



Informatica®
10.4.0

Developer トランスフォー メーションガイド

Informatica Developer トランスフォーメーションガイド

10.4.0

2019 年 12 月

© 著作権 Informatica LLC 2009, 2020

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 PowerCenter 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: 2020-02-07

目次

序文	30
Informatica のリソース	30
Informatica Network	30
Informatica ナレッジベース	30
Informatica マニュアル	30
Informatica 製品可用性マトリックス	31
Informatica Velocity	31
Informatica Marketplace	31
Informatica グローバルカスタマサポート	31
第 1 章: トランスフォーメーションについて	32
トランスフォーメーションについての概要	32
アクティブなトランスフォーメーション	33
パッシブトランスフォーメーション	33
接続されていないトランスフォーメーション	33
複数ストラテジのトランスフォーメーション	33
トランスフォーメーションの説明	34
ネイティブ環境および非ネイティブ環境でのトランスフォーメーション	37
トランスフォーメーションデータ型の処理	39
Decimal データ型	39
Timestamp with Time Zone	41
Timestamp with Local Time Zone	41
トランスフォーメーションの開発	41
複数グループのトランスフォーメーション	42
複数グループトランスフォーメーションのルールとガイドライン	42
トランスフォーメーションの式	43
式エディタ	44
式内のポート名	44
ポートへの式の追加	45
式内のコメント	45
式の検証	45
テスト式	45
データ型の変換	46
ローカル変数	47
データの一時的な格納と複雑な式の簡素化	47
複数行の値の格納	48
ストアードプロシージャからの値の取得	48
変数ポートの設定のガイドライン	48
変数の初期化	49
ポートのデフォルト値	50

ユーザー定義のデフォルト値.	51
ユーザー定義のデフォルト入力値.	52
デフォルト値の検証.	53
ユーザー定義のデフォルト出力値.	53
デフォルト値の一般ルール.	55
デフォルト値の検証.	56
トレースレベル.	56
再利用可能なトランスフォーメーション.	57
再利用可能なトランスフォーメーションのインスタンスと継承される変更.	57
再利用可能なトランスフォーメーションの編集.	57
再利用可能なトランスフォーメーションのエディタのビュー.	58
再利用不可能なトランスフォーメーション.	58
再利用不可能なトランスフォーメーション用のエディタのビュー.	58
トランスフォーメーションの作成.	58
第2章: トランスフォーメーションポート.	60
トランスフォーメーションポートの概要.	60
ポートの作成.	60
ポートの設定.	61
ポートのリンク.	61
1 対 1 のリンク.	62
1 対多のリンク.	62
ポートの手動リンク.	62
ポートの自動リンク.	62
ポートのリンクに関するルールおよびガイドライン.	63
ポート属性のプロパゲート.	64
依存関係のタイプ.	64
リンクパスの依存関係.	64
暗黙の依存関係.	65
トランスフォーメーションでプロパゲートされるポート属性.	65
Excel からのポートのコピー.	67
Excel でのトランスフォーメーションの編集.	68
Developer Tool へのメタデータのコピー.	68
例: Excel でのトランスフォーメーションの編集.	69
Excel からのコピーに関するルールおよびガイドライン.	69
第3章: トランスフォーメーションキャッシュ.	70
トランスフォーメーションキャッシュの概要.	70
キャッシュタイプ.	71
キャッシュファイル.	71
キャッシュファイルディレクトリ.	72
キャッシュサイズ.	72
自動キャッシュサイズ.	72

固有のキャッシュサイズ.	74
データ統合サービスによるキャッシュサイズの増加.	74
パーティション化されたキャッシュのキャッシュサイズ.	75
キャッシュサイズの最適化.	75
手順 1. トレースレベルを [詳細 - 初期化] に設定する.	75
手順 2. 自動キャッシュモードでマッピングを実行する.	75
手順 3. キャッシングパフォーマンスを分析する.	76
手順 4. 固有のキャッシュサイズを設定する.	76
第 4 章: アドレスバリデータトランスフォーメーション.	78
アドレスバリデータトランスフォーメーションの概要.	79
アドレス参照データ.	79
アドレス参照データの種類.	79
モードとテンプレート.	81
ポートグループとポートの選択.	81
アドレスバリデータトランスフォーメーションの入力ポートグループ.	82
アドレスバリデータトランスフォーメーションの出力ポートグループ.	83
複数インスタンスのポート.	85
アドレス検証プロジェクト.	86
フォーマットされたアドレスと郵便事業者の住所表記基準.	87
部分的アドレスの入力.	88
アドレスバリデータのステータスポート.	89
要素のステータスコードの定義.	90
[アドレス解決コード] 出力ポートの値.	92
[要素入力ステータス] 出力ポートの値.	92
[要素の関連性] 出力ポートの値.	93
[要素の結果ステータス] 出力ポートの値.	94
[拡張要素の結果ステータス] 出力ポートの値.	95
[郵送可能スコア] の出力ポートの値.	96
[照合コード] の出力ポートの値.	97
[ジオコーディングのステータス] の出力ポートの値.	99
アドレスバリデータトランスフォーメーションの全般設定.	100
[設定] ウィンドウのアドレス検証のプロパティ.	101
アドレス検証データのプロパティ.	102
アドレス検証ライセンスのプロパティ.	103
アドレス検証エンジンのプロパティ.	103
アドレス検証の詳細プロパティ.	104
エイリアスの市区町村.	105
エイリアス番地.	105
大文字小文字表記.	105
国.	106
国のタイプ.	106
デフォルトの国.	107

