロセスです。データベース環境で動作するデータロードツールを使用します。

EMI テーブルにデータを入力する際、_EMI テーブルからのリンクをサポートするために、少なくとも 1 つのシステムカラムにデータを提供する必要があります。さらに、C_BaseObject_EMI テーブルには、一意のソースキーを持ち、他のテーブルへの外部キーを持たないフラットレコードを含める必要があります。

入力テーブルのデータ型の変換

MDM Hub では、ベースオブジェクトの DATE データ型のカラムが、関連する EMI テーブルの VARCHAR データ型のカラムとして表示されます。この問題は、MDM Hub がデータを比較するために日付を文字列に変換す

ることで発生します。EMI テーブルを取り込む前に、ベースオブジェクトの日付データ型を変換すると、この問題を回避できます。

1. 変換するベースオブジェクトのレコードを特定します。カラムと関連する rowid_object の両方の日付値を記録します。
2. クエリツールを使用して、ベースオブジェクトレコードに対応する日付データ型のレコードを STRP テーブルから選択します。
クエリは、SSA_DATA フィールドの日付とタイムスタンプに類似したデータを含むレコードを返します。
3. 日付形式マスク YYYY/MM/DD HH24:MI:SS を使用して、ベースオブジェクトの DATE 値を VARCHAR 値に変換します。

データベース	変換コマンド
Oracle と IBM DB2	(to_char(date_column, 'YYYY/MM/DD HH24:MI:SS'))
Microsoft SQL Server	(convert(VARCHAR(10), date_column, 111) + ' ' + convert(VARCHAR(8), date_column, 108))

4. 外部一致のためにデータを EMI テーブルにロードします。
5. 外部一致ジョブを実行します。
外部一致ジョブは、日付データ型と一致する値を返します。

外部一致ジョブの実行

1. MDM Hub の外部にあるデータロードプロセスを使用して、C_BaseObject_EMI テーブルにデータを入力します。
2. Hub コンソールで、以下のいずれかのツールを開始します。
 - バッチビューア
 - バッチグループ
3. ベースオブジェクトの外部一致ジョブを選択します。
4. 外部一致に使用する一致ルールセットを選択します。
5. 外部一致ジョブを実行します。
 - 外部一致ジョブは、C_BaseObject_EMI テーブルのすべてのレコードを、ベースオブジェクトのレコードと照合します。入力テーブルと出力テーブルに統合インジケータの概念はありません。
 - 一致グループの構築は、結果に対しては実行されません。
6. データ管理ツールを使用して、C_BaseObject_EMO テーブル内の結果を確認します。
7. 結果を保存する場合は、外部一致ジョブを再度実行する前にバックアップコピーを作成します。
注: C_BaseObject_EMO テーブルは、外部一致ジョブを実行するたびに MDM Hub によって破棄され作成し直されます。

一致トークンの生成ジョブ

一致トークンの生成ジョブでは、一致トークンを生成して、ベースオブジェクトに関連付けられた一致キーテーブルに格納するトークン化プロセスが実行されます。これにより、それらの一致トークンを後から一致プロ

セスで使用して一致候補を特定できます。一致トークンの生成ジョブは、あいまい一致ベースオブジェクトにのみ適用され、完全一致ベースオブジェクトには適用されません。

一致プロセスを実行するには、一致キーテーブル内の一致トークンが最新であることが必要です。一致トークンの更新が必要である場合（ロードプロセスでレコードが追加または更新された場合など）は、一致プロセスで一致ジョブが開始されるときに自動的にトークン化プロセスが実行されます。一致プロセスを迅速化するために、トークン化プロセスは以下のいずれかの方法で（一致プロセスの前に）個別に実行することをお勧めします。

- 一致トークンの生成ジョブを手動で実行する
- ロードプロセス完了後に自動的にトークン化プロセスが実行されるように設定する

状態が有効なベースオブジェクトのトークン化

状態が有効なベースオブジェクトの場合にのみ、状態が「削除済み」のレコードはトークン化プロセスでスキップされます。

このようなレコードは、トークン化 API によってトークン化できますが、バッチ処理では無視されます。保留レコードは、MATCH_PENDING_IND（デフォルトではオフ）を設定することで、ベースオブジェクトごとに一致できます。

一致トークンの生成ジョブの操作範囲を設定する

一致トークンの生成ジョブを実行する前に、すべてのレコードに対してトークンを生成するか、または未処理のテーブルに ROWID_OBJECT の値が指定された新規レコードと変更済みレコードに対してのみトークンを生成するかを決定します。

- ベースオブジェクト内のすべてのレコードに対して一致トークンを生成するには、**【すべての一致トークンの再生成】** チェックボックスを選択します。
- ベースオブジェクト内の新規レコードまたは更新済みレコードに対して一致トークンを生成するには、**【すべての一致トークンの再生成】** チェックボックスを選択解除します。

一致トークンの生成後、ベースオブジェクトの一致ジョブを実行できます。

ヒント: Generate Match Tokens プロセスで、クラス `ssa.ssaname3.jssan3cl` が初期化できないというエラーが返される場合は、システム管理者に連絡してください。

1. SSA-NAME3 の DLL（dynamic linked library）ファイルが含まれるディレクトリ <MDM installation directory>/hub/cleanse/lib へのパスが PATH 環境変数に含まれているかを確認します。
2. MDM Hub の検索および一致を実行するプロセスサーバーに Visual Studio 2019 の Microsoft Visual C++ 再頒布可能パッケージがインストールされているかを確認します。
3. Visual Studio 2019 の Microsoft Visual C++ 再頒布可能パッケージがインストールされている場合は、依存性チェッカー（Dependency Walker（depends.exe）など）を使用して `jssan3cl.dll` をロードし、Visual C++ 再頒布可能パッケージが正常に適用されていることを確認します。

ヒント: Visual Studio 2019 の Visual C++ 再頒布可能パッケージには、Windows Server にオペレーティングシステムパッチがインストールされていることが必要です。Visual C++ 再頒布可能パッケージをインストールする前に、オペレーティングシステム要件を確認します。たとえば、Windows Server 2012 のベースラインバージョン以降、Visual C++ 再頒布可能パッケージを正常にインストールするには、オペレーティングシステムに約 100 個のパッチ（合計約 2 GB）を適用する必要があります。

スマート検索データジョブの初期インデックス処理

スマート検索データの初期インデックス処理ジョブでは、ビジネスエンティティ内の検索可能なフィールドのすべての値に対してインデックスを作成します。検索ではインデックスを使用して、検索可能なフィールド内のデータを検索します。

スマート検索データの初期インデックス処理ジョブを実行して、検索可能なビジネスエンティティからレコードを抽出して、インデックスに追加します。データを1回以上インデックス処理した後、ロードジョブを実行すると、スマート検索データの初期インデックス処理ジョブが内部で実行されて、新しいデータと更新済みデータがインデックス処理されます。

スマート検索データの初期インデックス処理ジョブは、すべてのレコードのインデックス処理要求をキューに入れた後、レコードを非同期的にインデックス処理して、正常に完了したことを報告します。検索要求が予期される結果を返すには、インデックス処理要求が正常に完了している必要があります。これには、数分かかることがあります。

すべてのレコードをインデックス処理した後にレコードを追加または更新した場合は、新しいビジネスエンティティまたは更新されたビジネスエンティティをインデックス処理する必要があります。ベースオブジェクトレコードを削除すると、一部のインデックスが古くなり不適切になる場合があります。「スマート検索データの初期インデックス処理」バッチジョブを実行してデータを再インデックス化し、古くなったインデックスを削除することで、検索要求のパフォーマンスを向上させることができます。

整数フィールドの値が19桁を超えると、スマート検索データの初期インデックス処理ジョブでは値をインデックス処理しません。日時フィールドの値にタイムゾーン情報が含まれていない場合、スマート検索データの初期インデックス処理ジョブでは、時間がロケールのタイムゾーン内にあると判断されます。その後、時間がUTCに変換され、UTCタイムがインデックス処理されます。

スマート検索データメトリックの初期インデックス処理

次の表では、「スマート検索データの初期インデックス処理」ジョブの実行に成功すると、バッチビューアによって表示されるメトリックについて説明します。

メトリック	説明
挿入済み	インデックスに追加されたレコードの総数を示します。
却下されたレコード	却下されたレコードの総数を示します。HUB_STATE_IND 値が無効な場合、レコードが却下されます。
削除された BO レコード	削除された状態にあるレコードの総数を示します。HUB_STATE_IND 値が-1 の場合、レコードは削除された状態になっています。

キー一致ジョブ

プライマリキー一致ルールと常に併用されるキー一致ジョブでは、同じプライマリキー値を使用する2つ以上のソースシステムからのレコードに対して一致プロセスが実行されます。

キー一致ジョブでは、新しいレコードが互いに比較されるとともに既存のレコードと比較され、プライマリキー一致ルールで定義されているとおりに、ソースレコードキーの比較に基づいて一致候補が特定されます。

キー一致ジョブは、スキーママネージャでベースオブジェクトのプライマリキー一致ルールが作成または変更されると（一致/マージ設定の構成）自動的に作成されます。

ロードジョブ

ロードジョブでは、ステージングテーブルから Hub Store 内の対応するターゲットベースオブジェクトにデータが移動されます。さらにロードジョブでは、信頼されたカラムが定義されているベースオブジェクトの信頼値が計算され、検証ルール（定義されている場合）を適用して最終的な信頼値が決定されます。

ロードジョブと状態が有効なベースオブジェクト

状態が有効なベースオブジェクトの場合、ロードバッチプロセスではあらゆる状態のレコードがロードされます。状態は、ステージングテーブルの入力カラムとして指定されます。入力状態は、マッピングでランディングテーブルカラムとして指定するか、派生させることができます。マッピングに入力状態が指定されていない場合、状態はアクティブと見なされます。

以下の表に、入力状態が既存の XREF の状態に与える影響を示します。

	既存の XREF の状態:	アクティブ	保留	削除済み	XREF なし（行 ID でロード）	ベースオブジェクトなし
入力の XREF の状態:						
アクティブ		更新	更新+昇格	更新+リストア	挿入	挿入
保留		更新を保留	更新を保留	更新+リストアを保留	更新を保留	挿入を保留
削除済み		論理削除	物理削除	物理削除	Error	Error
未定義		アクティブとして扱う	保留として扱う	削除として扱う	アクティブとして扱う	アクティブとして扱う

注: HUB_STATE_IND 値が無効である場合は、レコードが却下されます。

以下の表に、Informatica MDM Hub が、ロード（PUT）時の特定の操作においてレコードをその状態に基づいてどのように処理するかをマトリックスで示します。

	入力レコードの状態	入力レコードの状態	注意事項
次の場合に XREF レコードを更新:	アクティブ	アクティブ	
	削除済み	アクティブ	
	保留	保留	
	アクティブ	保留	
	削除済み	削除済み	
	保留	削除済み	

	入力レコードの状態	入力レコードの状態	注意事項
	削除済み		ベースオブジェクト行 ID 削除レコードが入力されると、Informatica MDM Hub はベースオブジェクトおよびすべての XREF レコード (ROWID_SYSTEM に関係なく) を削除済み状態に更新します。
次の場合に XREF レコードを挿入:	保留	アクティブ	ペアのもう一方のレコードが作成されます。
	アクティブ	レコードなし	
	保留	レコードなし	
次の場合に XREF レコードを削除:	アクティブ	保留 (ペアになったレコードの場合)	ペアのアクティブなレコードが削除され、保留レコードが更新されます。 ペアのレコードは、同じ PKEY_SRC_OBJECT と ROWID_SYSTEM を持つ 2 つのレコードです。
	削除済み	保留	
次の場合に Informatica MDM Hub がエラーを表示:	保留	アクティブ (ペアになったレコードの場合)	ペアのレコードは、同じ PKEY_SRC_OBJECT と ROWID_SYSTEM を持つ 2 つのレコードです。

その他の注意事項:

- (ロードの更新に対して) 入力状態が指定されていない場合は、入力状態が現在の状態と同じであると見なされます。例えば、入力状態が NULL で、更新対象の XREF またはベースオブジェクトの既存の状態が保留である場合は、入力状態が NULL ではなく保留と見なされます。

ロードジョブの実行ルール

ロードジョブには以下のルールが適用されます。

- ロードジョブは、ロードジョブが使用するステージングテーブルをロードするステージジョブが正常に完了した場合にのみ、実行します。
- 子テーブルのロードジョブを実行する前に、親テーブルのロードジョブを実行します。
- 子オブジェクトのルックアップが定義されていない場合 (ルックアップテーブルおよびカラムが入力されていない場合)、データを正しくロードするために、ロードジョブを実行する前に子オブジェクトに対してステージジョブを繰り返す必要があります。
- 同じベースオブジェクトに対して一度に 1 つのロードジョブを実行できます。同じベースオブジェクトに対する複数のロードジョブを同時に実行することはできません。
- 共通の子ベースオブジェクトを共有している複数の親ベースオブジェクト上でロードジョブを並列で実行することはできません。

ロードジョブでの更新の強制

ロードジョブを実行する前に、**【更新の強制】** チェックボックスを使用して、ロードジョブでのステージングテーブルからターゲットベースオブジェクトへのデータのロード方法を設定することができます。デフォルトでは、Informatica MDM Hub がステージングテーブルの各レコードの最終更新日をチェックして、そのレコードがまだロードされていないことを確認します。この動作を変更するには、**【更新の強制】** チェックボックスを

オンに（選択）します。これにより、最終更新日が無視され、更新が強制的に行われ、ステージングテーブルからすでにロードされているかどうかにかかわらず各レコードがロードされます。ただし、この方法は慎重に使用してください。ロードするデータのボリュームによっては、更新を強制することで処理時間が増加する可能性があります。

ロードジョブでの状態管理オーバーライドシステム

1つのソースシステムを状態管理オーバーライドシステムとして設定することができます。状態管理オーバーライドシステムは、削除済み状態のレコードを保持することができます。状態管理オーバーライドシステム内の削除済みのレコードは、ロードジョブ中にデータベースオブジェクトレコード内のセル値を上書きすることができます。

ロードジョブ中にベースオブジェクトレコード内の状態管理オーバーライドシステムを使用してセル値を上書きするには、以下に示す条件を満たしてください。

- ランディングテーブルの HUB_STATE_IND カラムが、ステージングテーブルの HUB_STATE_IND カラムに割り付けられている。
- ランディングテーブルの DELETED_IND カラムが、ステージングテーブルの DELETED_IND カラムに割り付けられている。
- Hub 状態インジケータが削除済み状態である。
- ステージングテーブルの DELETED_IND カラムの値が-999999999 である。

ロードジョブでの一致トークンの生成

スキーマツールでベースオブジェクトの詳細プロパティを設定する際に、**【ロード時に一致トークンを生成する】** チェックボックスをオンに（選択）して、ロードジョブでレコードがベースオブジェクトにロードされた後に一致トークンが生成されるようにすることができます。

デフォルトでは、このチェックボックスがオフに（クリア）され、一致トークンは一致プロセスで生成されます。

システムカラムでのロードバッチジョブ

ロードバッチジョブの影響は、一部のシステムカラムで異なります。

下表に、ロードバッチジョブ挿入操作と更新操作がシステムカラムに与える影響を示します。

システムカラム	ロードバッチジョブ操作のタイプ	システムカラムへの影響
CREATE_DATE	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。
CREATE_DATE	更新	ベースオブジェクトのカラムは、元の値を保ちます。
CREATOR	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。

システムカラム	ロードバッチジョブ操作のタイプ	システムカラムへの影響
CREATOR	更新	ベースオブジェクトのカラムは、元の値を保ちます。
UPDATE_BY	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。
UPDATE_BY	更新	ベースオブジェクトのカラムは、元の値を保ちます。 ロードバッチジョブは、相互参照テーブルのカラムにステージングテーブルのデータを設定します。
LAST_UPDATE_DATE	挿入	ロードバッチジョブは、ベースオブジェクトテーブルと相互参照テーブルの LAST_UPDATE_DATE カラムに SYSDATE を設定します。
LAST_UPDATE_DATE	更新	ロードバッチジョブは、ベースオブジェクトテーブルと相互参照テーブルの LAST_UPDATE_DATE カラムに SYSDATE を設定します。
SRC_LUD	挿入	ロードバッチジョブは、SRC_LUD カラムに LAST_UPDATE_DATE カラムの値を設定します。
SRC_LUD	更新	ロードバッチジョブは、SRC_LUD カラムに LAST_UPDATE_DATE カラムの値を設定します。
DELETED_IND	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。
DELETED_IND	更新	ベースオブジェクトのカラムは、元の値を保ちます。
DELETED_BY	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。
DELETED_BY	更新	ベースオブジェクトのカラムは、元の値を保ちます。
DELETED_DATE	挿入	ロードバッチジョブは、これらのカラムにステージングテーブルのデータを設定します。ステージングテーブルのカラムが null 値の場合は、ロードバッチジョブはこれらのカラムにカラムのデフォルト値を設定します。
DELETED_DATE	更新	ベースオブジェクトのカラムは、元の値を保ちます。

ロードジョブメトリック

ロードジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリックが表示されます（該当する場合）。

メトリック	説明
総レコード数	ロードジョブによって処理されたレコードの数。
挿入済み	ロードジョブによってターゲットオブジェクトに挿入されたレコードの数。
更新済み	ロードジョブによって更新されたターゲットオブジェクト内のレコードの数。
アクションなし	アクションが実行されなかったレコードの数（レコードがベースオブジェクトにすでに存在していたため）。
更新された相互参照	このベースオブジェクトの相互参照テーブルを更新したレコードの数。増分ロードでレコードをロードしたときに、すでに統合されていた（相互参照にのみ存在し、ベースオブジェクトにない）レコードです。
トークン化されたレコード	ロードジョブによってトークン化されたレコードの数。スキーマツールで「ロード時に一致トークンを生成する」チェックボックスが選択されている場合にのみ適用されます。
マージに関与した XREF レコード	他の rowid_objects にマージされた、更新済み相互参照レコードの数。更新された相互参照レコードの総数と、更新されたベースオブジェクトレコードの数の差を表します。
ルックアップなし/無効な rowid_object レコード	ルックアップ情報がない rowid_object レコードが無効なソースレコードの数。

手動マージジョブ

一致ジョブが実行された後に、一致ジョブで手動マージの対象としてキューに追加されたレコードを、データスチュワードがマージマネージャを使用して処理できます。

手動によるマージジョブは、バッチビューアではなくマージマネージャで実行されます。バッチビューアでは、マージマネージャで実行された手動マージジョブのジョブ実行ログを確認することのみ可能です。

手動統合の最大一致数

スキーママネージャで、手動統合を準備するための一致の最大数を設定できます。これにより、データスチュワードが、多数の手動統合を処理する手間を軽減することができます。この制限に達すると、一致の数が減少するまで一致ジョブ、自動一致ジョブ、およびマージジョブは実行されません。

マージマネージャでの手動マージジョブの実行

手動マージジョブを開始すると、進行状況インジケータを含んだダイアログが表示されます。手動マージが完了するまでしばらく時間がかかることがあります。処理中に問題が発生した場合は、完了時にエラーメッセージが表示されます。このエラーは、バッチビューアで表示される手動マージジョブ実行ログにも示されます。

マージマネージャでは、プロセスダイアログに「プロセスを未完了とマーク」ボタンが表示されます。このボタンを使用すると、手動マージジョブのステータスが更新されますが、ジョブが強制終了されることはありません。このボタンをクリックすると、マージプロセスがバックグラウンドで続行されます。この時点で、バッチビューアにこのプロセスのエントリが表示されます。プロセスが完了すると、実行の成功または失敗が報告されます。マージマネージャの詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

手動マージ解除ジョブ

マージスタイルのベースオブジェクトの場合のみ、手動マージジョブが実行された後に、データスチュワードがデータマネージャを使用して、手動でマージされたレコードを手動でマージ解除できます。手動マージ解除ジョブは、バッチビューアではなくデータマネージャで実行されます。バッチビューアでは、データマネージャで実行された手動マージジョブのジョブ実行ログを確認することのみ可能です。データマネージャの詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

データマネージャでの手動によるマージ解除ジョブの実行

手動によるマージ解除ジョブを開始すると、データマネージャによって進行状況インジケータを含んだダイアログが表示されます。手動によるマージ解除には、完了までにある程度の時間がかかることがあります（特に問題のレコードが多数の構成レコードから生成されている場合）。処理中に問題が発生した場合は、完了時にエラーメッセージが表示されます。このエラーは、バッチビューアで表示される手動によるマージ解除のジョブ実行ログにも示されます。

データマネージャでは、プロセスダイアログに **【プロセスを未完了とマーク】** ボタンが表示されます。このボタンを使用すると、手動によるマージ解除ジョブのステータスが更新されますが、ジョブが強制終了されることはありません。このボタンをクリックすると、マージ解除プロセスがバックグラウンドで続行されます。この時点で、バッチビューアにこのプロセスのエントリが表示されます。プロセスが完了すると、実行の成功または失敗が報告されます。

一致ジョブ

一致ジョブではベースオブジェクトの検索キーが生成され、データ内の一致候補が検索されて、一致候補に一致ルールが適用されます。次に、一致が生成され、自動統合または手動統合のために一致がキューに追加されます。

ORS で新しいベースオブジェクトを作成すると、MDM Hub で一致ジョブが作成されます。各一致ジョブでは、ベースオブジェクト内の新しいレコードまたは更新されたレコードが、そのベースオブジェクト内のすべてのレコードと比較されます。

一致ジョブを実行すると、一致した行に自動統合および手動統合のフラグが設定されます。MDM Hub で、自動マージまたはオートリンクレコードを統合するジョブが作成されます。レコードに手動統合のフラグが設定されている場合は、データスチュワードがマージマネージャを使用して手動統合を実行する必要があります。手動統合の詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

一致ジョブの設定は、スキーママネージャの **【一致/マージ設定】** ノードで行います。

テーブル間一致パスまたはテーブル内一致パスでレコード間のリレーションを定義するために使用されるベースオブジェクトでは、一致ジョブを実行しないでください。実行すると、リレーションデータが変更され、レコード間の関連性が失われます。

注: 環境に Windows を実行するアプリケーションサーバーと Linux を実行するデータベースサーバーがある場合、一致ジョブは応答しなくなることがあります。

一致テーブル

Informatica MDM Hub の一致ジョブをベースオブジェクトに対して実行すると、ベースオブジェクトの一致テーブルに一致したレコードのペアが取り込まれます。

一致テーブルには、通常 *Base_Object_MTCH* という名前が付けられます。

一致ジョブと状態が有効なベースオブジェクト

以下の表に、状態が有効なベースオブジェクトの入力状態に対する一致バッチプロセスの動作を詳しく説明します。

ソースベースオブジェクトの状態	ターゲットベースオブジェクトの状態	操作の結果
アクティブ	アクティブ	レコードが一致の目的で分析されます。
保留	アクティブ	バッチ一致で保留レコードが無視されるかどうかは、テーブルレベルのパラメータです。これを設定すると、指定したベースオブジェクトの保留レコードがバッチ一致の処理対象となります。ただし、保留レコードは一致のソースレコードにすることのみ可能です。
削除済み	あらゆる状態	削除済みレコードは、バッチ一致で無視されます。
あらゆる状態	保留	保留レコードを一致のターゲットにすることはできません。

注: 一致グループの構築（BMG）では、保留中のレコードがあるグループは構築されません。保留中のレコードが個別の一致として保持されます。一致した保留レコードには、automerge_ind=2 が設定されます。

自動一致とマージジョブ

マージスタイルのベースオブジェクトの場合に限り、ベースオブジェクトに対して自動一致とマージジョブを実行することができます。

自動一致とマージジョブでは、一致ジョブの後に自動マージジョブを実行するサイクルが、一致するレコードがなくなるか、手動による統合で処理できるレコード数の上限に達するまで繰り返されます。

バッチジョブの設定の制限

ベースオブジェクトの一致ジョブでは、ベースオブジェクトの各レコードとベースオブジェクトの他の各レコードとの一致は行われません。

代わりに、（スキーマツールで）以下の項目を指定します。

- ジョブを実行するたびに一致させるレコード数
- 手動の統合に対して許可される一致数

これにより、データスチュワードが多数の手動統合を処理する手間を軽減することができます。この制限値に到達すると、手動統合用に準備するレコード数が減少するまで、一致ジョブは実行されません。

一致ルールセットの選択

一致ジョブの場合、ジョブを実行する前に、一致評価のために使用する一致ルールセットを選択できます。

このベースオブジェクトのデフォルトの一致ルールセットが自動的に選択されています。他の一致ルールセットを選択するには、ドロップダウンリストをクリックして、このベースオブジェクトに定義されている他の一致ルールセットを選択します。

一致ジョブメトリック

一致ジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリックが表示されます（該当する場合）。

メトリック	説明
一致したレコード	一致ジョブによって一致したレコード数。
トークン化されたレコード	一致ジョブによってトークン化されたレコード数。
自動マージ用にキューに追加	一致ジョブによってキューに追加されたレコード数。これらのレコードを処理するには、自動マージジョブを使用します。
手動マージ用にキューに追加	一致ジョブによって手動マージ用にキューに追加されたレコード数。これらのレコードを処理するには、Hub コンソールのマージマネージャを使用します。

一致分析ジョブ

一致分析ジョブでは、メトリックを収集するための検索が実行されますが、実際の一致は実施されません。

大量の一致要件（ホットスポット）が生じる可能性があるデータ領域が見つかった場合は、一致過多を防ぐために、Informatica MDM Hub によって対象のレコードが保留中ステータスに移行されます。保留中のレコードには、統合インジケータ 9 が設定されます。これによりデータスチュワードは、一致と統合を進める前にデータマネージャでデータを手動で確認できます。一致分析ジョブは、通常は一致ルールを調整するために使用するか、あるいはベースオブジェクトのデータが一致過多の状態でないか、または一致過多をもたらす大きなデータの共通部分（ホットスポット）がないかを確認するために使用します。

一致分析ジョブの依存関係

各一致分析ジョブを実行するには、ベースオブジェクト内の新しいレコードと更新されたレコードがトークン化され、一致処理のためにキューに追加されている必要があります。また、テーブル間一致が有効になっているベースオブジェクトに対して一致分析ジョブを実行するには、すべての子テーブルに対してデータトークン化ジョブが正常に終了している必要があり、データトークン化ジョブを実行するには子テーブルに対するロードジョブが正常に終了している必要があります。

保留中レコードの数の制限

一致分析ジョブによって保留中ステータスに移行されるレコードの数を制限することができます。デフォルトでは制限が設定されていません。制限を設定するには、cmxcleanse.properties ファイルを編集し、次の設定を追加します。

```
cmx.server.match.threshold_to_move_range_to_hold = n
```

この n は、一致分析ジョブで保留中ステータスに移行することのできるレコードの最大数です。

cmxcleanse.properties ファイルの詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

一致分析ジョブメトリック

一致分析ジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリックが表示されます（該当する場合）。

メトリック	説明
トークン化されたレコード	一致分析ジョブによってトークン化されたレコード数。
保留ステータスに移行したレコード	一致過多を回避するために「保留」ステータス（統合インジケータ=9）に移行されたレコードの数。これらのレコードは通常はデータ内のホットスポットを示し、一致プロセスでは処理されません。データスチュワードは、データマネージャで保留ステータスを解除できます。
（一致用に）分析されたレコード	一致のために分析されたレコードの数。
必要な一致比較数	このベースオブジェクトを処理するために必要となる実際の一致の数。

実行ログのメトリック

メトリック	説明
保留ステータスに移行したレコード	保留に移動するレコードの数
（一致用に）分析されたレコード	一致に対して分析されたレコードの数
必要な一致比較数	このベースオブジェクトを処理するために必要となる実際の一致の数

統計

統計	説明
範囲内の上位 10 件	指定された検索範囲内の上位 10 件のレコード数。
範囲内の上位 10 件の比較	指定された検索範囲内に対して実行される必要のある上位 10 件の一致比較数。
保留に移動した総レコード数	保留に移動したレコードの数。
保留に移動した総一致数	レコードを保留に移動する必要がある一致の合計数。
処理される合計範囲数	ベースオブジェクトのすべての一致を処理する必要がある範囲の数。
合計候補数	このベースオブジェクトのすべての一致を処理する必要がある一致候補の合計数。
分析時間	分析に実行必要な合計時間。

重複データの一致ジョブ

重複データの一致ジョブでは、完全な重複が検索されて、それらが一致していると見なされます。

完全な重複の最大数は、ベースオブジェクトごとにスキーママネージャで重複一致しきい値プロパティに定義されるベースオブジェクトカラムに基づいています。

注: 重複一致しきい値が 1 に設定されていて、かつベースオブジェクトで非同等一致が有効になっている場合は、バッチビューアに重複データの一致ジョブが表示されません。

重複データを一致させる手順

1. 重複データの一致ジョブは、ロードジョブの終了直後に実行します。
2. 重複データの一致ジョブが完了したら、自動マージジョブを実行して、重複データの一致ジョブで検出された重複データを処理します。
3. 自動マージジョブが完了したら、通常的一致/マージプロセスを実行します（一致ジョブ、自動マージジョブの順に実行するか、または自動一致とマージジョブを実行）。

複数マージジョブ

複数マージジョブは、複数のレコードを 1 つのジョブにマージします。基本的に、1 つのバッチとしてマージするレコード全体を組み込みます。このバッチジョブは、SIF MultiMergeRequest 要求を呼び出す外部アプリケーションによってのみ開始されます。詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

昇格ジョブ

状態が有効なオブジェクトに対して、昇格ジョブは PROMOTE_IND カラムを XREF テーブルから読み取り、カラムの値が 1 のすべての行に対して、システムの状態をアクティブにします。

Informatica MDM Hub は昇格ジョブの実行後に PROMOTE_IND をリセットします。

注: レコードが昇格されなかった場合、昇格バッチプロセスの実行中にレコードの PROMOTE_IND カラムは 0 に変更されません。

昇格ジョブの動作の詳細を以下に示します。

昇格前の XREF の状態	昇格前のベースオブジェクトの状態	XREF に対する Hub のアクション	BO に対する Hub のアクション	BVT を更新するか	結果の BO の状態	操作の結果
保留	アクティブ	昇格	更新	はい	アクティブ	Informatica MDM Hub は保留の XREF を昇格し、昇格した XREF を含めて BVT を再計算します。
保留	保留	昇格	昇格	はい	アクティブ	Informatica MDM Hub は保留の XREF およびベースオブジェクトを昇格します。BVT は昇格した XREF に基づいて計算されます。

昇格前の XREF の状態	昇格前のベースオブジェクトの状態	XREF に対する Hub のアクション	BO に対する Hub のアクション	BVT を更新するか	結果の BO の状態	操作の結果
削除済み	この操作はベースオブジェクトレコードの状態にかかわらず同様に動作します。	なし	なし	いいえ	結果のベースオブジェクトレコードの状態はこの操作によって変化しません。	Informatica MDM Hub はバッチ昇格の削除済みレコードを無視します。このシナリオは、昇格のフラグが付けられたレコードが昇格バッチプロセスの実行前に削除された場合に発生します。
アクティブ	この操作はベースオブジェクトレコードの状態にかかわらず同様に動作します。	なし	なし	いいえ	結果のベースオブジェクトレコードの状態はこの操作によって変化しません。	Informatica MDM Hub はバッチ昇格のアクティブレコードを無視します。このシナリオは、昇格のフラグが付けられたレコードが昇格バッチプロセスの実行前にアクティブになった場合にのみ発生します。

注: 昇格操作と削除操作は子レコードに指示するためにカスケードします。

Hub コンソールを使用した昇格ジョブの実行

昇格ジョブを実行する手順

- Hub コンソールで、以下のいずれかのツールを開始します。
 - バッチビューア
 - バッチグループ
- 対象のベースオブジェクトの昇格ジョブを選択します。
- 昇格ジョブを実行します。
- 昇格ジョブの結果を表示します。

Informatica MDM Hub に昇格ジョブの結果が表示されています。

昇格ジョブメトリック

昇格ジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリックが表示されます（該当する場合）。

メトリック	説明
自動昇格したレコード数	昇格ジョブによって昇格されたレコード数。
削除された相互参照レコード	昇格ジョブによって削除された XREF レコード数。
昇格していないアクティブなレコード	昇格されなかったアクティブなレコードの数。
昇格していない保護されたレコード	昇格されなかった保護されたレコードの数。

昇格ジョブを実行すると、バッチビューアのジョブサマリページでこれらの統計を表示できます。

ベースオブジェクトの再計算ジョブ

ROWID_OBJECT_TABLE パラメータを使用して再計算するベースオブジェクトを指定していない場合、すべてのベースオブジェクトのベストバージョンオブトゥールズを再計算します。

信頼設定または検証設定を変更した後で、ベースオブジェクトの再計算ジョブを実行します。メタデータを同期していても、MDM Hub は、信頼設定または検証設定が変更された後にベストバージョンオブトゥールズを再計算しません。信頼設定または検証設定を変更した後にベストバージョンオブトゥールズを再計算しないと、ベストバージョンオブトゥールズが古いものとなる可能性があります。

ベースオブジェクトの再計算ジョブは、ROWID_OBJECT_TABLE パラメータを使用しても使用しなくても実行できます。ROWID_OBJECT_TABLE パラメータを使用してこのジョブを実行すると、MDM Hub は、テーブルまたはインラインビュー内の ROWID_OBJECT カラムによって特定されるすべてのベースオブジェクトのベストバージョンオブトゥールズを再計算します。インラインビューの前後には括弧が必要です。

ROWID_OBJECT_TABLE パラメータを使用せずにこのジョブを実行すると、MDM Hub は、ベースオブジェクト内のすべてのレコードのベストバージョンオブトゥールズを再計算します。MDM Hub は、バッチサイズが MATCH_BATCH_SIZE のレコードまたは、テーブル内のレコード数の 4 分の 1 のうち、少ないほうを再計算します。

BVT の再計算ジョブ

指定された ROWID_OBJECT のベストバージョンオブトゥールズを再計算します。

信頼設定または検証設定を変更した後で、BVT の再計算ジョブを実行します。メタデータを同期していても、MDM Hub は、信頼設定または検証設定が変更された後にベストバージョンオブトゥールズを再計算しません。信頼設定または検証設定を変更した後にベースオブジェクトを再計算しないと、ベストバージョンオブトゥールズが古いものとなる可能性があります。

一致テーブルのリセットジョブ

一致テーブルのリセットジョブは、一致ジョブの実行後、以下の状況が存在する場合に自動的に作成されます。レコードが consolidation_ind = 2 に更新済みで、一致ルールを変更した場合。

一致処理の後に一致ルールを変更した場合、一致をリセットするように求められます。一致をリセットした場合、一致テーブルのデータはすべて削除されます。さらに、一致テーブルのリセットジョブによって、consolidation_ind が 2 から 4 にリセットされます。

変更をスキーマ一致カラムに保存すると、メッセージボックスが表示され、既存の一致をリセットしてバッチビューアに一致テーブルのリセットジョブを作成するように求められます。**【はい】** をクリックする必要があります。

注: 既存の一致をリセットしない場合、次の一致ジョブを実行するのに時間がかかります。一致ジョブを実行する前に Informatica MDM Hub で一致トークンを再生成する必要があります。

注: このジョブは、バッチビューアからは実行できません。

再検証ジョブ

再検証ジョブは、ロードプロセス中の最初の検証以降に変更されたレコードに対して検証ロジック/ルールを実行します。再検証は、ロードプロセスの最初の検証手順以降にレコードが変更されたときに実行できます。レコードが変更されていない場合は、レコードは更新されません。一部のレコードが変更され、既存の検証ルールで検出されると、結果がメトリックで示されます。

注: 再検証ジョブは、カラムに対して検証が有効になっている場合、初期ロード後および検証ルールを設定したベースオブジェクトのマージ前にのみ実行できます。

再検証は、ベースオブジェクトのバッチビューアを使用して手動で実行します。

ステージジョブ

ステージジョブは、Informatica MDM Hub マッピングでランディングテーブルとステージングテーブル間に設定されたクレンジングを実行し、ランディングテーブルからステージングテーブルにデータを移動します。

ステージジョブは、クレンジングジョブを並行して実行します。ステージの状態は、ステージング中にどのプロセスサーバーがヒットするかを示します。

状態が有効なベースオブジェクトに対して、HUB_STATE_IND 値が有効ではない場合、レコードは却下されません。

注: ステージジョブがグレー表示されている場合、ステージングテーブル、カラムマッピング、またはクレンジング関数の変更により、マッピングは無効になっています。マッピングツールを使用して特定のマッピングを開き、確認して保存します。

ステージジョブのガイドライン

ステージジョブのバッチジョブを実行する場合:

- ステージジョブは、ステージジョブが使用するランディングテーブルをロード処理する ETL プロセスが正常に完了した場合にのみ実行します。
- ステージジョブ間に依存関係がないことを確認します。
- 複数のプロセスサーバーがジョブを実行するように設定されている場合、複数のステージジョブを同時に実行できます。

ステージジョブメトリック

ステージジョブの実行後、バッチビューアのジョブ実行ログに以下のメトリックが表示されます。

メトリック	説明
総レコード数	ステージジョブによって処理されたレコードの数。
挿入済み	ステージジョブによってターゲットオブジェクトに挿入されたレコードの数。
却下	ステージジョブによって却下されたレコードの数。

同期ジョブ

同期ジョブによって、ベースオブジェクトのメタデータが更新されます。スキーマの信頼設定を変更した後、およびロードジョブを実行する前に、同期ジョブを実行します。

ベースオブジェクトを保存する際に、次の条件が当てはまる場合に同期ジョブを実行するよう、MDM Hub によって求められます。

- ベースオブジェクトにデータが含まれる。
- ベースオブジェクトに、新たに追加された、信頼が有効なカラムが存在する。

同期ジョブを実行するには、バッチビューアに移動して、実行するベースオブジェクトの同期ジョブを見つけます。次に、**【バッチの実行】** ボタンをクリックしてジョブを実行します。初期ロードが実行されると、MDM Hub によって、信頼が有効なカラムを持つベースオブジェクトのメタデータが更新されます。同期ジョブを実行したら、ロードジョブを実行できます。

ベースオブジェクト内で定義された信頼が有効なカラムが多すぎる場合、またはカラム名が長すぎる場合、同期ジョブは失敗します。カラム名が、最大許容長さの 26 文字に近い場合、そのカラム名は長すぎると見なされます。ジョブの失敗を防ぐには、短いカラム名にして、信頼が有効なカラム数を 48 個以下にします。また

は、ベースオブジェクトを保存する前に、すべての信頼および検証カラムを有効にする場合は、同期ジョブの必要性を省くことができます。

第 28 章

ユーザー出口

この章では、以下の項目について説明します。

- [ユーザーイグジットの概要, 586 ページ](#)
- [ユーザー出口の処理, 587 ページ](#)
- [ユーザー出口 JAR ファイル, 588 ページ](#)
- [UserExitContext クラス, 589 ページ](#)
- [ステージプロセスユーザーイグジット, 591 ページ](#)
- [ロードプロセスユーザーイグジット, 594 ページ](#)
- [一致プロセスユーザーイグジット, 595 ページ](#)
- [マージプロセスユーザーイグジット, 597 ページ](#)
- [アンマージプロセスユーザー出口, 598 ページ](#)
- [タスク管理ユーザー出口, 601 ページ](#)
- [ユーザー出口内でのサービス統合フレームワークの API の使用, 602 ページ](#)
- [ユーザーイグジットの実装のガイドライン, 605 ページ](#)

ユーザーイグジットの概要

ユーザーイグジットとは、バッチまたはサービス統合フレームワーク（SIF）の API プロセスの特定の時点で実行して MDM Hub の機能を拡張するためにユーザーが開発するカスタム Java コードです。MDM Hub は通常の MDM Hub 処理を終了して、ユーザーが開発したコードを実行します。

例えば、ランディング後ユーザーイグジットを使用して、差分検出プロセスの前にアドレスの事前クレンジングを実行することができます。

ユーザーイグジットは、すべて 1 つの JAR ファイルで MDM Hub にアップロードできます。各オペレーショナル参照ストアに 1 つの JAR ファイルをアップロードできます。

MDM Hub では、ユーザーイグジットの呼び出し時に入力パラメータの値が指定されます。パフォーマンスが不必要に低下しないように、ユーザーイグジットはガイドラインに従って実装してください。

ユーザーイグジットは、以下の MDM Hub プロセスに実装できます。

ステージプロセス

ステージプロセスでは、ランディングテーブルから、ベースオブジェクトに関連付けられたステージングテーブルにデータを移動します。ステージプロセスでは、以下のユーザーイグジットを実行できます。

- ランディング後ユーザーイグジット。ETL プロセスを通じてランディングテーブルに入力した後で、ランディング後ユーザーイグジットを使用してランディングテーブルのデータを絞り込みます。
- ステージング前ユーザーイグジット。差分プロセスのカスタム処理を実行するには、ステージング前ユーザーイグジットを使用します。
- ステージング後ユーザーイグジット。ステージジョブの終了時にカスタム処理を実行するには、ステージング後ユーザーイグジットを使用します。

ロードプロセス

ロードプロセスでは、ステージングテーブルからベースオブジェクトテーブルにデータを移動します。ロードプロセスでは、以下のユーザーイグジットを実行できます。

- ロード後ユーザーイグジット。ロードプロセス後にカスタム処理を実行するには、ロード後ユーザーイグジットを使用します。

一致プロセス

一致プロセスでは、重複候補であるベースオブジェクトレコードが特定されます。一致プロセスでは、以下のユーザーイグジットを実行できます。

- マッチ前ユーザーイグジット。一致プロセスの前にカスタム処理を実行するには、マッチ前ユーザーイグジットを使用します。
- マッチ後ユーザーイグジット。マッチテーブルでカスタム処理を実行するには、マッチ後ユーザーイグジットを使用します。

マージプロセス

マージプロセスでは、重複するベースオブジェクトレコードが1つのマスタベースオブジェクトレコードに統合されます。マージプロセスでは、以下のユーザーイグジットを実行できます。

- マージ後ユーザーイグジット。マージプロセス後にカスタム処理を実行するには、マージ後ユーザーイグジットを使用します。

アンマージプロセス

アンマージプロセスでは、1つのマスタベースオブジェクトレコードが、レコードがマージされる前に存在した個々のベースオブジェクトレコードにアンマージされます。アンマージプロセスでは、以下のユーザーイグジットを実行できます。

- アンマージ前ユーザーイグジット。アンマージプロセスの前にカスタム処理を実行するには、アンマージ前ユーザーイグジットを使用します。
- アンマージ後ユーザーイグジット。アンマージプロセス後にカスタム処理を実行するには、アンマージ後ユーザーイグジットを使用します。

ユーザー出口の処理

MDM Hub は、ブロック、バッチ、または SIF の API トランザクションの一部としてユーザー出口を処理します。ユーザー出口で例外が生成された場合、MDM Hub はブロック、バッチ、または SIF の API 呼び出しの処理をロールバックします。SIF の API およびバッチプロセスは同じユーザー出口を呼び出します。

バッチプロセス中にユーザー出口を実行すると、ロード後およびマージ後ユーザー出口の場合を例外として、ユーザー出口はバッチジョブの実行前または実行後に実行されます。ロード後およびマージ後ユーザー出口

は、MDM Hub による各ブロックの処理後に実行されます。プロセスサーバーを設定する場合は、cmxserver.properties ファイルでブロックサイズを設定します。

ユーザー出口 JAR ファイル

ユーザー出口インタフェースクラスに基づいて、カスタムユーザー出口クラスを開発できます。カスタムユーザー出口クラスを別個の JAR ファイルに実装し、JAR ファイルを MDM Hub にアップロードしてユーザー出口を登録します。

MDM Hub のインストールでは、mdm-ue.jar という JAR ファイルが <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/lib に含まれています。mdm-ue.jar ファイルには、各ユーザー出口に対する Java インタフェースクラスが含まれています。

ユーザー出口 JAR ファイルとカスタムコードを MDM Hub にアップロードするには、Hub コンソールを使用します。各オペレーショナル参照ストアについて、1 つのユーザー出口 JAR ファイルをアップロードできます。MDM Hub でユーザー出口の実装を変更する場合は、最初に MDM Hub からユーザー出口を削除します。次に、最新の実装のユーザー出口が含まれている JAR ファイルをアップロードします。

ユーザーイグジット JAR ファイルの実装

ユーザーイグジット JAR ファイルを実装するには、カスタム Java コードを作成して、そのコードを JAR ファイルにパッケージ化します。

1. Java 開発ツールを使用して、ユーザーイグジットインタフェースを実装するクラスを作成します。
2. 実装したカスタムユーザーイグジットクラスを JAR ファイルとしてエクスポートします。これは Java 開発ツールを使用して、または次の例のように Java コマンドを使用して行えます。

```
<java_project_folder>\bin>jar -cf <user_specified_JAR_file_name> .\com\userexit\*.class
```

MDM Hub へのユーザーイグジットのアップロード

ユーザーイグジットをオペレーショナル参照ストアにアップロードするには、Hub コンソールのユーザーオブジェクトレジストリを使用します。

1. ユーザーイグジットをアップロードするオペレーショナル参照ストアに接続していることを確認します。
2. **【ユーティリティ】** ワークベンチで **【ユーザーオブジェクトレジストリ】** を選択します。
3. 書き込みロックを取得します。
4. ナビゲーションペインで **【ユーザーイグジット】** を選択します。
5. **【追加】** ボタンをクリックします。
6. **【ユーザーイグジットの追加】** ウィンドウで、**【参照】** をクリックします。
7. **【開く】** ウィンドウで、ユーザーイグジットが含まれている JAR ファイルを参照します。 **【開く】** をクリックします。
8. **【ユーザーイグジットの追加】** ウィンドウで、任意で説明を入力し、**【OK】** をクリックします。
プロパティペインのユーザーイグジットテーブルに、ユーザーイグジットが表示されます。

MDM Hub からのユーザーイグジットの削除

MDM Hub からすべてのユーザーイグジットを削除するには、ユーザーオブジェクトレジストリからユーザーイグジットを削除します。ユーザーイグジットを個別に削除することはできません。

1. **【ユーティリティ】** ワークベンチで **【ユーザーオブジェクトレジストリ】** を選択します。
2. 書き込みロックを取得します。
3. ナビゲーションペインで **【ユーザーイグジット】** を選択します。
4. プロパティペインでユーザーイグジットテーブルを選択し、**【削除】** ボタンをクリックします。 **【OK】** をクリックします。

UserExitContext クラス

UserExitContext クラスには、MDM Hub でユーザー出口に対して指定されるパラメータが含まれています。

UserExitContext クラスは、以下のパラメータをユーザー出口に渡します。

batchJobRowid

バッチジョブのジョブ ID。UserExitContext クラスは、バッチプロセスの実行中に存在するすべてのユーザー出口に、BatchJobRowid パラメータを渡します。

接続

MDM Hub プロセスで使用されるデータベース接続。

stagingTableName

ロードジョブのソーステーブル。UserExitContext クラスは、ロードおよびステージプロセス中に stagingTableName パラメータを渡します。

tableName

MDM Hub プロセスが使用するテーブルの名前。

以下のコードは、UserExitContext クラスを示しています。

```
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 stagingTableName;
    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;
    }

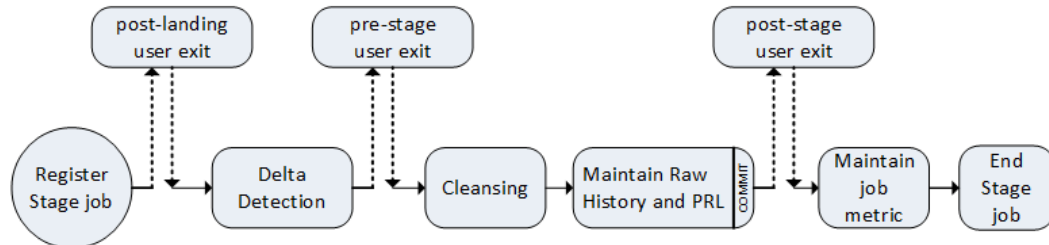
    /**
     * Set the database connection in the context.
     *
     * @param connection the database connection used in the hub process
     */
    public void setDBConnection(Connection connection) {
        this.conn = connection;
    }
}

```

ステージプロセスユーザーイグジット

ステージプロセスでは、ランディング後、ステージング前、およびステージング後ユーザーイグジットを呼び出すことができます。

以下の図に、ステージプロセスで呼び出すことができるステージプロセスとユーザーイグジットを示します。



ユーザーイグジットは、ステージプロセスで以下の順序で実行されます。

1. MDM Hub がステージジョブを登録します。
2. ランディング後ユーザーイグジットが実行されます。
3. MDM Hub がデルタ検出を実行します。
4. ステージング前ユーザーイグジットが実行されます。
5. MDM Hub がデータクレンジングを実行します。
6. MDM Hub がステージテーブルに入力します。
7. 監査証跡を有効にすると、MDM Hub が RAW テーブルに入力します。
8. MDM Hub がステージテーブルの変更をコミットします。
9. ステージング後ユーザーイグジットが実行されます。
10. ステージジョブが終了します。

ランディング後ユーザー出口

MDM Hub でステージジョブが登録されると、MDM Hub がランディング後ユーザー出口を呼び出します。

ETL プロセスを通じてランディングテーブルに入力した後で、ランディング後ユーザー出口を使用してランディングテーブルのデータを絞り込みます。ランディング後ユーザー出口を使用して、差分検出の前にランディングテーブルでカスタム処理を実行することができます。例えば、物理削除の検出を実行したり、制御文字を印刷可能な文字に置き換えたり、アドレスに対するクレンジング前処理を実行したりすることができます。

ランディング後ユーザー出口インタフェース

ランディング後ユーザー出口を実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

ランディング後ユーザー出口では、以下の完全修飾インタフェース名を使用します。

`com.informatica.mdm.userexit.PostLandingUserExit`

メソッド

ランディング後ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String landingTableName, String previousLandingTableName) throws Exception;
```

コンセンサスタイプの物理削除の検出を使用する場合は、メソッドに次のロジックを追加します。

```
ConsensusFlagUpdate consensusProcess = new ConsensusFlagUpdate(userExitContext.getBatchJobRowid(),
stagingTableName);
consensusProcess.startConsensusFlagUpdate(userExitContext.getDBConnection());
```

パラメータ

ランディング後ユーザー出口では、以下のパラメータを使用します。

landingTableName

ステージジョブのソーステーブル。

previousLandingTableName

前のランディングテーブル名。前回のステージジョブの実行からステージングテーブルにマップされたソースデータのコピーが含まれています。

stagingTableName

ステージジョブのターゲットテーブル。

userExitContext

パラメータをユーザー出口に渡します。

ステージング前ユーザー出口

MDM Hub は、ステージングテーブルにデータをロードする前に、ステージング前ユーザー出口を呼び出します。

差分プロセスのカスタム処理を実行するには、ステージング前ユーザー出口を使用します。ステージング前ユーザー出口を使用して、差分量が定義済みの許容制限値を超えていないかどうかを判定できます。例えば、ソースシステムからの差分量が 500,000 を超えた場合には、ユーザー出口を使用してステージプロセスを停止することができます。

ステージング前ユーザー出口インタフェース

ステージング前ユーザー出口を実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

ステージング前ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.PreStageUserExit
```

メソッド

ステージング前ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String landingTableName,
String deltaTableName) throws Exception;
```

パラメータ

ステージング前ユーザー出口では、以下のパラメータを使用します。

deltaTableName

差分テーブルの名前。差分テーブルには、MDM Hub が差分として特定したレコードが含まれています。

landingTableName

ステージジョブのソーステーブル。

stagingTableName

ステージジョブのターゲットテーブル。

userExitContext

パラメータをユーザー出口に渡します。

ステージング後ユーザー出口

MDM Hub は、MDM Hub がステージングテーブルにデータをロードした後で、ステージング後ユーザー出口を呼び出します。

ステージジョブの終了時にカスタム処理を実行するには、ステージング後ユーザー出口を使用します。ステージング後ユーザー出口を使用して、ステージジョブで却下されたレコードを処理することもできます。例えば、既知の重大でない条件に基づいて MDM Hub が却下したレコードが削除されるように、ユーザー出口を設定することができます。

ステージング後ユーザー出口インタフェース

ステージング後ユーザー出口を実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

ステージング後ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.PostStageUserExit
```

メソッド

ステージング後ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String landingTableName,
    String previousLandingTableName) throws Exception;
```

物理削除の検出を使用する場合は、メソッドに次のロジックを追加します。

```
HardDeleteDetection hdd = new HardDeleteDetection(rowidJob, stagingTableName);
hdd.startHardDeleteDetection(userExitContext.getDBConnection());
```

パラメータ

ステージング後ユーザー出口では、以下のパラメータを使用します。

landingTableName

ステージジョブのソーステーブル。

previousLandingTableName

前のランディングテーブル名。前回のステージジョブの実行からステージングテーブルにマップされたソースデータのコピーが含まれています。

stagingTableName

ステージジョブのターゲットテーブル。

userExitContext

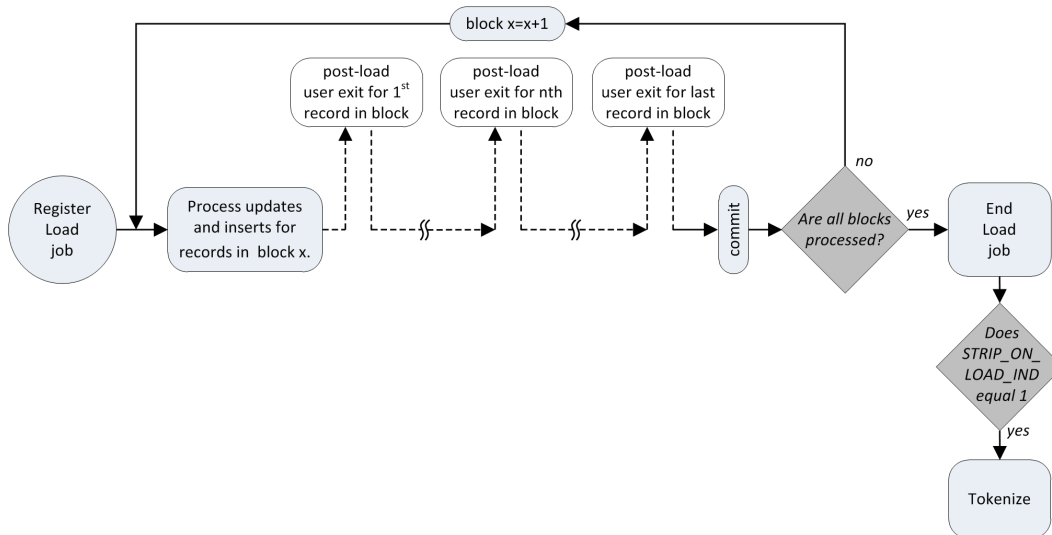
パラメータをユーザー出口に渡します。

ロードプロセスユーザーイグジット

ロードプロセスでは、ロード後ユーザーイグジットを呼び出すことができます。

ロードしたジョブの数が 0 より大きい場合、ロード後ユーザーイグジットが呼び出されます。ロード後ユーザーイグジットは、バッチ処理の最後ではなく、MDM Hub が各ブロックを処理した後で実行されます。ロード後ユーザーイグジットは、MDM Hub がブロック内のすべてのレコードの更新と挿入を処理した後で、ブロック内のレコードごとに実行されます。

以下の図に、ロードプロセスで呼び出すことができるロードプロセスとユーザーイグジットを示します。



ロード後ユーザーイグジットは、ロードプロセスで以下の順序で実行されます。

1. MDM Hub がロードジョブを登録します。
2. MDM Hub がレコードを更新または最初のブロックにレコードを挿入します。
3. ロード後ユーザーイグジットは、ブロック内のレコードごとに実行されます。
4. MDM Hub が変更をコミットします。
5. MDM Hub にロードが必要なブロックが他にもある場合は、プロセスが手順 2 に戻り、次のブロックを処理します。
6. MDM Hub がすべてのブロックをロードすると、ロードジョブが終了します。
7. strip_on_load_ind の値が 1 である場合には、トークン化ジョブによって、あいまいマッチに必要なマッチトークンが生成されます。

ロード後ユーザーイグジット

MDM Hub はロードプロセス後にロード後ユーザーイグジットを呼び出します。

ロードプロセス後にカスタム処理を実行するには、ロード後ユーザーイグジットを使用します。

注: ロード後のユーザーイグジットの再帰呼び出しを避けるために、ロード後のユーザーイグジットに MultiMerge という SIF の API を使用しないでください。ロード後のユーザーイグジットで Put という API を使用するときは、ロード後のユーザーイグジットの再帰呼び出しを防ぐために、Put の API の BypassPostLoadUE パラメータを true に設定してください。

ロード後ユーザーインターフェース

ロード後ユーザーを実装する場合は、適切なインターフェース名、メソッド、パラメータを使用します。

インターフェース名

ロード後ユーザーでは、以下の完全修飾インターフェース名を使用します。

```
com.informatica.mdm.userexit.PostLoadUserExit
```

メソッド

ロード後ユーザーでは、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, ActionType actionType,  
    Map<String, Object>baseObjectDataMap, Map<String, Object> xrefDataMap,  
    List<Map<String, Object>> xrefDataMapList) throws Exception;
```

パラメータ

ロード後ユーザーでは、以下のパラメータを使用します。

actionType

ロードプロセスによってレコードの挿入とレコードの更新のどちらが行われたかを示します。

baseObjectDataMap

ロードプロセスによって更新または挿入が行われた、ベースオブジェクト内のデータが含まれます。

userExitContext

パラメータをユーザー出口に渡します。

xrefDataMap

更新または挿入の要因となった、相互参照レコード内のデータが含まれます。

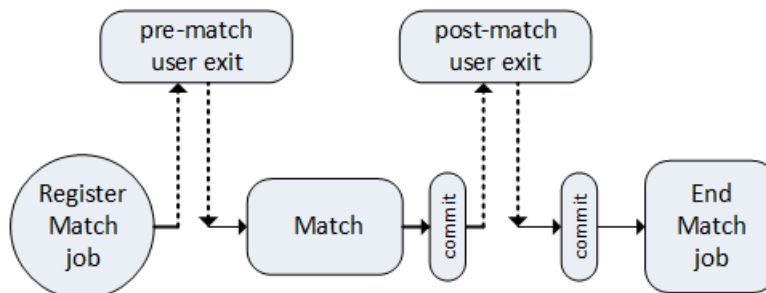
xrefDataMapList

有効期間が重複しているために MDM Hub によって変更された、相互参照レコード内のデータが含まれます。MDM Hub は、タイムラインが有効なベースオブジェクトの xrefDataMapList パラメータに入力します。

一致プロセスユーザーイグジット

一致プロセスでは、マッチ前およびマッチ後ユーザーイグジットを呼び出すことができます。

以下の図に、一致プロセスで呼び出すことができる一致プロセスとユーザーイグジットを示します。



ユーザーイグジットは、一致プロセスで以下の順序で実行されます。

1. MDM Hub が一致ジョブを登録します。
2. マッチ前ユーザーイグジットが実行されます。
3. MDM Hub が一致ジョブを実行します。
4. MDM Hub がマッチの変更をコミットします。
5. マッチ後ユーザーイグジットが実行されます。
6. MDM Hub がマッチの変更をコミットします。
7. 一致ジョブが終了します。

マッチ前ユーザー出口

MDM Hub が、マッチプロセスの前にマッチ前ユーザー出口を呼び出します。

マッチプロセスの前にカスタム処理を実行するには、マッチ前ユーザー出口を使用します。

マッチ前ユーザー出口インタフェース

マッチ前ユーザー出口を実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

マッチ前ユーザー出口では、以下の完全修飾インタフェース名を使用します。

`com.informatica.mdm.userexit.PreMatchUserExit`

メソッド

マッチ前ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws Exception;
```

パラメータ

マッチ前ユーザー出口では、以下のパラメータを使用します。

`matchSetName`

マッチルールセットの名前。

`userExitContext`

パラメータをユーザー出口に渡します。

マッチ後ユーザー出口

MDM Hub がマッチプロセス後にマッチ後ユーザー出口を呼び出します。

マッチテーブルでカスタム処理を実行するには、マッチ後ユーザー出口を使用します。例えば、マッチ後ユーザー出口を使用して、マッチキューのマッチを操作することができます。

マッチ後ユーザー出口インタフェース

マッチ後ユーザー出口を実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース

マッチ後ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.PostMatchUserExit
```

メソッド

マッチ後ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws Exception;
```

パラメータ

マッチ後ユーザー出口では、以下のパラメータを使用します。

matchSetName

MDM Hub がマッチの検索に使用するマッチルールセットの名前。

userExitContext

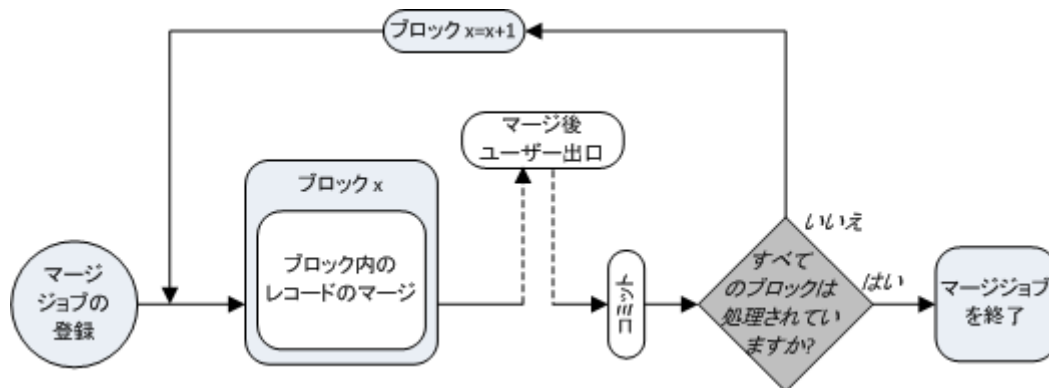
パラメータをユーザー出口に渡します。

マージプロセスユーザーイグジット

マージプロセスでは、マージ後ユーザーイグジットを呼び出すことができます。

Merge SIF API と MultiMerge SIF API は、マージプロセスが完了した後、マージ後ユーザーイグジットを呼び出します。

以下の図に、マージプロセスで呼び出すことができるバッチマージプロセスとユーザーイグジットを示します。



ユーザーイグジットは、バッチマージプロセス内で以下の順序で実行されます。

1. MDM Hub がマージジョブを登録します。
2. MDM Hub がブロック内の一致レコードをマージします。
3. マージ後ユーザーイグジットは、ブロック内の影響を受けるすべてのレコードに対して実行されます。
4. MDM Hub が変更をコミットします。
5. MDM Hub は残りのブロックに対して手順 2 から 4 を繰り返します。
6. MDM Hub がすべてのブロックを処理すると、マージジョブが終了します。

マージ後ユーザーイグジット

MDM Hub はマージプロセス後にマージ後ユーザーイグジットを呼び出します。

マージプロセス後にカスタム処理を実行するには、マージ後ユーザーイグジットを使用します。例えば、マージ後ユーザーイグジットを使用して、親レコードのマッチとマージの影響を受ける子レコードのマッチとマージを実行することができます。

注: マージ後のユーザーイグジットの再帰呼び出しを避けるために、マージ後のユーザーイグジットに MultiMerge という SIF の API を使用しないでください。

マージ後ユーザーインタフェース

マージ後ユーザーを実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

マージ後ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.PostMergeUserExit
```

メソッド

マージ後ユーザー出口では、以下のメソッドを使用します。

```
void processUserExit(UserExitContext userExitContext, Map<String, List<String>> baseObjectRowIds) throws Exception;
```

パラメータ

マージ後ユーザー出口では、以下のパラメータを使用します。

baseObjectRowIds

マージに関連するベースオブジェクトの行 ID のリスト 最初のエントリはターゲットベースオブジェクトレコードです。 リスト内の残りのエントリは、ターゲットにマージされたソースベースオブジェクトです。

Map<String, List<String>>

ターゲット行 ID をキーにしたマップ

userExitContext

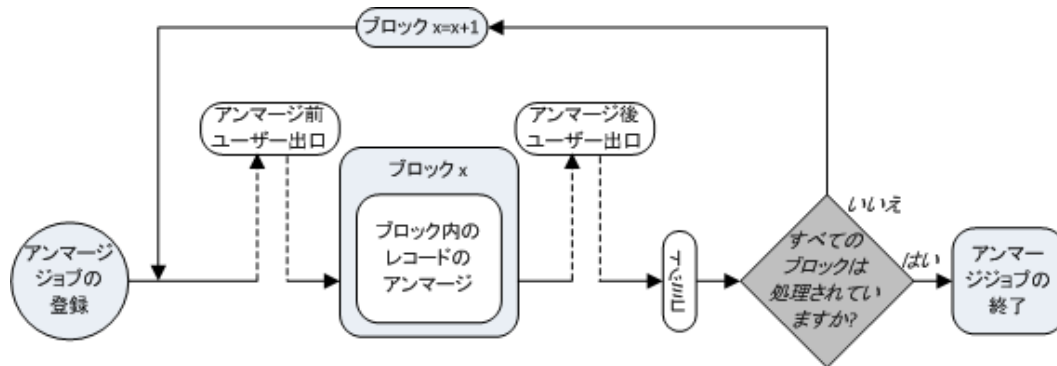
パラメータをユーザー出口に渡します。

アンマージプロセスユーザー出口

アンマージプロセスでは、アンマージ前およびアンマージ後ユーザー出口を呼び出すことができます。

Unmerge という SIF の API が、アンマージプロセスが開始する前にアンマージ前ユーザー出口を呼び出します。Unmerge という SIF の API が、アンマージプロセスが完了した後、アンマージ後ユーザー出口を呼び出します。

以下の図に、バッチアンマージプロセスで呼び出すことができるバッチアンマージプロセスとユーザー出口を示します。



ユーザー出口は、バッチアンマージプロセスで以下の順序で実行されます。

1. MDM Hub がアンマージジョブを登録します。
2. アンマージ前ユーザー出口が実行されます。
3. MDM Hub が、最初のブロック内のレコードをアンマージします。
4. アンマージ後ユーザー出口が実行されます。
5. MDM Hub が変更をコミットします。
6. MDM Hub が、残りのブロックに対して手順 2～5 を繰り返します。
7. MDM Hub がすべてのブロックを処理すると、アンマージジョブが終了します。

アンマージ前ユーザーイグジット

MDM Hub はアンマージプロセスの前にアンマージ前ユーザーイグジットを呼び出します。

アンマージプロセスの前にカスタム処理を実行するには、アンマージ前ユーザーイグジットを使用します。

注: アンマージ前のユーザーイグジットの再帰呼び出しを避けるために、アンマージ後のユーザーイグジットに Unmerge という SIF の API を使用しないでください。

アンマージ前ユーザーイグジットインタフェース

アンマージ前ユーザーイグジットを実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

アンマージ前ユーザーイグジットでは、以下の完全修飾インタフェース名を使用します。

`com.informatica.mdm.userexit.PreUnmergeUserExit`

メソッド

アンマージ前ユーザーイグジットでは、以下のメソッドを使用します。

`void processUserExit(UserExitContext userExitContext, Set<UnmergeKey> unmergeKeys) throws Exception;`

パラメータ

アンマージ前ユーザーイグジットでは、以下のパラメータを使用します。

`pkeySourceObject`

ソースオブジェクトのプライマリキー。UnmergeKeys によってブロック内の各レコードに渡されます。

rowidSystem

マージ解除で処理されるベースオブジェクトの相互参照レコードのシステム行 ID。UnmergeKeys によってブロック内の各レコードに渡されます。

userExitContext

パラメータをユーザーイグジットに渡します。

アンマージ後ユーザーイグジット

MDM Hub はアンマージプロセス後にアンマージ後ユーザーイグジットを呼び出します。

アンマージプロセス後にカスタム処理を実行するには、アンマージ後ユーザーイグジットを使用します。

アンマージ後ユーザーイグジットインタフェース

アンマージ後ユーザーイグジットを実装する場合は、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

アンマージ後ユーザーイグジットでは、以下の完全修飾インタフェース名を使用します。

`com.informatica.mdm.userexit.PostUnmergeUserExit`

メソッド

アンマージ後ユーザーイグジットでは、以下のメソッドを使用します。

`void processUserExit(UserExitContext userExitContext, Set<PostUnmergeResponse> responses) throws Exception;`

パラメータ

アンマージ後ユーザーイグジットでは、以下のパラメータを使用します。

boTableName

マージ解除されたレコードが含まれるベースオブジェクトテーブルの名前。PostUnmergeResponse によってブロック内の各レコードに渡されます。

rowidObject

回復したベースオブジェクトレコードの行 ID。PostUnmergeResponse によってブロック内の各レコードに渡されます。

userExitContext

パラメータをユーザーイグジットに渡します。

タスク管理ユーザー出口

AssignTasks ユーザー出口と GetAssignableUsersForTask ユーザー出口をタスク管理に使用できます。

AssignTasks ユーザー出口インタフェース

AssignTasks ユーザー出口を実装するとき、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

AssignTasks ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.AssignTasksUserExit
```

メソッド

AssignTasks ユーザー出口では、以下のメソッドを使用します。

```
void processAssignTasks(UserExitContext userExitContext, int maxTasks)
```

パラメータ

AssignTasks ユーザー出口では、以下のパラメータを使用します。

userExitContext

パラメータをユーザー出口に渡します。

maxTasks

各ユーザーに割り当てるタスクの最大数です。

GetAssignableUsersForTask ユーザー出口インタフェース

GetAssignableUsersForTask ユーザー出口を実装するとき、適切なインタフェース名、メソッド、パラメータを使用します。

インタフェース名

GetAssignableUsersForTask ユーザー出口では、以下の完全修飾インタフェース名を使用します。

```
com.informatica.mdm.userexit.GetAssignableUsersForTaskUserExit
```

メソッド

GetAssignableUsersForTask ユーザー出口では、以下のメソッドを使用します。

```
public List<String> processGetAssignableUsersForTask(String taskType, String rowidSubjectArea, UserExitContext userExitContext)
```

パラメータ

GetAssignableUsersForTask ユーザー出口では、以下のパラメータを使用します。

rowidSubjectArea

サブジェクト領域の行 ID。

taskType

タスクのタイプ。

userExitContext

パラメータをユーザー出口に渡します。

ユーザー出口内でのサービス統合フレームワークの API の使用

サービス統合フレームワーク（SIF）の API を呼び出すためのユーザー出口を作成することができます。ユーザー出口は SIF クライアントを使用し、これがユーザー出口コードで実装した SIF の API を呼び出します。

SIF クライアントを作成するには、mdm-ue.jar ファイルに含まれる UserExitSifClient という Java クラスが使用できます。ユーザー出口の実装内でのみ SIF クライアントを使用して SIF の API を呼び出すことができます。データベース接続情報を入力パラメータとして使用します。

ユーザー出口と SIF の API の呼び出しは、同じデータベース接続を使用するため、同じトランザクションの一部です。ユーザー出口が SIF の API を呼び出すときの処理は、トランザクショナルになります。まだコミットされていない変更は表示することができます。エラーが発生すると、処理全体をロールバックすることができます。

サービス統合フレームワークの API を呼び出すためのユーザーイグジットの作成

カスタムコードを追加して、サービス統合フレームワーク（SIF）の API を呼び出すユーザーイグジットを作成することができます。SIF の API を呼び出す SIF クライアントを作成するには、このコードを追加する必要があります。

1. Java の統合開発環境（Eclipse など）で Java プロジェクトを作成します。
2. 次に示す MDM Hub JAR ファイルを、その Java プロジェクトに追加します。
 - mdm-ue.jar
 - siperian-api.jar
 - log4j-<version number>.jarjar ファイルは次のディレクトリにあります。
 - UNIX の場合: <MDM Hub のインストールディレクトリ>/hub/server/lib
 - Windows の場合: <MDM Hub のインストールディレクトリ>\hub\server\lib
3. ユーザーイグジットインタフェースを実装するために、mdm-ue.jar ファイルの中にユーザーイグジットの Java クラスを作成します。
4. SIF の API を呼び出すカスタムコードを追加します。
 - a. mdm-ue.jar ファイルで UserExitSifClient クラスを使用してユーザーイグジットクライアントを作成します。
 - b. 呼び出す必要のある SIF の API リクエストを定義します。
 - c. ユーザーイグジットの SIF クライアントを利用して SIF の API を呼び出すコードを指定します。
5. この Java クラスをコンパイルして、クラスファイルをカスタムユーザーイグジットの JAR ファイルにパッケージ化します。
6. Hub コンソールで【ユーザーオブジェクトレジストリ】ツールを使用してカスタムユーザーイグジットの JAR ファイルを MDM Hub にアップロードします。

ユーザーイグジットの例

会社で、あるレコードに対してあいまい一致を実行する必要があります。あいまい一致を実行できるようにするために、事前にマッチトークンを生成し、それをベースオブジェクトに関連するマッチキーテーブルに保存しておく必要があります。

マッチトークンを生成するには、トークン化 API を呼び出す必要があります。トークン化 API を呼び出すユーザーイグジットを設定して、あいまい一致を実行する必要のあるレコードに対するマッチトークンを生成することができます。

次に示すユーザーイグジットのサンプルコードでは、SIF クライアントを使用してトークン化 API を呼び出し、レコードに関するマッチトークンを生成しています。

```
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
```

サービス統合フレームワークの API

サービス統合フレームワーク（SIF）の API の呼び出しに、ユーザーイグジットの JAR ファイルに含まれる UserExitSifClient という Java クラスを使用することができます。

次のテーブルで、ユーザーイグジットが呼び出すことのできる SIF の API について説明します。

SIF の API	説明
CanUnmergeRecords	指定する相互参照レコードが統合ベースオブジェクトレコードからマージ解除できるかどうかを決定します。
CleansePut	キーが識別する単一レコードをベースオブジェクトに挿入または更新します。
Cleanse	リクエストの中で指定する入力レコードを、選択する MDM Hub クレンジング関数で指定された出力形式に変換します。
CleanTable	オペレーショナルリファレンスストアおよびそれに関連付けられたすべてのテーブルから、データを削除します。
CreateTask	タスクを作成します。
削除	ベースオブジェクトからレコードを削除します。deleteBORecord フラグを指定すると、ソースキーとシステム名のみを指定した場合でも、API の Delete によってベースオブジェクトレコードが削除されます。
FlagForAutomerge	一致テーブルで自動マージのレコードにフラグを付けます。一致テーブルの中の q wレコードの自動マージインジケータの値が 1 の場合、次の自動マージプロセスの中でそのレコードがマージされます。そのレコードが存在しない場合、FlagForAutomerge という API が一致テーブルにレコードを作成し、自動マージインジケータを 1 に設定します。
GetAssignableUsersForTasks	タスクの割り当て先に指定できるユーザーのリストを取得します。
GetEffectivePeriods	ベースオブジェクトレコードの集計有効期間を取得します。
GetMatchedRecords	レコードの一致候補を取得します。
GetMergeHistory	ベースオブジェクトのマージ履歴を表示するツリーを取得します。
Get	既知のキーを使用してパッケージから単一レコードを取得します。
GetSearchResults	検索クエリで見つかったレコードの数が、検索クエリから返ってくると期待するレコードの数を越えたときに、追加のデータを取得します。
GetTasks	タスクのリストとタスクの詳細を取得します。
GetTrustScore	ベースオブジェクトレコードのカラムに関して現在の信頼スコアを取得します。
GetXrefForEffectiveDate	有効な日付に対し複数の相互参照レコードを取得します。
マージ	同じオブジェクトを代表する 2 つのベースオブジェクトレコードをマージします。

SIF の API	説明
MultiMerge	<p>同じオブジェクトを代表する複数のベースオブジェクトレコードをマージします。マージされたレコードに対してフィールドレベルのオーバーライドを指定できます。</p> <p>重要: ロード後およびマージ後のユーザーイグジットでこの API を使用する場合、MDM Hub が再帰呼び出しを生成します。他のユーザーイグジットを使用して MultiMerge の API を呼び出します。</p>
PreviewBvt	<p>指定したレコードの組み合わせがマージされたり保留中の更新が発生したときに、ベースオブジェクトレコードの概略が分かるプレビューを作成します。</p>
PromotePendingXrefs	<p>相互参照レコードを昇格させるか、それに昇格のフラグを付けます。</p>
Put	<p>キーで識別される単一レコードをベースオブジェクトに挿入または更新します。</p> <p>重要: ロード後のユーザーイグジットでこの API を使用すると、Informatica MDM Hub が再帰呼び出しを生成します。ロード後のユーザーイグジットを迂回するために PUT の API の BypassPostLoadUE パラメータを true に設定します。</p>
RemoveMatchedRecords	<p>ベースオブジェクトレコードに関連する一致を、一致テーブルから削除します。</p>
Restore	<p>相互参照レコードを復帰させます。</p>
SearchHmQuery	<p>階層マネージャのエンティティとリレーションを検索します。</p>
SearchMatch	<p>一致カラムとルール定義に基づいてパッケージ内でレコードを検索します。</p>
SearchQuery	<p>パッケージから、指定した基準を満たすレコードの組み合わせを取得します。</p>
SetRecordState	<p>指定したキーで識別されるベースオブジェクトレコードに対して統合インジケータを設定します。</p>
Tokenize	<p>MDM Hub が更新または挿入するベースオブジェクトレコードに対してマッチトークンを生成します。</p>
マージ解除	<p>ベースオブジェクトレコードをマージ解除します。</p> <p>重要: PreUnmerge ユーザーイグジットでこの API を使用する場合、MDM Hub が再帰呼び出しを生成します。他のユーザーイグジットを使用して Unmerge という API を呼び出してください。</p>
UpdateMatchRecord	<p>一致テーブルのレコードを作成または更新します。</p>
UpdateTask	<p>タスクを更新します。</p>

ユーザーイグジットの実装のガイドライン

ユーザーイグジットは、MDM Hub のパフォーマンスを不必要に低下させない方法で実装します。

ユーザーイグジットを実装するときは、次のガイドラインを考慮してください。

ユーザーイグジットがパフォーマンスに与える影響を考慮する。

ユーザーイグジットを実装すると、MDM Hub のパフォーマンスに影響します。どうしてもユーザーイグジットを使用して目的を達成する必要があるか、ユーザーイグジットに依存することなく同じことを非同期的にできないかどうかを検討してください。

例えば、データをロードするときに、保留レコードに対するタスクを作成する必要があるのではないかと感じる場合があります。ユーザーイグジットを使用したタスク作成は可能ですが、そうする必要はありません。プロセスの中にボトルネックが生じ、不必要にパフォーマンスを下げてしまいます。これらのタスクは、プロセスのその時点では特に意味を持っていないため、後で作成してもかまいません。

データベース接続を使用してデータベースに対しクエリまたは更新を行う。

Java のユーザーイグジットのコンテキストがデータベース接続を提供します。このデータベース接続を使用して、JDBC の直接呼び出しまたは SIF の API の呼び出しでデータベースを更新するか、データベースにクエリを実行します。

ユーザーイグジットからの SIF の API の呼び出しがバッチパフォーマンスにどのように影響を与えるかを考慮する。

ユーザーイグジットから SIF の API を呼び出すときは、そのような呼び出しがバッチプロセスにどのような影響を及ぼすかを考慮してください。

可能なときは SIF の API を使用する。

相当する SIF の API が使用できるときは、直接 MDM Hub のテーブルを使用しないでください。

ユーザーイグジットで行った変更に対して、明示的にコミットやロールバックを行わない。

データベース接続を使用して SIF の API の呼び出しまたは JDBC の直接呼び出しを行う場合、その呼び出しはユーザーイグジットの呼び出し元と同じトランザクションに関与します。ユーザーイグジットの呼び出し元はトランザクションで行われた変更点を維持します。ユーザーイグジットに渡された接続に対して、明示的なコールコミットやロールバックを行わないでください。

MDM Hub はユーザーイグジットの認証や承認を行わない。

ユーザーイグジットは、MDM Hub のオブジェクトにフルアクセスして実行されます。ユーザーイグジットは、そのときの状況に関わらず、MDM Hub のオブジェクトにフルアクセスして実行されます。ユーザーイグジットの SIF クライアントを使用してサポート対象の SIF の API の一部を呼び出すことができます。ユーザーイグジットの呼び出し元と同じトランザクションに関与できるのは、サポート対象の SIF の API のみです。ユーザーイグジットの SIF クライアントを使用してサポート外の SIF の API にアクセスすることはできません。

パート VI: アプリケーションアクセスの設定

この部には、以下の章があります。

- [ORS 固有の API, 608](#) ページ
- [ORS 固有のメッセージスキーマ, 613](#) ページ
- [登録済みのカスタムコードの表示, 617](#) ページ

第 29 章

ORS 固有の API

この章では、以下の項目について説明します。

- [ORS 固有の API の概要, 608 ページ](#)
- [パフォーマンスに関する考慮事項, 609 ページ](#)
- [サポートされているリポジトリオブジェクト, 609 ページ](#)
- [アーカイブテーブル, 610 ページ](#)
- [ORS 固有の SIF API の生成とデプロイ, 611 ページ](#)
- [ORS 固有の SIF API の名前変更, 611 ページ](#)
- [ORS 固有のクライアント JAR ファイルのダウンロード, 611 ページ](#)
- [ORS 固有のクライアント JAR ファイルと SIF SDK の併用, 612 ページ](#)
- [ORS 固有の SIF API の削除, 612 ページ](#)

ORS 固有の API の概要

オペレーショナル参照ストア内のオブジェクト（ベースオブジェクト、パッケージ、クレンジング関数など）の API を生成できます。ORS 固有の API を生成するには、Hub コンソールの SIF マネージャツールを使用します。

ORS 固有の API を生成する前に、オペレーショナル参照ストアのベースオブジェクトとパッケージを設定します。ORS 固有の API は、SiperianClient でクライアント JAR ファイルから使用できるほか、Web サービスとしても使用できます。ORS 固有の API を生成するときには、MDM Hub によってバージョン ID が ORS 固有の API クライアント JAR ファイルに関連付けられます。

ORS 固有の API は、特定のオペレーショナル参照ストアオブジェクトに対して機能します。例えば、汎用 API はユーザーが指定するデータベースレコード内にデータを配置することがあります。ORS 固有の API は、名前および電子メールアドレスと同じデータを識別し、それらのフィールドをオペレーショナル参照ストアに定義されているとおりに顧客レコードに配置します。

ORS 固有の API は、サービス統合フレームワーク（SIF）SDK で使用するか、または SOAP Web サービスとして使用します。SIF SDK を使用する場合は、Java Development Kit と Apache Jakarta Ant ビルドシステムをインストールする必要があります。

パフォーマンスに関する考慮事項

ORS 固有の API の生成パフォーマンスは、選択するオブジェクトの数によって異なります。最適なパフォーマンスを実現するには、API を生成する必要があるオブジェクトのみ選択します。

注: 場合によっては、関連する SIF API Javadoc のヒープ領域の不足を防ぐために、ヒープサイズを増やす必要があります。デフォルトのヒープサイズは 256M です。このデフォルト値は、cmxserver.properties ファイルで `sif.jvm.heap.size` パラメータを設定してオーバーライドすることも可能です。

サポートされているリポジトリオブジェクト

安全な特定のリポジトリオブジェクトに対して SIF API を生成できます。オブジェクトを保護状態にするには、Hub コンソールのセキュアリソースツールを使用します。

次のリポジトリオブジェクトには、ORS 固有の SIF API を生成できます。

- ベースオブジェクト
- パッケージ
- マッピング
- クレンジング関数
- 一致カラム
- 一致ルールセット

注: 一致カラムと一致ルールセットに対して API を生成する場合は、関連パッケージを選択したことを確認します。関連パッケージを選択しない場合、一致カラムと一致ルールセットに対して ORS 固有の SIF API が MDM Hub によって生成されません。

ORS 固有の SIF API プロパティ

Hub コンソールで SIF マネージャユーティリティを使用して、ORS 固有の SIF API のプロパティを設定します。

ORS 固有の SIF API の次のプロパティを設定できます。

論理名

ORS の論理名。

論理名を編集し、編集後の名前を一意にすることができます。ORS 固有の SIF API の論理名を編集したら、ORS 固有の SIF API を再生成してデプロイします。

Java 名

ORS の Java 名。

ORS 固有の SIF API のクライアント JAR ファイル名には、Java 名が含まれます。論理名を編集して、Java 名を変更します。編集後の論理名を確実に一意にします。

WSDL URL

ORS 固有の SIF API をデプロイするときに、MDM Hub が生成する WSDL ファイルの URL。

API 生成時刻

ORS 固有の SIF API を生成したときの日付と時間。この形式は `mm/dd/yy hh:mm tt` です。

バージョン ID

MDM Hub が生成しデプロイする ORS 固有の SIF API の一意の ID。

MDM Hub は、次の要素でバージョン ID を使用します。

- エンタープライズマネージャツールの **【環境レポート】** タブと **【ORS データベース】** タブでのプロパティ。
- クライアント JAR ファイルの名前。
- クライアント JAR の MANIFEST.MF ファイル。

リポジトリオブジェクトのステータス

リポジトリオブジェクトのステータスは、MDM Hub がリポジトリオブジェクトに対して ORS 固有の SIF API を生成、デプロイできるかどうかを示します。

Hub コンソールにある SIF マネージャユーティリティの **【選択済みの非同期オブジェクト】** テーブル内の **【ステータス】** カラムに、リポジトリオブジェクトのステータスが表示されます。オブジェクトの更新後、オブジェクトのステータスを更新できます。オブジェクトのステータスを更新するには、SIF マネージャユーティリティの **【SIF API マネージャ】** タブで **【オブジェクトステータスの更新】** をクリックします。

リポジトリオブジェクトは、以下に示すステータスのいずれかになります。

新規

オブジェクトが新しく、SIF API が生成、デプロイされていないことを示します。オブジェクトに対して ORS 固有の API を生成してデプロイすると、ステータスは最新に変更されます。

最新

SIF API の生成後、オブジェクトは変更されておらず、最新の状態になっていることを示します。

同期していない

SIF API の生成後、オブジェクトは変更され、同期していない状態になっていることを示します。SIF API を再生成して、ステータスを最新に変更します。

安全でない

オブジェクトが安全ではなく、そのオブジェクトに対して SIF API を生成できないことを示します。Hub コンソールのセキュアリソースツールに、Not secure ステータスのオブジェクトが非公開リソースとして表示されます。

削除済み

オブジェクトが削除され、そのオブジェクトに対して API を生成できないことを示します。Hub コンソールのモデルワークベンチのいずれかのツールを使用してオブジェクトを削除すると、オブジェクトのステータスは Deleted になります。ORS 固有の SIF API を生成してデプロイするときに、ステータスが Deleted のオブジェクトは削除されます。

アーカイブテーブル

CMX_DATA テーブルスペースに格納された C_REPAR_SIF_ORS_CONFIG テーブルに生成するすべての ORS 固有の SIF API をアーカイブすることができます。ORS 固有の SIF API のバージョン ID を使用して、アーカイブを識別します。

アーカイブテーブルのレコードには、文字ベースのレコードより大きい可能性のあるプロプデータが含まれ、時間の経過とともに非常に大きなサイズになることがあります。データベース管理者は、アーカイブテーブルを定期的にアーカイブまたはパージしてデータベースをクリーニングできます。

ORS 固有の SIF API の生成とデプロイ

Hub コンソールで SIF マネージャユーティリティを使用して、安全なリポジトリオブジェクトに対して ORS 固有の SIF API を生成およびデプロイします。ORS 固有の SIF API は、特定のリポジトリオブジェクトを選択して生成できます。

MDM Hub で ORS 固有の SIF API を生成してデプロイするには、アプリケーションサーバーがインストールされたコンピュータ上の Java コンパイラにアクセスする必要があります。ORS 固有の SIF API を生成、デプロイする前に、ORS のベースオブジェクトとパッケージを必ず設定してください。

1. Hub コンソールを起動して、ORS に接続します。
2. ユーティリティワークベンチを展開して、**[SIF マネージャ]** をクリックします。
SIF マネージャユーティリティが表示されます。
3. 書き込みロックを取得するには、**[書き込みロック]** メニューで **[ロックの取得]** をクリックします。
4. リポジトリオブジェクトのステータスを更新するには、**[SIF API マネージャ]** タブで **[オブジェクトステータスの更新]** をクリックします。
リポジトリオブジェクトのステータスが **[選択済みの非同期オブジェクト]** テーブル内で更新されます。
5. SIF API を生成およびデプロイするリポジトリオブジェクトを選択します。
6. **ORS 固有の SIF API の生成とデプロイ** をクリックします。
ORS 固有のクライアント JAR ファイルと WSDL ファイルが生成されます。

ORS 固有の SIF API の名前変更

Hub コンソールで SIF マネージャユーティリティを使用して、ORS 固有の SIF API を名前変更します。

1. SIF マネージャユーティリティの **[書き込みロック]** メニューで **[ロックの取得]** をクリックします。
2. **[SIF API マネージャ]** タブの **[論理名]** ボックスで、**[編集]** ボタンをクリックして論理名を編集します。
3. **[許可]** ボタンをクリックします。
論理名が保存され、その論理名に一致するように Java 名が更新されます。
4. **ORS 固有の SIF API の生成とデプロイ** をクリックします。
ORS 固有のクライアント JAR ファイルと WSDL ファイルが再生成されます。

ORS 固有のクライアント JAR ファイルのダウンロード

特定のリポジトリオブジェクト用の ORS 固有の SIF API を生成したら、SiperianClient クラスと SIF API 参照マニュアルを含んだクライアント JAR ファイルをダウンロードします。

1. SIF マネージャユーティリティの **[SIF API マネージャ]** タブで、**[クライアント JAR ファイルのダウンロード]** をクリックします。
[クライアント JAR ファイルを保存するディレクトリを選択] ダイアログボックスが表示されます。

2. クライアント JAR ファイルを保存するディレクトリを選択し、**【保存】** をクリックします。
<Java Name>Client_<Version ID>.jar ファイルがダウンロードされ、選択したディレクトリに保存されます。

ORS 固有のクライアント JAR ファイルと SIF SDK の併用

ORS 固有のクライアント JAR ファイルを SIF SDK とともに使用することができます。

1. 統合開発環境（IDE）を使用する場合に、Web サービスを構築するためのプロジェクトファイルがあるときには、ダウンロードしたクライアント JAR ファイルをビルドクラスパスに追加します。
2. 次のディレクトリから build.xml ファイルを開きます。
 - Windows の場合:<Resource Kit Installation Directory>\hub\resourcekit\sdk\sifsdk
 - UNIX の場合:<Resource Kit Installation Directory>/hub/resourcekit/sdk/sifsdk
3. ダウンロードしたクライアント JAR ファイルが build_war マクロに含まれるように、build.xml ファイルをカスタマイズします。
4. build.xml ファイルを保存して閉じます。

ORS 固有の SIF API の削除

Hub コンソールで SIF マネージャユーティリティを使用して、ORS 固有の SIF API を削除します。

1. SIF マネージャユーティリティの **【書き込みロック】** メニューで、**【ロックの取得】** をクリックします。
2. **【SIF API マネージャ】** タブで **【ORS 固有の API の削除】** をクリックします。
ORS 固有のクライアント JAR ファイルと WSDL が削除されます。

第 30 章

ORS 固有のメッセージスキーマ

この章では、以下の項目について説明します。

- [ORS 固有のメッセージスキーマの概要, 613 ページ](#)
- [JMS イベントスキーママネージャツールについて, 613 ページ](#)
- [JMS イベントスキーママネージャツールの起動, 614 ページ](#)
- [SIF マネージャツールの起動, 614 ページ](#)
- [ORS 固有のスキーマの生成およびデプロイ, 615 ページ](#)

ORS 固有のメッセージスキーマの概要

Hub コンソールの SIF マネージャツールを使用し、ORS 固有の JMS イベントメッセージスキーマを生成できます。メッセージスキーマを生成できるオブジェクトは、SIF マネージャツールの「非同期オブジェクト」セクションに表示されます。

ORS 固有の JMS イベントメッセージスキーマの生成パフォーマンスは、その生成とデプロイに MDM Hub が使用するオブジェクトの数によって異なります。

ORS 固有の JMS イベントメッセージスキーマは、URL を介してダウンロードまたはアクセスできる XSD ファイルとして使用できます。従来のイベント XML メッセージスキーマを使用する場合は、ORS 固有の JMS イベントメッセージスキーマを生成する必要はありません。SIF SDK を使用する場合は、Java Development Kit と Apache Jakarta Ant ビルドシステムをインストールする必要があります。

JMS イベントスキーママネージャツールについて

JMS イベントスキーママネージャでは、Hub が JMS メッセージの生成に使用するメッセージ構造を定義する XML スキーマを使用します。

この XML スキーマは、Informatica MDM Hub Resource Kit に含まれています（ORS 固有のスキーマは、URL からダウンロード可能なファイルとして提供されています）。

注: JMS イベントスキーマを生成するには、セキュアパッケージまたはリモートパッケージが少なくとも 1 つ定義されている必要があります。

重要: 同じスキーマ（CMX_ORIS など）を含むデータベースが 2 つある場合、設定を最初に保存した時点で、JMS イベントの論理名（スキーマ名と同じ）が重複することになります。そのため、SIF の API に合わせて、最初の論理名として、一意であるデータベースの表示名をスキーマ名の代わりに使用する必要があります。この論理名は、スキーマを生成する前に変更する必要があります。

また、各 ORS には、共通の XSD ファイル（siperian-mrm-events.xsd）の要素を使用する ORS に固有の XSD ファイルが含まれます。ORS 固有の XSD には、<ors-name>-siperian-mrm-event.xsd という形式の名前が付けられます。XSD では、スキーマ内のパッケージおよびリモートパッケージごとに 2 つのオブジェクトが定義されます。

オブジェクト名	説明
[packageName]Event	EventMetadata 型の要素と [packageName] を含む複合型。
[packageName]Record	パッケージとそのフィールドを表す複合型。SipMetadata 型の要素も含まれます。この複合型は、Informatica MDM Hub の Services Integration Framework (SIF) で定義されるパッケージのレコード構造に似ています。

注: 従来の XML イベントメッセージオブジェクトを使用する場合は、ORS 固有のメッセージオブジェクトを生成する必要はありません。

JMS イベントスキーママネージャツールの起動

JMS イベントスキーママネージャツールを起動する手順

1. Hub コンソールで、オペレーショナル参照ストア（オペレーショナル参照ストア）に接続します。
2. Informatica ユーティリティワークベンチを展開して、**[SIF マネージャ]** をクリックします。
3. **[JMS イベントスキーママネージャ]** タブをクリックします。

Hub コンソールに、JMS イベントスキーママネージャツールが表示されます。

JMS イベントスキーママネージャツールには以下の領域が表示されます。

領域	説明
JMS ORS 固有のイベントメッセージスキーマ	<p>ORS のイベントメッセージスキーマが表示されます。</p> <p>この機能を使用して、現在の ORS 用の ORS 固有の JMS イベントメッセージを生成してデプロイします。論理名を使用して、デプロイメントのコンポーネントに名前が付けられます。スキーマは URL を使用してダウンロードまたはアクセスできます。</p> <p>注: 従来の XML イベントメッセージオブジェクトを使用する場合は、ORS 固有のメッセージオブジェクトを生成する必要はありません。</p>
非同期オブジェクト	生成済みスキーマと非同期のスキーマのデータベースオブジェクトが表示されます。

SIF マネージャツールの起動

SIF マネージャツールは、Hub コンソールのユーティリティワークベンチで起動できます。

1. Hub コンソールを起動し、オペレーショナル参照ストア（ORS）に接続します。
2. ユーティリティワークベンチを展開して、**[SIF マネージャ]** をクリックします。

SIF マネージャツールが表示されます。

ORS 固有のスキーマの生成およびデプロイ

Java Software Development Kit (SDK) の tools.jar に、コンパイラが用意されています。

この操作を行うには、アプリケーションサーバーマシンで Java コンパイラにアクセスする必要があります。Java Software Development Kit (SDK) の tools.jar に、コンパイラが用意されています。Java ランタイム環境 (JRE) にはコンパイラが含まれていません。SDK を利用できない場合は、tools.jar ファイルをアプリケーションサーバーのクラスパスに追加する必要があります。

重要: 同じスキーマ (CMX_ORIS など) を含むデータベースが 2 つある場合、設定を最初に保存した時点で、JMS イベントの論理名 (スキーマ名と同じ) が重複することになります。そのため、SIF の API に合わせて、最初の論理名として、一意であるデータベースの表示名をスキーマ名の代わりに使用する必要があります。この論理名は、スキーマを生成する前に変更する必要があります。

以下の手順は、ORS のベースオブジェクト、パッケージ、およびマッピングが設定済みであることを前提としています。これらの設定を後から変更する場合は、ORS 固有のスキーマを再生成してください。

また、JMS イベントスキーマを生成するには、少なくとも 1 つのセキュアパッケージまたはリモートパッケージが必要です。

ORS 固有のスキーマを生成してデプロイする手順

1. JMS イベントスキーママネージャを起動します。
Hub コンソールに、JMS イベントスキーママネージャツールが表示されます。
2. イベントスキーマの [論理名] フィールドに値を入力します。
スキーマに変更を加えるには、書き込みロックを設定する必要があります。
3. **[ORS 固有のスキーマの生成とデプロイ]** をクリックします。

注: スキーマを生成するには、少なくとも 1 つのセキュアパッケージまたはリモートパッケージが設定されている必要があります。生成するセキュアオブジェクトがない場合は、Informatica MDM Hub でランタイムエラーメッセージが生成されます。

XSD ファイルのダウンロード

XSD ファイルは、XML ファイルの構造を定義したもので、XML ファイルの検証にも使用できます。

例えば、XML ファイルに XSD への参照が含まれている場合は、XML 検証ツールを使用して、XML 内のタグが XSD での定義と一致しているかを検証できます。

XSD ファイルをダウンロードする手順

1. JMS イベントスキーママネージャを起動します。
Hub コンソールに、JMS イベントスキーママネージャツールが表示されます。
2. 書き込みロックを取得します。
スキーマに変更を加えるには、書き込みロックを設定する必要があります。
3. **[XSD ファイルのダウンロード]** をクリックします。
または、スキーマ URL に指定されている URL を使用して XSD ファイルにアクセスできます。

非同期オブジェクトの検出

イベントスキーマを再生成してシステムの変更を反映させる必要があるかどうかを判別するには、[非同期オブジェクトの検出] を使用します。

JMS イベントスキーママネージャに、前回のスキーマ生成後に変更されたパッケージおよびリモートパッケージの一覧が表示されます。

注: 非同期オブジェクト検出機能によって、生成された API がスキーマ内のデータベースオブジェクトと比較されるため、非同期オブジェクトを検出するにはこの両方が存在していなければなりません。

非同期オブジェクトを検出する手順

1. JMS イベントスキーママネージャを起動します。
Hub コンソールに、JMS イベントスキーママネージャツールが表示されます。
2. 書き込みロックを取得します。
スキーマに変更を加えるには、書き込みロックを設定する必要があります。
3. **【非同期オブジェクトの検出】** をクリックします。
JMS イベントスキーママネージャの下部パネルに、すべての非同期オブジェクトが表示されます。

注: 非同期オブジェクトの影響を評価した後に、スキーマを再生成するかどうかを決定します（通常は、生成されたスキーマの特定のバージョンを操作するために、Hub と対話する外部コンポーネントが作成されます）。スキーマを再生成すると、これらの外部コンポーネントが機能しなくなることがあります。

JMS イベントスキーママネージャから非同期オブジェクトが返された場合は、[ORS 固有のスキーマの生成とデプロイ] をクリックしてイベントスキーマを再生成します。

非同期オブジェクトの自動検索

非同期オブジェクトを定期的に検索し、必要に応じてスキーマを再生成するように、Informatica MDM Hub を設定することができます。

この自動ポーリング機能は、指定したポーリング間隔（ミリ秒）を自動的に適用するデータ変更監視スレッド内で行われます。この間隔は、メッセージキューツールの [メッセージチェック間隔] で指定します。監視スレッドがアクティブになると、この自動サービスでは、まず非同期の間隔が経過しているかどうかを確認し、経過している場合は非同期チェックを実行します。その後、必要に応じてイベントスキーマを再生成します。

非同期オブジェクトを定期的に検索するように Hub を設定する手順

1. JMS イベントスキーママネージャで生成するスキーマの論理名を設定します。
注: この手順を行わないと、サーバーログに、スキーマの生成を設定するように求める警告が発行されます。
2. データ変更監視メッセージのキューステータスを有効にします。
3. ルートノード [メッセージキュー] を選択し、非同期チェック間隔（ミリ秒）を設定します。
非同期の自動ポーリング機能は、事実上メッセージチェック間隔に依存するため、非同期チェック間隔の値はメッセージチェック間隔の値と同じかそれよりも大きくする必要があります。
注: 非同期チェックを無効にするには、非同期チェック間隔を 0 に設定します。

第 31 章

登録済みのカスタムコードの表示

この章では、以下の項目について説明します。

- [概要, 617 ページ](#)
- [ユーザーオブジェクト, 617 ページ](#)
- [ユーザーオブジェクトレジストリツールの起動, 618 ページ](#)
- [ユーザー出口の表示, 618 ページ](#)
- [カスタム Java クレンジング関数の表示, 618 ページ](#)
- [カスタムボタン関数の表示, 619 ページ](#)

概要

この章では、ユーザーオブジェクトレジストリツールを使用して、登録済みのカスタムコードを表示する方法を説明します。

ユーザーオブジェクト

ユーザーオブジェクトは、Hub 機能を拡張するために Informatica MDM Hub に登録できるカスタム関数です。

ユーザーオブジェクトレジストリツールは、MDM Hub で使用および登録するために開発されたユーザーオブジェクトを追跡する読み取り専用のツールです。ユーザーオブジェクトレジストリにある次のユーザーオブジェクトにアクセスできます。

ユーザーイグジット

事前に定義された一連の固定のパラメータを含む、ユーザーがカスタマイズした、暗号化されない Java コード。バッチジョブ中に特定の時点で実行するユーザー出口。各ベースオブジェクトに対してユーザー出口を個別に設定できます。

カスタム Java クレンジング関数

標準のクレンジングライブラリを補完するためにカスタムロジックを使用する Java クレンジング関数。カスタム Java クレンジング関数は JAR ファイルです。Informatica MDM Hub によってデータベースにバイナリの large object 型として保存されます。

カスタムボタン関数

データマネージャ、マージマネージャ、および階層マネージャに追加のアイコンやロジックを提供するユーザーインターフェース関数。

ユーザーオブジェクトレジストリツールの起動

ユーザーオブジェクトレジストリツールを起動する手順

1. Hub コンソールでオペレーショナル参照ストア（ORS）に接続します。
2. Informatica ユーティリティワークベンチを展開して、[ユーザーオブジェクトレジストリ] をクリックします。

ユーザーオブジェクトレジストリツールが表示されます。

ユーザーオブジェクトレジストリツールに以下の領域が表示されます。

カラム	説明
登録済みユーザーオブジェクトのタイプ	選択した ORS に登録されているユーザーオブジェクトの階層ツリー。以下のカテゴリがあります。 <ul style="list-style-type: none">- ユーザーイグジット- カスタム Java クレンジング関数。- カスタムボタン関数。
ユーザーオブジェクトのプロパティ	選択されているユーザーオブジェクトのプロパティ。

ユーザー出口の表示

この節では、ユーザーオブジェクトレジストリツールでユーザー出口を表示する方法について説明します。

ユーザー出口について

ユーザー出口は Java コードで構成され、バッチもしくは SIF の API プロセスの特定の時点で実行し、MDM Hub の機能性を拡張します。

ユーザー出口は、カスタムの操作を Hub サーバーのプロセス（post-load、post-merge、post-match など）と統合するメカニズムを提供する Informatica MDM Hub のバックエンドプロセスで起動されます。

ユーザー出口の表示

ユーザーオブジェクトレジストリツールで Informatica MDM Hub のユーザー出口を表示する手順

1. ユーザーオブジェクトレジストリツールを起動します。
2. ユーザーオブジェクトのリストで、[ユーザー出口] を選択します。
ユーザーオブジェクトレジストリツールにユーザー出口が表示されます。

カスタム Java クレンジング関数の表示

この節では、ユーザーオブジェクトレジストリツールで登録されているカスタム Java クレンジング関数を表示する方法について説明します。

カスタム Java クレンジング関数について

Java ライブラリ（ユーザーライブラリではない）に追加されたカスタムクレンジング関数の詳細は、ユーザーオブジェクトレジストリに公開されます。

Informatica MDM Hub では、データをクレンジングするクレンジング関数を構築して実行することができます。クレンジング関数は、標準化や検証のためにレコード内のデータ値に適用される関数です。例えば、挨拶の文句に使用するデータの列がある場合、クレンジング関数を使用して“Doctor”という語をすべて“Dr.”に標準化することができます。クレンジング関数は継続的に適用することも、ステージングテーブルの列に単に出力値を割り当てすることもできます。

カスタム Java クレンジング関数の登録方法

クレンジング関数は、Hub コンソールのクレンジング関数ツールを使用して設定されます。

登録されているカスタム Java クレンジング関数の表示

ユーザーオブジェクトレジストリツールで登録されているカスタム Java クレンジング関数を表示する手順

1. ユーザーオブジェクトレジストリツールを起動します。
2. ユーザーオブジェクトのリストで、**[カスタム Java クレンジング関数]** を選択します。

ユーザーオブジェクトレジストリツールに、登録されているカスタム Java クレンジング関数が表示されます。

カスタムボタン関数の表示

この節では、ユーザーオブジェクトレジストリツールで登録されているカスタムボタン関数を表示する方法について説明します。

カスタムボタン関数について

Informatica MDM Hub の実装環境では、Hub コンソールのユーザーに対し、Informatica MDM Hub 実装を拡張するためのカスタムボタンを提供することができます。ユーザーはカスタムボタンを使用して、特定の外部サービス（データの取得や結果の計算など）の呼び出しや特殊な操作（ワークフローの起動など）の実行などのタスクを行うことができます。カスタムボタンは、Hub コンソールのマージマネージャ、データマネージャ、および階層マネージャの各ツールに追加できます。

サーバーおよびクライアントベースのカスタム関数は、ユーザーオブジェクトレジストリに表示されます。

カスタムボタン関数の登録方法

Informatica MDM Hub 実装の Hub コンソールにカスタムボタンを追加するには、以下の作業を実行します。

1. 要求メッセージと応答メッセージの形式やパラメータなど、呼び出す外部サービスの詳細を確認します。
2. カスタムボタンで実行するビジネスロジックを記述してパッケージ化します。
3. Hub コンソールのアプリケーションツールに表示されるようにパッケージをデプロイします。

登録されているカスタムボタン関数の表示

ユーザーオブジェクトレジストリツールで登録されているカスタムボタン関数を表示する手順

1. ユーザーオブジェクトレジストリツールを起動します。
2. **【カスタムボタン関数】** を選択します。

ユーザーオブジェクトレジストリツールに、登録されているカスタムボタン関数が表示されます。

付録 A

MDM Hub のプロパティ

この付録では、以下の項目について説明します。

- [MDM Hub のプロパティの概要, 621 ページ](#)
- [Hub サーバーのプロパティ, 621 ページ](#)
- [Hub サーバーのプロパティファイルのサンプル, 641 ページ](#)
- [プロセスサーバーのプロパティ, 644 ページ](#)
- [プロセスサーバーのプロパティファイルのサンプル, 652 ページ](#)
- [オペレーショナルリファレンスストアのプロパティ, 653 ページ](#)

MDM Hub のプロパティの概要

MDM Hub プロパティファイルには Hub サーバーおよび Process サーバーの構成設定が含まれます。

テキストエディタを使用して、プロパティファイルを表示または編集できます。Hub サーバーおよび Process サーバーを最初にインストールする際に、インストーラにより、プロパティファイルの一部のプロパティに値が設定されます。インストール後、プロパティを編集して、MDM Hub の動作を変更することができます。例えば、Hub サーバーのデータ暗号化を設定するには、`encryption.plugin.jar` プロパティのデータ暗号化 JAR のパスおよびファイル名を指定する必要があります。

一部のプロパティはオプションなので、MDM Hub プロパティファイルに手動で追加する必要があります。例えば、WebSphere でのクラスターリングを有効にするには、`cluster.flag` プロパティを Hub サーバープロパティに追加してから、その値を `true` に設定する必要があります。

プロパティファイルを編集した後、変更を有効にするためアプリケーションサーバーを再起動します。

Hub サーバーのプロパティ

`cmxserver.properties` ファイルの Hub サーバープロパティを設定します。

`cmxserver.properties` ファイルは次のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/resources

MDM Hub コンソールのシングルサインオンのプロパティ

次のプロパティを使用すると、MDM Hub コンソールのログインページでシングルサインオン（SSO）オプションを有効または無効にすることができます。

hubconsole.show.login.with.sso

このプロパティを有効にすると、MDM Hub コンソールのログインページで SSO オプションを使用できるようになります。デフォルトは false です。

ビジネスエンティティのダウンロードプロパティ

次のプロパティは、ビジネスエンティティのダウンロードにおけるデフォルトの制限を変更します。

pdf_num_of_children

オプション。手動で追加する必要があります。親ビジネスエンティティに対してエクスポートできる子の最大数を指定します。デフォルト値は 1000 です。

pdf-depth

オプション。手動で追加する必要があります。ビジネスエンティティに対してエクスポートできる子レコードの最大深度を指定します。デフォルトは 20 です。

タスクのエクスポートプロパティ

タスクのダウンロードにおけるデフォルトの制限を変更するには、次の Hub Server プロパティを設定します。
cmx.e360.taskdata.export.limit

Excel ファイルにエクスポートするタスクの数（最大 10,000 レコード）。デフォルトは 1000 です。

選択的フィルタリングのプロパティ

次のプロパティを使用すると、親ビジネスエンティティの子レコードまたは孫レコードのフィルタリングを有効または無効にすることができます。

cmx.e360.legacysearch.filterchildoutput

親ビジネスエンティティの子レコードまたは孫レコードにフィルタリングを適用するには、このプロパティを有効にします。デフォルトは false です。

Hub サーバー環境のプロパティ

次のプロパティは、Hub サーバーの場所と、アプリケーションサーバーおよびデータベースの接続詳細を設定します。

cmx.home

Hub サーバーのインストールディレクトリ。このプロパティは Hub サーバーのインストール中に設定されます。

cmx.appserver.hostname

MDM Hub のデプロイ先にする EJB クラスタの名前。次の形式でクラスタサーバーのホスト名を指定します。

`<host_name>.<domain>.com`

クラスタ環境での Hub サーバーのデプロイの詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

cmx.appserver.naming.protocol

アプリケーションサーバータイプの命名規則。Websphere の場合のデフォルト値は、iiop、JBoss 5 の場合は jnp、JBoss 7 の場合は remote および WebLogic の場合は t3 です。このプロパティは Hub サーバーのインストール中に設定されます。

cmx.appserver.rmi.port

アプリケーションサーバーのポート。デフォルト値は Websphere の場合は、2809、WebLogic の場合は 7001、JBoss 5 の場合は 1099 および JBoss 7 の場合は 4447 です。クラスタ環境での Hub サーバーのデプロイの詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

cmx.appserver.type

アプリケーションサーバーのタイプ。このプロパティは以下のいずれかの値になります。JBoss、WebSphere または WebLogic。このプロパティは Hub サーバーのインストール中に設定されます。

cmx.server.attachment.temp.ttl_minutes

ファイルが TEMP ストレージに作成されてから期限が切れるまでの時間（分）。ファイルの期限が切れな
いようにするには 0 に設定します。デフォルトは 60 です。

cmx.server.masterdatabase.type

MDM Hub マスターデータベースのタイプ。このプロパティは以下のいずれかの値になります。DB2、
Oracle または MSSQL。このプロパティは Hub サーバーのインストール中に設定されます。

cmx.server.masterdatabase.schemaname

IBM DB2 環境でのみ必要。名前が cmx_system 以外である場合、プロパティを使用して MDM Hub マスター
データベースの名前を指定します。デフォルトは cmx_system です。

cookie-secure

Data Director セッションの Cookie を保護します。安全な IDD セッションの Cookie を有効にするには、
cookie-secure フラグのコメントを解除し、値を true に設定します。デフォルトは false です。

変更を有効にするには、Hub コンソールを再起動する必要があります。

http-only

HTTP 用の Data Director セッションの Cookie のみを保護します。セッションの Cookie を有効にするに
は、http-only フラグのコメントを解除し、値を true に設定します。デフォルトは false です。

変更を有効にするために Hub コンソールを再起動します。

locale

Hub サーバーおよび Hub コンソールのロケールです。このプロパティの値は、最初に Hub サーバーをイ
ンストールするときに設定されます。

JBoss のアプリケーションサーバーのプロパティ

Hub サーバーが JBoss アプリケーションサーバーで実行されている場合は、次のプロパティが使用されます。

cmx.appserver.version

アプリケーションサーバーの JBoss のバージョン。このプロパティは以下のいずれかの値になります。5
または 7。このプロパティは Hub サーバーのインストール中に設定されます。

cmx.jboss7.management.port

JBoss ネイティブ管理ポート。JBoss の場合のデフォルトは、9990 です。このプロパティは Hub サーバ
ーのインストール中に設定されます。

cmx.jboss7.security.enabled

JBoss の EJB セキュリティを有効にします。JBoss EJB セキュリティを有効にする場合は、true に設定し
ます。JBoss EJB セキュリティを無効にする場合は、false に設定します。デフォルトは true です。
JBoss セキュリティの詳細については、『*Multidomain MDM のインストールガイド*』を参照してくださ
い。

cmx.server.ejb3

JBoss 7 の場合のみ。アプリケーションサーバー EJB3 のルックアップを有効にする場合は、true に設定
します。デフォルトは false です。このプロパティは Hub サーバーのインストール中に設定されます。

ejb-client-version

オプション。手動で追加する必要があります。EJB クライアントのバージョンを指定します。デフォルトの JBoss EJB クライアントを使用しない場合は、`ejb-client-version` を使用して、別の EJB クライアントを指定します。Hub コンソールの EJB クライアントの設定の詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

jboss.cluster

JBoss 7 の場合のみ。EJB サーバーが Hub サーバーに対してクラスタ化されているかどうかを指定します。EJB クラスタリングを有効にする場合は、`true` に設定します。クラスタ EJB サーバーがない場合は、`false` に設定します。デフォルトは `false` です。

WebSphere のアプリケーションサーバーのプロパティ

Hub サーバーが WebSphere アプリケーションサーバーで実行されている場合は、次のプロパティが使用されます。

cluster.flag

オプション。手動で追加する必要があります。WebSphere 環境の場合のみ。クラスタリングが有効かどうかを指定します。クラスタリングを有効にする場合は、`true` に設定します。クラスタリングを無効にする場合は、`false` に設定します。デフォルトは `false` です。

cmx.appserver.password

WebSphere 管理者のパスワード。このプロパティは、WebSphere 管理セキュリティが有効な場合に使用可能になります。

cmx.appserver.username

WebSphere 管理者のユーザー名。このプロパティは、WebSphere 管理セキュリティが有効な場合に使用可能になります。

cmx.appserver.soap.connector.port

WebSphere 環境の場合のみ。SOAP コネクタポート。WebSphere の場合のデフォルト値は 8880 です。クラスタ環境での Hub サーバーのデプロイの詳細については、WebSphere 用の『*Multidomain MDM のインストールガイド*』を参照してください。

cmx.websphere.security.enabled

WebSphere のセキュリティが有効かどうかを指定します。WebSphere 管理セキュリティを有効にする場合は、`true` または `yes` に設定します。デフォルトは `no` です。WebSphere 管理セキュリティの有効化の詳細については、『*Multidomain MDM のアップグレードガイド*』を参照してください。

cmx.websphere.security.sas.config.name

WebSphere 環境の場合のみ。`sas.client.props` ファイルのカスタム名。セキュアな EJB ルックアップを使用する環境の使用。

デフォルトは `sas.client.props` です。

cmx.websphere.security.sas.config.url

WebSphere 環境の場合のみ。`sas.client.props` ファイルの場所。セキュアな EJB ルックアップを使用する環境の使用。

デフォルトは `https://yourdomain.com:9443/cmx/filesx/Security/WebSphere/sas.client.props` です。

cmx.websphere.security.ssl.config.name

WebSphere 環境の場合のみ。`ssl.client.props` ファイルのカスタム名。セキュアな EJB ルックアップを使用する環境の場合。

デフォルトは `ssl.client.props` です。

cmx.websphere.security.ssl.config.url

WebSphere 環境の場合のみ。ssl.client.props ファイルの場所。セキュアな EJB ルックアップを使用する環境の使用。

デフォルトは `https://yourdomain.com:9443/cmx/filesx/Security/WebSphere/ssl.client.props` です。

was.jms.log.dir

オプション。手動で追加する必要があります。WebSphere 環境の場合のみ。WebSphere のリソースである、SIB のログディレクトリの場所を指定します。

was.jms.permanent_store.dir

オプション。手動で追加する必要があります。WebSphere 環境の場合のみ。WebSphere のリソースである、SIB の永続的な格納ディレクトリの場所を指定します。

was.jms.temp_store.dir

オプション。手動で追加する必要があります。WebSphere 環境の場合のみ。WebSphere のリソースである、SIB の一時的な格納ディレクトリの場所を指定します。

データベースのプロパティ

データベースには、次のプロパティを設定できます。

cmx.server.loadWorker.max.joins.optimization

オプション。手動で追加する必要があります。IBM DB2 の場合のみ。クエリで使用される結合の最大数を指定します。DB2 で、ロードジョブに 12 を超えるルックアップテーブルがあり、クエリの実行に時間がかかりすぎる場合は、20 に設定します。デフォルトは 30 です。

全般プロパティ

次のプロパティは、Hub サーバプロセスの動作を設定します。

cmx.outbound.bypass.multixref.insystem

オプション。手動で設定する必要があります。SIF API が複数の相互参照レコードでベースオブジェクトを更新する際に、Hub サーバーでのメッセージの作成をバイパスするには、true に設定します。デフォルトは false です。

cmx.server.datalayer.cleanse.execution

クレンジングジョブの実行場所を指定します。クレンジングジョブをアプリケーションサーバーで実行する場合、LOCAL に設定します。クレンジングジョブをデータベースサーバーで実行する場合、DATABASE に設定します。デフォルトは LOCAL です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cmx.server.datalayer.cleanse.working_files

クレンジングジョブの間に作成された一時ファイルを保存するかどうかを指定します。一時ファイルはトラブルシューティングや監査を目的として使用できます。一時作業ファイルを削除する場合は、FALSE に設定します。一時作業ファイルを保存する場合は、KEEP に設定します。デフォルトは、KEEP です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cmx.server.datalayer.cleanse.working_files.location

クレンジングジョブの作業ファイルの場所。MDM Hub はこのプロパティを Hub サーバーの初期化ルーチンの間に使用します。このプロパティは Hub サーバーのインストール中に設定されます。このプロパティは変更しないでください。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cmx.server.encryptionMethod

オプション。手動で追加する必要があります。SSL 暗号化を有効にするには、SSL に設定します。

cmx.server.load.nonsmos.sourcesystem.enddate.like.smos

状態管理オーバーライドシステム（SMOS）ではないシステムのリレーション終了日を SMOS と同じ日に設定します。true に設定すると、リレーションシップの終了日を SMOS と同じにすることができます。

cmx.server.met.max_send_size

リポジットリマネージャが送信できる最大ファイルサイズ（単位: バイト）。デフォルトは 9000000 です。

cmx.server.met.promotion_snapshot

オプション。手動で追加する必要があります。.meta ファイルの生成を有効にする場合は、true に設定します。.meta ファイルの生成を無効にする場合は、false に設定します。デフォルトは true です。

cmx.server.multi_data_set_schema

オプション。手動で追加する必要があります。メッセージトリガ XML メッセージを有効にして、親レコードと対応するすべての子レコードを含めるには、true に設定します。メッセージトリガ XML メッセージを無効にして、親レコードと対応するすべての子レコードを含めるには、false に設定します。デフォルトは false です。

cmx.server.poller.monitor.interval

全サーバーにポーリングする間隔（秒）。サーバーのポーリングを無効にするには、0 に設定します。デフォルトは 30 です。

cmx.server.put.autopopulate.missing.user.columns.bo.list

手動で追加する必要があります。ベースオブジェクトのカラムの [NULL 可能] プロパティを無効にした場合は、プロパティを設定します。

カラムの [NULL 可能] プロパティを無効にした場合は、Data Director または SIF PUT の操作中にレコードが更新されると、更新したフィールドの値のみを使用して相互参照レコードが作成されます。null 以外が指定されたフィールドを含む他のすべての相互参照レコードのフィールドには、null 値が使用されます。レコードの最善データ（BVT）の計算中、null 以外が指定されたフィールドで null 値を使用すると、エラーが発生します。

BVT の計算で null 以外が指定されたフィールドが考慮されるようにするには、プロパティ値を [NULL 可能] プロパティが無効にされたカラムを使用するベースオブジェクトの名前のカンマ区切りリストに設定します。プロパティが設定されると、MDM Hub は、関連付けられたベースオブジェクトの値を含む相互参照レコードの null 値を更新します。これにより、BVT 計算中、null 以外が指定されたフィールドで null 値は使用されません。

cmx.server.selective.bvt.enabled

オプション。手動で追加する必要があります。MDM Hub が SIF 要求に含まれるフィールドに対してのみ BVT 計算を適用するかどうかを指定します。MDM Hub が SIF 要求で指定されたフィールドのみを更新するように true に設定します。デフォルトは false です。

cmx.server.validateServerCertificate

オプション。手動で追加する必要があります。サーバー証明書の検証を無効にするには、false に設定します。デフォルトは true です。

com.informatica.mdm.message.queue.max.bytes.threshold

オプション。手動で追加する必要があります。メッセージキューに送信されるメッセージの上限をバイト単位で指定します。メッセージが指定したサイズを超えると、メッセージは送信されず、メッセージのステータスが [失敗] に設定されます。

com.informatica.mdm.sifapi.xref.edit.sys0.only

オプション。手動で追加する必要があります。管理ソースシステムのみを介して編集相互参照レコードを作成するには、true に設定します。すべてのソースシステムを介して編集相互参照レコードを作成するには、false に設定します。デフォルトは true です。

重要: Hub サーバーとプロセスサーバーの両方のプロパティを設定する必要があり、プロパティ値は両方のプロパティファイルで同じである必要があります。

<connection factory name>.qcf.password

オプション。手動で追加する必要があります。アプリケーションサーバーに設定されたパスワードを使用して JMS セキュリティを設定するように MDM Hub を設定します。

<connection factory name>.qcf.username

オプション。手動で追加する必要があります。アプリケーションサーバーに設定されたユーザー名を使用して JMS セキュリティを設定するように MDM Hub を設定します。メッセージキューの保護の詳細については、「[「JMS セキュリティの設定」 \(ページ 504\)](#)」を参照してください。

databaseld.password

オプション。手動で追加する必要があります。パスワードの暗号化ツールを使用するための暗号化されたパスワードを指定します。パスワードの暗号化ツールの使用の詳細については、『*Multidomain MDM のリソースキットガイド*』を参照してください。

databaseld.username

オプション。手動で追加する必要があります。パスワードの暗号化ツールを使用するユーザー名を指定します。パスワードの暗号化ツールの使用の詳細については、『*Multidomain MDM のリソースキットガイド*』を参照してください。

encryption.plugin.jar

データ暗号化 JAR ファイルのパスおよびファイル名。Hub サーバーのデータ暗号化の設定の詳細については、『[「手順 3. Hub サーバーのデータ暗号化の設定」 \(ページ 190\)](#)』を参照してください。

mq.data.change.monitor.thread.start

マルチノード環境で、個々のノードに対してメッセージキューのポーリングを行うかどうかを指定します。メッセージキューのポーリングを無効にするには、false に設定します。MDM Hub EAR ファイルがデプロイされているすべての Java 仮想マシン上で、デフォルトは true です。

searchQuery.buildBvtTemp.MaxRowCount

オプション。手動で追加する必要があります。BVT の計算時に GetOneHop API で使用するレコードの最大数を指定します。デフォルトは 5000 です。GetOneHop の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

sif.api.hm.flyover.max.record.count

オプション。手動で追加する必要があります。階層ビューリレーションテーブルに表示されるリレーションレコード数を制限するよう最大レコードカウントを設定します。デフォルトは 10000 です。

サーバーのプロパティを更新した後は、スキーマを検証してから Data Director アプリケーションを再デプロイする必要があります。階層ビューリレーションテーブルレコードの詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

sif.jvm.heap.size

オプション。手動で追加する必要があります。API に対し、デフォルトのヒープサイズを設定します（単位: メガバイト）。デフォルトは 256 です。

sif.search.result.query.temptableTimeToLive.seconds

オプション。手動で追加する必要があります。GetOneHop の場合。検索クエリ中に一時テーブル内のデータが存在する秒数を指定します。デフォルトは 30 です。GetOneHop の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

sip.hm.entity.font.size

オプション。手動で追加する必要があります。階層マネージャでのフォントサイズを設定します。値の範囲は、6 から 100 までです。階層マネージャのプロパティの設定の詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

sip.hm.entity.max.width

オプション。手動で追加する必要があります。階層マネージャでのエンティティボックスの最大幅を設定します。値の範囲は、20 から 5000 までです。階層マネージャのプロパティの設定の詳細については、『*Multidomain MDM のデータスチュワードガイド*』を参照してください。

sip.lookup.dropdown.limit

データマネージャツールとマージマネージャツールのメニューに表示されるエントリ数。このプロパティには、最小や最大の制限がありません。デフォルトは 100 です。

cmx.match.training.confidence.threshold

オプション。プロビジョニングツールで一致ルールセットを作成するために必要な最小の一致信頼度スコア。デフォルトは 85 です。

cmx.match.training.data.encoding

オプション。プロビジョニングツールでの一致トレーニングのエンコーディングを設定します。一致トレーニングに使用するエンコードを有効にする場合は、1 に設定します。デフォルトは 0 です。

cmx.server.match.server_encoding プロパティが使用するのと同じ値を使用していることを確認します。

バッチプロセスのプロパティ

次のプロパティはバッチプロセスに影響を与えます。

cmx.server.automerge.block_size

自動マージバッチプロセスのブロックサイズ。デフォルトは 250 です。

cmx.server.automerge.threads_per_job

自動マージのバッチプロセスのスレッド数。入力する推奨最大値は $n-1$ で、ここでの n は Hub サーバーで使用可能な CPU の数です。デフォルトは 1 です。

cmx.server.batch.acceptunmatchedrecordsasunique.block_size

一致しないレコードを一意的なバッチジョブとして受け入れるために、各ブロックで処理するレコードの最大数。デフォルトは 250 です。

cmx.server.batch.acceptunmatchedrecordsasunique.threads_per_job

一致しないレコードの受け入れを一意的なバッチジョブとして処理するために、MDM Hub が使用するスレッドの数。デフォルトは 20 です。

cmx.server.batch.batchunmerge.block_size

バッチマージ解除プロセスのブロックサイズ。デフォルトは 250 です。

cmx.server.batch.load.block_size

ロードジョブに対して各ブロックで処理するレコードの最大数。デフォルトは 250 です。

cmx.server.batch.recalculate.block_size

BVT の再計算とバッチジョブの再検証に対して各ブロックで処理するレコードの最大数。デフォルトは 250 です。

cmx.server.batch.threads_per_job

MDM Hub がロード、BVT の再計算、バッチジョブの再検証の処理（自動バッチジョブを除く）に使用するスレッドの数。また、バッチマージ解除プロセスで MDM Hub が使用するスレッドの数も指定します。

入力する推奨最大値は $n-1$ で、ここでの n は Hub サーバーで使用可能な CPU の数です。デフォルトは 10 です。

cmx.server.batch.max_concurrent_job_groups

処理する同時インポートジョブグループの最大数。デフォルトは 10 です。

cmx.server.jobControl.noOfDays

オプション。手動で追加する必要があります。Hub コンソールのバッチグループツールで、バッチグループジョブログ用に処理される履歴の日数。デフォルトは -1 で、ログにすべての履歴の詳細が含まれることを示します。

cmx.server.strp_clean.execution_mode

オプション。手動で追加する必要があります。一致キーテーブルのバックグラウンドクリーンアッププロセスの操作範囲を設定します。

操作範囲に次の値のいずれかを指定します。

- ALL。すべての登録されているオペレーショナル参照ストアのすべての一致キーテーブルから、invalid_ind=1 がある一致トークンを削除します。
- CONFIGURED_ORs。指定したオペレーショナル参照ストアのすべての一致キーテーブルから、invalid_ind=1 がある一致トークンを削除します。操作範囲を CONFIGURED_ORs に設定する場合は、cmx.server.strp_clean.ors プロパティを cmxserver.properties ファイルに追加します。
- CONFIGURED_STRP。特定のオペレーショナル参照ストアの特定のベースオブジェクトの一致キーテーブルから invalid_ind=1 がある一致トークンを削除します。操作範囲を CONFIGURED_STRP に設定する場合は、cmx.server.strp_clean.strp プロパティを cmxserver.properties ファイルに追加します。

cmx.server.strp_clean.ors

オプション。手動で追加する必要があります。無効な一致トークンを削除するために、バックグラウンドクリーンアッププロセスを実行する必要があるオペレーショナル参照ストアの名前を指定します。例えば、invalid_ind=1 がある一致トークンを cmx_ors1 および cmx_ors2 のすべての一致キーテーブルから削除するには、cmx.server.strp_clean.ors=cmx_ors1,cmx_ors2 を追加します。

cmx.server.strp_clean.strp

オプション。手動で追加する必要があります。一致キーテーブルをクリーンアップするために、バックグラウンドクリーンアッププロセスを実行する必要があるオペレーショナル参照ストアとベースオブジェクトの組み合わせを指定します。例えば、invalid_ind=1 がある一致トークンを、cmx_ors1 の BO1 および cmx_ors2 の BO2 の一致キーテーブルから削除するには、cmx.server.strp_clean.strp=cmx_ors1.C_BO1,cmx_ors2.C_BO2 を追加します。

cmx.server.strp_clean.delete_records_count

オプション。手動で追加する必要があります。一致キーテーブルからクリーンアップするレコード数を指定します。

cmx.server.strp_clean.retry_sec

オプション。手動で追加する必要があります。MDM Hub が、一致キーテーブルで無効な一致トークンがあるレコードを検索する時間（秒）を指定します。デフォルトは 60 です。

cmx.server.strp_clean.threads_count

オプション。手動で追加する必要があります。MDM Hub が、一致キーテーブルで無効な一致トークンがあるレコードを検索するときに使用するスレッド数を指定します。デフォルトは 20 です。

mq.data.change.threads

パブリッシュプロセス中に JMS メッセージを処理するために使用するスレッドの数。デフォルトは 1。

mq.data.change.batch.size

パブリッシュプロセスの各バッチで処理する JMS メッセージの数。デフォルトは 500。

mq.data.change.timeout

JMS メッセージを処理できる時間（秒）。デフォルトは 120。

セキュリティマネージャのプロパティ

次のプロパティは、セキュリティマネージャに影響を与えます。

cmx.server.clock.tick_interval

1 クロックティックとする時間（ミリ秒）。デフォルトは 60000 です。

cmx.server.provider.userprofile.cacheable

データをキャッシュできるかどうかを指定します。データのキャッシュによりパフォーマンスが改善されます。データのキャッシュを有効にする場合は、true に設定します。データのキャッシュを無効にする場合は、false に設定します。デフォルトは true です。

cmx.server.provider.userprofile.expiration

有効期限が来る前にキャッシュされたデータが存続している時間（ミリ秒）。デフォルトは 60000 です。

cmx.server.provider.userprofile.lifespan

有効期限が来る前にキャッシュされたデータが存続している時間（ミリ秒）。デフォルトは 60000 です。

cmx.server.sam.cache.resources.refresh_interval

セキュリティアクセスマネージャ（SAM）のリソースデータがデータベースから再ロードされる場合の、1 間隔のクロックティックの数。デフォルトは 5 です。更新間隔の変更については、「*Multidomain MDM のセキュリティガイド*」を参照してください。1 クロックティックとする時間（ミリ秒）を指定するには、cmx.server.clock.tick_interval プロパティを使用します。

cmx.server.sam.cache.user_profile.refresh_interval

ユーザープロファイルに対する SAM のリソースデータがデータベースから再ロードされる場合の、1 間隔のクロックティックの数。デフォルトは 30 です。1 クロックティックとする時間（ミリ秒）を指定するには、cmx.server.clock.tick_interval プロパティを使用します。

Oracle データベースのユーザーイグジット

Oracle データベースでは次のプロパティを使用できます。このプロパティを使用するには、プロパティを cmxserver.properties ファイルと cmxcleanse.properties ファイルの両方に追加する必要があります。

cmx.server.dbuserexit.load.PostLoadUserExit

オプション。MDM Hub がロードプロセス後にデータベース postload ユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。PL/SQL ユーザーイグジットについては、使用環境に対応した『*Multidomain MDM のアップグレードガイド*』を参照してください。

cmx.server.dbuserexit.put.PostLoadUserExit

オプション。MDM Hub が配置要求の実行後に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PostMergeUserExit

オプション。MDM Hub がマージ要求または自動マージバッチの実行後に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PreUnmergeUserExit

オプション。MDM Hub がマージ解除要求またはマージ解除バッチの実行前に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PostUnmergeUserExit

オプション。MDM Hub がマージ解除要求またはマージ解除バッチの実行後に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PreUserMergeAssignment

オプション。MDM Hub がレビューするマージ解除レコードの割り当て前にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.AssignTask

オプション。MDM Hub がタスクをユーザーに割り当てる前に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.GetAssignableUserForTask

オプション。MDM Hub がタスクをユーザーに割り当てる前に、データベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

Data Director の全般プロパティ

次のプロパティは、Data Director の動作に影響を与えます。

注: 次のサーバーのプロパティを更新した後、スキーマを検証してから IDD アプリケーションを再デプロイする必要があります。

com.siperian.dsapp.mde.common.idd2coccs.Many2ManyChild.name.version

手動で追加する必要があります。MDM 管理者がビジネスエンティティスキーマを生成すると、一部の子レベルのサブジェクト領域名の大文字が小文字に変更されます。大文字と小文字を維持するには、プロパティを 10.2 に設定します。

cmx.server.sa2be.sort

オプション。手動で追加する必要があります。サブジェクト領域のビジネスエンティティ定義を生成する際に、ビジネスエンティティのサブジェクト領域レイアウトのフィールドの順序を保持するかどうかを指定します。元のレイアウトからのソート順を保持する場合は、true に設定します。デフォルトは false です。

case.insensitive.search

true に設定すると、ベースオブジェクトの個別のカラムの大文字と小文字を区別しない属性を有効にできます。これにより、Data Director で大文字と小文字を区別しないクエリ検索が有効になります。この設定が有効になっている各カラムには、新しいインデックスが作成されます。インデックス管理にはパフォーマンスの低下が伴うため、このプロパティは注意して使用してください。デフォルトは false です。

cmx.bdd.redirect_to_login_after_logout

オプション。手動で追加する必要があります。Data Director での Google SSO 認証の場合のみ。ログアウトするとログイン画面に戻るよう Data Director を設定するには、true に設定します。ログアウトするとデフォルトのログアウト画面にリダイレクトするよう Data Director を設定するには、false に設定します。デフォルトは false です。

cmx.bdd.server.traffic.compression_enabled

Data Director サーバーのトラフィック圧縮が有効かどうかを指定します。トラフィックの圧縮によりパフォーマンスが改善されます。圧縮を有効にする場合は、true に設定します。圧縮を無効にする場合は、false に設定します。デフォルトは true です。

cmx.dataview.enabled

MDM 管理者がサブジェクト領域モデルを実装した場合、IDD ユーザーは【データ】タブを使用してレコードを検索、編集、および管理します。このオプションは、【データ】タブと関連要素が IDD アプリケーションに表示されるかどうかを指定します。

新規インストールでは、デフォルトは false です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは true です。

cmx.dataview.enabled=true の場合、次のユーザーインターフェース要素が IDD アプリケーションに表示されます。

- 【新規】タブ（【新規】ウィンドウをサブジェクト領域を含めて開く）
- 【データ】タブ（次の一時的インターフェースを含む）：
 - 【データ】ワークスペースタブ（レコードを編集および管理するためのサブジェクト領域レコードビューを含む）
 - 検索タブ（検索クエリおよび検索クエリ結果を含む）
 - タスクタブ（タスクを管理）
- 他のビューのメニューから【データ】ビューへのリンク
- カスタムタブ（構成されている場合）

プロパティ cmx.dataview.enabled および cmx.e360.view.enabled が true に設定されており、サブジェクト領域を持つ Data Director に関連して【相互参照】ビュー、【履歴】ビュー、および【一致レコード】ビューを有効化する場合は、プロパティ cmx.e360.match_xref.view.enabled を false に設定します。

MDM 管理者がエンティティ 360 フレームワークを実装した場合、Data Director ユーザーは【検索】ボックスを使用して、マスターデータを編集および管理するためのレコードやエンティティタブを探します。その場合、類似する機能によってユーザーが混乱しないように、【データ】タブと関連要素を非表示にできます。例えば、cmx.e360.view.enabled=true と設定した場合は、cmx.dataview.enabled=false と設定します。

cmx.bdd.enable_url_authentication

オプション。手動で追加する必要があります。Data Director の URL の認証を有効にします。認証が有効な場合、ユーザーがログインするときに、そのユーザー名とパスワードが Data Director の URL に渡されます。認証を有効にするには、true に設定します。認証を無効にするには、false に設定します。デフォルトは false です。

cmx.bdd.password_blowfish_encrypted

オプション。手動で追加する必要があります。Data Director の URL で認証が有効になっている場合に、ユーザーパスワードに対して Blowfish 暗号化が有効になります。有効にすると、パスワードは Data Director の URL で公開されません。暗号化を有効にするには、true に設定します。暗号化を無効にするには、false に設定します。デフォルトは false です。

cmx.display.deployed.invalid.met.app

オペレーショナルリファレンスストアのメタデータが有効でない場合、Data Director ではデプロイされたアプリケーションのリストが表示されません。別の有効なオペレーショナル参照ストアを使用するアプリケーションも使用できなくなります。デプロイされたアプリケーションのリストを表示するには、このプロパティを追加して、true に設定します。

cmx.e360.view.enabled

MDM 管理者がエンティティ 360 フレームワークを実装した場合、IDD ユーザーは **【検索】** ボックスを使用して、レコードを編集および管理するためのレコードやエンティティタブを探します。新規インストールでは、デフォルトは true です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは false です。

cmx.e360.view.enabled=true の場合、次のユーザーインターフェース要素が Data Director アプリケーションに表示されます。

- **【新規】** タブ（**【新規】** ウィンドウがビジネスエンティティを含めて開かれる）。
- **【タスクマネージャ】** タブ（タスクを管理する）。
- **【検索】** タブ（検索結果を含む）。
- エンティティタブ（ビジネスエンティティレコードを編集および管理）。エンティティタブが表示されるのは、ビジネスエンティティレコードを新規追加した場合か、ビジネスエンティティレコードを検索結果から開いた場合です。タブのラベルは動的で、ワークスペースを開くアクションに基づいて決まります。
- 他のビューのメニューからの **【ビジネスエンティティ】** ビューへのリンク
- カスタムタブ（構成されている場合）

cmx.e360.submit_all

オプション。手動で追加する必要があります。true に設定すると、Data Director によって未編集のフィールドを含むすべてのフィールドがサーバーに送信され、ビジネスエンティティレコードのルートフィールドまたは子フィールドをユーザーが編集する際に検証および保存操作が実行されます。デフォルトは false です。

例えば、あるレコードにルートフィールドとして個人のビジネスエンティティ、子フィールドとして住所と電子メールアドレスが含まれているとします。住所フィールドを編集すると、個人、住所、電子メールアドレスの子フィールドがサーバーに送信されて、レコードが検証および保存されます。

cmx.dataview.taskmanager.enabled

オプション。cmx.e360.view.enabled プロパティが false に設定されている場合にのみ適用できます。サブジェクト領域を使用する Data Director アプリケーションでタスクマネージャが表示されるかどうかを示します。true に設定するとタスクマネージャが表示されます。デフォルトは false です。

cmx.dataview.taskmanager.enabled が false に設定されており、Data Director アプリケーションのプロビジョニングツールでホームページを作成しない場合、アプリケーションには従来の開始ページが表示されます。

cmx.e360.match_xref.view.enabled

ビジネスエンティティを含む Data Director に対して **【相互参照】** ビューおよび **【一致レコード】** ビューを有効化するかどうかを指定します。ビューを有効化するには、true に設定します。デフォルトは true です。

プロパティ cmx.dataview.enabled および cmx.e360.view.enabled が true に設定されており、サブジェクト領域を持つ Data Director に関連して **【相互参照】** ビュー、**【履歴】** ビュー、および **【一致レコード】** ビューを有効化する場合は、プロパティ cmx.e360.match_xref.view.enabled を false に設定します。

cmx.server.override_orstitle

オプション。手動で追加する必要があります。Data Director へのログイン時、現在のタスクに優先デフォルトタイトルを指定します。cmx.server.override_orstitle プロパティを cmxserver.properties ファイル内の優先タイトルに設定します。

例えば、プロパティを「すべてのサブジェクト領域」に設定した場合、画面上のタイトルは「**すべてのサブジェクト領域のタスク**」として表示されます。

cmx.server.be-import.task-limit

タスク承認ワークフローをトリガするためにユーザーがインポートできるレコードの最大数を指定します。cmx.server.be-import.task-limit 値を好ましい最大数に設定します。例えば、ユーザーが最大 10000 レコードをインポートし、タスク承認ワークフローがトリガされるようにするには、cmx.server.be-import.task-limit=10000 を設定します。ユーザーが 10000 を超えるレコードをインポートしようすると、タスク承認ワークフローはトリガされず、エラーが表示されます。

cmx.server.find-replace.record-limit

ユーザーが一括操作で置換できるレコードの最大数を指定します。cmx.server.find-replace.record-limit プロパティを好ましい最大数に設定します。例えば、プロパティを cmx.server.find-replace.record-limit=10000 に設定した場合、ユーザーは最大 10000 レコードを検索および置換することができます。

cmx.server.find-replace.task-limit

タスク承認ワークフローをトリガする、置換されたレコードの最大数を指定します。cmx.server.find-replace.task-limit プロパティを好ましい最大数に設定します。例えば、プロパティを cmx.server.find-replace.task-limit=500 に設定した場合、ユーザーが最大 500 レコードを置換すると、タスク承認ワークフローがトリガされます。ユーザーが 500 を超えるレコードを置換しようすると、エラーが表示されます。

cmx.server.find-replace.entity-record-limit

ユーザーが「スマート検索」画面から「検索と置換」画面にコピーできるレコードの最大数を指定します。cmx.server.find-replace.entity-record-limit プロパティを好ましい最大数に設定します。例えば、プロパティを cmx.server.find-replace.entity-record-limit=1000 に設定した場合、ユーザーは「スマート検索」画面から「検索と置換」画面に最大 1000 件のレコードをコピーできます。ユーザーが 1000 を超えるレコードをコピーしようすると、エラーが表示されます。

cmx.file_import.job_group.ttl

MDM Hub がファイルインポートジョブグループを保存してから削除するまでの最大時間を指定します。デフォルトは 180 日です。値の後にサフィックスを追加する必要があります。サフィックスのオプションは、day、hour、min または sec です。例えば、プロパティを cmx.file_import.job_group.ttl=180day に設定した場合、インポートジョブグループは MDM Hub に 180 日間保存されます。

cmx.file_import.job_group_control.ttl

ファイルインポートジョブグループ管理ログを MDM Hub に保存してから削除するまでの最大時間を指定します。デフォルトは 30 日です。値の後にサフィックスを追加する必要があります。サフィックスのオプションは、day、hour、min または sec です。例えば、プロパティを cmx.file_import.job_group_control.ttl=30day に設定した場合、インポートジョブグループ管理ログは MDM Hub に 30 日間保存されます。

cmx.file_import.mapping.temp.ttl

ファイルインポートマッピングを MDM Hub の一時ストレージに保存してから削除するまでの最大時間を指定します。デフォルトは 20 分です。値の後にサフィックスを追加する必要があります。サフィックスのオプションは、day、hour、min または sec です。例えば、プロパティを cmx.file_import.mapping.temp.ttl=20min に設定した場合、インポートジョブマッピングは MDM Hub に 20 分間保存されます。

cmx.file_import.mapping.system.ttl

ファイルインポートマッピングを MDM Hub の永続ストレージに保存してから削除するまでの最大時間を指定します。デフォルトは 20 日です。値の後にサフィックスを追加する必要があります。サフィックスのオプションは、day、hour、min または sec です。例えば、プロパティを
cmx.file_import.mapping.system.ttl=20day に設定した場合、インポートジョブマッピングは MDM Hub に 20 日間保存されます。

cmx.file.allowed_file_extensions

Data Director アプリケーションでレコードやタスクに添付することのできるファイルの拡張子を一覧表示します。デフォルトでは、.pdf ファイルおよび.jpg ファイルを添付できます。拡張子を複数指定する場合は、各値をカンマで区切ります。

例: cmx.file.allowed_file_extensions=pdf,jpg,png,txt,zip,exe.

cmx.file.max_file_size_mb

Data Director アプリケーションで添付できるファイルのサイズ制限を指定します。

注: サブジェクト領域データモデルを使用する Data Director アプリケーションには 20 MB の静的サイズ制限があります。20 MB より大きいサイズ制限を指定しても、サブジェクト領域データモデルを使用する Data Director アプリケーションでは 20 MB の静的サイズ制限が維持されます。ビジネスエンティティデータモデルを使用する Data Director アプリケーションでは、cmx.file.max_file_size_mb プロパティで定義されているサイズ制限に維持されます。

Data Director の検索プロパティ

次のプロパティは、Data Director での検索の動作に影響を与えます。

ex.max.conn.per.host

ホストに接続する Elasticsearch ノードの最大数を設定します。ホスト上の Elasticsearch クラスターノードの数に設定します。

ex.max.threads

Apache Elasticsearch クラスターでノードごとに非同期ノンブロッキングレシーバを使用するスレッドの最大数を設定します。デフォルトは 1 です。

この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

es.index.refresh.interval

「スマート検索データの初期インデックス処理」バッチジョブが実行された後に Elasticsearch がデータへの変更をコミットする間隔（秒）を設定します。そのデータを検索できるようになるのは、この時間間隔の後です。デフォルトは 30 です。

このプロパティは、初回のインデックス処理でインデックス処理量が大きくなった場合に影響します。この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

cmx.task.search.records.return

ユーザーがビジネスエンティティを使用して Data Director のタスクマネージャでタスクを検索するときの Elasticsearch ページネーションを制御します。デフォルトは 1000 です。

この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

cmx.server.security.sql.injection.check

オプション。手動で追加する必要があります。SQL インジェクションチェック機能を設定します。この機能を無効にするには、cmxserver.properties ファイルで値を false に設定してから、サーバーを再起動します。デフォルトは true です。

cmx.server.batch.smartsearch.initial.block_size

スマート検索データの初期インデックス処理のバッチジョブが各ブロックで処理できる最大レコード数。デフォルトは 250 です。大きなデータセットをインデックス処理する場合は、レコード数を増やします。推奨最大値は 1000 です。

cmx.server.match.max_time_searcher

オプション。手動で追加する必要があります。1 つの検索が実行できる最長継続時間を指定します。デフォルトは 99999999 秒です。

cmx.server.remove_duplicates_in_search_query_results

重複レコードを検索クエリの結果に表示させるかどうかを指定します。検索クエリの結果に重複レコードを表示させる場合は、true に設定します。検索クエリの結果で重複レコードを非表示にする場合は、false に設定します。デフォルトは false です。

cmx.server.enrichcopager.thread_pool

プロパティを手動で追加します。ReadCO 操作を同時に実行できるように、スレッドプールから EnrichCoPager プロパティが使用するスレッドの数を設定します。デフォルトは 30 です。スレッドの数を 1 に設定すると、プロパティは無効になります。

cmx.server.enrichcopager.min_rec_for_multithreading

EnrichCoPager プロパティがマルチスレッドを使用するまでに返すレコードの最小数を設定します。デフォルトは 2 です。

cmx.ss.enabled

検索を有効にするかどうかを示します。新規インストールでは、デフォルトは true です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは false です。

search.provisioning.numshards

オプション。ご使用の Elasticsearch 環境で作成するシャードの数。値はシャードの最大数およびノードの合計数により異なります。例えば、シャードの最大数が 1 で、ノードの数が 3 の場合は、3 つのシャードを作成することができます。デフォルトは Hub サーバーの合計数です。

search.provisioning.numreplicas

オプション。別々のノードで作成する Elasticsearch エンジンドキュメントの部数。異なるノードのシャードにドキュメントの複数のコピーを作成するには、レプリケーション要素を使用します。1 つ以上のノードが予期せずシャットダウンした場合の高可用性を実現するには、ドキュメントの複数のコピーが必要です。例えば、レプリケーション要素が 2 の場合、2 つのノードにあるドキュメントの 2 つのコピーを取得します。Elasticsearch のデフォルトは 0 です。

sif.search.result.drop.batch.interval.milliseconds

オプション。手動で追加する必要があります。検索結果クレンジングの各バッチが処理された後に、SearchResultManager デーモンが一時停止する間隔（ミリ秒）を指定します。デフォルトは 0 です。

sif.search.result.drop.batch.record.count

オプション。手動で追加する必要があります。SearchResultManager デーモンが一度にクリーンアップするキャッシュされた検索の数を指定します。デフォルトは 200 です。

sif.search.result.query.timeToLive.seconds

オプション。手動で追加する必要があります。未使用のクエリがキャッシュされたままにしている秒数を指定します。デフォルトは 900 です。

sif.search.result.refresh.interval.seconds

オプション。手動で追加する必要があります。キャッシュされた検索に対するクリーンアッププロセスの実行後に、SearchResultManager デーモンが一時停止する間隔（秒）を指定します。デフォルトは1です。SIF 検索 API の設定の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

ssl.keyStore

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルの絶対パスおよびファイル名。

ssl.keyStore.password

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルのプレーンテキストパスワード。

ssl.trustStore

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルの絶対パスおよびファイル名。

ssl.trustStore.password

アプリケーションサーバーの HTTPS ポートを使用して Hub サーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルのプレーンテキストパスワード。

階層 REST API プロパティ

次のプロパティは、ビジネスエンティティサービスの階層 REST API に影響します。

cmx.server.hierarchy.max-search-depth

階層 REST API を使用して階層パスを検索するときに検索される最大深度。デフォルト値は 100 です。

cmx.server.hierarchy.max-search-width

階層 REST API を使用してエクスポートするときに含める検索される階層の最大幅。デフォルト値は 1000000 です。

com.informatica.mdm.bulk.relationship.changes.limit

一括タスク管理 REST API を使用するときの要求の変更の最大数。デフォルト値は 1000 です。

Data Director のタスク、ワークフロー、レポートのプロパティ

次のプロパティは、タスク、レビュープロセスワークフロー、およびレポートに影響します。

activevos.jndi

オプション。手動で追加する必要があります。レポートサービスを ActiveVOS に接続するための JNDI ルックアップ文字列を指定します。ActiveVOS EAR ファイルを編集して JNDI ルックアップ文字列をカスタマイズした場合は、このプロパティを使用してカスタム JNDI ルックアップ文字列を指定します。デフォルトの JNDI ルックアップ文字列は java:/jdbc/ActiveVOS です。

activevos.merge.workflow.service.name

Informatica ActiveVOS に対する MDM サービス呼び出しの名前を指定する必要があります。デフォルトでは、このプロパティは定義されていません。このプロパティが定義されていない場合、自動マージタスクは作成されません。

activevos.workflow.startup.timeout.seconds

Informatica ActiveVOS がタスクを作成してタスク ID を返すまで待機する時間（秒）。デフォルトは 10 です。

cmx.e360.BPMProcess.view.enabled

オプション。手動で追加する必要があります。ActiveVOS abAdmin ロールが割り当てられているユーザーに対しタスクマネージャのタスクに関連付けられているワークフローの図を表示するかどうかを示します。true に設定するとワークフローの図が表示されます。デフォルトは false です。

cmx.e360.BPMProcess.view.autologout.seconds

オプション。手動で追加する必要があります。タスクマネージャのワークフローの図にアクセスした際に ActiveVOS セッションがアクティブになっている秒数。指定した時間が過ぎるとセッションは終了します。デフォルトは 30 です。

cmx.server.task.grouppotentialmergebyruleid

オプション。手動で追加する必要があります。複数の一致が生成される手動一致タスクで同じ ROWID を持つ複数のタスクエントリを作成するかどうかを指定します。一致エントリごとにタスクを 1 つ作成するには、false に設定します。デフォルトは true です。

sip.task.assignment.interval

自動タスク割り当ての時間間隔（分）。自動タスク割り当てを無効にするには、0 に設定します。デフォルトは 0 です。

sip.task.assignment.start.delay

MDM Hub の初期化後、自動タスク割り当て開始までの待機時間（分）。遅延を設定しないと、ユーザーがタスクを作成する際にエラーが発生する場合があります。デフォルトは 10 分です。

sip.task.digest.interval

タスク通知の間隔（単位: 時間）タスク通知を無効にするには、0 に設定します。デフォルトは 0 です。

sip.task.maximum.assignment

自動タスク割り当てが有効なときに各ユーザーに自動的に割り当てられるタスクの数。デフォルトは 25 です。

task.creation.batch.size

オプション。手動で追加する必要があります。自動タスク割り当てプロセスの各反復の各一致テーブルの処理レコードの最大数を設定します。デフォルトは 1000 です。

マージタスクのプロパティの設定の詳細については、『*Multidomain MDM Business Process Manager Adapter SDK の実装ガイド*』を参照してください。

task.creation.maximum

オプション。手動で追加する必要があります。各照合テーブルで MDM Hub が作成するタスクの最大数を設定します。デフォルトは 50 です。タスク数がこの値を超えると、一致テーブルに関連付けられているタスク数が閉じられるまで、一致テーブルのレコードにマージタスクを作成できなくなります。

Siperian ワークフローエンジンのメールサーバーのプロパティ

Siperian ワークフローエンジンを使用する場合、次のプロパティはタスク通知に使用されるメールサーバーの動作に影響を与えます。

mail.smtp.auth

指定したメールサーバーで送信メッセージの認証が必要かどうかを決定します。MDM Hub メールサーバーを使用する場合は、mail.smtp.auth を true に設定します。デフォルトは false です。

タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

mail.smtp.host

タスク通知メールのためのメールサーバーの名前。サーバーのプロパティを更新した後は、スキーマを検証してから Data Director アプリケーションを再デプロイする必要があります。タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

mail.smtp.password

指定した mail.smtp.user のパスワード。mail.smtp.auth が true の場合、mail.smtp.password に値を指定する必要があります。タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

mail.smtp.port

メールサーバーのポート番号。デフォルトは 25 です。タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

mail.smtp.sender

タスク通知メールの送信者の電子メールアドレスを指定します。タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

mail.smtp.user

送信メールサーバーのユーザー名。mail.smtp.auth が true の場合、mail.smtp.user に値を指定する必要があります。タスク通知メールの設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

パスワードのハッシュ化およびカスタムハッシュのプロパティ

次のプロパティは、パスワードのハッシュ化およびカスタムハッシュアルゴリズムに影響します。

password.security.hash.algo

MDM Hub でパスワードを暗号化するために使用されるハッシュアルゴリズム (ALGO_NAME) を設定します。このプロパティは Hub サーバーのインストール中に設定されます。SHA3 ハッシュアルゴリズムを使用するには、SHA3 に設定します。カスタムハッシュアルゴリズムの場合は、他の名前に設定します (特殊文字や空白は使用できません)。

ハッシュアルゴリズムの設定の詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

password.security.hash.algo.<ALGO_NAME>.class

password.security.hash.algo プロパティで指定されたハッシュアルゴリズムの基本の実装クラスを含みます。このプロパティは Hub サーバーのインストール中に設定されます。

password.security.hash.algo.property.<param-name>

オプション。設定されたハッシュアルゴリズムのカスタムプロパティを指定します。デフォルトでは、SHA3 ハッシュアルゴリズムのサイズプロパティを指定します。次の値のいずれかに設定します: 224、256、384、または 512。デフォルトは 512。

com.informatica.mdm.security.certificate.provider.class

MDM Hub のデフォルトの証明書プロバイダでは、com.siperian.sam.security.certificate.PKIUtilDefaultImpl に設定します。このプロパティは Hub サーバーのインストール中に設定されます。

informatica.security.unique.id

パスワードのハッシュ化に使用されるカスタムハッシュキー。最大 32 の 16 進文字のシーケンス (区切り文字は含まない) を含むハッシュキーを使用することをお勧めします。カスタムハッシュキーの使用の詳細については、『*Multidomain MDM のセキュリティガイド*』を参照してください。

重要: カスタマハッシュキーの機密性を保護してください。カスタマハッシュキーの値が失われた場合、すべてのパスワードをリセットする必要があります。

セキュリティ設定ユーティリティのプロパティ

セキュリティ設定ユーティリティを使用するには、次のプロパティを設定します。

mdm.mail.server.host

MDM 管理者が使用する電子メールクライアントの SMTP サーバーホストに設定します。例えば、smtp.gmail.com に設定します。パスワードをリセットした場合、ユーザーアカウントの関連付けられた電子メールアドレスに、セキュリティ設定ユーティリティから新しい一時パスワードが送信されます。

mdm.mail.server.port

MDM 管理者が使用する電子メールクライアントで使用するポートに設定します。

mdm.mail.server.user

MDM 管理者の電子メールアドレスに設定します。例えば、MDM_Hub_admin@gmail.com に設定します。

mdm.mail.server.password

MDM 管理者の電子メールアドレスのパスワードを入力します。

mdm.mail.server.smtpauth

SMTP 認証を有効にするには、true に設定します。Gmail SMTP サーバーに接続する場合は必須です。

mdm.mail.server.ttls

TLS 認証を有効にするには、true に設定します。Gmail SMTP サーバーに接続する場合は必須です。

ビジネスエンティティリソースプロパティ

Entity 360 UI で設定された特権に基づいてビジネスエンティティサービスリソースを使用するには、次のプロパティを設定します。

<ORS_ID>.resourceMapping

オプション。ビジネスエンティティサービスリソースを有効または無効にします。Entity 360 UI で、次のビジネスエンティティサービスリソースに設定された特権を確認できます。特権に基づいてビジネスエンティティサービスリソースを有効にするには、<ORS_ID>.resourceMapping プロパティの値を true に設定し、MDM Hub Server を再起動します。デフォルトは false です。

cmxserver.properties ファイルにリストされているオペレーショナル参照ストアに複数の値を指定できます。例:

```
orcl-SAMPLE_ORS.resourceMapping=true
orcl-TSR_HUB.resourceMapping=false
```

<ORS_ID>.resourceMapping プロパティを有効にすると、Entity 360 UI での [アドホック照合]、[ファイルのインポート]、[マイジョブ]、[検索と置換]、[タスクマネージャ]、[クエリ結果のエクスポート]、[ビジネスエンティティのコピー]、[PDF としてエクスポート]、および [タスクのエクスポート] のワークスペース要素の表示が実行権限に応じて変化します。

<ORS_ID>.resourceMapping プロパティを有効にすると、Entity 360 UI でのクエリ管理要素の表示は、Security Access Manager (SAM) 内の次のビジネスエンティティサービスリソース特権によって変わります。

作成

作成権限がある場合、[+] アイコンと [名前を付けて保存] オプションを使用して、新しいクエリを作成できます。

更新

更新権限がある場合、[クエリの編集] アイコンから既存のクエリを変更して保存できます。

削除

削除権限がある場合、既存のクエリを削除できます。

実行

実行権限がある場合、新規または既存のクエリを実行できます。

注: Entity 360 UI で [検索と置換]、[ファイルのインポート]、[アドホック照合]、[ビジネスエンティティのコピー]、[PDF としてエクスポート]、および [タスクのエクスポート] ワークスペース要素のジョブステータスを表示するには、Security Access Manager (SAM) 内でマイジョブビジネスエンティティサービスの実行特権が有効になっていることを確認してください。

子レコードのコピーのプロパティ

コピーする子レコードの数と深度パラメータをカスタマイズするには、cmxserver.properties ファイルで次のプロパティを設定します。

copy-entity-record.num-of-children-limit

オプション。コピーする子レコードの数を設定します。デフォルトは 25 です。

copy-entity-record.depth-limit

オプション。返す子レベルの数を設定します。デフォルトは 5 です。

Hub サーバーのプロパティファイルのサンプル

Hub サーバーのプロパティファイルは cmxserver.properties と呼ばれます。

次に、通常の cmxserver.properties ファイルの内容の例を示します。

```
# Installation directory
cmx.home=$USER_INSTALL_DIR$

# Master database settings
cmx.server.masterdatabase.type=$SIP.DB.TYPE$

# Server settings
# Application server type: jboss, websphere or weblogic
cmx.appserver.type=$SIP.APPSERVER.TYPE$
cmx.appserver.version=$SIP.APPSERVER.VERSION$

# Application server hostname. Optional property to be used for deploying MRM 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=$SIP.APPSERVER.RMI.PORT$
#If the default management port 9999 for jboss is changed. This should be configured accordingly.
cmx.jboss7.management.port=$SIP.JBOSS7.MANAGEMENT.PORT$
# default setting: iiop for websphere, jnp for jboss5, remote for jboss7, t3 for weblogic
cmx.appserver.naming.protocol=$SIP.APPSERVER.NAMING.PROTOCOL$
# default setting: 8880 for websphere only (this is not being used in jboss and weblogic
cmx.appserver.soap.connector.port=$SIP.WEBSPPHERE.SOAP.PORT$
# default setting: 'No' for websphere only (this is not being used in jboss and weblogic
cmx.websphere.security.enabled=$SIP.WEBSPPHERE.SECURITY.ENABLED$
## 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

# interval sleeping between polling all databases for writelock, in seconds, default=10, 0 will disable
cmx.server.writelock.monitor.interval=10

# DO NOT EDIT SETTINGS BELOW
cmx.server.datalayer.cleansse.execution=SERVER
cmx.server.datalayer.cleansse.working_files.location=$USER_INSTALL_DIR$$/logs
cmx.server.datalayer.cleansse.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 before automated task assignment starts. 0 means immediately on Hub initialization
sip.task.assignment.start.delay=10
# 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=$INSTALLER_LOCALE$

# 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=IDDMergeTask
#The wait time for ActiveVOS to create task for the process and return task ID
activevos.workflow.startup.timeout.seconds=10

#Properties for Smart Search
# Master switch to enable/disable Smart Search
cmx.ss.enabled=true

#E360 Properties
# Master switch to enable/disable E360 View tab
cmx.e360.view.enabled=true
# Switch to enable/disable new Match and Xref screens.
cmx.e360.match_xref.view.enabled=true

# Enable to apply filters on child output as well when searching
cmx.e360.legacysearch.filterchildoutput=false

#Dataview Properties
# Master switch to enable/disable IDD DataView tab
cmx.dataview.enabled=false
cmx.dataview.taskmanager.enabled=false

#selected hashing algorithm
password.security.hash.algo=$HASHING_ALGO_NAME$

#custom hashing algorithm data: algorithm name and implementation class name
password.security.hash.algo.$HASHING_ALGO_NAME$.class=$HASHING_CLASS_NAME$

#custom hashing algorithm data: implementation class name of hub certificate jar.
com.informatica.mdm.security.certificate.provider.class=$CERTIFICATE_CLASS_NAME$

#hashing pepper value, default is empty
informatica.security.unique.id=$HASHING_PEPPER$

user_industry=$USER_INDUSTRY$

cmx.file.max_file_size_mb=20
cmx.file.max_concurrent_uploads=20
cmx.file.allowed_file_extensions=pdf,jpg,svg,xls,xlsx,svg

# Workflow view properties
cmx.e360.BPMProcess.view.enabled=false
#cmx.e360.BPMProcess.view.autologout.seconds=

# maximum number of BE records user can take from SMART search screen to find/replace screen. No error, just
scope is limited.
cmx.server.find-replace.entity-record-limit=10000

# maximum number of records that can be updated in one bulk operation. prevents "change the whole world" cases.
cmx.server.find-replace.record-limit=10000

```

```

# maximum number of records that can be processed synchronously or maximum number of records can be processed
without triggering workflow
# for bulk edit find replace only
cmx.server.find-replace.task-limit=25

# maximum number of records can be processed without triggering workflow
# for file import operation, counting on lines of file
cmx.server.be-import.task-limit=25

# set to "true" for backward compatibility with version < 10.4. For 10.4 it is better to set it to "false"
cmx.server.bes.generate-ors-specific-xsd=false

# how deep hub traverses hierarchy while trying to find hierarchy root
cmx.server.hierarchy.max-search-depth=100
# how many records will be included into export (how wide tree is traversed)
cmx.server.hierarchy.max-search-width=1000000

#Hub console session timeout in minutes
hubconsole.session.timeout=30

#Use <ORS_ID>.resourceMapping to enable or disable Business Entity Services Resources.
#After you enable the <ORS_ID>.resourceMapping flag and restart MDM Hub Server, the UI resources in the E360
Data Director tool shows resources based on the privileges that you set in the E360 UI.
#You can #set the values to true or false. Default is false.
#<ORS_ID>.resourceMapping=true

#Export the number of tasks records from the taskManger
cmx.e360.taskdata.export.limit=10000

# maximum number of children that can be loaded for each parent when creating a PDF file for download
BusinessEntity
pdf_num_of_children=1000

# maximum number of depth that can be loaded when creating a PDF file for download BusinessEntity
pdf-depth=20

# maximum number of children that can be copied for each parent when copying BE
copy-entity-record.num-of-children-limit=25
# maximum number of depth when copying BE
copy-entity-record.depth-limit=5

```

プロセスサーバーのプロパティ

cmxcleanse.properties ファイルでプロセスサーバーのプロパティを設定できます。

cmxcleanse.properties ファイルは次のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/resources

cmx.server.datalayer.cleansse.working_files.location

カスタム Java ライブラリファイルの場所（デフォルトでは、Process サーバーインストールディレクトリの場所に設定されている）。別の場所を指定する場合は、変更後にアプリケーションサーバーを再起動してください。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジアダプタガイド*』を参照してください。

cmx.server.datalayer.cleansse.working_files

クレンジングジョブの間に作成された一時ファイルを保存するかどうかを指定します。一時ファイルはトラブルシューティングや監査を目的として使用できます。一時作業ファイルを削除する場合は、FALSE に設定します。一時作業ファイルを保存する場合は、KEEP に設定します。デフォルトは、KEEP です。クレンジ

ングエンジンの統合の詳細については、『*Multidomain MDM のクレンジアダプタガイド*』を参照してください。

cmx.server.datalayer.cleanser.execution

クレンジングジョブの実行場所を指定します。クレンジングジョブをアプリケーションサーバーで実行する場合、LOCAL に設定します。クレンジングジョブをデータベースサーバーで実行する場合、DATABASE に設定します。デフォルトは LOCAL です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジアダプタガイド*』を参照してください。

cmx.home

Process サーバーのインストールディレクトリ。このプロパティは Process サーバーのインストール中に設定されます。

cmx.appserver.type

アプリケーションサーバーのタイプ。このプロパティは以下のいずれかの値になります。JBoss、WebSphere または WebLogic。このプロパティは Process サーバーのインストール中に設定されます。

cmx.appserver.version

アプリケーションサーバーの JBoss のバージョン。このプロパティは以下のいずれかの値になります。5 または 7。このプロパティは Process サーバーのインストール中に設定されます。

cmx.appserver.soap.connector.port

WebSphere 環境の場合のみ。SOAP コネクタポート。WebSphere の場合のデフォルト値は 8880 です。

cmx.websphere.security.enabled

WebSphere のセキュリティが有効かどうかを指定します。WebSphere 管理セキュリティを有効にする場合は、true または yes に設定します。デフォルト値は No です。

cmx.jboss7.management.port

JBoss 管理ポート。JBoss の場合のデフォルトは、9990 です。このプロパティはプロセスサーバーインストール中に設定されます。

cmx.server.load.nonsmos.sourcesystem.enddate.like.smos

状態管理オーバーライドシステム（SMOS）ではないシステムのリレーション終了日を SMOS と同じ日に設定します。true に設定すると、リレーションシップの終了日を SMOS と同じにすることができます。

cmx.server.match.lwm

オプション。手動で追加する必要があります。軽量一致機能を制御します。軽量一致機能と一致レコードでの完全スコアリングを有効にするには、Y に設定します。一致レコードでの完全スコアリングは行わずに軽量一致機能を有効にするには、ONLY に設定します。デフォルトは N です。

このプロパティは cmx.server.match.lwm_param および cmx.server.match.stats プロパティとともに使用します。

cmx.server.match.lwm_param

オプション。手動で追加する必要があります。cmx.server.match.lwm プロパティは Y または ONLY に設定する必要があります。このプロパティ値は次の形式で SSA-NAME3 コントロールに設定します。

LWM=Y LWM_FIELDS=<field1>,<weight1>[,...,<fieldn>,<weightn>] LWM_LIMIT=<Reject>[,<Accept>]

cmx.server.match.stats

オプション。手動で追加する必要があります。cmx.server.match.lwm プロパティは Y または ONLY に設定する必要があります。

cmx.server.match.server_encoding

一致処理に使用するエンコードの設定。一致処理に使用するエンコードを有効にする場合は、1 に設定します。デフォルトは 0 です。

cmx.server.match.max_records_per_ranger_node

一致処理ノード当たりのレコード数。一致処理当たりのレコード数が多いと、使用するメモリも多くなります。一致処理当たりの最適なレコード数は、Process サーバーで使用可能なメモリと処理能力によって異なります。デフォルトは 3000 です。

cmx.server.match.disable_subtype_match_child

重複の検索プロセスで、厳密でない子検証ストラテジを使用するかどうかを指定します。その結果、検索によって子レコードレベルで誤った一致が返される場合があります。このような一致を許可する場合は、1 に設定します。デフォルトは 0 です。

cmx.server.match.max_return_records_searcher

あいまい検索操作時の検索スレッドに対して、スコア付けされる候補レコードの数の制限を設定します。手動で追加する必要があります。デフォルトは -1 です。

あいまい検索操作が時間依存であるかまたは CPU 負荷が高い場合のプロパティを設定します。MDM Hub は、オペレーショナル参照ストア (ORS) の GETLIST Limit プロパティの値を考慮して、検索スレッドを停止するタイミングを決定します。Hub コンソールのデータベースツールを使用して、GETLIST Limit プロパティを設定します。

cmx.server.match.max_return_records_searcher プロパティの値を設定すると、あいまい検索操作がより高速に完了することがあります。次の条件のいずれかが満たされると、検索スレッドは停止します。

- スコア付けされた候補レコードの数が、cmx.server.match.max_return_records_searcher プロパティに設定された値に達する。
- 一致するレコードの数が、GETLIST Limit プロパティに設定された値に達する。

プロパティを設定しない場合、またはデフォルト値の -1 を使用する場合、あいまい検索操作は cmx.server.match.max_return_records_searcher プロパティを無視し、GETLIST Limit プロパティに基づいて実行されます。次の条件のいずれかが満たされると、検索スレッドは停止します。

- 一致するレコードの数が、GETLIST Limit プロパティに設定された値に達する。
- スコア付けの対象の候補レコードがない。

com.informatica.mdm.sifapi.xref.edit.sys0.only

オプション。手動で追加する必要があります。管理ソースシステムのみを介して編集相互参照レコードを作成するには、true に設定します。すべてのソースシステムを介して編集相互参照レコードを作成するには、false に設定します。デフォルトは true です。

重要: Hub サーバーとプロセスサーバーの両方のプロパティを設定する必要があり、プロパティ値は両方のプロパティファイルで同じである必要があります。

cmx.ss.enabled

検索を有効にするかどうかを示します。新規インストールでは、デフォルトは true です。アップグレード時に、このプロパティが設定されている場合は、値がアップグレード前の値に設定されたままになります。このプロパティが設定されていない場合、デフォルトは false です。

JBoss 6.4.0 のみ。JBoss 6.4.0 を使用している環境で検索を有効化する場合は、cmx.server.match.file_load を false に設定する必要があります。この設定により、プロセスサーバーで一致のためにネイティブデータベースユーティリティではなく JDBC アップローダーが強制的に使用されます。

cleanse.library.addressDoctor.property.SetConfigFile

Informatica Address Verification の構成ファイルのパス。例えば、C:/infamdm/Hub/cleanse/resources/AddressDoctor/5/SetConfig.xml です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cleanse.library.addressDoctor.property.ParametersFile

Informatica Address Verification のパラメータファイルのパス。例えば、C:/infamdm/Hub/cleanse/resources/AddressDoctor/5/Parameters.xml です。

cleanse.library.addressDoctor.property.DefaultCorrectionType

Informatica Address Verification の修正タイプ (PARAMETERS_DEFAULT に設定される必要あり)。

cleanse.library.trilliumDir.property.config.file.1

Trillium Director クレンジングライブラリの構成ファイル 1 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td_default_config_Global.txt です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cleanse.library.trilliumDir.property.config.file.2

Trillium Director クレンジングライブラリの構成ファイル 2 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td11_default_config_US_detail.txt です。

cleanse.library.trilliumDir.property.config.file.3

Trillium Director クレンジングライブラリの構成ファイル 3 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/Trillium/samples/director/td11_default_config_US_summary.txt です。

cleanse.library.trilliumDir11.property.config.file.1

Trillium Director 11 クレンジングライブラリの構成ファイル 1 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_Global.txt です。クレンジングエンジンの統合の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cleanse.library.trilliumDir11.property.config.file.2

Trillium Director 11 クレンジングライブラリの構成ファイル 2 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_US_detail.txt です。

cleanse.library.trilliumDir11.property.config.file.3

Trillium Director 11 クレンジングライブラリの構成ファイル 3 のファイルパス。例えば、C:/infamdm/Hub/cleanse/resources/TrilliumDirector11/samples/director/td11_default_config_US_summary.txt です。

cleanse.library.trilliumDir.property.set_maximum_retry_count

オプション。MDM Hub がレコードを処理するために Trillium サーバーに接続しようとする最大回数を設定します。デフォルトは 5 です。ネットワーク接続の再試行回数を増やすことについては、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cleanse.library.group1EntServer.property.config.file

Group1Software Enterprise Server の構成ファイル。このプロパティは Process サーバーのインストール中に設定されます。

cleanse.library.group1CDQ.property.config.file

Group1Software CDQ Server の構成ファイル。このプロパティは Process サーバーのインストール中に設定されます。

cleanse.library.firstLogicDirect.property.config.file

FirstLogicDirect の構成ファイル。このプロパティは Process サーバーのインストール中に設定されます。

cmx.server.match.distributed_match

オプション。手動で追加する必要があります。Process サーバーがクレンジング操作およびあいまい一致プロセスのための分散処理環境に参加するかどうかを指定します。デフォルトは 1 で、有効になります。分散処理を無効にするには 0 に設定します。

複数のプロセスサーバーの設定の詳細については、『*Multidomain MDM のインストールガイド*』を参照してください。

cmx.server.cleansize.min_size_for_distribution

オプション。手動で追加する必要があります。ジョブがプロセスサーバーの間で分散されときのサイズを指定します。デフォルトは 1000 です。

cmx.server.java_jdbc_loader

オプション。手動で追加する必要があります。SQL ローダーのかわりに JDBC ファイルローダーを使用するかどうかを指定します。JDBC ファイルローダーを使用するには true に設定します。デフォルトは false です。

cmx.server.tokenize.file_load

オプション。手動で追加する必要があります。中間ファイルを使用して、データをトークン化するためにデータベースにロードするかどうかを指定します。true に設定すると、中間ファイルを使用してデータをロードします。直接データロードの場合は、false に設定します。Oracle 環境と IBM DB2 環境の場合、デフォルトは true で、中間ファイルの使用によりパフォーマンスが改善されます。Microsoft SQL Server 環境の場合、デフォルトは false です。

cmx.server.tokenize.loader_batch_size

オプション。手動で追加する必要があります。トークン化プロセスのダイレクトロード中にデータベースに送る INSERT 文の最大数。デフォルトは 1000 です。

cmx.server.match.file_load

オプション。手動で追加する必要があります。中間ファイルを使用して、マッチングを行うためにデータをデータベースにロードするかどうかを指定します。true に設定すると、中間ファイルを使用してデータをロードします。直接データロードの場合は、false に設定します。Oracle 環境と IBM DB2 環境の場合、デフォルトは true です。外部一致に設定されている Microsoft SQL Server 環境と IBM DB2 環境の場合、デフォルトは false です。

注: プロパティ cmx.server.match.file_load が false に設定されている場合、クレンジングログにある一致の数がバッチビューアと異なることがあります。一致の数が異なる場合は、バッチビューアに一覧表示されている一致の数を参照してください。

cmx.server.match.loader_batch_size

オプション。手動で追加する必要があります。一致プロセスのダイレクトロード中にデータベースに送る INSERT 文の最大数。デフォルトは 1000 です。

cmx.server.match.exact_match_fuzzy_bo_api

オプション。手動で追加する必要があります。あいまい一致ベースオブジェクトで完全一致を実行する場合は、1 に設定します。あいまい一致ベースオブジェクトで完全一致を無効にする場合は、0 に設定します。デフォルトは 0 です。

変更を有効にするためにはアプリケーションサーバーを再起動する必要があります。あいまい一致ベースオブジェクトでの完全一致の設定の詳細については、『*Multidomain MDM サービスの統合フレームワークガイド*』を参照してください。

encryption.plugin.jar

オプション。手動で追加する必要があります。データ暗号化 JAR ファイルのパスおよびファイル名。データの暗号化の設定の詳細については、『[手順 3. Hub サーバーのデータ暗号化の設定](#) (ページ 190)』を参照してください。

cmx.server.bmg.use_longs

オプション。手動で追加する必要があります。Process サーバーが長い ROWID_OBJECT 値を使用できるようにする場合は、1 に設定します。Process サーバーが長い ROWID_OBJECT 値を使用できないようにする場合は、0 に設定します。デフォルトは 0 です。

cmx.server.match.threshold_to_move_range_to_hold

オプション。手動で追加する必要があります。一致分析ジョブによって保留中ステータスに移行できるレコード数の上限を設定します。デフォルトは 1000000 です。

cmx.server.dbuserexit.load.PostLoadUserExit

オプション。cmxserver.properties ファイルと cmxcleanse.properties ファイルの両方に手動で追加する必要があります。Oracle の場合のみ。MDM Hub がロードプロセス後にデータベース postload ユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。PL/SQL ユーザーイグジットについては、使用環境に対応した『*Multidomain MDM のアップグレードガイド*』を参照してください。

cmx.server.dbuserexit.PostLandingUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub がランディング後ユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

PL/SQL ユーザーイグジットの有効化の詳細については、『*Multidomain MDM のアップグレードガイド*』を参照してください。

cmx.server.dbuserexit.PreStageUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub がステージ要求の実行前にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PostStageUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub がステージ要求の実行後にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PreMatchUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub が一致要求の実行前にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PostMatchUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub が一致要求の実行後にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cmx.server.dbuserexit.PostMergeUserExit

オプション。手動で追加する必要があります。Oracle の場合のみ。MDM Hub がマージ要求の実行後にデータベースのユーザーイグジットを呼び出すかどうかを指定します。このプロパティを有効にする場合は、true に設定します。デフォルトは false です。

cluster.flag

オプション。手動で追加する必要があります。WebSphere 環境の場合のみ。クラスタリングが有効かどうかを指定します。クラスタリングを有効にする場合は、true に設定します。クラスタリングを無効にする場合は、false に設定します。デフォルトは false です。

cmx.server.cleanser.number_of_rec_batch

オプション。手動で追加する必要があります。バッチに含まれるクレンジング用のレコードの最大数を設定します。デフォルトは 50 です。

Process サーバーでのランタイム動作の設定の詳細については、『*Multidomain MDM のクレンジングアダプタガイド*』を参照してください。

cmx.server.match.searcher_search_level

オプション。手動で追加する必要があります。Data Director での拡張検索の検索レベルを設定します。値は、Narrow、Typical、Exhaustive または Extreme のいずれかです。デフォルトは Narrow です。

サーバーのプロパティを更新した後は、スキーマを検証してから Data Director アプリケーションを再デプロイする必要があります。検索レベルの詳細については、「[「検索レベル」 \(ページ 410\)](#)」を参照してください。拡張検索の設定の詳細については、『*Multidomain MDM Data Director の実装ガイド*』を参照してください。

cmx.server.match.searcher.database.worker.multithreaded

オプション。手動で追加する必要があります。true に設定すると、検索範囲の処理と SearchMatch API のパフォーマンスの最適化に、複数の並行スレッドが使用されます。デフォルトでは、マルチスレッド範囲の処理は無効になっています。

cmx.server.match.searcher.database.worker.multithreaded プロパティを設定する場合は、cmx.server.match.searcher_thread_count プロパティを設定することでスレッド数も必ず設定します。

cmx.server.match.searcher.dbfiltered.max.key.size

オプション。SearchMatch API のパフォーマンスを最適化するには、DBFILTERED のしきい値を指定します。SearchMatch レコードに cmx.server.match.searcher.dbfiltered.max.key.size プロパティの値以下の SSA_KEY がある場合に、DBFILETRED 機能が起動されます。

cmx.server.match.searcher.resultset.size

SearchMatch データベースクエリの ResultSet サイズを指定します。

cmx.server.match.searcher_thread_count

オプション。手動で追加する必要があります。SearchMatch API のスレッド数を設定します。デフォルトは 1 です。SearchMatch API に対して 1 つのスレッドを使用する場合は、1 に設定します。

cmx.server.match.searcher_thread_count プロパティをデフォルト値以外の値に設定する場合は、cmx.server.match.searcher.database.worker.multithreaded プロパティを必ず true に設定します。

SearchMatch API のパフォーマンスの最適化の詳細については、次に示す Informatica MySupport ポータルの H2L を参照してください。

- *Multidomain MDM for IBM DB2 パフォーマンスのチューニング*
<https://mysupport.informatica.com/docs/DOC-11208>
- *Multidomain MDM for Oracle パフォーマンスのチューニング*
<https://mysupport.informatica.com/docs/DOC-11207>
- *Multidomain MDM for Microsoft SQL Server パフォーマンスのチューニング*
<https://mysupport.informatica.com/docs/DOC-11149>

ex.max.conn.per.host

ホストに接続する Elasticsearch ノードの最大数を設定します。ホスト上の Elasticsearch クラスターノードの数に設定します。

ex.max.threads

Apache Elasticsearch クラスターでノードごとに非同期ノンブロッキングレシーバを使用するスレッドの最大数を設定します。デフォルトは 1 です。

この値を変更するのは、Informatica グローバルカスタマサポートから提案された場合のみにしてください。

`search.provisioning.numshards`

オプション。ご使用の Elasticsearch 環境で作成するシャードの数。値はシャードの最大数およびノードの合計数により異なります。例えば、シャードの最大数が 1 で、ノードの数が 3 の場合は、3 つのシャードを作成することができます。デフォルトは、検索を有効化しているプロセスサーバーの合計数です。

`search.provisioning.numreplicas`

オプション。別々のノードで作成する Elasticsearch エンジンドキュメントの部数。異なるノードのシャードにドキュメントの複数のコピーを作成するには、レプリケーション要素を使用します。1 つ以上のノードが予期せずシャットダウンした場合の高可用性を実現するには、ドキュメントの複数のコピーが必要です。例えば、レプリケーション要素が 2 の場合、2 つのノードにあるドキュメントの 2 つのコピーを取得します。Elasticsearch のデフォルトは 0 です。

`MAX_INITIAL_RESULT_SIZE_TO_CONSIDER`

オプション。プロパティを手動で追加します。Data Director アプリケーションで表示する検索結果の合計数。推奨最大値は 250 です。デフォルトは 130 です。130 を超える値に設定すると、Data Director アプリケーションのパフォーマンスに影響を与えます。

`mdm.smartsearch.cache.ttl`

オプション。プロパティを手動で追加します。ビジネスエンティティの検索 Web サービス要求のキャッシュされた検索結果が、有効期限切れになるまでの時間（ミリ秒）。デフォルトは 60000 です。

`min_rec_for_multithreading`

マルチスレッドバッチ操作をバッチジョブに適用するための、MDM Hub の最小バッチサイズ。次のタイプのバッチジョブに適用されます。自動マージ、マージ解除、ロード、スマート検索データの初期インデックス処理、ステージ、分散一致、およびトークン化プロセス。デフォルトは 1000 です。

`mq.data.change.monitor.thread.start`

マルチノード環境で、個々のノードに対してメッセージキューのポーリングを行うかどうかを指定します。メッセージキューのポーリングを無効にするには、`false` に設定します。MDM Hub EAR ファイルがデプロイされているすべての Java 仮想マシン上で、デフォルトは `true` です。

`ssl.keyStore`

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルの絶対パスおよびファイル名。

`ssl.keyStore.password`

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。キーストアファイルのプレーンテキストパスワード。

`ssl.trustStore`

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルの絶対パスおよびファイル名。

`ssl.trustStore.password`

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。プロパティを手動で追加します。トラストストアファイルのプレーンテキストパスワード。

`cmx.websphere.security.ssl.config.url`

アプリケーションサーバーの HTTPS ポートを使用してプロセスサーバーを設定する場合は必須です。WebSphere 環境の場合のみ。プロパティを手動で追加します。`ssl.client.props` ファイルの絶対パス（ファイル名を含む）。

cmx.outbound.bypass.multixref.insystem

オプション。手動で設定する必要があります。バッチジョブで複数の相互参照レコードでベースオブジェクトを更新する際に、プロセスサーバーでのメッセージの作成をバイパスするには、true に設定します。デフォルトは false です。

cmx.server.stage.sqlldr.charset

オプション。データのアップロードに SQL*Loader を使用していて、アップロードしたデータが破損している場合、このプロパティをデータに一致する文字セット（AL32UTF8 など）に設定します。ステージジョブを実行すると、指定した文字セットを使用する SQL*Loader の制御ファイルが生成されます。その後、データを再ロードできます。デフォルトは UTF8 です。

cmx.server.stripDML.blockSize

MDM Hub が各ブロックで処理するレコード数。デフォルトは 100 です。

cmx.server.stripDML.noOfThreadsForDelete

MDM Hub が一致キーテーブルからレコードを削除するために使用するスレッド数。デフォルトは 30 です。

cmx.server.stripDML.noOfThreadsForInsert

MDM Hub が一致キーテーブルにレコードを挿入するために使用するスレッド数。デフォルトは 50 です。

cmx.server.stripDML.noOfThreadsForUpdate

MDM Hub が一致キーテーブルのレコードを更新するために使用するスレッド数。デフォルトは 30 です。

cmx.server.stripDML.useDeleteInsertLock

オプション。手動で設定する必要があります。MDM Hub が一致ジョブ中にトークン化を実行する、または多数のレコードが含まれるベースオブジェクトでトークン化 API 呼び出しを実行するには、true に設定します。デフォルトは false です。

cmx.server.stripDML.useUpdate

オプション。手動で設定する必要があります。IBM DB2 の場合のみ。true に設定すると、再トークン化中の IBM DB2 環境のパフォーマンスが向上します。デフォルトは false です。

com.siperian.common.xml.sdo.sdo_resolver_pool_size

オプション。手動で追加する必要があります。SDO の生成に使用するスレッドプールの数を指定します。値の範囲は 1 から 20 までです。指定された値が範囲外の場合は、デフォルト値が使用されます。デフォルトは 3 です。

プロセスサーバーのプロパティファイルのサンプル

Process サーバーのプロパティファイルは cmxcleanse.properties と呼ばれます。

以下に、通常の cmxcleanse.properties ファイルの内容の例を示します。

```
# 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

```

オペレーショナルリファレンスストアのプロパティ

オペレーショナル参照ストアデータベースのプロパティには、データベースのベンダーとバージョン、C_REPOS_DB_RELEASE テーブルからの情報が含まれています。

次のリストに、エンタープライズマネージャツールに表示されるオペレーショナル参照ストアプロパティを示します。

データベース製品名

データベースシステムの名前。

データベース製品バージョン

使用しているデータベースバージョン。

環境 ID

環境の ID。

TNS 名

オペレーショナル参照ストアデータベースの TNS 名。

接続 URL

オペレーショナル参照ストアデータベースに接続するための URL。

データベース ID

データベース ID。

システム間の時間差

差分検出値（秒単位）。これにより、入力データが現在より後のものかどうかを判別します。

バイト単位のカラム長

Oracle 環境の場合。

SQL*Loader でロード中のデータベースが UTF-8 データベースかどうかを判別するために使用されます。

デフォルト値は 1 で、データベースが UTF-8 であることを示します。

ロードテンプレート

IBM DB2 環境用。ログ取りモードのときに db2 ロードコマンドが生成するログファイルの、データベースサーバー上のパス。

MTIP の再生成が必要

true に設定されている場合、マッチ/マージプロセスの前に MTIP ビューが再生成されます。

デフォルト値は false で、ビューは再生成されません。

エンタープライズマネージャツールの [MTIP の再生成] ボタンで、MTIP ビューを再生成できます。

注: ORS がプロダクションモードの場合は、MTIP ビューを再生成できません。エンタープライズマネージャツールから MTIP を再生成する前に、プロダクションモード設定をオフにします。

グローバルなロギングなし

Oracle および IBM DB2 環境用。テーブル作成時にデータベースリカバリの目的でロギングを有効にする場合に使用されます。デフォルト値は true で、ロギングは行われません。

モデルリポジトリサービス URL

MDM Hub がモデルリポジトリサービスに接続するために必要な接続パラメータ。エンタープライズマネージャツールで接続パラメータを指定します。

付録 B

設定の詳細の表示

この付録では、以下の項目について説明します。

- [設定の詳細の表示の概要, 655 ページ](#)
- [エンタープライズマネージャの開始, 655 ページ](#)
- [エンタープライズマネージャのプロパティ, 656 ページ](#)
- [環境レポート, 658 ページ](#)
- [エンタープライズマネージャでのバージョン履歴の表示, 658 ページ](#)
- [アプリケーションサーバーのログの使用, 659 ページ](#)
- [クライアントに対する Hub コンソールログの使用, 661 ページ](#)

設定の詳細の表示の概要

Hub コンソールでエンタープライズマネージャツールを使用して、Informatica MDM Hub 実装の設定の詳細を設定し、表示することができます。

エンタープライズマネージャツールを使用して、Hub サーバー、プロセスサーバー、ORS データベース、および MDM Hub マスターデータベースのプロパティ、バージョン履歴、環境レポートを表示します。また、エンタープライズマネージャを使用して、ORS データベースのデータベースログを設定および表示することもできます。

エンタープライズマネージャの開始

Hub コンソールでエンタープライズマネージャツールを起動するには、設定ワークベンチを展開して、**【エンタープライズマネージャ】**をクリックします。また、Hub コンソールツールバーのエンタープライズマネージャツールのクイック起動ボタンをクリックすることもできます。

エンタープライズマネージャツールが Hub コンソールに表示されます。

エンタープライズマネージャのプロパティ

Hub サーバー、プロセスサーバー、MDM Hub マスターデータベース、または ORS データベースのプロパティを表示するには、エンタープライズマネージャツールを使用します。表示するサーバーやデータベースを選択する前に、エンタープライズマネージャを起動しておく必要があります。

エンタープライズマネージャツールを使用して次のプロパティを表示します。

Hub サーバー

Hub サーバータブを選択すると、エンタープライズマネージャに Hub サーバースプロパティが表示されます。これらのプロパティの詳細については、cmxserver.properties ファイルを参照してください。

プロセスサーバー

プロセスサーバータブを選択すると、エンタープライズマネージャにプロセスサーバーのリストが表示されます。特定の Process サーバーを選択すると、エンタープライズマネージャにそのプロパティが表示されます。これらのプロパティの詳細については、cmxcleanse.properties ファイルを参照してください。

マスターデータベース

[マスターデータベース] タブを選択すると、エンタープライズマネージャに MDM Hub マスターデータベースプロパティが表示されます。表示されるデータベースプロパティは、データベースのベンダーとバージョンだけです。

ORS データベース

[オペレーショナル参照ストアデータベース] タブを選択すると、エンタープライズマネージャにオペレーショナル参照ストアデータベースのリストが表示されます。オペレーショナル参照ストアデータベースを選択すると、エンタープライズマネージャにそのオペレーショナル参照ストアのプロパティが表示されます。

上部パネルには、MDM Hub マスターデータベースに登録されているオペレーショナル参照ストアデータベースのリストが表示されます。下部パネルには、上部パネルで選択したオペレーショナル参照ストアデータベースのプロパティとバージョン履歴が表示されます。オペレーショナル参照ストアのプロパティには、データベースのベンダーとバージョン、C_REPOS_DB_RELEASE テーブルからの情報が含まれます。バージョン履歴は C_REPOS_DB_VERSION テーブルにも保持されます。

注: エンタープライズマネージャには、現在のバージョンの MDM Hub で有効なオペレーショナル参照ストアデータベースのみが表示されます。エンタープライズマネージャがオペレーショナル参照ストアデータベースに関する情報を取得できない場合、エンタープライズマネージャには、オペレーショナル参照ストアデータベースがリストに含まれていない理由を説明するメッセージが表示されます。

C_REPOS_DB_RELEASE テーブル

C_REPOS_DB_RELEASE テーブルにはオペレーショナル参照ストアプロパティが含まれています。

次のリストでは、エンタープライズマネージャにより、ユーザーの設定に応じてオペレーショナル参照ストアデータベースに表示される C_REPOS_DB_RELEASE のプロパティについて説明します。

DEBUG_LEVEL

ORS データベースのデバッグレベル。デバッグレベルには以下のレベルがあり、それぞれの整数値で指定されます。

100 = エラーレベルのデバッグ

200 = 警告レベルのデバッグ

300 = 情報レベルのデバッグ

400 = デバッグレベルのデバッグ

500 = すべてのレベルのデバッグ

ENVIRONMENT_ID

環境の ID。

DEBUG_FILE_PATH

ORS データベースデバッグログの場所へのパス。

DEBUG_FILE_NAME

ORS データベースデバッグログの名前。

DEBUG_IND

デバッグが有効かどうかを示すフラグ。

0 = デバッグは無効です。

1 = デバッグは有効です。

DEBUG_LOG_FILE_SIZE

Oracle 環境用。 データベースログファイルのサイズ (MB 単位)。デフォルトは 5 です。

DEBUG_LOG_FILE_NUMBER

Oracle 環境用。 ログローリングに使用されるログファイルの数。デフォルトは 5 です。

TNSNAME

ORS データベースの TNS 名。

CONNECTION_PORT

ORS データベースがリスンするポート。

ORACLE_SID

Oracle データベースの ID。

DATABASE_HOST

データベースがインストールされているホスト。

INTER_SYSTEM_TIME_DELTA_SEC

データベースサーバーのシステム時刻と別のマシンのシステム時刻の差を補うために必要な秒数。

COLUMN_LENGTH_IN_BYTES_IND

Oracle 環境用。 SQL Loader でロード中のデータベースが UTF-8 データベースかどうかを判別するために使用されるフラグ。デフォルト値の 1 は、データベースが UTF-8 であることを意味します。

LOAD_TEMPLATE

IBM DB2 環境用。 ログ取りモードのときに db2 ロードコマンドが生成するログファイルの、データベースサーバー上のパス。

MTIP_REGENERATION_REQUIRED_IND

一致/マージプロセスの前に MTIP ビューが再生成されることを示すフラグ。デフォルト値の 0 (ゼロ) は、ビューが再生成されないことを意味します。

GLOBAL_NOLOGGING_IND

Oracle および IBM DB2 環境用。 テーブル作成時に DB リカバリの目的でロギングを有効にする場合に使用されます。デフォルト値の 1 は、ロギングが行われないことを意味します。

環境レポート

【環境レポート】タブを選択すると、関連するエラーメッセージに加えて、すべてのハブコンポーネントタブのプロパティのサマリがエンタープライズマネージャによって表示されます。レポートには、次の順序でプロパティが並んでいます。

- Hub サーバー
- プロセスサーバー
- マスターデータベース
- ORS データベース

MDM Hub 環境レポートの保存

MDM Hub 環境レポートを保存するには、Hub コンソールでエンタープライズマネージャツールを使用します。

1. Hub コンソールの【設定】ワークベンチでエンタープライズマネージャツールを選択します。
2. エンタープライズマネージャツールで【環境レポート】タブを選択します。
3. 【保存】をクリックします。
4. 【Hub 環境レポートの保存】ダイアログボックスで、環境レポートを保存するディレクトリに移動します。
5. 【保存】をクリックします。

エンタープライズマネージャでのバージョン履歴の表示

エンタープライズマネージャツールを使用して、Hub サーバー、プロセスサーバー、MDM Hub マスターデータベース、または ORS データベースのバージョン履歴を表示することができます。表示するサーバーやデータベースを選択する前に、エンタープライズマネージャを起動しておく必要があります。

1. エンタープライズマネージャ画面で、表示する情報タイプのタブを選択します。
 - Hub サーバー
 - プロセスサーバー
 - マスターデータベース
 - ORS データベース

エンタープライズマネージャに、選択項目に固有のバージョン履歴が表示されます。

2. 【バージョン履歴】タブを選択します。

エンタープライズマネージャに、その特定のコンポーネントのバージョン情報が表示されます。バージョン履歴は、インストールの開始時刻の降順にソートされます。

アプリケーションサーバーのログの使用

log4j.xml ファイルを使用して、Hub サーバーまたはプロセスサーバーのアプリケーションサーバーのログファイルを設定します。エンタープライズマネージャを使用して、アプリケーションサーバーのログを設定することはできません。

アプリケーションサーバーのログレベル

アプリケーションサーバーのログの log4j.xml コンフィギュレーションファイルには、デバッグして情報を取得するために一連のログレベルが含まれます。

次の表に、アプリケーションサーバーのログレベルを示します。

名前	ORS メタデータテーブルの値	説明
ALL	500	関連するすべての情報をログに記録します。
デバッグ (DEBUG)	400	デバッグに使用されます。(デフォルト)
情報 (INFO)	300	アプリケーションサーバーのログ情報。
警告	200	警告メッセージ。
エラー (ERROR)	100	エラーメッセージ。

ログファイルのローリング

アプリケーションサーバーのログファイルが log4j.xml 設定の MaxFileSize パラメータで定義されている最大サイズに達すると、エンタープライズマネージャはログのローリングプロシージャを実行して既存のログ情報をアーカイブし、ログファイルが新しいアプリケーションサーバー情報で上書きされないようにします。

- Hub サーバーおよびプロセスサーバーの log4j.xml ファイルにある次のアプリケーションサーバーログ構成設定を定義します (それぞれ `<infamdm_install_dir>\hub\server\conf` と `<infamdm_install_dir>\cleanse\server\conf` にあります)。
 - ファイル。C:\infamdm\hub\server\logs\cmxserver.log など、ログファイルの名前。
WebLogic および WebSphere では、次のファイル名を使用してファイルを互いに区別できるようにしながら、ログを同じ場所に収集することが推奨されます。
Hub Server: C:\<infamdm_install_dir>\hub\server\logs\cmxserver.log
プロセスサーバー: C:\<infamdm_install_dir>\hub\server\logs\cmxcleanse.log
 - MaxFileSize。アプリケーションサーバーのログファイルの最大ファイルサイズ。
 - MaxBackupIndex。最大ファイル数。
 - しきい値。しきい値ログレベル。このしきい値レベルより高いログレベルをすべて上書きします。
- Hub サーバーとプロセスサーバーに使用されるログファイルにより、さまざまなアプリケーションサーバーメッセージがアプリケーションサーバーのログファイルに追加されます。
- 物理ログファイルのサイズが MaxFileSize を超えると、Hub がログローリングプロシージャをアクティブ化します。
 - アクティブなログファイルの名前が、<filename>.hold に変更されます。
 - <filename>.(n) というファイルの名前がすべて<filename>.(n+1) に変更されます。

- c. n+1 が MaxBackupIndex より大きい場合は、ファイルが削除されます。
- d. ロールオーバーがアプリケーションサーバーファイルの最大数を超えると、MDM Hub は元のログファイルの名前を変更します。例えば、MDM Hub によって cmxserver.log が cmxserver.log.1 に、cmxserver.log.1 が cmxserver.log.2 に名前が変更されます。その後に、cmxserver.log が新しいログ情報で上書きされます。
- e. <filename>.hold ファイルの名前は<filename>.1 に変更されます。

注: また、Hub でロールオーバーが実行される前に、log.rolling ファイルが作成されます。ログファイルが期待どおりにローリングされない場合は、ログファイルディレクトリを確認し、ログのローリングを続行できるように log.rolling ファイルを削除します。

アプリケーションサーバーのログの設定

デバッグするには、アプリケーションサーバーの設定を行う必要があります。

アプリケーションサーバーのログは、アプリケーションサーバーレベルで管理されます。アプリケーションサーバーのログファイル内に含まれるログエントリは、ORS 固有ではなく、アプリケーションサーバー固有のもので、Hub サーバーのログファイルはアプリケーションサーバー固有の cmxserver.log ファイルに書き込まれます。複数のアプリケーションサーバーを使用する場合、アプリケーションサーバーごとに独自の構成ファイルとログファイルがあります。同様に、WebSphere および WebLogic にインストールされているプロセスサーバーは、それぞれのアプリケーションサーバー上のログファイルに書き込みます。JBoss 上に MDM Hub がインストールされている場合、Hub サーバーとプロセスサーバーは同じ JBoss インスタンス上にあり、共有されている同じアプリケーションサーバーログファイルに書き込みます。

表示するサーバーやデータベースを選択する前に、エンタープライズマネージャを起動しておく必要があります。アプリケーションサーバーのログを設定するには、次の手順を実行します。

1. log4j.xml ファイルを編集用を開きます。
log4j.xml ファイルは<infamdm_install_dir>\hub\server\conf にあります。
2. log4j.xml ファイルを編集します。
 - デフォルト値を DEBUG に変更します。
 - log4j.xml ファイルにあるログカテゴリの名前を com.siperian、com.informatica、および com.delos に設定します。
 - ログカテゴリ siperian.performance を「OFF」に設定します。このログカテゴリは、Informatica からそのように要求があった場合のみ「ON」にする必要があります。

次の例に、log4j.xml ファイルの設定を示します。

```
<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>
```

クライアントに対する Hub コンソールログの使用

コンソールログには、コンソールイベントから、およびコンソールとサーバーの間の通信から発生したメッセージおよびエラーが含まれています。コンソールログは、クライアントマシンの以下のディレクトリにあります。

`<Windows_users_home_dir>\Siperian\console.log`

同じフォルダ内にある `log4j.properties` ファイルを使用して、ログレベル、ログファイルサイズ、古いログを保持する数を管理できます。`log4j.properties` ファイルの最後の行は以下のように設定する必要があります。

`log4j.rootCategory=DEBUG, FileLog, ConsoleLog`

付録 C

行レベルのロック

この付録では、以下の項目について説明します。

- [行レベルのロックの概要, 662 ページ](#)
- [行レベルのロックについて, 662 ページ](#)
- [行レベルのロックの設定, 663 ページ](#)
- [SIF 要求とバッチプロセスの間のロックの相互作用, 664 ページ](#)

行レベルのロックの概要

この付録では、バッチおよび API プロセス（SIF 要求を含む）または API/API プロセスを非同期的に実行する際にデータの整合性を保護するために行レベルのロックを有効にして使用方法について説明します。

注: Informatica MDM Hub 実装でバッチジョブと API 呼び出しを同時に実行しない場合は、この節をスキップできます。

行レベルのロックについて

Informatica MDM Hub では、行レベルのロックを使用して、ベースオブジェクトのレコードに対してバッチと SIF の操作を同時に実行する場合のデータの整合性を管理します。行レベルのロックの特徴は次のとおりです。

- ベースオブジェクトのデータ（同じベースオブジェクト内の異なるレコード）をバッチプロセスと SIF プロセスの両方で更新できます。
- オンラインプロセスとバッチプロセスの競合がなくなります。
- データへの高度な同時アクセスが可能になります。
- ミラーリング環境に必要なハードウェアの重複を回避します。

行レベルのロックは、バッチと API のプロセスまたは API と API のプロセスを Informatica MDM Hub 実装で非同期的に実行する場合に有効にする必要があります。行レベルのロックは、非同期の SIF とバッチの処理に適用されます。同期バッチの処理については、従来のアプリケーションレベルのロックで制限されます。

デフォルトの動作

行レベルのロックはデフォルトで無効になっています。無効である間は、API プロセス（SIF 要求を含む）およびバッチプロセスを同時に同じベースオブジェクトで非同期的に実行できません。オペレーショナル参照ス

トアに対して行レベルのロックを明示的に有効にすると、Informatica MDM Hub は行レベルのロックメカニズムを使用して、トークン化プロセス、マッチプロセス、およびマージプロセスでの同時更新を管理します。

ロックのタイプ

Informatica MDM Hub のデータ管理には、次のタイプのロックが含まれます。

排他ロック

他のすべてのジョブ（API プロセスまたはバッチプロセス）がロックされているベースオブジェクトに対して処理を実行できないようにします。

共有ロック

特定のジョブのみ実行できないようにします。例えば、あるバッチジョブでベースオブジェクトに対する非排他ロックが発行されると、相互運用が有効（オン）になっている場合は、この共有ロックによって他のバッチジョブが禁止されますが、API ジョブはベースオブジェクトに対して処理を試行できます。

行レベルのロック

影響を受けるベースオブジェクトの行をロックする共有ロック。

行レベルのロックの使用に関する考慮事項

Informatica MDM Hub 実装で行レベルのロックを使用するときは、以下の点に注意してください。

- バッチプロセスは個々のレコードを長時間ロックできるため、Web 経由での SIF のアクセスが妨げられる可能性があります。
- SIF 要求が短時間ロックアウトされることがあります。ただし、これはほとんどなく、あったとしても 10 分以内です。
- Hub ストアのサイズは、バッチプロセスと SIF 要求を同時に実行した場合のリソース需要の負荷に十分に対応できるサイズにする必要があります。

注: 複数のバッチジョブをまとめて実行する場合は、相互運用性を有効にする必要があります。複数のバッチジョブを実行する際に同じ子（または親）レコードへのアクセスを試行する親が複数ある場合、他のバッチで処理中のレコードをロックしようとし、それらのレコードを保持している時間がバッチの待機時間を超えると、そのジョブは失敗します。最大待機時間は C_REPOS_TABLE で定義されます。

行レベルのロックの設定

ここでは、行レベルのロックを設定する方法について説明します。

ORS での行レベルのロックの有効化

デフォルトでは、行レベルのロックは有効になっていません。オペレーショナル参照ストアで行レベルのロックを有効にする手順

1. Hub コンソールで、データベースツールを起動します。
2. 設定するオペレーショナル参照ストアを選択します。
3. オペレーショナル参照ストアのプロパティを編集します。
4. **【バッチと API のロックの相互運用性】** チェックボックスを選択（オンに）します。

5. 変更を保存します。

注: 行レベルのロックを有効にすると、[PUT 時にトークン化] プロパティを有効にできなくなります。

ロック待機時間の設定

有効にすると、スキーママネージャを使用して、オペレーショナル参照ストアのベースオブジェクトに対する SIF およびバッチのロック待機時間を設定できます。この待機時間を過ぎると、Informatica MDM Hub でエラーメッセージが表示されます。

SIF 要求とバッチプロセスの間のロックの相互作用

この節では、SIF 要求とバッチプロセスの間のロックの相互作用について説明します。

API とバッチの相互運用性が有効になっている場合の対話

以下の表に、行レベルのロックが有効になっている場合の API とバッチプロセスの対話を示します。

既存の ロック/新 しい入 力呼び 出し	バッチ- 排他ロッ ク	バッチ-共有ロック	API-行レベルのロック
バッチ-排 他ロッ ク	即座にエ ラーメッ セージを 表示しま す。	即座にエ ラーメッ セージを 表示しま す。	Batch_Lock_Wait_Seconds の時間だけ待機して、ロックの存在をチェックします。ロックが待機時間中に解除されない場合は、エラーメッセージを表示します。ロックされるテーブルごとに呼び出されます。
バッチ-共 有ロッ ク	即座にエ ラーメッ セージを 表示しま す。	即座にエ ラーメッ セージを 表示しま す。	Batch_Lock_Wait_Seconds の時間だけ待機して、FOR UPDATE SELECT を使用して行レベルのロックを適用します。テーブルでロックが管理されない場合は、エラーメッセージを表示します。ロックされるテーブルごとに呼び出されます。
API-行 レベル のロッ ク	即座にエ ラーメッ セージを 表示しま す。	API_Lock_Wait_Seconds の時間だけ待機して、FOR UPDATE SELECT を使用して行レベルのロックを適用します。テーブルでロックが管理されない場合は、エラーメッセージを表示します。	API_Lock_Wait_Seconds の時間だけ待機して、FOR UPDATE SELECT を使用して行レベルのロックを適用します。テーブルでロックが管理されない場合は、エラーメッセージを表示します。ロックされるテーブルごとに呼び出されます。

注: 承認タスクを実行する際、XREF テーブルの INTERACTION_ID カラムのカスタムインデックスによってサービス側のパフォーマンスは向上しますが、特にロードおよび自動マージなどの大規模バッチプロセスでパフォーマンスが低下するというトレードオフがあります。

API とバッチの相互運用性が無効になっている場合の対話

以下の表に、API とバッチの相互運用性が無効になっている場合の API プロセスとバッチプロセスの対話を示します。このシナリオでは、バッチプロセスが排他ロックを発行する一方で、SIF 要求は排他ロックをチェックするだけでロックの発行は行いません。

既存のロック/新しい入力呼び出し	バッチ	API
バッチ-排他ロック	即座にエラーメッセージを表示します。	「デフォルトの動作」 (ページ 662) を参照してください。
API	即座にエラーメッセージを表示します。	「デフォルトの動作」 (ページ 662) を参照してください。

付録 D

MDM Hub ロギング

この付録では、以下の項目について説明します。

- [MDM Hub のロギングの概要, 666 ページ](#)
- [ロギングの設定, 667 ページ](#)
- [Hub コンソールのログ, 667 ページ](#)
- [Hub Server ログ, 667 ページ](#)
- [プロセスサーバーログ, 668 ページ](#)
- [エンティティ 360 ログ, 668 ページ](#)
- [プロビジョニングツールのログ, 668 ページ](#)

MDM Hub のロギングの概要

MDM Hub で発生したメッセージとエラーはすべて、ログファイルに格納されます。MDM Hub は、MDM Hub のインストールまたはアップグレード中にログファイルを作成します。

次の表に、MDM Hub コンポーネントのログファイルを示します。

MDM Hub コンポーネント	ログファイル
Hub コンソール	console.log
Hub サーバー	cmxserver.log
プロセスサーバー	cmxserver.log
エンティティ 360	entity360view.log
プロビジョニングツール	provisioning.log

ロギングの設定

ログが記録されるように Hub サーバーを設定することができます。log4j.xml ファイルにロギングの設定を指定します。

1. 以下のディレクトリにある log4j.xml を開きます。
<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/conf
2. 次のカテゴリ名の<priority name>要素でロギングレベルを設定します。
 - com.siperian
 - com.delos
 - com.informaticaロギングレベルは、次の値のいずれかに設定できます。
 - DEBUG。最も詳細なログの記録方法。
 - INFO。やや詳細なログの記録方法。
 - ERROR。最も簡易なログの記録方法。デフォルトは INFO です。
3. Threshold パラメータを DEBUG に設定します。

Hub コンソールのログ

MDM Hub を使用すると、Hub コンソールを起動したオペレーティングシステム環境で console.log ファイルが作成されます。Hub コンソールを Windows 環境で起動した場合は、C:\Documents and Settings\<user_home>\siperian ディレクトリに Hub コンソールログが作成されます。Hub コンソールを UNIX 環境で起動した場合は、/<user_home>/siperian ディレクトリに Hub コンソールログが作成されます。

Hub コンソールログには、Hub コンソールからのログメッセージが含まれています。MDM Hub とアプリケーションサーバーとの通信で発生したエラーや、アプリケーションサーバーからのエラーメッセージ、コンソールエラーメッセージなどが、このファイルに記録されます。デフォルトでは、MDM Hub によって console.log が作成されます。

console.log ファイルはローリングログファイルです。ログのサイズが 5 MB に達すると console.log.1 にコピーされ、ログが再開されます。Hub サーバーはこの処理を無限に行うため、多数のログファイルが作成される可能性があります。古いファイルを定期的に削除するか、別のストレージ領域に転送する必要があります。

Hub Server ログ

Hub Server では、アプリケーションサーバーのログファイルである cmxserver.log が生成されます。

Hub Server ログは以下のディレクトリに表示されます。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/logs

Hub Server ログには、アプリケーションサーバーのログ処理とデバッグ処理に関する情報が書き込まれます。デフォルトでは、Hub Server によって cmxserver.log ファイルが作成されます。cmxserver.log はローリングログファイルであり、サイズが 5 MB に達すると、Hub Server によって cmxserver.log.1 にコピーされ、ログが再

開されます。Hub サーバーはこの処理を無限に行うため、多数のファイルが作成される可能性があります。古いファイルを定期的に削除するか、別のストレージ領域に転送する必要があります。

プロセスサーバーログ

プロセスサーバーでは、クレンジング、トークン化、およびシミュレーション関数のために `cmxserver.log` が生成されます。

プロセスサーバーログは、アプリケーションサーバーのログファイルです。 `cmxserver.log` ファイルは以下のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/logs

`cmxserver.log` ファイルには、クレンジングプロセスのデバッグ処理とエラーメッセージが書き込まれます。デフォルトでは、プロセスサーバーによって `cmxserver.log` ファイルが作成されます。 `cmxserver.log` ファイルはローリングログファイルで、最大ファイルサイズに達すると、プロセスサーバーによって `cmxserver.log.1` にコピーされ、ログが再開されます。

プロセスサーバーのログファイルの最大サイズは、デフォルトでは 5MB です。ただし、次のディレクトリで最大サイズを設定できます。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/conf/log4j.xml

エンティティ 360 ログ

MDM Hub では、エンティティ 360 フレームワークのすべてのメッセージ、エラー、およびフルスタックトレースを記録する `entity360view.log` が生成されます。

`entity360view.log` ファイルは、次のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/logs

`log4j-entity360view.xml` ファイルには、エンティティ 360 フレームワークの設定情報が保存されますが、Hub サーバーの設定情報は含まれません。

`log4j-entity360view.xml` ファイルは、次のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/conf

プロビジョニングツールのログ

MDM Hub は、プロビジョニングツールの設定ログメッセージを格納する `provisioning.log` を生成します。

`provisioning.log` ファイルは次のディレクトリに配置されます。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/logs

`log4j-provisioning.xml` ファイルには、プロビジョニングツールの設定情報が保存されますが、Hub サーバーの設定情報は含まれません。

`log4j-provisioning.xml` ファイルは、次のディレクトリにあります。

<MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/conf

付録 E

テーブルのパーティション化

- [テーブルのパーティション化のサポート, 670 ページ](#)

テーブルのパーティション化のサポート

MDM 設計チームではまだテストしていませんが、Multidomain MDM でサポートし、パーティション化されたテーブルで正常に機能することが期待されています。Informatica では、テーブルのパーティション化によって MDM への一部のリアルタイムアクセスのパフォーマンスが向上する一方、パーティション境界をまたがるレコードの照合やマージのパフォーマンスに影響を及ぼす可能性があるかと予測しています。このため、顧客の通常の開発プロセスの一環として、全体的なパフォーマンスへの影響をテストし、測定することをお勧めします。

付録 F

製品使用ツールキットを使用した MDM 環境情報の収集

この付録では、以下の項目について説明します。

- [製品使用ツールキットを使用した MDM 環境情報の収集の概要, 671 ページ](#)
- [Hub サーバーでの MDM Hub データ収集の有効化, 673 ページ](#)
- [プロセスサーバーでの MDM Hub データ収集の有効化, 673 ページ](#)
- [Hub サーバーでの MDM Hub データ収集の無効化, 674 ページ](#)
- [プロセスサーバーでの MDM Hub データ収集の無効化, 674 ページ](#)

製品使用ツールキットを使用した MDM 環境情報の収集の概要

MDM Hub による MDM Hub 環境に関する情報の Informatica への送信を有効または無効にできます。製品使用ツールキットは、Hub サーバーやプロセスサーバーのシステムコンポーネントに関する情報や、MDM Hub 環境に関する情報を送信できます。

製品使用ツールキットによって送信される情報は、MDM Hub 実装の重要なインサイトを示す、詳細なヘルスチェックです。Informatica のエキスパートは、ユーザーの環境に合わせてベストプラクティスの推奨事項を調整したり、プロジェクトを迅速に実装できるよう提案したりすることができます。

データ収集を有効にすると、アプリケーションサーバーの起動から 10 分後に製品使用ツールキットが Informatica に情報を送信します。それ以降は、30 日ごとに情報が送信されます。

重要: MDM Hub をインストールまたはアップグレードする場合、データ収集はデフォルトで有効です。インストールまたはアップグレード後は、いつでもデータ収集を無効にできます。

システム設定情報

次の表に、データ収集を有効にしたときに収集される Hub サーバーコンポーネントとプロセスサーバーコンポーネントに関する情報を示します。

収集される情報	説明
メモリ	合計物理メモリ、使用可能な物理メモリ、最大仮想メモリ、使用可能な仮想メモリ、および使用中の仮想メモリ
CPU	CPU 名、最大クロック速度、および負荷率
環境変数	環境変数の名前と値
ディスクドライブ	ドライブ文字 ID、ドライブサイズ、および空き領域
オペレーティングシステム情報	オペレーティングシステム名、バージョン、製造元、構成、ビルドタイプ
パッチに関するオペレーティングシステム情報	オペレーティングシステムのパッチのリスト
サービス	各サービスの名前とステータス
ネットワーク構成	ネットワーク構成についての詳細
仮想化	ノード ID、バージョン、およびシリアル番号

MDM Hub 環境情報

次の表に、データ収集を有効にした場合に収集される MDM Hub 環境に関する情報を示します。

収集される情報	説明
ホストタイプ	ホストのタイプ
ビルド番号	MDM Hub のビルド番号
アプリケーションサーバー	アプリケーションサーバーのプロパティ
Java	Java のプロパティ
Java オプション	Java のオプション
バージョン履歴	製品バージョン、コンポーネントバージョン、サーバー名、バージョン番号、インストール開始時刻、インストール終了時刻、インストールステータス
ライセンス	各 MDM Hub コンポーネントのライセンスステータス、ライセンスタイプ、ライセンス制限、発効日、有効期限、ライセンスキー
Log4j サーバー	Log4j サーバーのプロパティ
MDM Hub マスタデータベース	MDM Hub マスタデータベースのプロパティ

収集される情報	説明
データベース	データベース URL、データベースバージョン、データベースタイプ、データベースユーザー
オペレーショナル参照ストア	データベースバージョン、ワークフローエンジン名、パッケージ情報、サブジェクト領域名、SSA ポピュレーション情報、リポジトリテーブル情報、Informatica Data Director ユーザー数、システム名

Hub サーバーでの MDM Hub データ収集の有効化

Hub サーバーで MDM Hub データ収集を有効にするには、mdmsupport.properties ファイルを編集します。

- 次の場所に移動します。
 - UNIX の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/support
 - Windows の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\server\support
- テキストエディタで mdmsupport.properties ファイルを開きます。
- phonehome.send_csm_files_to_informatica を 1 に設定します。
- ファイルを保存して閉じます。
- アプリケーションサーバーを再起動します。

プロセスサーバーでの MDM Hub データ収集の有効化

プロセスサーバーで MDM Hub データ収集を有効にするには、mdmsupport.properties ファイルを編集します。

- 次の場所に移動します。
 - UNIX の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/support
 - Windows の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\cleanse\support
- テキストエディタで mdmsupport.properties ファイルを開きます。
- phonehome.send_csm_files_to_informatica を 1 に設定します。
- ファイルを保存して閉じます。
- アプリケーションサーバーを再起動します。

Hub サーバーでの MDM Hub データ収集の無効化

Hub サーバーで MDM Hub データ収集を無効にするには、mdmsupport.properties ファイルを編集します。

1. 次の場所に移動します。
 - UNIX の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/server/support
 - Windows の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\server\support
2. テキストエディタで mdmsupport.properties ファイルを開きます。
3. phonehome.send_csm_files_to_informatica を 0 に設定します。
4. ファイルを保存して閉じます。
5. アプリケーションサーバーを再起動します。

プロセスサーバーでの MDM Hub データ収集の無効化

プロセスサーバーで MDM Hub データ収集を無効にするには、mdmsupport.properties ファイルを編集します。

1. 次の場所に移動します。
 - UNIX の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>/hub/cleanse/support
 - Windows の場合: <MDM Hub installation directory: MDM Hub のインストールディレクトリ>\hub\cleanse\support
2. テキストエディタで mdmsupport.properties ファイルを開きます。
3. phonehome.send_csm_files_to_informatica を 0 に設定します。
4. ファイルを保存して閉じます。
5. アプリケーションサーバーを再起動します。

付録 G

用語集

accept limit: 許容限度

一致の許容度を決定する数値。許容限度は、Informatica によって、ポピュレーション内にその一致目的に応じて定義されています。

active state (records): アクティブな状態（レコード）

ベースオブジェクトレコードまたは相互参照レコードに関する状態。ベースオブジェクトレコードは、少なくとも 1 つの相互参照レコードがアクティブな場合にアクティブになります。相互参照レコードは、アクティブな状態の場合にのみ、統合されたベースオブジェクトにデータを渡します。

アクティブなレコードは、デフォルトで MDM Hub のプロセスの対象に含まれます。これらのレコードは、どの操作にも含めることができます。承認プロセスが必要なレコードの場合、これらのレコードは、すでにそのプロセスを経て承認されています。

Admin source system: 管理ソースシステム

デフォルトのソースシステム。データマネージャツールまたはマージマネージャツールからの手動による信頼のオーバーライドやデータの編集に使用されます。 [source system: ソースシステム \(ページ 697\)](#) を参照してください。

auditable events: 監査可能イベント

内部監査メカニズムによって追跡できる MDM Hub のアクティビティ。MDM Hub は監査情報を監査ログテーブル C_REPOS_AUDIT に格納します。

authentication: 認証

ユーザーが指定した ID がそのユーザーのものであるかどうかを確認するプロセス。Informatica MDM Hub では、ユーザーが指定した資格情報（ユーザー名/パスワードまたはセキュリティペイロード、あるいはその両方の組み合わせ）に基づいてユーザーが認証されます。Informatica MDM Hub では、内部認証メカニズムが用意されているほか、サードパーティの認証プロバイダを使用したユーザー認証もサポートされています。

authorization: 認証

ユーザーが要求した Informatica MDM Hub のリソースにアクセスするために必要な特権を持っているかどうかを確認するプロセス。Informatica MDM Hub では、リソース特権はロールに割り当てられています。ユーザーとユーザーグループはロールに割り当てられます。ユーザーのリソース特権は、ユーザーが割り当てられているロール、およびユーザーが属するユーザーグループに割り当てられているロールで決まります。

automerge: 自動マージ

レコードを自動的にマージするプロセス。マージスタイルのベースオブジェクトでのみ使用されます。一致ルールの結果として、自動マージまたは手動マージを選択できます。自動マージを実行するように一致ルールで指定すると、ベースオブジェクトテーブルの複数のレコードが手動操作なしで自動的に結合されます。

base object: **ベースオブジェクト**

ビジネスに関するエンティティ（顧客やアカウントなど）に関する情報を格納するテーブル。

batch group: **バッチグループ**

1つのコマンドで実行できる個々のバッチジョブ（ステージジョブ、ロードジョブ、一致ジョブなど）の集合。グループ内の各バッチジョブは、他のジョブと順番に実行することも並行して実行することも可能です。

batch job: **バッチジョブ**

バッチモードで処理されるコマンドのシーケンス。これは、コマンドのシーケンスが、バッチファイルというファイルに追加され、単一プロセスとして実行するために実行依頼されたことを意味します。

batch mode: **バッチモード**

バッチジョブを介して MDM Hub とやり取りする方法。バッチジョブは、Hub コンソールで実行できます。また、サードパーティの管理ツールを使用して、バッチジョブをスケジュール設定して実行することもできます。

best version of the truth (BVT): **ベストバージョン オブ トゥルース (BVT)**

ソースレコードのデータのうちの最適なセルと統合されているレコード。

マージスタイルのベースオブジェクトの場合、ベースオブジェクトレコードは BVT レコードで、対応するソースレコードのうちの最も信頼度が高いセルの値と統合して構築されます。

BI vendor: **BI ベンダ**

ビジネスインテリジェンスソフトウェア製品の製造元。

bulk merge: **一括マージ**

[automerge: 自動マージ \(ページ 675\)](#) を参照してください。

business process: **ビジネスプロセス**

ビジネスプロセスとは、組織の目標を達成し、ビジネス機能を実行するワークフローです。個々のビジネスプロセスは、目標を達成するうえで必要なアクティビティを含み、またアクティビティを通して実行するパスを定義します。Multidomain MDM には、事前定義されている Informatica ActiveVOS ビジネスプロセス（ActiveVOS サーバーによって管理）が同梱されています。このプロセスの組織上の目的は、ビジネスマネージャやデータスチュワードなどの権限を持ったスタッフにマスターデータの更新をすべて確認させることです。

business process management (BPM): **ビジネスプロセス管理 (BPM)**

ビジネスプロセス管理は、組織のプロセスの適合理化に焦点を当てています。Informatica MDM は、ビジネスプロセス管理エンジンが組み込まれた状態で提供されます。このエンジンを使用すると、マスターデータの確認と承認のプロセスを自動化できます。

BVT

[best version of the truth \(BVT\): ベストバージョン オブ トゥルース \(BVT\) \(ページ 676\)](#) を参照してください。

cascade delete: **カスケード削除**

親オブジェクトが削除されたときに、子オブジェクトの関連レコードを削除する操作。カスケード削除操作を有効にするには、CASCADE_DELETE_IND パラメータを 1 に設定します。

cascade unmerge: カスケードマージ解除

親ベースオブジェクトのレコードがマージされない場合に行う処理を指定する機能。この機能を有効にすると、親オブジェクトのレコードがマージされない場合、子ベースオブジェクトの関連レコードも MDM Hub によってマージ解除されます。

cell: セル

テーブル内のカラムとレコードが交差する位置。セルにはデータ値または NULL が含まれます。

change list: 変更リスト

ターゲットリポジトリに対する変更のリスト。変更リストにおける **変更**とは、ターゲットリポジトリに対して実行される操作（ベースオブジェクトの追加や一致ルールのプロパティの更新など）のことです。変更リストは、Hub のリポジトリ間の差異のリストと言えます。

cleanse: クレンジング

[data cleansing: データクレンジング \(ページ 679\)](#) を参照してください。

cleanse engine: クレンジングエンジン

Informatica MDM Hub でデータクレンジングを実行するために使用されるサードパーティの製品。

cleanse function: クレンジング関数

ステージジョブで入力データを変更し、それぞれの入力文字列を出力文字列に変換するコード。通常、これらの関数は、データを標準化して一致プロセスを最適化するために使用されます。複数のクレンジング関数を組み合わせることで、複雑なフィルタリングや標準化を実行できます。

cleanse list: クレンジングリスト

クレンジングプロセスで入力文字列の一部を置き換えるルールの論理的なグループ。

column: カラム

テーブル内にある特定のタイプのデータ値のセット（テーブルの各行に 1 つ）。[system column: システムカラム \(ページ 698\)](#)、[user-defined column: ユーザー定義カラム \(ページ 700\)](#) を参照してください。

comparison change list: 比較変更リスト

2 つのリポジトリの内容を比較してターゲットリポジトリに対して行う変更をまとめた変更リスト。比較変更リストは、リポジトリマネージャでデザインオブジェクトを昇格またはインポートするときに使用されます。

complete match tracking: 完全一致追跡

2 つのレコードが中間レコードを経て一致する結果となった、完全なまたは元の一致チェーンの表示。

conditional mapping: 条件マッピング

ランディングテーブルとステージングテーブルの間のカラムのマッピングの 1 つ。このマッピングでは、SQL WHERE 句を使用して、ランディングテーブルからフィルタ条件に一致するレコードだけを選択します。

Configuration workbench: 設定ワークベンチ

各種の MDM Hub オブジェクト（オペレーショナル参照ストア、ユーザー、セキュリティ、メッセージキュー、メタデータの検証など）を設定するためのツールが含まれています。

consolidated record: **統合されたレコード**

[*master record*: マスタレコード \(ページ 687\)](#) を参照してください。

consolidation indicator: **統合インジケータ**

ベースオブジェクトのレコードの統合状態を表します。CONSOLIDATION_IND カラムに格納されます。

consolidation process: **統合プロセス**

重複レコードを 1 つのレコードにマージまたはリンクするプロセス。Informatica MDM Hub の目標は、すべての重複データを特定して取り除き、完全なトレーサビリティを保ちながら、それらをまとめて 1 つの統合されたレコードにマージまたはリンクすることです。

content metadata: **コンテンツメタデータ**

Informatica MDM Hub で処理されたビジネスデータについて記述したデータ。コンテンツメタデータは、相互参照テーブルや履歴テーブルなど、ベースオブジェクトのサポートテーブルに格納されます。コンテンツメタデータを使用すると、ベースオブジェクトのデータのソースやデータの経時的な変化を特定するのに役立ちます。

contiguous effective period: **連続した有効期間**

レコードの有効期間に空白期間がないように、他のレコードバージョンの有効期間と連続するレコードの有効期間。

control table: **制御テーブル**

Informatica MDM Hub によってベースオブジェクトに対して自動的に作成されるオペレーショナル参照ストアのシステムテーブル。制御テーブルは、ロード、マージ、およびマージ解除の各プロセスのサポートに使用されます。Informatica MDM Hub では、ベースオブジェクト内の信頼が有効なそれぞれのカラムについて、対応する制御テーブルでレコード（最終更新日とソースシステムの識別子）を管理します。

creation change list: **作成変更リスト**

リポジトリのコンテンツのエクスポート後に生成される変更リスト。作成変更リストは、リポジトリマネージャでデザインオブジェクトをインポートするために使用されます。

cross-reference table: **相互参照テーブル**

Informatica MDM Hub によってベースオブジェクトに対して自動的に作成されるオペレーショナル参照ストアのシステムテーブル。相互参照テーブルには、ベースオブジェクトの各レコードについて、ソースシステムあたり 0～n 個のレコードが格納されます。このレコードには、ソースシステムのプライマリキーと、ベースオブジェクトテーブル内の各セルに対してソースシステムから提供された最新の値が格納されます。

Data Access Services: **データアクセスサービス**

これらのアプリケーションサーバーレベルの機能により、Informatica MDM Hub で複数のモードのデータアクセスをサポートできるようになり、Informatica MDM Hub の多数のデータサービスを Informatica サービス統合フレームワーク（SIF）を介して公開できます。これは、リアルタイムの同期統合にも非同期統合にも役立ちます。

database: **データベース**

Hub Store 内の整理されたデータの集まり。Informatica MDM Hub では、2 種類のデータベースをサポートしています。マスターデータベースとオペレーショナル参照ストア（ORS）です。

data change event: **データ変更イベント**

ある期間内で有効なデータに対する変更。

data cleansing: **データクレンジング**

データのコンテンツとレイアウトを標準化するプロセス。このプロセスでは、テキスト値を識別可能な要素に分解および解析し、データライブラリと照合して識別可能な値（郵便番号など）を検証し、正しくない値をデータライブラリの正しい値に置き換えます。

Data Manager: **データマネージャ**

すべてのマージ（自動マージを含む）の結果を確認し、必要に応じてデータのコンテンツを修正するためのツール。このツールでは、各ベースオブジェクトレコードのデータリネージを確認できます。また、既存のマージ済みレコードをマージ解除したり、統合された各レコードに関する各種の履歴を表示したりできます。

データマネージャツールでは、レコードの検索、それらの相互参照の表示、レコードのマージ解除、レコードのリンク解除、履歴レコードの表示、新しいレコードの作成、レコードの編集、および信頼の設定のオーバーライドを行うことができます。データマネージャには、定義した検索条件を満たすすべてのレコードが表示されます。

datasource: **データソース**

アプリケーションサーバー環境において、データベースに関する情報を識別する JDBC リソース。この情報には、データベースサーバーの場所、データベースの名前、データベースユーザーの ID とパスワードなどが含まれます。Informatica MDM Hub では、オペレーショナル参照ストアとの通信にこの情報が必要となります。

data steward: **データスチュワード**

データ品質に関する主な責任を担う Informatica MDM Hub ユーザー。データスチュワードは、Informatica MDM Hub に Hub コンソールからアクセスし、Informatica MDM Hub のツールを使用して、Hub ストア内のオブジェクトの設定を行います。

Data Steward workbench: **データスチュワードワークベンチ**

Informatica MDM Hub の UI の一部。統合されたデータや例外処理のキューに追加された一致データを確認するために、データのセマンティクスを理解した、組織内のデータの信頼度を確保する役割を担うデータアナリストまたはデータスチュワードによって使用されます。

データマネージャ、マージマネージャ、および階層マネージャを使用するためのツールが含まれています。

data type: **データ型**

テーブルカラムで使用できる値の特性を定義します。文字、数値、日付、バイナリデータなどがあります。Informatica MDM Hub では、Informatica MDM Hub 実装で使用するデータベースプラットフォームのデータ型にそのままマップできる共通のデータ型のセットを使用しています。

decay curve: **減衰曲線**

信頼の経時的な減衰を視覚的に示したものの。この形状は、設定されている減衰タイプと減衰期間によって決まります。

decay period: **減衰期間**

信頼レベルが最大信頼レベルから最小信頼レベルに減衰するまでの時間（単位は日、週、月、四半期、および年）。

decay type: **減衰タイプ**

減衰期間に信頼レベルがどのように低下するかを示すタイプ。

deleted state (records): **削除済み状態（レコード）**

削除済み状態のレコードは、Hub のデータとして必要がなくなったレコードです。これらのレコードはプロセスで使用されません（ただし、明示的に要求された場合は除く）。レコードの削除は明示的に行う必要があり、削除したレコードは必要に応じてリストアできます。保留状態のレコードを削除すると、そのレコードは完全に削除され、リストアすることはできません。

delta detection: **差分検出**

この機能を有効にすると、MDM Hub のステージプロセスで、新しいレコードと変更されたレコードだけが処理されます。差分検出は、レコード全体を比較するか、日付カラムに基づいて実行できます。

design object: **デザインオブジェクト**

実装のスキーマやその他の構成設定を定義するためのメタデータの一部。デザインオブジェクトには、Informatica MDM Hub オブジェクトの各種のインスタンスが含まれます。例えば、ベースオブジェクトとそのカラム、ランディングテーブルとステージングテーブル、カラム、インデックス、リレーション、マッピング、クレンジング関数、クエリとパッケージ、信頼の設定、検証ルールと一致ルール、Security Access Manager の定義、階層マネージャの定義など、さまざまな設定が含まれます。

distinct mapping: **個別マッピング**

ランディングテーブルとステージングテーブルの間のカラムのマッピングの 1 つ。このマッピングでは、ランディングテーブルから個別にレコードを選択します。単一のランディングテーブルで複数のステージングテーブルにデータを入力し、そのランディングテーブルが正規化されていない場合、個別のマッピングを使うと便利です。例えば、ランディングテーブルに顧客データとアドレスデータの両方が含まれていることがあります。

distinct source system: **個別ソースシステム**

統合されずにベースオブジェクトに挿入されるデータを提供するソースシステム。

distribution: **配布**

調整によって最善データが確立された後に、マスタレコードのデータを他のアプリケーションやデータベースに配布するプロセス。

downgrade: **ダウングレード**

ロードプロセスまたは API（cleansePut および Put）を使用してデータを挿入または更新するときに、検証ルールであるレコードの信頼を一定のパーセント下げる操作。

duplicate: **重複**

特定のカラムのデータ（名前、住所、または組織のデータなど）が同一またはほぼ同一である 1 つ以上のレコード。一致プロセス中に実行される一致ルールにより、2 つのレコードの類似性が確認され、それらを統合プロセスで重複と見なすかどうかが判定されます。

Dynamic Data Masking

クライアントとデータベースの間で動作して、機密情報への不正アクセスを防止するデータセキュリティ製品。Dynamic Data Masking は、データベースに送信された要求をインターセプトし、その要求にデータマスキングルールを適用してから、要求に対する結果をクライアントに返します。

effective period: **有効期間**

レコードが有効な期間です。有効期間は、開始日と終了日によって定義されます。

entity: **エンティティ**

階層マネージャでは、エンティティとは、他のエンティティに関連付けることができるタイプが指定されたオブジェクトです。エンティティの例には、個人、組織、製品、世帯などが含まれます。

entity base object: **エンティティベースオブジェクト**

エンティティベースオブジェクトは、階層マネージャのエンティティに関する情報を格納するためのベースオブジェクトです。

entity type: **エンティティタイプ**

階層マネージャでは、エンティティタイプとは、階層マネージャを使用して関連付け可能なオブジェクトの種類を定義したものです。例えば、個人、組織、製品、世帯などがあります。エンティティタイプが同じエンティティは、すべて同じエンティティベースオブジェクトに格納されます。HM 設定ツールでは、エンティティタイプは、ナビゲーションツリー内のそのタイプが関連付けられたエンティティオブジェクトの下に表示されます。

exact match: **完全一致**

同一のレコードだけを一致させる一致/検索ストラテジ。完全一致を指定した場合、そのベースオブジェクトに対しては完全一致カラムだけを定義できます（完全一致ベースオブジェクトにあいまい一致カラムを含めることはできません）。完全一致/検索ストラテジを使用するベースオブジェクトは、「完全一致ベースオブジェクト」と呼ばれます。

exclusive lock: **排他ロック**

Hub コンソールにおいて、基本となるスキーマを排他的に変更するために必要なロック。排他ロックにより、他の Hub コンソールユーザーによって同時にターゲットデータベースが変更されることを防ぐことができます。排他ロックは、その排他ロックを保持するユーザーが解除する必要があります。別のユーザーがクリアすることはできません。

execution path: **実行パス**

Informatica MDM Hub でバッチグループ全体を実行する際に、それぞれのバッチジョブが実行される順序。実行パスは開始ノードで始まり、終了ノードで終わります。バッチグループツールでは実行シーケンスは検証されません。実行シーケンスは、ユーザーが正しく指定する必要があります。

export process: **エクスポートプロセス**

リポジトリマネージャにおいて、リポジトリ内のメタデータを移植可能な変更リスト XML ファイルにエクスポートするプロセス。エクスポートしたファイルは、デザインオブジェクトを別のリポジトリにインポートする際に使用できます。また、アーカイブ用にソース制御システムに保存しておくこともできます。エクスポートプロセスでは、サポートされているすべてのデザインオブジェクトが変更リスト XML ファイルにコピーされます。

external application user: **外部アプリケーションユーザー**

MDM Hub ハブのデータにサードパーティのアプリケーションを介して間接的にアクセスする MDM ユーザー。ユーザー設定の詳細については、*Multidomain MDM のセキュリティガイド*を参照してください。

external cleanse: **外部クレンジング**

ランディングテーブルに取り込む前にデータをクレンジングするプロセス。外部クレンジングは、通常、抽出-変換-ロード（ETL）ツールなどのデータクレンジングユーティリティを使用して Informatica MDM Hub の外部で実行されます。

external match: **外部一致**

新しいデータ（個別の入力テーブルに格納されるデータ）をあいまい一致ベースオブジェクトの既存のデータと一致させ、一致するかどうかをテストして結果を確認するプロセス。ベースオブジェクトのデータが実際に変更されることはなく、ベースオブジェクトに関連付けられた一致テーブルが変更されることもありません。

extract-transform-load (ETL) tool: **抽出-変換-ロード（ETL）ツール**

ソースシステムからデータを抽出し、データを変換して適切な状態にし（ルールやルックアップテーブルなどの機能を使用）、そのデータをターゲットデータベースにロードする（書き込む）ソフトウェアツール（Informatica MDM Hub の外部のツール）。Informatica MDM Hub 実装では、ソースシステムからデータを抽出してランディングテーブルに取り込むために ETL ツールを使用します。

foreign key: **外部キー**

リレーショナルデータベースにおいて、別のテーブル（まれに同じテーブルの場合もある）のプライマリキー値に対応する値が格納されるカラム（またはカラムのセット）。外部キーは、他のテーブルへのポインタとして機能します。例えば、Employee テーブルの Department_Number カラムは、Department テーブルのプライマリキーを参照する外部キーです。

function: **関数**

[*cleanse function: クレンジング関数\(ページ 677\)*](#)を参照してください。

fuzzy match: **あいまい一致**

確率的な手法で一致を判定する一致/検索ストラテジ。このストラテジでは、スペルの差異や誤りなど、一致するレコードを同一でなくする差異が考慮されます。あいまい一致を選択した場合、ベースオブジェクトに特殊なカラム（あいまい一致キー）が追加されます。あいまい一致/検索ストラテジを使用するベースオブジェクトのことを、あいまい一致ベースオブジェクトと呼びます。あいまい一致を使用する場合は、ポピュレーションを選択する必要があります。

fuzzy match key: **あいまい一致キー**

一致カラムであいまい一致/検索ストラテジを使用する場合にスキーママネージャによって追加される、ベースオブジェクトの特別なカラム。このカラムは、検索や一致の実行時にこのベースオブジェクトの一致候補を生成するために使用されるプライマリフィールドです。どのあいまい一致ベースオブジェクトにも、あいまい一致キーが必ず 1 つだけ含まれます。

geocode: **ジオコード**

地理的な座標（緯度、経度、および標高。標高は省略可能）に基づくロケーションアドレス。近接検索を行うにはジオコードが必要です。

global business identifier (GBID): **グローバルビジネス識別子 (GBID)**

ビジネスニーズに基づいてレコードを一意かつグローバルに識別するための共通の識別子（キー値）が格納されるカラム。例えば、次のようなものが含まれます。

- Informatica MDM Hub の外部のアプリケーション（ERP システムや CRM システムなど）によって定義された識別子。
- 外部の組織で定義されている識別子（AMA 番号や DEA 番号などの業界固有のコードなど）あるいは、政府が発行した識別子（社会保障番号、税金 ID 番号、運転免許証番号など）。

hard delete: **物理削除**

ベースオブジェクトレコードや相互参照レコードをデータベースから物理的に削除すること。

Hierarchies Tool: **階層ツール**

Informatica MDM Hub 管理者は、設計時に階層ツール（以前の「階層マネージャ設定ツール」）を使用して、階層マネージャでデータのリレーションを表示および操作するために必要な構造を設定します。階層ツールを使用して、Informatica MDM Hub 実装で使用する階層マネージャのコンポーネント（エンティティタイプ、階層、リレーションタイプ、パッケージ、およびプロファイル）を定義します。階層ツールには、モデルワークベンチからアクセスできます。

hierarchy: **階層**

階層マネージャで、リレーションタイプをまとめたもの。これらのリレーションタイプは、階層のエンティティの位置に基づいてランク付けされるわけではなく、相互に関連するとも限りません。単に分類や識別がしやすいようにグループ分けされたリレーションタイプです。

Hierarchy Manager: **階層マネージャ**

階層マネージャにより、MDM Hub で管理されているレコードに関連付けられた階層データをユーザーが管理できるようになります。詳細については、『*Multidomain MDM 設定ガイド*』を参照してください。

hierarchy type: **階層タイプ**

階層マネージャで、階層の論理的な分類。階層タイプは、特定のリレーションが含まれる（該当する）階層の一般的な分類です。 [hierarchy: 階層 \(ページ 683\)](#) を参照してください。

history table: **履歴テーブル**

関連するテーブルへの変更に関する履歴情報を格納するオペレーショナル参照ストア内のテーブルのタイプ。履歴テーブルには、詳細な変更追跡オプションが用意されています。例えば、マージとマージ解除の履歴、事前クレンジング済みデータの履歴、ベースオブジェクトの履歴、相互参照の履歴などがあります。

HM package: **HM パッケージ**

階層マネージャパッケージは、MDM パッケージのサブセットであり、階層マネージャに必要なメタデータを格納します。

hotspot: **ホットスポット**

ビジネスデータにおいて、一致過多のデータ（共通の一致が多いデータ）を表すレコードのグループ。

Hub Console: **Hub コンソール**

管理者およびデータスチュワード向けの一連のツールで構成される Informatica MDM Hub のユーザーインタフェース。それぞれのツールで、特定のアクション、または関連する一連のアクションを実行することができ

ます。例えば、データモデルの構築、バッチジョブの実行、データフローの設定、Informatica MDM Hub のリソースに対する外部アプリケーションからのアクセスの設定など、システムの設定や操作に関するタスクを実行できます。

hub object: Hub **オブジェクト**

ビジネスエンティティに関する情報を格納する、Hub で定義されている各種のオブジェクトの総称。例えば、ベースオブジェクト、相互参照テーブルなど、レポートのメトリックに関連付けることができる Hub のすべてのオブジェクトが含まれます。

Hub Server

共通のコアサービス（アクセス、セキュリティ、セッションの管理など）に使用される中間層（アプリケーションサーバー）のランタイムコンポーネント。

Hub Store

Informatica MDM Hub 実装で、マスターデータベースと 1 つ以上のオペレーショナル参照ストア（ORS）データベースを格納するデータベース。

immutable source: **不変ソース**

ベースオブジェクトに、常に最善の、最終的なデータを提供するデータソース。不変ソースのレコードは一意として受け入れられ、不変ソースのレコードが完全に統合された後は、マージの実行時にも変更されなくなります。不変ソースは個別システムでもあります。不変ソースシステムからのソースレコードでは、ロードおよび PUT の統合インジケータが常に 1（統合されたレコード）になります。

implementer: **実装者**

組織の要件に基づく Informatica MDM Hub の設計、開発、テスト、およびデプロイに関する主な責任を担う Informatica MDM Hub ユーザー。デザインオブジェクトの作成、スキーマの構築、一致ルールの定義、パフォーマンスのチューニングなどを初めとするさまざまなタスクを担当します。

import process: **インポートプロセス**

リポジトリマネージャで、ライブラリまたは変更リストのデザインオブジェクトをリポジトリに追加するプロセス。デザインオブジェクトはターゲットリポジトリにまだ存在していません。

incremental load: **増分ロード**

ベースオブジェクトの初期データロード後に実行されるロードプロセス。新しいデータと更新されたデータだけがベースオブジェクトにロードされることから、増分ロードと呼ばれています。重複データは無視されます。

indirect match: **間接一致**

[*transitive match*: 遷移一致 \(ページ 699\)](#)を参照してください。

initial data load: **初期データロード**

空のベースオブジェクトへの最初のデータのロード。初期データロードでは、ステージングテーブルのすべてのレコードが新しいレコードとしてベースオブジェクトに挿入されます。

internal cleanse: **内部クレンジング**

ステージプロセスで、ランディングテーブルのデータが適切なステージングテーブルにコピーされるときに実行される、データをクレンジングするプロセス。内部クレンジングは、設定されたクレンジング関数を使用し

て Informatica MDM Hub の内部で実行されます。クレンジング関数の実行は、プロセスサーバーが、サポートされているクレンジングエンジンと連携して行います。

job execution log: **ジョブ実行ログ**

バッチビューアツールおよびバッチグループツールで、ジョブの完了ステータスと関連メッセージ（成功、失敗、警告など）を表示するログ。

key match job: **キー一致ジョブ**

複数のソースで同じプライマリーキーが使用されている場合に、それらのソースのレコードを照合する Informatica MDM Hub のバッチジョブ。キー照合ジョブでは、新しいレコードを他の新しいレコードおよび既存のレコードと比較し、プライマリーキー照合ルールで定義されたソースレコードのキーの比較に基づいて一致候補を特定します。プライマリーキー照合ルールが定義された後、キー照合バッチジョブを実行します。

key type: **キータイプ**

一致キーに関する重要な特性を定めたもの。この情報に基づいて、Informatica MDM Hub でキーが正しく生成され、検索の効率が向上します。Informatica MDM Hub に用意されている一致キータイプには、Person_Name、Organization_Name、および Address_Part1 があります。

key width: **キー幅**

一致処理の際の検索の速度、返される一致候補の数、およびキーによるディスク領域の使用量を決定する設定。キー幅は、[標準]、[拡張]、[限定]、[優先] から選択できます。キー幅はあいまい一致オブジェクトにのみ適用されます。

landing table: **ランディングテーブル**

Informatica MDM Hub で処理されるデータをソースシステムから PUT するテーブル。

land process: **ランディングプロセス**

ソースシステムからランディングテーブルにデータを取り込むプロセス。

lineage: **リネージ**

Hub ストア内の統合されたレコードにデータを提供したシステムおよびシステムのレコード。

linear decay: **線形減衰**

信頼レベルが最大信頼度から最小信頼度まで直線的に減少すること。

linear unmerge: **線形マージ解除**

ベースオブジェクトレコードをマージ解除し、既存のマージツリー構造から削除すること。マージツリー構造から削除されるのは、マージ解除したそのベースオブジェクトレコードだけです。マージツリーのそれより下位にあるベースオブジェクトレコードは、元のマージツリーにすべてそのまま残ります。

load insert: **ロードの挿入**

ターゲットベースオブジェクトにレコードを挿入すること。ロードプロセス時、ステージングテーブルのレコードがターゲットテーブルにまだ存在しない場合、ターゲットテーブルにレコードが挿入されます。

load process: **ロードプロセス**

ステージングテーブルのデータを Hub Store 内の対応するベースオブジェクトにロードするプロセス。新しいデータが Hub Store 内の既存のデータと重複する場合は、信頼の設定と検証ルールを使用して、どちらの値の

信頼度が高いかが判定されます。 [trust: 信頼 \(ページ 699\)](#)、 [validation rule: 検証ルール \(ページ 701\)](#)、 [load insert: ロードの挿入 \(ページ 685\)](#)、 [load update: ロードの更新 \(ページ 686\)](#) を参照してください。

load update: **ロードの更新**

ターゲットベースオブジェクトにレコードを挿入すること。ロードプロセス時、ステージングテーブルのレコードがターゲットテーブルにまだ存在しない場合、ターゲットテーブルにレコードが挿入されます。

lock: **ロック**

[書き込みロック \(ページ 703\)](#)、 [exclusive lock: 排他ロック \(ページ 681\)](#) を参照してください。

lookup: **ルックアップ**

ロードジョブで親テーブルからデータの値を取得するプロセス。MDM Hub では、ベースオブジェクトに関連付けられているステージングテーブルを設定する際に、ステージングテーブル（子テーブル）の外部キーカラムが親テーブルのプライマリキーに関連付けられていれば、その親テーブルからデータを取得するようにルックアップを設定することができます。

manual merge: **手動マージ**

レコードを手動でマージするプロセス。一致ルールの結果として、自動マージまたは手動マージを選択できます。手動マージを実行するように指定された一致ルールは、データスチュワードによる確認対象として十分な類似点があるが、システムで自動的にマージできるほどではないレコードを特定します。

manual unmerge: **手動マージ解除**

レコードを手動でマージ解除するプロセス。

mapping: **マッピング**

ソースデータに対して適用される一連の変換を定義したもの。マッピングは、ステージプロセスで（または SiperianClient CleansePut API 要求を使用して）、ランディングテーブルからステージングテーブルにデータを転送するときに使用されます。マッピングでは、ランディングテーブル内のソースカラム、データを取り込むステージングテーブル内のターゲットカラム、およびデータのクレンジングに使用する中間のクレンジング関数（ある場合）を指定します。 [conditional mapping: 条件マッピング \(ページ 677\)](#)、 [distinct mapping: 個別マッピング \(ページ 680\)](#) を参照してください。

mapplet: **マップレット**

Mapplet Designer を使用して作成できるトランスフォーメーションのセット。複数のマッピングでロジックを再利用する場合はマップレットを作成します。

master data: **マスターデータ**

企業のビジネスに必須と見なされるエンティティのうち、複数のシステムまたはビジネスプロセスで使用する必要がある共通のコアエンティティ（およびその属性と値）の集まり。マスターデータの例には、顧客、製品、従業員、サプライヤ、場所のデータなどがあります。

Master Database: **マスターデータベース**

Informatica MDM Hub 環境の構成設定を格納するデータベース。これには、ユーザーアカウント、セキュリティ構成、ORS レジストリ、メッセージキューの設定などが含まれます。マスターデータベースは、Informatica MDM Hub 環境ごとに 1 つだけ含めることができます。マスターデータベースのデフォルトの名前は CMX_SYSTEM です。 [オペレーショナル参照ストア \(ORS\) \(ページ 701\)](#) も参照してください。

Master Data Management: **マスターデータ管理**

エンタープライズ用のレコードのシステムとしてマスターデータが作成および管理される制御されたプロセス。MDM は、マスターデータが正しいこと、一貫性があること、および完全であることを確実に検証するために実装されます。また、必要に応じて、内部または外部のビジネスプロセス、アプリケーション、またはユーザーが使用できるようにそれらのデータが配布されます。

master record: **マスタレコード**

特定のエンティティ（特定の組織や個人など）の「最善データ」を表すベースオブジェクト内の単一のレコード。マスタレコードは、エンティティの完全に統合されたデータを表します。

match: **一致**

2つのレコードの特定のカラムに同一または類似の値があるため、2つのレコードを自動的にマージすべきか、手動マージの対象とすべきかを判定するプロセス。

match candidate: **一致候補**

一致する可能性があるベースオブジェクト内のレコード。あいまい一致ベースオブジェクトのみに適用されます。

match column: **一致カラム**

比較のために一致ルールで使用されるカラム。各一致カラムは、ベースオブジェクトの1つ以上のカラムに基づいています。

match column rule: **一致カラムルール**

一致カラムとして定義したカラム（姓、名、住所1、住所2など）の値に基づいてレコードを一致させるために使用される一致ルール。

match key: **一致キー**

ベースオブジェクトのあいまい一致キーカラムのデータを表すエンコードされた文字列。一致キーは、関連する差異の一致キーの値が同じになるように名前や住所に含まれる単語と数字の組み合わせから構築された、圧縮およびエンコードされた固定長の値で、関連性のある類似データは同じ一致キー値となります。一致キーは一致トークンの一部です。一致トークンは、トークン化プロセスで生成されて一致キーテーブルに格納され、一致プロセスで一致する候補を特定するために使用されます。

match key table: **一致キーテーブル**

トークン化プロセスで生成された一致トークン（一致キーとエンコードされていない RAW データの組み合わせ）を格納するシステムテーブル。このデータは、一致プロセスで一致する候補を特定するために使用されます。重複するレコードを特定するために定義された一致ルールに従って一致キーが比較されます。

match list: **一致リスト**

カスタムの標準化リストを定義したもの。関数には、特殊なクレンジング機能（住所の検証や住所の分解など）を利用するための事前に定義された関数を使用します。

match process: **一致プロセス**

2つのレコードを比較して類似点を確認するプロセス。2つのレコードが相互に重複する可能性を示す十分な類似点が見つかったら、Informatica MDM Hub によってそれらのレコードにマージのフラグが設定されます。

match purpose: 一致目的

あいまい一致ベースオブジェクトについて、一致ルールの主要な目的を定義したもの。例えば、人物の一致を特定するときに、2つのレコードが同じ人物のものかどうかを判別するための重要な要素が住所である場合は、「住居」という名前の一致目的を使用します。各一致目的には、2つのレコードの比較方法について、その目的に合った最適な方法に関する情報が含まれています。Informatica MDM Hub は、選択された一致目的に基づいて、一致するレコードを特定するための一致ルールを適用します。ルールの動作は、選択した目的によって異なります。

match rule: 一致ルール

レコードが重複する可能性があるかどうかを Informatica MDM Hub で判定する際の基準を定義したもの。一致ルールに一致カラムが組み合わされて、2つのレコードがマージの対象として十分に類似していると見なす条件が決定されます。各一致ルールは、類似点を調べる必要がある一致カラムの組み合わせを指定します。

match rule set: 一致ルールセット

一致ルールの論理的な集まり。ユーザーは、一致プロセスのさまざまな段階で異なるルールのセットを実行することができます。一致ルールセットには、検索ストラテジを決定する検索レベル、および任意の数の自動と手動の一致ルールが含まれます。また、必要に応じて、一致プロセスの対象に含めるレコードと除外するレコードを選択するためのフィルタも含まれます。一致ルールセットは、一致カラムルールの実行に使用されますが、プライマリキーの一致ルールの実行には使用されません。

match search strategy: 一致検索ストラテジ

使用する一致の信頼度。必要なパフォーマンスも考慮して指定します。あいまいまたは完全を指定できます。完全一致/検索ストラテジの方が処理は高速ですが、完全一致では、データに誤りがあると一致が見落とされる可能性があります。[fuzzy match: あいまい一致 \(ページ 682\)](#)、[exact match: 完全一致 \(ページ 681\)](#)、[match process: 一致プロセス \(ページ 687\)](#)を参照してください。

match search strategy: 一致検索ストラテジ

使用する一致の信頼度。必要なパフォーマンスも考慮して指定します。あいまいまたは完全を指定できます。完全一致/検索ストラテジの方が処理は高速ですが、完全一致では、データに誤りがあると一致が見落とされる可能性があります。[fuzzy match: あいまい一致 \(ページ 682\)](#)、[exact match: 完全一致 \(ページ 681\)](#)、[match process: 一致プロセス \(ページ 687\)](#)を参照してください。

match subtype: 一致サブタイプ

顧客、ベンダ、およびパートナーのレコードを含む Organization ベースオブジェクトなど、さまざまなタイプのデータを含むベースオブジェクトで使用されます。一致サブタイプを使用すると、同じベースオブジェクト内の特定のタイプのデータに一致ルールを適用することができます。各一致ルールに対し、その一致ルールで無視するレコードをフィルタで除外する「サブタイプ」カラムとして機能する、完全一致カラムを指定します。

match table: 一致テーブル

ベースオブジェクトに関連付けられるシステムテーブルの1つ。このテーブルは、一致プロセスをサポートします。ベースオブジェクトに対する一致ジョブの実行時、関連付けられている一致テーブルに、一致するレコードの各ペアの ROWID_OBJECT 値、一致を特定した一致ルールの識別子、および自動マージインジケータが入力されます。

match token: 一致トークン

ベースオブジェクトの一致カラムのエンコードされた値（一致キー）とエンコードされていない値（RAW）の両方を表す文字列。一致トークンは、トークン化プロセスで生成されて一致キーテーブルに格納され、一致プロセスで一致する候補を特定するために使用されます。

match type: **一致タイプ**

各一致カラムには、一致比較用に一致カラムのトークン化方法を決定する一致タイプがあります。

maximum trust: **最大信頼度**

データ値が変更された場合の信頼レベル。例えば、ソースシステム A で電話番号フィールドが 555-1234 から 555-4321 に変更された場合、新しい値では、電話番号フィールドに対してシステム A の最大信頼度レベルが与えられます。最大信頼度レベルを高く設定することによって、ソースシステムにおける変更が常にベースオブジェクトに適用されるようにすることができます。

Merge Manager: **マージマネージャ**

手動マージ用にキューに追加されるレコードに対して確認や処理を行うために使用するツール。

merge process: **マージプロセス**

指定した一致カラムに同じ値（または極めて類似した値）を持つために、ベースオブジェクトテーブルの複数のレコードを結合するプロセス。[consolidation process: 統合プロセス \(ページ 678\)](#)、[automerge: 自動マージ \(ページ 675\)](#)、[manual merge: 手動マージ \(ページ 686\)](#)を参照してください。

merge-style base object: **マージスタイルのベースオブジェクト**

Informatica MDM Hub の一致機能やマージ機能で使用するベースオブジェクトのタイプ。

message: **メッセージ**

Informatica MDM Hub では、Java Message Service (JMS) メッセージを示します。メッセージキューサーバーは以下の 2 種類の JMS メッセージを処理します。

- 受信メッセージ。Informatica MDM Hub サービスの起動の非同期処理に使用されます。
- 送信メッセージ。JMS を介してソースシステムまたはその他のシステムにデータ変更を配信する通信チャネルを提供します。

message queue: **メッセージキュー**

データのあるプロセスから別のプロセス（例えば、Informatica MDM Hub から外部アプリケーション）に転送するメカニズム。

message queue rule: **メッセージキュールール**

ベースオブジェクトイベントを識別し、影響されるレコードを更新するために内部システムに転送するメカニズム。メッセージキュールールは、更新、マージ、および一意としてとして受け取られるレコードに対してサポートされています。

message queue server: **メッセージキューサーバー**

Informatica MDM Hub では、Java Message Service (JMS) サーバー。アプリケーションサーバー環境で定義され、Informatica MDM Hub が受信および送信 JMS メッセージの管理に使用します。

message trigger: **メッセージトリガ**

Informatica MDM Hub 内で特定のアクションが発生した場合に起動されるルール。ルールが定義されているアクションが実行されると、送信メッセージキューに JMS メッセージが配置されます。メッセージトリガは、（オブジェクトへのアクションについて）メッセージを生成してキューに配置する状況を識別します。

metadata: **メタデータ**

他のデータを記述するために使用されるデータ。Informatica MDM Hub では、Informatica MDM Hub 実装で
使用されるスキーマ（データモデル）を、メタデータや関連する構成設定を使用して記述します。

metadata validation: **メタデータの検証**

[validation process: 検証プロセス \(ページ 701\)](#) を参照してください。

minimum trust: **最小信頼度**

データ値が（減衰期間の経過後に）「古く」なったときに移行する信頼レベル。この値は最大信頼度以下である
必要があります。最大信頼度と最小信頼度が同じ場合、減衰曲線は平らになり、減衰期間と減衰タイプは影
響しなくなります。 [decay period: 減衰期間 \(ページ 679\)](#) も参照してください。

Model workbench: **モデルワークベンチ**

Informatica MDM Hub の UI の一部。実装者がデプロイメントの実行中にソリューションの設定に使用したり、
データアーキテクトが変化するビジネスニーズに応じてさまざまなタイプのメタデータおよびルールを継
続的に設定する際に使用します。

クエリグループの作成、パッケージおよびその他のスキーマオブジェクトの定義、および現在のスキーマの表
示用のツールが含まれます。

noncontiguous effective period: **連続しない有効期間**

他のレコードバージョンの有効期間と連続していないレコードバージョン（レコードの有効期間に中断がある
ものなど）の有効期間。

non-contributing cross reference: **非貢献相互参照**

ベースオブジェクトレコードの BVT（最善データ）に提供しない相互参照（XREF）レコード。したがって、
相互参照レコードの値はベースオブジェクトレコードに表示されません。状態が有効なレコードにのみ該当す
ることに注意してください。

non-equal matching: **非同等一致**

一致ルールを設定するとき、1つのカラム内の同等の値が相互に一致することを防ぎます。非同等一致は完全
一致カラムにのみ適用されます。

overmatching: **一致過多**

関係のない一致を含め、結果上の一致が多すぎる状態。あいまい一致ベースオブジェクトに関してのみ発生し
ます。一致を設定する際の目標は、データに関する一致を適切な個数で見つけることです。

package: **パッケージ**

パッケージとは、Informatica MDM Hub の 1 つ以上の基本テーブルの公開されたビューです。パッケージは、
それらのテーブル内のカラムのサブセットを、そのテーブルに結合された他のテーブルと一緒に示します。パ
ッケージはクエリに基づきます。基になるクエリで、テーブルまたは別のパッケージからレコードのサブセッ
トを選択することができます。

password policy: **パスワードポリシー**

Informatica MDM Hub ユーザーアカウントのパスワードの特性（パスワード長、有効期限、ログイン設定、パ
スワードの再利用およびその他の要件）を指定します。Informatica MDM Hub 実装のすべてのユーザーアカ
ウントに対するグローバルパスワードポリシーを定義して、個別のユーザーについてこれらの設定を上書きす
ることができます。

pending state (records): **保留状態（レコード）**

保留レコードは、Hub での一般的な使用についてまだ承認されていないレコードです。保留レコードにはほとんどの操作を実行可能ですが、操作では保留レコードを明示的に指定する必要があります。レコードを承認プロセスに渡す必要がある場合、これらのレコードはまだ承認されておらず、承認プロセスの途中にあります。

period end date: **期間の終了日**

レコードバージョンの有効期間が終わる日。

period start date: **期間の開始日**

レコードバージョンの有効期間が始まる日。

policy decision points (PDPs): **ポリシー決定ポイント（PDP）**

ユーザー ID を認証し、MDM Hub リソースへのユーザーアクセスを許可する特定のセキュリティチェックポイント。

policy enforcement points (PEPs): **ポリシー適用ポイント（PEP）**

実行時にセキュリティポリシーを認証要求および承認要求に適用する、特定のセキュリティチェックポイント。

population: **ポピュレーション**

一致させるレコードのデータに関する特性を定義します。Informatica MDM Hub には、デフォルトで US（米国）ポピュレーションが付属していますが、Informatica では各国の標準ポピュレーションを提供しています。ポピュレーションは、名前、アドレス、およびその他の識別データに存在する必然的な差異およびエラーを明確にします。Informatica MDM Hub での、一致トークンの作成方法を指定し、また、一致させるデータのポピュレーションに対する検索ストラテジおよび一致目的の挙動を指定します。あいまい一致/検索ストラテジでのみ使用されます。

primary key: **プライマリキー**

リレーショナルデータベーステーブルで、レコードを一意に識別する値を持つカラム（またはカラムセット）。例えば、部門番号カラムは、部門テーブルのプライマリキーです。

primary key match rule: **プライマリキーの一致ルール**

同じプライマリキーを使用する 2 つのシステムのレコードを一致させるために使用される一致ルール。 [match column rule: 一致カラムルール \(ページ 687\)](#) も参照してください。

private resource: **非公開リソース**

ロールツールに公開されていない Informatica MDM Hub リソース。Services Integration Framework（SIF）の操作からアクセスできません。Hub コンソールで新しいリソース（新しいベースオブジェクトなど）を追加すると、デフォルトで非公開リソースに設定されます。

privilege: 特権

MDM Hub のリソースにユーザーがアクセスする権限。MDM Hub の内部承認では、各ロールに以下のいずれかの特権が割り当てられます。

特権	許可する操作
READ	データの表示。
CREATE	Hub Store 内でのデータレコードの作成。
UPDATE	Hub Store 内のデータレコードの更新。
MERGE	データのマージおよびマージ解除。
EXECUTE	クレンジング関数およびバッチグループの実行。
DELETE	Hub Store からのデータレコードの削除。

外部アプリケーションユーザーが MDM Hub リソースに対して持つアクセス権は、特権によって決まります。例えば、特定のパッケージおよびパッケージカラムに、読み取り、作成、更新、およびマージ特権を設定できます。設定は Hub コンソールの使用にある程度影響しますが、これらの特権は Hub コンソールを使用する場合に実行されません。

process: プロセス

[business process: ビジネスプロセス \(ページ 676\)](#) を参照してください。

profile: プロファイル

階層マネージャでは、HM ユーザーが表示、編集、または追加できるフィールドおよびレコードを表します。例えば、2 つのプロファイルを用意し、一方ではすべてのエンティティおよびリレーションに対する完全な読み取り/書き込みアクセスを許可し、もう一方は読み取り専用にする（追加や編集の操作を許可しない）ことも可能です。

promotion process: 昇格プロセス

コンテキストに応じて、以下のものを示します。

- **リポジトリマネージャ:** デザインオブジェクトの変更内容をリポジトリ間でコピーするプロセス。昇格は、リポジトリ間の差分変更のコピーに使用されます。
- **状態管理:** Informatica MDM Hub の個々のレコードのシステム状態を保留状態からアクティブ状態に変更するプロセス。

provider: プロバイダ

[security provider: セキュリティプロバイダ \(ページ 696\)](#) を参照してください。

provider property: プロバイダのプロパティ

セキュリティプロバイダによって提供されるサービスにアクセスするために必要な名前と値のペア。

proximity search: 近接検索

ジオコード半径を指定してその半径内のレコードを照合するプロセス。近接検索は、カラムに緯度、経度、および標高（標高は省略可能）が表示されるベースオブジェクトに対して実行されます。

publish: **パブリッシュ**

他のアプリケーション、データベースなどに配布するために Informatica MDM Hub メッセージをメッセージキューに送信するプロセス。

query: **クエリ**

データを Hub ストアから取得する要求。Informatica MDM Hub では、管理者が条件を指定してデータを取得することができます。クエリは、選択したカラムを返すように設定できます。WHERE 句で結果セットをフィルタしたり、複雑なクエリ構文（GROUP BY、ORDER BY、HAVING 句など）や集計関数（SUM、COUNT、AVG など）も使用できます。

query group: **クエリグループ**

クエリの論理グループ。クエリグループは、クエリを整理するための単なるメカニズムです。[query: クエリ \(ページ 693\)](#)を参照してください。

raw table: **RAW テーブル**

ランディングテーブルからデータをアーカイブするテーブル。

real-time mode: **リアルタイムモード**

サードパーティのアプリケーションを使用して Informatica MDM Hub を操作する方法。Services Integration Framework (SIF) インタフェースから Informatica MDM Hub 操作を呼び出します。SIF では、レコードの読み取り、クレンジング、一致、挿入、および更新などのさまざまなサービスに対する操作を実行できます。[batch mode: バッチモード \(ページ 676\)](#)、[Services Integration Framework \(SIF\) \(ページ 696\)](#)も参照してください。

reconciliation: **調整**

指定されたエンティティに関して、Informatica MDM Hub は 1 つ以上のソースシステムからデータを取得し、「複数の適正データ」を調整して、最善データがそのエンティティのマスターレコードに渡されるようにします。調整では、一致のプロセスを最適化するための事前のデータクレンジング、およびベースオブジェクトに対するレコードの統合が行われます。[distribution: 配布 \(ページ 680\)](#)を参照してください。

record: **レコード**

オブジェクトのインスタンスを表す、テーブル内の行。例えば、アドレステーブルでは 1 つのレコードに 1 つのアドレスが含まれます。[source record: ソースレコード \(ページ 697\)](#)、[consolidated record: 統合されたレコード \(ページ 678\)](#)も参照してください。

record version: **レコードバージョン**

特定の期間有効であるレコードのバージョン。各レコードには複数のバージョンを作成でき、各バージョンにはそれぞれ異なる有効期間を設定できます。

referential integrity: **参照整合性**

設定済みの外部キーリレーションに基づく、全体的なテーブル間への親子リレーションルールの適用。

regular expression: **正規表現**

テキストデータを一般的に使用される構文規則および記号パターンに従って一致させ、操作するために使用される計算式。Informatica MDM Hub では、正規表現関数によって、クレンジング操作に正規表現を使用することができます。正規表現の構文やパターンなどの詳細については、java.util.regex.Pattern の Javadoc を参照してください。

relationship: **リレーション**

階層マネージャで、2つの特定のエンティティ間の関係を表します。階層マネージャのリレーションは、リレーションタイプ、階層タイプ、リレーションの属性、およびリレーションがアクティブになる日付を指定することによって定義します。[relationship type: リレーションタイプ\(ページ 694\)](#)、[hierarchy: 階層\(ページ 683\)](#)を参照してください。

relationship base object: **リレーションベースオブジェクト**

リレーションベースオブジェクトは、階層マネージャのリレーションについての情報を格納するために使用されるベースオブジェクトです。

relationship type: **リレーションタイプ**

リレーションの一般的なクラスを示します。リレーションタイプは以下の項目を定義します。

- このタイプのリレーションに含めることができるエンティティのタイプ
- リレーションの方向（ある場合）
- Hub コンソールにリレーションを表示する方法

[relationship: リレーション\(ページ 694\)](#)、[hierarchy: 階層\(ページ 683\)](#)を参照してください。

request: **要求**

Informatica MDM Hub 要求 (API)。サービス統合フレームワーク (SIF)、要求/応答 API モデルを使用した、外部アプリケーションから特定の Informatica MDM Hub 機能へのアクセスを可能にします。

resource: **リソース**

Informatica MDM Hub 実装によって使用される Informatica MDM Hub オブジェクト。特定のリソースをセキュアリソースとして設定することができます (ベースオブジェクト、マッピング、パッケージ、リモートパッケージ、クレンジング関数、HM プロファイル、監査テーブル、ユーザーテーブル)。さらに、コンテンツメタデータ、一致ルールセット、メタデータ、バッチグループ、監査テーブル、およびユーザーテーブルなどの SIF 操作によってアクセス可能なセキュアリソースを、管理者が設定することができます。

resource group: **リソースグループ**

特権の割り当てを簡素化するセキュアリソースの集まり。リソースグループを使用すると、リソースグループの簡単なロールへの割り当てなど、特権を複数のリソースに一度に割り当てることができます。[resource: リソース\(ページ 694\)](#)、[privilege: 特権\(ページ 692\)](#)を参照してください。

Resource Kit

Informatica MDM Hub Resource Kit は、Informatica MDM Hub に実装して機能を拡張する例を提供する、ユーティリティ、例、およびライブラリのセットです。

RISL decay: RISL **減衰**

RISL (Rapid Initial Slow Later) 減衰。ほとんどの減衰が減衰期間の最初に発生します。信頼レベルは凹の逆放物曲線を描きます。ソースシステムにこの減衰タイプがある場合、システムからの新しい値は信頼される可能性があります、その後上書きされるようになっていきます。

row: **行**

[record: レコード\(ページ 693\)](#)を参照してください。

rule: **ルール**

有効な動作を定義する文、または計算や比較を定義する文。MDM Hub はルールを管理し、マスタデータにルールを適用します。 [match rule: 一致ルール \(ページ 688\)](#)、 [message queue rule: メッセージキュールール \(ページ 689\)](#)、 [state transition rules: 状態遷移ルール \(ページ 697\)](#)、 [timeline rules: タイムラインのルール \(ページ 699\)](#)、および [validation rule: 検証ルール \(ページ 701\)](#) も参照してください。

rule set: **ルールセット**

[match rule set: 一致ルールセット \(ページ 688\)](#) を参照してください。

rule set filtering: **ルールセットのフィルタリング**

一致ルールセットで処理するレコードを除外する機能。例えば、さまざまなタイプの組織（顧客、ベンダ、見込み客、パートナーなど）を含む組織ベースオブジェクトがある場合、ベンダのみを選択的に処理する一致ルールセットを定義できます。

schema: **スキーマ**

お客様の Informatica MDM Hub 実装に使用されるデータモデル。Informatica MDM Hub には、必須のスキーマはありません。スキーマはソースシステムから独立しています。

Schema Manager: **スキーママネージャ**

スキーママネージャは、Hub コンソールが、スキーマの定義や、ステージングテーブルおよびランディングテーブルの定義に使用する設計時コンポーネントです。スキーママネージャは、一致およびマージ、検証、メッセージキューに対するルールの定義でも使用します。

Schema Viewer Tool: **スキーマビューアツール**

スキーマビューアツールは、Informatica MDM Hub 実装のスキーマ設定を可視化する Hub コンソールの設計時コンポーネントです。スキーマビューアは複雑なスキーマを可視化する際に特に便利です。

secure resource: **セキュアリソース**

ロールツールに公開されている、保護されている Informatica MDM Hub リソース。特定の特権を持つロールに追加可能なリソースです。ユーザーアカウントを特定のロールに割り当てると、そのユーザーアカウントは、そのロールに関連付けられている特権に従って、SIF を使用してセキュアリソースにアクセスすることができます。外部アプリケーションが SIF 操作を使用して Informatica MDM Hub リソースにアクセスできるようにするには、リソースをセキュアに設定する必要があります。すべての Informatica MDM Hub リソースはデフォルトで非公開になるため、セキュアリソースはリソースの追加後に明示的に指定する必要があります。

ります。 [private resource: 非公開リソース \(ページ 691\)](#)、 [resource: リソース \(ページ 694\)](#) も参照してください。

ステータス設定	説明
セキュア	この Informatica MDM Hub リソースをロールツールに公開し、特定の特権を持つロールにリソースを追加できるようします。特定のロールにユーザーアカウントを割り当てると、そのユーザーアカウントは、ロールに関連付けられた特権に従って、SIF 要求を使用してセキュアリソースにアクセスできるようになります。
非公開	ロールツールでこの Informatica MDM Hub リソースを非公開にします。デフォルト。Services Integration Framework (SIF) 操作を使用したアクセスを禁止します。Hub コンソールで新しいリソース（新しいベースオブジェクトなど）を追加すると、デフォルトでリソースは「非公開」に設定されます。

security: セキュリティ

Informatica MDM Hub の実装内のデータおよびその他リソースに対する不正アクセスや改ざんを防ぐことで、情報のプライバシー、機密性、およびデータ整合性を保護する機能。

Security Access Manager workbench: Security Access Manager ワークベンチ

ユーザー、グループ、リソース、およびロールに対するツールを含みます。

security payload: セキュリティペイロード

以降の認証や承認に必要な補足データを含めることができる、MDM Hub 操作の要求に提供される RAW バイナリデータ。

security provider: セキュリティプロバイダ

ユーザーが Informatica MDM Hub にアクセスする際のセキュリティサービス（認証、承認、およびユーザープロファイルサービス）を提供するサードパーティ製アプリケーション。

segment matching: セグメント一致

一致ルールを特定のデータのサブセットに限定する方法。例えば、セグメント一致を使用して別個の国の顧客に別々の一致ルールを定義して、特定の国コードに特定のルールを限定的に使用することができます。セグメント一致は、ルールベースで設定され、完全一致とあいまい一致のベースオブジェクト両方に適用されます。

Services Integration Framework (SIF)

クライアントプログラムとのインタフェースとなる Informatica MDM Hub の一部。論理的に、クライアント/サーバーモデルの中間層として提供されます。以下のアーキテクチャのいずれかを使用して要求/応答処理を実装できます。

- SOAP プロトコルを使用する緩やかに結合された Web サービス。
- Enterprise JavaBeans (EJB) または XML をベースとする、緊密に結合された Java リモートプロシージャ呼び出し。
- 非同期の Java Message Service (JMS) ベースメッセージ。
- Hypertext Transfer Protocol (HTTP) を介してやり取りされる XML ドキュメント。

SIRL decay: **SIRL 減衰**

SIRL (Slow Initial Rapid Later) 減衰。ほとんどの減衰が減衰期間の最後に発生します。信頼レベルは凸の放物曲線に従います。ソースシステムがこの減衰タイプである場合、値が減衰期間の最後に近づくまで、他のシステムがこの値を上書きする可能性は比較的低くなります。

soft delete: **論理削除**

ベースオブジェクトまたは相互参照レコードは、ユーザー属性または HUB_STATE_IND で削除済みとマークされます。

source record: **ソースレコード**

ソースシステムからの raw レコード。

source system: **ソースシステム**

データを Informatica MDM Hub に提供する外部システム。

stage process: **ステージプロセス**

ランディングテーブルからデータを読み取り、設定されているクレンジングを実行し、さらにクレンジング済みデータを対応するステージングテーブルに移動するプロセス。差分検出を有効にした場合、Informatica MDM Hub では、新規または変更されたレコードのみが処理されます。

staging table: **ステージングテーブル**

クレンジングされたデータが、ロードジョブを介してベースオブジェクトにロードされる前に一時的に格納されるテーブル。

state-enabled base object: **状態が有効なベースオブジェクト**

状態管理が有効になっているベースオブジェクト。

state management: **状態管理**

MDM データフロー全体で処理ロジックに影響する、ベースオブジェクトレコードおよび相互参照レコードのシステム状態を管理するプロセス。データフローの各ステージにあるベースオブジェクトレコードおよび相互参照レコードには、そのレコードを操作する Hub ツールを使用してシステム状態を割り当てることができます。さらに、スキーマ管理用のさまざまな Hub ツールを使用して、ベースオブジェクトの状態管理を有効にしたり、レコードの状態を変更可能なユーザーを管理するためにユーザー権限を設定することができます。

状態管理の対象となる状態は、アクティブ、保留、および削除済みです。

state transition rules: **状態遷移ルール**

レコードがある状態から別の状態に変化可能かどうか、またいつ変化するかを決定するルール。状態遷移ルールは、ベースオブジェクトと相互参照レコードで異なります。

stripping: **削除**

この用語は使用されなくなりました。

strip table: **テーブルの削除**

この用語は使用されなくなりました。

surviving cell data: **存続セルデータ**

2つのレコードからマージするセルの評価時に、Informatica MDM Hubによって、どちらのセルを存続させ、もう一方を破棄するかが判定されます。**存続セルデータ**（勝利セル）は、2つのセルにおけるより良好なデータを表します。最終的には、1つの統合されたレコードに最適な存続セルデータが含まれ、**最善データ**となります。

survivorship: **存続性**

2つのレコードからマージするセルを評価する際に Informatica MDM Hub によって判定される項目。Informatica MDM Hub によって、どちらのセルデータを存続させ、どちらを破棄するかが判定されます。存続性は、信頼が有効なカラムおよび信頼が有効ではないカラム両方に適用されます。2つの異なるレコードのセルを比較する際、Informatica MDM Hub はデータのプロパティに基づいて存続性を判定します。例えば、2つのカラムの信頼が有効な場合は信頼が高い方のセルが勝ちます。信頼スコアが同じ場合、LAST_UPDATE_DATE が新しい方のセルが勝ちます。LAST_UPDATE_DATE が同じ場合、Informatica MDM Hub では他の条件によって存続性が判定されます。

system column: **システムカラム**

Informatica MDM Hub が自動的に作成して保持するテーブル内のカラム。システムカラムはメタデータを含みます。ベースオブジェクトのシステムカラムは、ROWID_OBJECT、CONSOLIDATION_IND、および LAST_UPDATE_DATE です。

system state: **システムの状態**

ベースオブジェクトレコードが Informatica MDM Hub でどのようにサポートされるかを示します。サポートされている状態は、アクティブ、保留、および削除済みです。

table: **テーブル**

データベースにおける、行（レコード）およびカラムに編成されたデータの集合。テーブルは、オブジェクトに対応する値の2次元セットと見なすことができます。テーブルのカラムはオブジェクトの特性を表し、行はオブジェクトのインスタンスを表します。Hub ストアでは、マスターデータベースと各オペレーショナル参照ストア（オペレーショナル参照ストア）がテーブルの集合を表します。オペレーショナル参照ストアにはベースオブジェクトがテーブルとして格納されます。

target database: **ターゲットデータベース**

Hub コンソールにおける、現在のツールのターゲットとなるマスターデータベースまたはオペレーショナル参照ストア（オペレーショナル参照ストア）。マスターデータベースに格納されているデータを管理するツール（ユーザーツールなど）では、ターゲットデータベースはマスターデータベースである必要があります。オペレーショナル参照ストアに格納されているデータを管理するツールでは、使用するオペレーショナル参照ストアを指定する必要があります。

timeline action: **タイムラインアクション**

データ変更イベントの追跡対象であるエンティティに対して実行されるアクション。レコードの追加、レコードの編集、有効期間の編集などのアクションを実行できます。

timeline granularity: **タイムラインの粒度**

レコードバージョンの有効期間の定義に使用される、時間の計測単位。例えば、年単位、月単位、秒単位などで有効期間を選択できます。

timeline rules: **タイムラインのルール**

データ変更イベントを追跡するために MDM Hub によって適用される事前定義されたルール。タイムラインルールに基づいて、レコードバージョンおよびそれらの有効期間におけるデータ変更イベントがキャプチャされます。

tokenize process: **トークン化プロセス**

一致比較の実行前に実行される特殊な形式のデータ標準化。最も基本的なタイプの一致では、トークン化によってスペースや句読点などの「ノイズ」文字が単に削除されます。より複雑なタイプの一致では、必要な類似度に基づいて、高度な一致コード（比較するデータの内容を表す文字列）が生成されます。

token table: **トークンテーブル**

この用語は使用されなくなりました。

traceability: **トレーサビリティ**

統合されたレコードのソースとなっているのはどのシステムか、また、そのシステム内のどのレコードかを特定できるようにするためのデータの保守。

transactional data: **トランザクションデータ**

アプリケーションによって実行されるアクションを表します。一般的には、通常の操作の一部としてアプリケーションによってキャプチャまたは生成されます。通常は、1つのレコードシステムでのみ保持され、そのコンテキストで正確かつ信頼できるデータになる傾向があります。例えば、銀行はおそらく、当座預金口座に対して行われる引き出し、預け入れ、および振替から生じるトランザクションデータを管理するためのアプリケーションを1つだけ使用しています。

transitive match: **遷移一致**

一致グループの構築（BMG）プロセス中に、他の一致の動作が原因で*間接的*に行われる一致。例えば、レコード1とレコード2を一致し、レコード2とレコード3を一致し、レコード3とレコード4を一致した場合、BMG プロセスによって冗長な一致が削除された後に、レコード2、3、および4とレコード1を一致した場合の結果が生成されたとします。この例では、レコード4をレコード1に一致させる明示的なルールはありません。代わりに、一致が間接的に行われています。

tree unmerge: **ツリーマージ解除**

マージされたベースオブジェクトレコードのツリーをそのままサブ構造としてマージ解除します。マージ解除されたベースオブジェクトレコードをルートとして持つサブツリーは、元のマージツリー構造から取得されます。例えば、a1とa2をaにマージし、b1とb2をbにマージし、最後にaとbをcにマージしたとします。この場合、aに対してツリーマージ解除を実行し、aをa1からマージ解除すると、a2がサブツリーになり、元のツリーcから取得されます。その結果、aはマージ解除後にツリーのルートになります。

trust: **信頼**

ソースシステム、変更履歴、およびその他のビジネスルールに基づいて、各セルに関連付けられた信頼度を測定するメカニズム。信頼では、データの経過時間（時間の経過とともに信頼度がどれだけ減衰したか）とデータの有効性が考慮されます。

trust level: **信頼レベル**

Informatica MDM Hub にレコードを提供するソースシステムに割り当てられた0～100の範囲の数値。この数値は、他のソースシステムと比較した際のそのソースシステムの信頼度を示します。信頼レベルは、別のソースシステムの信頼レベルと比較されたときにのみ意味を持ちます。

trust score: **信頼スコア**

特定のレコードの現在の信頼度。ロードジョブの実行中に、Informatica MDM Hub によって各レコードの信頼スコアが計算されます。ベースオブジェクトに対して検証ルールが定義されている場合は、ロードジョブによってこの検証ルールがデータに適用され、場合によっては信頼スコアがさらにダウングレードされます。統合プロセスの実行中に、2つのレコードがマージまたはリンクの候補となっている場合は、信頼スコアが高いレコードの値が残ります。マージマネージャツールで、データスチュワードによって信頼スコアを手動でオーバーライドできます。

undermatching: **一致不足**

一致が少なすぎるために関連する一致が見落とされる一致（あいまい一致ベースオブジェクトの場合のみ）。一致を設定する際の目標は、データに関する一致を適切な個数で見つけることです。

unmerge: **マージ解除**

前にマージしたレコードをマージ解除するプロセス。マージスタイルのベースオブジェクトでのみ使用されます。

user: **ユーザー**

Informatica MDM Hub リソースにアクセスできる個人（人またはアプリケーション）。Informatica MDM Hub におけるユーザーは、マスターデータベースで定義されるユーザーアカウントで表されます。

user-defined column: **ユーザー定義カラム**

テーブル内のシステムカラムではない任意のカラム。ユーザー定義カラムはスキーママネージャで追加され、このカラムには一般的にビジネスデータが含まれます。

user exit: **ユーザー出口**

ユーザー出口は Java コードで構成され、バッチもしくは SIF の API プロセスの特定の時点で実行し、MDM Hub の機能性を拡張します。

開発者は、バッチジョブの実行前および実行後の処理のために適切なユーザーイグジットにカスタムコードを追加して、Informatica MDM Hub バッチプロセスを拡張できます。

user group: **ユーザーグループ**

ユーザーアカウントの論理的な集まり。

user object: **ユーザーオブジェクト**

機能を拡張するために MDM Hub に登録される、ユーザーによって定義された機能。

MDM Hub には次のタイプのユーザーオブジェクトがあります。

ユーザーオブジェクト	説明
ユーザーイグジット	修正されたパラメータおよび事前定義されたパラメータ式を含む Java コード。ユーザーイグジットは、Informatica MDM Hub のバッチプロセス実行時の特定の時点で実行されるように、ベースオブジェクト単位で設定します。
カスタム Java クレンジング関数	標準のクレンジングライブラリをカスタムロジックで補完するための Java クレンジング関数。これらの関数は基本的には JAR ファイルで、データベースに BLOB として格納されています。
カスタムボタン関数	データマネージャ、マージマネージャ、および階層マネージャに追加のアイコンやロジックを設定するためのカスタム UI 関数。

validation process: 検証プロセス

リポジトリを定義するメタデータの完全性と整合性を検証するプロセス。検証プロセスでは、リポジトリの論理モデルと物理スキーマを比較します。問題が発生した場合は、注意を必要とする問題のリストがリポジトリマネージャによって生成されます。

validation rule: 検証ルール

データ値が無効と見なされる条件を Informatica MDM Hub に示すルール。データが検証ルールで指定された基準を満たすと、そのデータの信頼値が検証ルールで指定された割合だけダウングレードされます。カラムに対して「最小信頼度の保持」フラグが設定されている場合、信頼をカラムの最小信頼度未満にダウングレードすることはできません。

workbench: ワークベンチ

Hub コンソールにおける、同様のツールをグループ化するメカニズム。ワークベンチとは、関連するツールの論理的集合です。例えば、モデルワークベンチには、スキーマ、クエリ、パッケージ、マッピングといった、データモデリングのためのツールが含まれています。

オペレーショナル参照ストア (ORS)

マスタデータを含むデータベースとマスタデータに適用されるルール。ルールには、マスタデータ処理のルール、マスタデータオブジェクトセット管理のルール、および MDM Hub がベストバージョン オブ トールースを定義するために使用する処理ルールと補助ロジックが含まれます。MDM Hub の構成には、1 つ以上のオペレーショナル参照ストアを含めることができます。ORS のデフォルト名は CMX_ORIS です。

システムと信頼ツール

システムと信頼ツールは、Informatica MDM Hub で統合するためのデータを提供できるソースシステムの名前を示すために使用される設計時ツールです。このツールを使用して、ベースオブジェクト内の信頼が有効なカラムごとに各ソースシステムに関連付けられた信頼設定を定義します。

セキュリティアクセスマネージャ (SAM)

セキュリティアクセスマネージャ (SAM) は、MDM Hub リソースを未承認アクセスから保護するセキュリティモジュールです。実行時に、SAM は MDM Hub 実装に対する組織のセキュリティポリシー決定を実施し、セキュリティ設定に従ってユーザーの認証とアクセス承認を処理します。

タイムライン

一定の期間におけるビジネスエンティティとそのリレーションのデータ変更イベント。データ変更イベントは、有効期間全体にわたって定義します。

データ統合サービス

Informatica Developer に対してデータ統合ジョブを実行するアプリケーションサービス。データ統合ジョブには、データのプレビューやマッピングの実行が含まれます。

データ統合ステージング

ソースシステムからデータを直接読み取り、データ品質トランスフォーメーションを使ってデータをクレンジングし、MDM Hub の対応するステージングテーブルにクレンジングしたデータを移動するプロセス。

パス

[一致パス \(ページ 703\)](#) を参照してください。

ビジネスエンティティ

ベースオブジェクトのネスト構造。ビジネスエンティティのルートベースオブジェクトに関連するすべての情報を表示するには、Informatica Data Director でエンティティ 360 フレームワークを使用します。ビジネスエンティティ内のデータを検索するには、Informatica Data Director で検索を実行します。

ビジネスエンティティサービス

ビジネスエンティティサービスは、MDM Hub コードを実行して、ビジネスエンティティのベースオブジェクトレコードを作成、更新、削除、検索する一連の操作です。

ファームिंगデプロイメント

JBoss クラスタにアプリケーションをデプロイする、デプロイメントの種類の 1 つ。アプリケーションのアーカイブファイルをクラスタメンバのファームディレクトリにデプロイすると、アプリケーションは同じクラスタ内のすべてのノードにわたって自動的に複製されます。

プロセスサーバー

クレンジング、マッチング、バッチジョブを行うサーバーです。プロセスサーバーはアプリケーションサーバー環境にデプロイされます。プロセスサーバーは、データを標準化する Trillium Director などのクレンジングエンジンと連携します。プロセスサーバーは、各インスタンスで同時に複数の要求を処理できるようにマルチスレッド対応になっています。

モデルリポジトリ

Developer ツールからアクセス可能なプロジェクトやフォルダのメタデータを格納するリレーショナルデータベース。

モデルリポジトリサービス

モデルリポジトリを実行および管理する Informatica ドメインのアプリケーションサービス。モデルリポジトリにより、Informatica 製品で作成されたメタデータがリレーショナルデータベースに格納され、製品間の協力関係が強化されます。

ユーティリティワークベンチ

アプリケーションイベントの監査、バッチグループの設定と実行、および SIF API の生成を行うためのツールが含まれています。

リポジトリ

オペレーショナル参照ストア（オペレーショナル参照ストア）。オペレーショナル参照ストアは、自身のスキーマおよび関連付けられたプロパティ設定に関するメタデータを保管します。リポジトリマネージャで、リポジトリ間でメタデータをコピーする場合、コピーするデザインオブジェクトを含むソースリポジトリとデザインオブジェクトのコピー先であるターゲットリポジトリが常に必要です。

リポジトリマネージャ

Hub コンソールのリポジトリマネージャツールは、リポジトリのメタデータ検証、あるリポジトリから別のリポジトリへのデザインオブジェクトの昇格、リポジトリへのデザインオブジェクトのインポート、変更リストへのリポジトリのエクスポートに使用されます。。

ロール

セキュアな Informatica MDM Hub リソースにアクセスするための一連の特権を定義します。

ワークフロー

Informatica Multidomain MDM では、ワークフローは組織内のビジネスプロセスを表します。[business process: ビジネスプロセス \(ページ 676\)](#)を参照してください。

一致パス

レコード間で階層を横断できます。ベースオブジェクト間の階層（テーブル間パス）にも、1つのベースオブジェクト内に存在する階層（テーブル内パス）にも対応します。一致パスは、別々のテーブルまたは同じテーブルにある関連レコードが関係する一致カラムルールを設定するために使用されます。

制約テーブル

一意または外部キー制約が定義されたデータベーステーブル。

却下テーブル

Informatica MDM Hub がターゲットテーブルに挿入できなかったレコードを含むテーブル。以下の種類があります。

- 指定したランディングテーブルのレコードに対して指定されたクレンジングの実行後のステー징ングテーブル（ステージプロセス）
- Hub Store テーブル（ロードプロセス）

セルの値が長すぎる場合、またはレコードの更新日が現在の日付より後の場合にレコードは却下されます。

操作

この用語は使用されなくなりました。[request: 要求 \(ページ 694\)](#)を参照してください。

書き込みロック

Hub コンソールにおける、基になるスキーマに変更を加えるために必要なロック。データスチュワード以外のすべてのツール（オペレーショナル参照ストアセキュリティツールを除く）は、書き込みロックを取得しない限り読み取り専用モードになります。書き込みロックによって、複数の同時ユーザーがスキーマに変更を加えることができるようになります。

検索レベル

Informatica MDM Hub がどの程度厳密に一致を検索するかを定義します（低、標準、高、または最高）。一致の目標は、データの一致を適切な数だけ見つけることです。少なすぎると重大な一致を見逃し（一致不足）、多すぎると重要ではないものも含めた過剰な一致が生成されます（一致過多）。

物理データオブジェクト

読み取り、検索、またはリソースへの書き込みに使用するデータの物理的な表現。

論理データオブジェクト

組織内の論理エンティティを表すオブジェクト。論理データオブジェクトには属性とキーがあり、属性間のリレーションを表します。

論理データオブジェクトマッピング

論理データオブジェクトを 1 つ以上の物理データオブジェクトにリンクするマッピング。論理データオブジェクトマッピングには、トランスフォーメーションロジックを含めることができます。

論理データオブジェクトモデル

組織内のデータとデータ間のリレーションを表すデータモデル。論理データオブジェクトが含まれ、オブジェクト間のリレーションが定義されます。

論理データオブジェクト書き込みマッピング

論理データオブジェクトを入力として使用し、ターゲットにデータを書き込むマッピング。このマッピングには 1 つ以上の論理データオブジェクトが入力として、物理データオブジェクトがターゲットとして含まれます。

論理データオブジェクト読み取りマッピング

論理データオブジェクト経由でデータを表示するマッピング。これにはソースとして 1 つ以上の物理データオブジェクトが含まれ、マッピング出力として論理データオブジェクトが含まれます。

索引

A

Address_Part1 キータイプ [405](#)
Address 一致目的 [418](#)
AssignTasks ユーザー出口
 インタフェース [601](#)
AUTOMERGE_IND
 外部マッチング出力テーブルカラム [567](#)

B

BPM ワークフローツール [178](#)
build_war マクロ [611](#)
BVT スナップショットジョブ [565](#)

C

C_REPOS_DB_RELEASE テーブル
 カラム [656](#)
C_REPOS_EXT_HARD_DEL_DETECT
 プライマリキーソースカラム [330](#)
C_REPOS_SYSTEM テーブル
 相互参照テーブル内の ROWID_SYSTEM による参照カラム [103](#)
CM_DIRTY_IND
 ベースオブジェクトのカラム [100](#)
cmxcleanse.properties
 概要 [644](#)
cmxserver.log
 説明 [667](#), [668](#)
cmxserver.properties
 概要 [621](#)
console.log
 説明 [667](#)
CONSOLIDATION_IND
 ベースオブジェクトのカラム [100](#)
Contact 一致目的 [418](#)
Corporate_Entity 一致目的 [418](#)
CREATE_DATE
 外部マッチング出力テーブルカラム [567](#)
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)
CREATOR
 外部マッチング出力テーブルカラム [567](#)
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)

D

DataEncryptor
 インタフェース [189](#)
 実装 [189](#)

DELETED_BY
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)
DELETED_DATE
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)
DELETED_IND
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)
Division 一致目的 [418](#)

E

Elasticsearch
 高可用性 [470](#)
elasticsearch アーカイブ
 抽出 [471](#)
Elasticsearch のインストール
 インストール前のタスク [470](#)
 前提条件 [470](#)
EMO テーブル、参照項目外部一致出力テーブル
entity360view.log
 説明 [668](#)

F

Family 一致目的 [418](#)
Fields 一致目的 [418](#)
FILE_NAME
 外部マッチング出力テーブルカラム [567](#)

G

GBID
 カラムのプロパティ [113](#)
GBID カラム [115](#)
GetAssignableUsersForTask ユーザー出口
 インタフェース [601](#)
GOV
 相互参照テーブルカラム [103](#)

H

HM パッケージ
 エンティティタイプへの割り当て [224](#)
 削除 [222](#)
 編集 [222](#)
Household 一致目的 [418](#)
HPCT テーブル、参照項目保留中の制御テーブル

HTTPS
プロセスサーバー向け [345](#)

Hub Store
オペレーショナルレコードストア (ORS) [63](#)
マスターデータベース [63](#)

HUB_STATE_IND
ステージングテーブルのカラム [363](#)
相互参照テーブルカラム [103](#)

HUB_STATE_IND カラム
概要 [175](#)

Hub コンソール
起動 [30](#)

Hub サーバー
cmxserver.log [667](#)
データ収集の無効化 [674](#)
データ収集の有効化 [673](#)
プロパティ [621](#)
ロギング設定 [667](#)

Hub ストア
スキーマ [79](#)
テーブルタイプ [80](#)
プロパティ [72](#)

Hub の状態
レコードの状態 [174](#)
関連項目レコードの状態

I

IBM DB2
データ型 [112](#)

Individual 一致目的 [418](#)

Informatica Data Director の基本検索
子レコードの欠如を許可する [399](#)

Informatica Data Quality
バッチジョブとマッピング [309](#)

INTERACTION_ID
相互参照テーブルカラム [103](#)

INTERACTION_ID カラム
概要 [176](#)

J

JAR ファイル
ORS 固有、使用 [612](#)
ORS 固有、ダウンロード [611](#)
ユーザーイグジット JAR ファイルの実装 [588](#)
ユーザー出口の [588](#)

Java アーカイブ (JAR) ファイル
tools.jar [615](#)

Java ユーザーイグジット
概要 [586](#)

JMS イベントスキーママネージャ
開始 [614](#)
概要 [613](#)
非同期オブジェクトの検出 [615](#)
非同期オブジェクトの自動検索 [616](#)

L

LAST_ROWID_SYSTEM
ベースオブジェクトのカラム [100](#)

LAST_UPDATE_DATE
ステージングテーブルのカラム [363](#)
相互参照テーブルカラム [103](#)
ベースオブジェクトのカラム [100](#)

log4j-entity360view.xml
説明 [668](#)

M

MDM Hub
ロギング [666](#)

MDM Hub コンソール
概要 [30](#)
ログイン [37](#)

MDM Hub コンソールインタフェース
ウィザードのようこそ画面 [41](#)
ウィンドウのサイズと位置 [41](#)
カスタマイズ [41](#)
[クイック起動] タブ [41](#)
[全般] タブ [41](#)
ツールバー [41](#)

MDM Hub ステージング
概要 [271](#), [302](#)

Microsoft SQL Server
カスタムインデックスの制限事項 [110](#)
行幅制限 [112](#)
データ型 [112](#)

Multidomain MDM
インストールの詳細 [41](#)

N

NULL 一致 [424](#)

NULL の外部キーを許可する
ステージングテーブルのカラムで [366](#)

NULL の更新を許可する
ステージングテーブルのカラムで [366](#)

NULL 可能
カラムのプロパティ [113](#)

NULL 値
カラムの NULL 値の許可 [113](#)

NULL 値を適用する
カラムのプロパティ [113](#)

O

Oracle
データ型 [112](#)

Organization_Name キータイプ [405](#)

Organization 一致目的 [418](#)

ORIG_ROWID_OBJECT
相互参照テーブルカラム [103](#)

ORS 固有の API
API アーカイブテーブル
メンテナンス [610](#)
build_war マクロ [612](#)
アーカイブテーブル [610](#)
概要 [608](#)
クライアント JAR のダウンロード [611](#)
使用 [612](#)
パフォーマンス [609](#)
プロパティ [609](#)
リポジトリオブジェクト [609](#)
リポジトリオブジェクトのステータス [610](#)

ORS 固有の SIF API
生成およびデブロイ [611](#)

ORS 固有のメッセージスキーマ
概要 [613](#)

P

PCTL テーブル、参照項目保留中の制御テーブル
PERIOD_END_DATE
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
PERIOD_START_DATE
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
Person_Name 一致目的 [418](#)
Person_Name キータイプ [405](#)
PKEY_SRC_OBJECT
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
PROMOTE_IND
 相互参照テーブルカラム [103](#)
provisioning.log
 説明 [668](#)
PUT_UPDATE_MERGE_IND
 相互参照テーブルカラム [103](#)
PUT 可能パッケージ [142](#)
PUT 対応パッケージ [142](#)
PUT 可能
 カラムのプロパティ [113](#)

R

RAW テーブル
 概要 [303](#)
Resident 一致目的 [418](#)
RISL (Rapid Initial Slow Later) 減衰 [372](#)
ROWID_MATCH_RULE
 外部マッチング出力テーブルカラム [567](#)
ROWID_OBJECT
 ステージングテーブルのカラム [363](#)
 相互参照テーブルカラム [103](#)
 ベースオブジェクトのカラム [100](#)
ROWID_OBJECT_MATCH
 外部マッチング出力テーブルカラム [567](#)
ROWID_SYSTEM
 相互参照テーブルカラム [103](#)
ROWID_XREF
 相互参照テーブルカラム [103](#)

S

S_
 相互参照テーブルカラム [103](#)
search
 stopwords [474](#)
 stopwords.txt ファイル [474](#)
 無視される単語 [474](#)
SIF の API
 ORS 固有、削除 [612](#)
 ORS 固有、名前変更 [611](#)
 ユーザーイグジットのサポート対象 [604](#)
SIF の API のユーザーイグジット
 例 [603](#)
 作成 [602](#)
SMOS
 状態管理オーバーライドシステム [296](#)
SOURCE_KEY
 外部マッチング出力テーブルカラム [567](#)
SOURCE_NAME
 外部マッチング出力テーブルカラム [567](#)

SQL

 カスタムクエリ [139](#)
SQL*Loader
 破損データ [49](#)
SRC_LUD
 相互参照テーブルカラム [103](#)
SRC_ROWID
 ステージングテーブルのカラム [363](#)
STG_ROWID_TABLE
 相互参照テーブルカラム [103](#)
stopwords.txt ファイル
 カスタマイズ [474](#)
 設定 [474](#)
synonyms.txt ファイル
 カスタマイズ [474](#)
 設定 [474](#)

U

Unicode
 NLS_LANG [49](#)
 UNIX とロケールに関する推奨事項 [48](#)
 クレンジング設定 [48](#)
UPDATED_BY
 ステージングテーブルのカラム [363](#)
 相互参照テーブル内のカラム [103](#)
 ベースオブジェクトのカラム [100](#)
UserExitContext
 概要 [589](#)

V

VERSION_SEQ
 ステージングテーブルのカラム [363](#)

W

Wide_Contact 一致目的 [418](#)
Wide_Household 一致目的 [418](#)

X

XSD ファイル
 ダウンロード [615](#)

あ

あいまい一致
 あいまい一致カラム [401](#)
 あいまい一致ベースオブジェクト [287](#), [404](#)
 あいまい一致/検索ストラテジ [415](#)
 あいまい一致ストラテジ [389](#)
あいまい一致ルール [287](#)
アクティブ状態、概要 [174](#)
アナライザ
 文字フィルタ
 トークナイザ [481](#)
 トークンフィルタ [481](#)
アプリケーションサーバーのログ
 概要 [659](#)
 設定 [660](#)
 レベル [659](#)
 ログファイルのローリング [659](#)

アンマージ後ユーザーイグジット
インタフェース [600](#)
概要 [599](#), [600](#)
アンマージプロセス
ユーザー出口 [598](#)
アンマージ前ユーザーイグジット
インタフェース [599](#)

い

一意のキー [313](#)
一致カラム
あいまい一致カラム [401](#)
あいまい一致ベースオブジェクト [404](#)
一致カラムの概要 [401](#)
一致キータイプ [405](#)
完全一致カラム [401](#)
完全一致ベースオブジェクト [407](#)
キー幅 [405](#)
子レコードの欠如 [399](#)
一致カラムルール
削除 [433](#)
追加 [431](#)
編集 [432](#)
一致キーテーブル
定義 [80](#)
一致グループの構築 (BMG) [289](#)
一致サブタイプ [424](#)
一致テーブルのリセットジョブ [583](#)
一致トークンの生成ジョブ [570](#)
一致パス
テーブル間パス [392](#)
テーブル内パス [394](#)
リレーションベースオブジェクト [392](#)
一致目的
フィールドタイプ [402](#)
フィールド名 [402](#)
一致ルールセットの検索レベル [410](#)
一般レベルの検索 [410](#)
インデックス
カスタムインデックス [109](#)
インデックスの設定
アナライザ [481](#)

え

エラスティック検索
インデックスの設定 [481](#)
エンタープライズマネージャ
オペレーショナルリファレンスストアのプロパティ [656](#)
エンティティ
概要 [203](#)
表示オブション [211](#)
エンティティ 360 表示
entity360view.log [668](#)
エンティティアイコン
削除 [203](#)
設定 [202](#)
追加 [202](#)
デフォルトのアップロード [202](#)
編集 [203](#)
エンティティタイプ
HM パッケージの割り当て [224](#)
概要 [207](#)
削除 [211](#)
作成 [210](#)

エンティティタイプ (続く)
編集 [210](#)
例 [209](#)
エンティティベースオブジェクト
概要 [203](#)
作成 [205](#)
ベースオブジェクトからの変換 [207](#)
ベースオブジェクトに戻す [211](#)

お

オペレーショナル参照ストア (ORS)
ORS について [63](#)
接続テスト [74](#)
登録解除 [76](#)
パスワード、変更 [74](#)
オペレーショナルリファレンスストア
データベースプロパティ [656](#)
登録 [67](#), [69](#)
オペレーショナルリファレンスストア (ORS)
GETLIST 制限 (行) [72](#)
JNDI データソース名 [72](#)
作成 [64](#)
編集 [72](#)
オペレーショナル参照ストア
登録 [70](#)
親がマージされたときにキューに再追加する [106](#)

か

階層
階層ノード [212](#)
概要 [194](#), [212](#)
削除 [212](#)
追加 [212](#)
例 [195](#)
階層ツール
設定の概要 [196](#)
階層マネージャ
エンティティアイコン、アップロード [202](#)
サンドボックス [226](#)
設定の概要 [196](#)
前提条件 [196](#)
データ設定 [222](#)
リポジトリベースオブジェクトテーブル [201](#)
外部一致ジョブ
外部一致ジョブについて [566](#)
実行 [569](#)
入力テーブル [566](#)
外部マッチングジョブ
出力テーブル [567](#)
外部一致テーブル
システムカラム [566](#)
外部キー
ルックアップ [277](#), [369](#)
外部キー関係
概要 [120](#)
仮想リレーション [121](#)
削除 [123](#)
作成 [120](#)
追加 [121](#)
編集 [121](#)
外部キーリレーション
作成 [121](#)
サポート対象 [541](#)
定義 [121](#)

外部キーリレーションベースオブジェクト

概要 [217](#)

作成 [217](#)

外部一致出力テーブル

概要 [567](#)

カラム [567](#)

書き込みロック

概要 [34](#)

取得 [36](#)

拡張キー幅 [405](#)

カスケードマージ解除 [493](#)

カスタム Java クレンジング関数

概要 [617](#)

表示 [618](#), [619](#)

カスタムインデックス

MDM Hub 外部で作成 [111](#)

概要 [109](#)

削除 [110](#)

作成 [110](#)

ノードへの移動 [110](#)

編集 [110](#)

カスタム関数

記述 [58](#)

クライアントベース [58](#)

サーバーベース [58](#)

削除 [60](#)

カスタムクエリ

SQL 構文 [139](#)

SQL の検証 [140](#)

クエリツール [129](#)

削除 [138](#)

追加 [140](#)

編集 [141](#)

概要 [139](#)

カスタムボタン

アイコン [60](#)

一覧表示 [60](#)

カスタム関数、記述 [58](#)

カスタムボタンの概要 [56](#)

クリック [57](#)

更新 [60](#)

タイプ変更 [60](#)

追加 [60](#)

テキストラベル [60](#)

デプロイ [60](#)

表示 [57](#)

プロパティファイル [60](#)

例 [58](#)

カスタムボタン関数

概要 [617](#)

登録 [619](#)

表示 [620](#)

カラム

データ型 [112](#)

テーブルへの追加 [112](#)

プロパティ [113](#)

予約語 [83](#)

環境レポート

保存 [658](#)

監査証跡、設定 [317](#)

完全一致

完全一致カラム [401](#), [430](#)

完全一致ベースオブジェクト [287](#)

フィルター致 [415](#)

完全一致/検索ストラテジ [415](#)

完全一致ストラテジ [389](#)

完全トークン化率 [106](#)

管理ソースシステム

管理ソースシステムの概要 [296](#)

名前の変更 [298](#)

き

キー

プライマリ [313](#)

プライマリ、単一カラムから [313](#)

プライマリ、複数カラムから [313](#)

キー一致ジョブ [571](#)

キータイプ [405](#)

キー幅 [405](#)

行 ID によるロード [316](#)

行レベルのロック

行レベルのロックの概要 [662](#)

使用に関する考慮事項 [663](#)

設定 [663](#)

待機時間 [664](#)

デフォルトの動作 [662](#)

有効化 [663](#)

近接検索

概要 [440](#)

設定 [441](#)

く

クエリ

影響分析、表示 [138](#)

概要 [128](#)

結果、表示 [138](#)

結合クエリ [144](#)

スキーマ変更 [130](#)

整理 [131](#)

パッケージの依存関係、分析 [138](#)

クエリグループ

概要 [131](#)

削除 [131](#)

追加 [131](#)

編集 [131](#)

クエリツール [129](#)

クエリの新規作成ウィザード [140](#)

グラフ関数

出力 [351](#)

条件付き実行コンポーネント [355](#)

入力 [351](#)

関数の追加 [352](#)

追加 [351](#)

グループ実行ログ

ステータス値 [559](#)

表示 [559](#)

クレンジング関数

Java クレンジングライブラリ [349](#)

可用性 [347](#)

関数モード [352](#)

グラフ関数 [350](#)

クレンジング関数ツール [347](#)

クレンジング関数の概要 [347](#)

クレンジングリスト [356](#)

集計 [309](#)

出力 [354](#)

条件付き実行コンポーネント [355](#)

セキュアリソース [347](#)

タイプ [347](#), [348](#)

定数 [354](#)

テスト [355](#)

クレンジング関数 (続く)

入力 [354](#)
プロパティ [348](#)
分解 [309](#)
マッピング [309](#)
ユーザークレンジングライブラリ [348](#)
ライブラリ [347](#)
ロギング [352](#)
ワークスペースボタン [353](#)
設定の概要 [348](#)

クレンジング関数ツール

開始 [347](#)
ワークスペースボタン [353](#)

クレンジングリスト

defaultValue プロパティ [358](#)
Input string プロパティ [358](#)
matchFlag プロパティ [359](#)
replaceAllOccurrences プロパティ [358](#)
searchType プロパティ [358](#)
SQL 一致 [360](#)
stopOnHit プロパティ [358](#)
Strip プロパティ [358](#)
一致出力文字列、インポート [361](#)
一致プロパティ [359](#)
一致文字列、インポート [360](#)
クレンジングリストの概要 [357](#)
追加 [357](#)
出力プロパティ [359](#)
プロパティ [357](#)
編集 [359](#)
完全一致 [360](#)
正規表現一致 [360](#)
文字列の一致 [360](#)
グローバル識別子 (GBID) カラム [115](#)

け

言語

Oracle 環境での設定 [49](#)

検証チェック [376](#)

検証ルール

カスタム、例 [381](#)
カスタム検証ルール [378](#), [380](#)
カラムの検証の有効化 [377](#)
検証チェック [376](#)
検証ルールの概要 [376](#)
最小信頼度の保持 [379](#)
削除 [382](#)
参照整合性 [378](#)
実行シーケンス [378](#)
状態が有効なベースオブジェクト [377](#)
存在確認 [378](#)
ダウングレード率 [379](#)
追加 [381](#)
定義 [376](#)
ドメイン確認 [378](#)
パターンの検証 [378](#)
必須カラム [377](#)
プロパティ [378](#)
編集 [382](#)
ルール SQL [379](#)
ルールカラムのプロパティ [379](#)
ルールタイプ [378](#)
ルール名 [378](#)
例 [379](#)
減衰曲線 [372](#)

減衰タイプ

RISL [372](#)
SIRL [372](#)
線形 [372](#)
限定キー幅 [405](#)

こ

高レベルの検索 [410](#)
個別ソースシステム [493](#)
個別マッピング [315](#)
コマンドボタン
オブジェクトの削除 [40](#), [41](#)
オブジェクトの追加 [40](#)
オブジェクトのプロパティの編集 [40](#), [41](#)
子レコードの欠如
概要 [399](#)

さ

再検証ジョブ [583](#)
最高予約キー
例 [366](#)
最高レベルの検索 [410](#)
最小信頼度 [372](#)
ベストバージョン オブ トールス (BVT)
概要 [292](#)
最善データ (BVT) [291](#)
最大信頼度 [372](#)
差分検出
使用に関する考慮事項 [320](#)
処理方法 [319](#)
設定 [318](#)
ランディングテーブル設定 [299](#)
サンドボックス [226](#)
サンプルアプリケーション
データ暗号化向け [188](#)

し

システムカラム
外部一致テーブル [566](#)
ステー징テーブル [363](#)
説明 [120](#)
ベースオブジェクト [100](#)
システムと信頼ツール [296](#)
システムリポジトリテーブル [296](#)
システム状態
レコードの状態 [174](#)
自動一致とマージジョブ
メトリック [564](#)
自動マージジョブ
自動一致とマージジョブ [564](#)
信頼が有効なカラム [564](#)
メトリック [565](#)
出力 [354](#)
手動
手動マージ解除ジョブ [577](#)
手動マージジョブ [576](#)
バッチジョブ [545](#)
昇格ジョブ [581](#), [664](#)
条件付き実行コンポーネント
条件付き実行コンポーネントの概要 [356](#)
使用するタイミング [356](#)
追加 [356](#)

条件マッピング [315](#)
照合ルールプロパティ
ジオコード半径 [423](#)
状態管理
HUB_STATE_IND カラム [175](#)
INTERACTION_ID カラム [176](#)
一致ジョブ [578](#)
概要 [173](#), [174](#)
状態遷移ルール、概要 [176](#)
タイムライン [150](#)
データのロードのルール [177](#)
ベースオブジェクトレコードの存続性 [177](#)
メッセージトリガ、有効化 [184](#)
有効化 [179](#)
レコードの昇格 [184](#), [185](#)
レコードの状態
状態遷移 [176](#)
ロードジョブ [572](#)
状態管理オーバーライドシステム
ロードジョブ [574](#)
状態管理オーバーライドシステム (SMOS)
概要 [296](#)
初期データロード (IDL) [273](#)

す

スキーマ
スキーマについて [79](#)
変更によるクエリおよびパッケージへの影響 [130](#)
スキーマ一致カラム [583](#)
スキーマオブジェクト [83](#)
スキーマの信頼カラム [584](#)
スキーマビューア
JPG で保存 [126](#)
印刷 [126](#)
オプション [125](#)
開始 [123](#)
階層ビュー [125](#)
概要ペイン [123](#)
カラム名 [125](#)
コマンドボタン [123](#)
コンテキストメニュー [125](#)
ズームアウト [124](#)
ズームイン [124](#)
図ペイン [123](#)
全体のズーム [124](#)
直交ビュー [125](#)
ビューの切り替え [125](#)
ペイン [123](#)
方向 [125](#)
スキーママネージャ
開始 [98](#)
外部キー関係 [120](#)
テーブルへのカラムの追加 [112](#)
ベースオブジェクト [98](#)
ステージジョブ
却下されたレコード [551](#)
ステージプロセス
ステージングテーブル [363](#)
テーブル [303](#)
ユーザーイグジット [591](#)
ステージング後ユーザー出口
インタフェース [593](#)
概要 [593](#)
ステージングテーブル
NULL の外部キーを許可する [366](#)
概要 [303](#), [363](#)

ステージングテーブル (続く)
カラム [116](#)
カラム、作成 [116](#)
カラムのプロパティ [366](#)
管理 [321](#)
構成 [363](#)
最高予約キー [365](#)
削除 [322](#)
システムカラム [363](#)
セルの更新 [366](#)
ソースシステムキーを保持する [365](#)
ソースシステムへの移動 [321](#)
定義 [80](#)
プロパティ [304](#)
編集 [321](#), [368](#)
ユーザー定義カラム [363](#)
ロードプロセス [272](#)
ステージングテーブルのカラム
NULL の更新を許可するプロパティ [366](#)
ルックアップカラムのプロパティ [366](#)
ルックアップテーブルのプロパティ [366](#)
ステージング前ユーザー出口
インタフェース [592](#)
概要 [592](#)
ステータス、設定 [547](#)

せ

正規表現関数
追加 [350](#)
制御テーブル [372](#)
生成
ロード時に一致トークンを生成する親がマージされたときにキューに再追加する [106](#)
制約
無効化 [106](#)
セキュリティアクセスマネージャーワークベンチ
概要 [43](#)
セグメント一致 [429](#)
設定ワークベンチ
概要 [42](#)
セルの更新 [366](#)
線形減衰 [372](#)

そ

相互参照昇格の履歴、有効化 [179](#)
相互参照テーブル
ROWID_XREF カラム [103](#)
カラム [103](#)
説明 [80](#)
相互参照テーブルの概要 [101](#)
定義 [80](#)
ベースオブジェクトへのリレーション [102](#)
履歴テーブル [105](#)
ロードプロセス [272](#)
クロスリファレンステーブル
昇格の履歴の有効化 [179](#)
増分ロード [273](#)
ソースシステム
管理ソースシステム [296](#)
キーの保持 [365](#)
個別ソースシステム [493](#)
最高予約キー [365](#)
システムと信頼ツール、開始 [296](#)
システムリポジトリテーブル (C_REPOS_SYSTEM) [296](#)

ソースシステム (続く)
ソースシステムの概要 [295](#)
不変ソースシステム [492](#)
プロパティ [297](#)
削除 [298](#)
追加 [297](#)
名前の変更 [298](#)
ソースシステムキーの保持 [365](#)

た

ターゲットデータベース
変更 [36](#)
選択 [30](#)
タイムライン
ガイドライン [146](#)
概要 [145](#)
構成 [160](#)
状態管理 [150](#)
有効化 [106](#), [109](#), [159](#)
粒度レベル [149](#)
ルール [150](#)
例 [146](#)
ロードバッチ [160](#)
タイムラインのルール
有効期間の重複 [150](#)
連続 [150](#)
タイムラインレコードバージョン
例 [148](#)
タスク
BPM ワークフローツール [178](#)
タスク管理
ユーザー出口 [601](#)

ち

重複一致しきい値 [106](#)
重複データ
一致 [288](#)
地理的座標
緯度 [401](#)
経度 [401](#)
標高 [401](#)

つ

ツール
クレンジング関数ツール [347](#)
スキーママネージャ [98](#)
ツールアクセスツール [54](#)
データベースツール [66](#)
バッチビューアツール [544](#)
マージマネージャツール [290](#)
マッピングツール [584](#)
ユーザーアクセス [53](#)
ユーザーツール [53](#)
ツールアクセスツール [54](#)

て

定義 [295](#)
定数 [354](#)
低レベルの検索 [410](#)

データアクセス
階層マネージャ [222](#)
データクレンジング
データクレンジングの概要 [341](#)
プロセスサーバー [342](#)
データ型
概要 [112](#)
カラムのプロパティ [113](#)
データスチュワードワークベンチ
概要 [43](#)
データソース
JDBC データソース [77](#)
削除 [77](#)
データソースについて [77](#)
データのクレンジング
MDM Hub 内 [341](#)
Unicode の設定 [48](#)
クレンジングの概要、データ [341](#)
設定タスク [341](#)
データのロード
初期データロード (IDL) [273](#)
増分ロード [273](#)
データベース
Unicode、設定 [46](#)
ターゲットデータベース [30](#)
データベース ID [72](#)
データベースオブジェクト名、制約 [83](#)
データベースツール
データベースツールの概要 [66](#)
データベースプロパティ
オペレーショナルリファレンスストア [656](#)
データ暗号化
Hub サーバーの設定 [190](#)
アーキテクチャ [187](#)
概念 [187](#)
サポートされている API 要求 [191](#)
サンプルのプロパティファイル [192](#)
設定 [189](#)
プロセスサーバーの設定 [191](#)
プロパティファイル [190](#)
ユーティリティ [188](#)
概要 [187](#)
制限 [188](#)
データ収集
Hub サーバー [673](#), [674](#)
MDM Hub 環境情報 [672](#)
システム設定情報 [672](#)
プロセスサーバー [673](#), [674](#)
概要 [671](#)
テーブル
Hub ストア [80](#)
一致キーテーブル [80](#), [279](#)
カラムの追加 [112](#)
システムリポジトリテーブル (C_REPOS_SYSTEM) [296](#)
ステージングテーブル [80](#)
制御テーブル [372](#)
相互参照テーブル [80](#)
バッチプロセスが使用するサポートテーブル [540](#)
ベースオブジェクト [80](#)
ランディングテーブル [80](#)
履歴テーブル [80](#)
テーブルカラム
グローバル識別子 (GBID) カラム [115](#)
削除 [120](#)
ステージングテーブル [116](#)
追加 [117](#)
テーブルカラムの概要 [112](#)
別テーブルからのインポート [118](#)

テーブルカラム (続く)
編集 [118](#)
テーブルカラム 定義のインポート [118](#)
テーブル間一致
説明 [430](#)
テーブル間パス [392](#)
テーブル内パス [394](#)
データ
破損、トラブルシューティング [49](#)
理解 [384](#)
デフォルト
カラムのプロパティ [113](#)
デフォルトあり
カラムのプロパティ [113](#)

と

同期ジョブ [376](#), [584](#)
統合
最善データ [291](#)
統合されたレコード [98](#)
統合プロセス
オプション [291](#)
概要 [290](#), [492](#)
トークナイザ
組み込み [482](#)
トークンフィルタ
組み込み [482](#)
トークン化プロセス
一致キー [279](#)
一致キーテーブル [279](#)
一致トークン [279](#)
主要な概念 [281](#)
トークン化プロセスの概要 [279](#)
未処理テーブル [281](#)
トラブルシューティング
cmxserver.log ファイル [667](#)
トレーサビリティ [290](#)

な

ナビゲーションツリー
親ノード [37](#)
項目の表示の変更 [39](#)
項目のフィルタリング [39](#)
子ノード [37](#), [38](#)
コマンドの実行 [40](#)
表示名によるソート [38](#)
フィルタリングオプション [38](#)
項目の検索 [39](#)
ナビゲーションペイン
概要 [37](#)
ナビゲーションペインの子ノード
非表示 [38](#)
表示 [38](#)

に

入力 [354](#)

は

排他ロック
概要 [34](#)

排他ロック (続く)
取得 [36](#)
バスコンポーネント
削除 [400](#)
追加 [400](#)
編集 [400](#)
パスワード
暗号化 [75](#)
変更 [37](#)
パスワードの暗号化 [75](#)
破損データ
トラブルシューティング [49](#)
パッケージ
PUT 可能 [142](#)
PUT 対応 [142](#)
概要 [128](#), [141](#), [221](#)
クエリ変更後の更新 [144](#)
結合クエリ [144](#)
更新パッケージ、概要 [142](#)
削除 [144](#)
スキーマ変更 [130](#)
追加 [142](#)
パッケージツール [129](#)
表示パッケージ、概要 [141](#)
編集 [143](#)
マージ [142](#)
パッケージツール [129](#)
バッチグループ
削除 [554](#)
実行 [557](#)
追加 [553](#)
編集 [554](#)
レベル、設定 [554](#)
バッチジョブ
BVT スナップショットジョブ [565](#)
一致テーブルのリセットジョブ [583](#)
一致トークンの生成ジョブ [570](#)
一致分析ジョブ [579](#)
外部一致ジョブ [566](#)
外部キーリレーション [541](#)
概要 [537](#)
キー一致ジョブ [571](#)
起動 [539](#)
却下されたレコード [551](#)
構成可能なオプション [546](#)
コマンドボタン [546](#)
再検証ジョブ [583](#)
サポートテーブル [540](#)
実行 [546](#)
自動一致とマージジョブ [563](#)
自動的に作成されるバッチジョブ [542](#)
自動マージジョブ [564](#)
手動による実行 [545](#)
手動マージ解除ジョブ [577](#)
手動マージジョブ [576](#)
昇格ジョブ [581](#)
ジョブ実行ステータス [547](#)
ジョブ実行ログ [547](#)
ジョブステータスを未完了に設定 [547](#)
ステージジョブ [584](#)
ステータス、設定 [547](#), [560](#)
ステータスの更新 [546](#)
設計に関する考慮事項 [541](#)
設定 [537](#)
選択 [545](#)
重複データの一致ジョブ [581](#)
同期ジョブ [376](#), [584](#)
バッチジョブの順序付け [540](#)

バッチジョブ (続く)
バッチマージ解除ジョブ [565](#)
複数マージジョブ [581](#)
プロパティ [545](#)
変更の発生時 [543](#)
マルチスレッドバッチジョブ [538](#)
履歴のクリア [552](#)
ロードジョブ [572](#)
一致しないレコードを一意とする [563](#)
一致ジョブ [577](#)
バッチジョブの順序付け [540](#)
バッチジョブの履歴のクリア [552](#)
バッチの相互運用性 [664](#)
バッチビューアツール
概要 [544](#)
パブリッシュプロセス
ORS 固有、スキーマファイル [293](#)
XSD ファイル [293](#)
概要 [292](#)
配布フロー [292](#)
メッセージキュー [293](#)
メッセージトリガ [293](#)
ランタイムフロー [294](#)
汎用クエリ
SQL 文、表示 [138](#)
概要 [132](#)
基本単位
カラム、選択 [134](#)
関数、定義 [135](#)
ソート順、定義 [137](#)
定数、定義 [135](#)
テーブル、選択 [134](#)
比較条件、定義 [136](#)
クエリ条件の絞り込み [133](#)
クエリツール [129](#)
削除 [138](#)
追加 [132](#)
編集 [133](#)

ひ

標準キー幅 [405](#)

ふ

フィルタ
削除 [398](#)
追加 [397](#)
フィルタの概要 [396](#)
プロパティ [396](#)
編集 [398](#)
フィルター一致ルール [287](#), [415](#)
複数マージジョブ [581](#)
不変行 ID オブジェクト [492](#)
プライマリキー
単一カラムから [313](#)
複数カラムから [313](#)
物理削除の検出 [330](#)
プライマリキーの一致ルール
削除 [437](#)
追加 [436](#)
編集 [436](#)
概要 [435](#)
プロセスサーバー
cmxserver.log [668](#)
HTTPS、有効化 [345](#)

プロセスサーバー (続く)
オンライン操作 [343](#)
概要 [342](#)
クレンジング要求 [343](#)
構成 [342](#)
削除 [346](#)
追加 [345](#)
データ収集の無効化 [674](#)
データ収集の有効化 [673](#)
テスト [346](#)
バッチジョブ [343](#)
プロセスサーバーツール [343](#)
プロパティ [343](#), [644](#)
編集 [346](#)
モード [342](#)
分散 [342](#)
プロセスビュー
概要 [33](#)
プロビジョニングツール
provisioning.log [668](#)
プロファイル
検証 [226](#)
コピー [227](#)
削除 [227](#)
追加 [226](#)
プロファイルの概要 [225](#)

へ

並列度 [106](#)
ベースオブジェクト
あいまい一致ベースオブジェクト [287](#)
影響分析 [111](#)
エンティティベースオブジェクト [203](#)
エンティティベースオブジェクトへの 変換 [207](#)
概要 [99](#)
カラムの追加 [83](#)
完全一致ベースオブジェクト [287](#)
削除 [111](#)
作成 [108](#)
システムカラム [100](#)
スタイル [106](#)
説明 [80](#)
定義 [99](#)
編集 [109](#)
予約接尾辞 [83](#)
予約特殊文字 [83](#)
リレーションベースオブジェクト [392](#)
リレーションベースオブジェクトから 戻す [216](#)
履歴テーブル [105](#)
レコードの存続性、状態管理 [177](#)
ロードの更新 [276](#)
ロードの挿入 [275](#)
ベースオブジェクトのスタイル [106](#)
ベースオブジェクトのプロパティ
一致プロセスの動作 [385](#)
ベストバージョン オブ トウールズ (BVT) [98](#)

ほ

ポピュレーション
US 以外のポピュレーション [46](#)
選択 [389](#)
複数のポピュレーション [47](#)
設定 [46](#)
保留中のレコード、マッチの有効化 [179](#)

ま

マージ

手動マージ解除ジョブ [577](#)

手動マージジョブ [576](#)

バッチマージ解除 [565](#)

マージ解除

親のマージ解除時に子をマージ解除 [493](#)

カスケードマージ解除 [493](#)

手動マージ解除 [577](#)

バッチマージ解除 [565](#)

マージ後ユーザーイグジット

概要 [598](#)

マージ後ユーザー

インタフェース [598](#)

マージパッケージ

[142](#)

マージプロセス

ユーザーイグジット [597](#)

マージプロセスの処理

概要 [492](#)

マージ前ユーザー出口

概要 [596](#)

マージマネージャツール

[290](#)

マスターデータベース

作成 [64](#)

パスワードの変更 [74](#)

マッチ後ユーザー出口

インタフェース [597](#)

概要 [596](#)

マッチプロセス

一致キーテーブル [288](#)

一致テーブル [288](#)

サポートテーブル [288](#)

マッチ前ユーザー出口

インタフェース [596](#)

マッピング

Informatica Data Quality のバッチジョブ [309](#)

カラム [314](#)

管理 [322](#)

行 ID によるロード [316](#)

クエリパラメータ [316](#)

クレンジング [309](#)

個別マッピング [315](#)

削除 [323](#)

条件マッピング [315](#)

図 [310](#)

スキーマへの移動 [323](#)

ステージングテーブルとランディングテーブル [80](#)

設定 [308](#)

追加 [311](#)

テスト [323](#)

パススルー [309](#)

プロパティ [305](#)

編集 [322](#)

マッピングツール

[310](#), [584](#)

め

メタデータ

信頼 [118](#)

同期 [118](#)

メタデータの同期

[118](#)

メッセージ

フィルタリング [511](#), [524](#)

メッセージフィールド [523](#)

要素 [510](#)

メッセージ (続く)

例

accept as unique メッセージ [511](#)

AmRule メッセージ [512](#)

BoDelete メッセージ [513](#)

BoSetToDelete メッセージ [513](#)

Insert メッセージ [515](#)

Merge Update メッセージ [516](#)

merge メッセージ [515](#), [528](#)

No Action メッセージ [517](#)

PendingInsert メッセージ [517](#)

PendingUpdateXref メッセージ [519](#)

PendingUpdate メッセージ [518](#)

XRefDelete メッセージ [521](#)

XRefSetToDelete メッセージ [522](#)

XREF の更新メッセージ [521](#)

更新メッセージ [520](#)

unmerge メッセージ [519](#)

例 (従来)

accept as unique メッセージ [524](#)

bo delete メッセージ [525](#)

bo set to delete メッセージ [525](#)

Delete メッセージ [514](#), [526](#)

Insert メッセージ [527](#)

Merge Update メッセージ [528](#)

Pending Insert メッセージ [529](#)

Pending Update XREF メッセージ [531](#)

Pending Update メッセージ [530](#)

XREF delete メッセージ [534](#)

XREF set to Delete メッセージ [534](#)

XREF の更新メッセージ [532](#)

更新メッセージ [531](#)

unmerge メッセージ [533](#)

メッセージキュー

削除 [503](#)

受信タイムアウト [499](#)

受信バッチサイズ [499](#)

ステータス [499](#)

追加 [502](#)

プロパティ [502](#)

編集 [503](#)

メッセージキューツール [499](#)

メッセージキューの概要 [293](#), [502](#)

メッセージチェック間隔 [499](#)

メッセージキューサーバー

削除 [502](#)

追加 [501](#)

編集 [501](#)

メッセージキューサーバーについて [500](#)

メッセージスキーマ

ORS 固有、生成とデプロイ [615](#)

メッセージトリガ

概要 [293](#)

削除 [508](#)

状態変更のために有効化 [184](#)

タイプ [505](#)

追加 [507](#)

編集 [508](#)

保留中の更新の有効化 [184](#)

メッセージトリガの概要 [504](#)

も

目的、一致 [416](#)

モデルワークベンチ

概要 [42](#)

ゆ

有効期間延長

例 [162](#)

有効期間短縮

例 [163](#)

ユーザー

ツールアクセス [53](#)

ユーザーイグジット

JAR ファイルの実装 [588](#)

SIF の API を呼び出すには [602](#)

アップロード [588](#)

アンマージ後 [599](#), [600](#)

アンマージ後インタフェース [600](#)

アンマージ前インタフェース [599](#)

一致プロセス [595](#)

ガイドライン [605](#)

概要 [586](#), [617](#)

削除 [589](#)

サポートされる SIF の API [604](#)

ステージプロセス [591](#)

マージ後 [598](#)

マージプロセス [597](#)

ユーザーイグジット

ロード後パラメータ [594](#)

例 [603](#)

ロード後 [594](#)

ロード後ユーザーイグジット

パラメータ [594](#)

ロードプロセス [594](#)

作成 [602](#)

ユーザーオブジェクト

概要 [617](#)

ユーザーオブジェクトレジストリ

概要 [617](#)

カスタム Java クレンジング関数、表示 [619](#)

カスタムボタン関数、表示 [620](#)

起動 [618](#)

ユーザー出口、表示 [618](#)

ユーザー定義カラム

ステージングテーブル [363](#)

ユーザー出口

AssignTasks インタフェース [601](#)

GetAssignableUsersForTask インタフェース [601](#)

JAR ファイル [588](#)

SIF の API を呼び出すには [602](#)

UserExitContext クラス [589](#)

アンマージプロセス [598](#)

マッチ後 [596](#)

マッチ前インタフェース [596](#)

概要 [618](#)

処理 [587](#)

ステージング後 [593](#)

ステージング後インタフェース [593](#)

ステージング前 [592](#)

ステージング前インタフェース [592](#)

タスク管理 [601](#)

表示 [618](#)

マージ後インタフェース [598](#)

マージ前 [596](#)

マッチ後インタフェース [597](#)

ランディング後 [591](#)

ランディング後ユーザー出口インタフェース [591](#)

ロード後インタフェース [595](#)

ロールバック [587](#)

優先キー幅 [405](#)

ユーティリティワークベンチ

概要 [44](#)

よ

用語解説 [675](#)

ら

ランディング後ユーザー出口

インタフェース [591](#)

概要 [591](#)

ランディングテーブル

概要 [303](#)

カラム [299](#)

削除 [301](#)

追加 [300](#)

定義 [80](#)

プロパティ [299](#)

編集 [300](#)

ランディングテーブルの概要 [299](#)

ランディングプロセス

C_REPOS_SYSTEM テーブル [296](#)

外部バッチプロセス [270](#)

概要 [269](#)

管理 [270](#)

ソースシステム [269](#)

抽出-変換-ロード (ETL) ツール [270](#)

ランディングテーブル [269](#)

ランディングテーブルにデータを入力する方法 [270](#)

リアルタイム処理 (API 呼び出し) [270](#)

り

リジェクトテーブル

概要 [303](#)

リポジトリベースオブジェクト (RBO) テーブル [201](#)

リレーション

外部キー関係 [120](#)

詳細 [122](#)

リレーションオブジェクト

概要 [213](#)

リレーションタイプ

概要 [217](#)

削除 [221](#)

作成 [220](#)

編集 [221](#)

例 [219](#)

リレーションベースオブジェクト

外部キーリレーションベースオブジェクト [217](#)

外部キーリレーションベースオブジェクトの作成 [217](#)

概要 [213](#)

更新例 [165](#), [166](#)

削除の例 [169](#), [170](#)

作成 [215](#)

終了の例 [168](#)

すべてのリレーション期間の削除 [170](#)

ベースオブジェクトに戻す [216](#)

変換 [215](#)

リレーション期間の削除 [169](#)

リレーションの更新 [165](#)

リレーションの終了 [168](#)

履歴

有効化 [106](#)

履歴テーブル

相互参照履歴テーブル [105](#)

定義 [80](#)

ベースオブジェクト履歴テーブル [105](#)

有効化 [109](#)

る

ルックアップ
外部キー [369](#)
構成 [369](#)
ルックアップの概要 [369](#)

れ

レコード状態 [173](#)
レコードの状態
概要 [174](#)
レコード
更新 [165](#)
編集 [165](#)
保留中のマッチ [179](#)
レコードバージョン
更新 [165](#)
追加 [164](#)
編集 [165](#)
レコードバージョンの追加
例 [164](#)

ろ

ロード後ユーザーイグジット
概要 [594](#)
ロード後ユーザー
インタフェース [595](#)
ロード時に一致トークンを生成する [574](#)
ロードジョブ
却下されたレコード [551](#)
強制的な更新、概要 [573](#)
ロード時に一致トークンを生成する [574](#)
ロードバッチサイズ [106](#)
ロードプロセス
概要 [272](#)

ロードプロセス (続く)
データ管理の手順 [274](#)
ユーザーイグジット [594](#)
ロードの挿入 [274](#)
ログ、アプリケーションサーバー
設定 [660](#)
レベル [659](#)
ログ、アプリケーションサーバーのデータベース
概要 [659](#)
ログファイルのローリング
概要 [659](#)
ロック
解除 [36](#)
書き込みロック [34](#), [36](#)
クリア [36](#)
サーバーキャッシュ [35](#)
取得 [34](#), [36](#)
排他ロック [34](#), [36](#)
非排他ロック [34](#)
ロックのタイプ [663](#)
ロックの有効期限 [35](#)
ロードバッチ
タイムラインの設定 [160](#)
複数のレコードバージョン [160](#)
例 [161](#)

わ

ワークフロー
レコードの状態 [174](#)
ワークフローエンジン
追加 [179](#)
ワークフローツール [178](#)
ワークベンチビュー
ツール [33](#)
概要 [33](#)