住所重複時の優先順位.	108
要素の略式表記.	108
実行インスタンス.	108
フレキシブル範囲拡大.	109
Geocode データ型.	110
グローバル最大フィールド長.	110
グローバル優先記述子.	111
入力形式の種類.	111
国を含む入力形式.	111
行セパレータ.	112
一致する代替用語.	112
拡張アーカイブのマッチング.	113
一致するスコープ.	113
最大結果カウント.	114
モード.	114
最適化レベル.	114
出力形式の種類.	115
国を含む出力形式.	115
優先される言語.	115
優先されるスクリプト.	121
拡大する範囲.	122
無効なアドレスの標準化.	123
トレースレベル.	124
認証レポート.	124
AMAS レポートのフィールド.	124
CASS レポートのフィールド.	125
SendRight レポート.	126
SERP レポートのフィールド.	126
アドレスバリデータトランスフォーメーションの設定.	127
アドレスバリデータトランスフォーメーションへのポートの追加.	127
ユーザー定義テンプレートの作成.	128
アドレスバリデータのモデルの定義.	128
認証レポートの定義.	129
非ネイティブ環境でのアドレスバリデータトランスフォーメーション.	129
Blaze エンジンでのアドレスバリデータトランスフォーメーション.	130
Spark エンジンでのアドレスバリデータトランスフォーメーション.	130
第 5 章 : アグリゲータトランスフォーメーション.	131
アグリゲータトランスフォーメーションの概要.	131
動的マッピングでのアグリゲータトランスフォーメーション.	132
アグリゲータトランスフォーメーションの開発.	132
アグリゲータトランスフォーメーションのポート.	132
集計式.	133

集計関数.	134
ネストされた集計関数.	134
集計式の条件句.	135
グループ化ポート.	135
グループ化ポートの設定.	136
グループ化パラメータ.	137
グループ化ポートのデフォルト値.	137
非集計式.	137
アグリゲータキャッシュ.	137
アグリゲータトランスフォーメーションの [ソート済み入力] オプション.	138
[ソート済み入力] オプションの条件.	138
アグリゲータトランスフォーメーションでのデータのソート.	139
アグリゲータトランスフォーメーションの詳細プロパティ.	140
再利用可能なアグリゲータトランスフォーメーションの作成.	141
再利用不可能なアグリゲータトランスフォーメーションの作成.	141
アグリゲータトランスフォーメーションに関するヒント.	142
アグリゲータトランスフォーメーションのトラブルシューティング.	142
非ネイティブ環境でのアグリゲータトランスフォーメーション.	143
Blaze エンジンでのアグリゲータトランスフォーメーション.	143
Spark エンジンでのアグリゲータトランスフォーメーション.	144
Databricks Spark エンジンでのアグリゲータトランスフォーメーション.	144

第 6 章 : 関連付けトランスフォーメーション..... 146

関連付けトランスフォーメーションの概要.	146
メモリ割り当て.	147
関連付けトランスフォーメーションの詳細プロパティ.	148

第 7 章 : 不良レコードの例外トランスフォーメーション..... 149

不良レコードの例外トランスフォーメーションの概要.	149
不良レコードの例外の出力レコードタイプ.	150
不良レコードの例外管理プロセスフロー.	151
不良レコードの例外マッピング.	151
不良レコードの例外の品質に関する問題.	153
ヒューマンタスク.	153
不良レコードの例外のポート.	154
不良レコードの例外トランスフォーメーションの入力ポート.	154
不良レコードの例外トランスフォーメーションの出力.	155
不良レコードの例外の [設定] ビュー.	155
不良レコードテーブルおよび問題テーブルの生成.	157
不良レコードの例外の問題の割り当て.	157
品質の問題に対するポートの割り当て.	158
例外トランスフォーメーションの詳細プロパティ.	158
不良レコードの例外トランスフォーメーションの設定.	158

不良レコードの例外マッピングの例.	159
不良レコードの例外のマプレット.	159
不良レコードの例外の入力グループの例.	160
不良レコードの例外の設定例.	161
不良レコードの例外のマッピングの出力例.	162
 第 8 章 : 大文字小文字変換プログラムトランスフォーメーション.	165
大文字小文字変換プログラムトランスフォーメーションの概要.	165
大文字小文字ストラテジのプロパティ.	165
大文字小文字変換プログラムストラテジの設定.	166
大文字小文字変換プログラムトランスフォーメーションの詳細プロパティ.	167
非ネイティブ環境での大文字小文字変換トランスフォーメーション.	167
 第 9 章 : 分類子トランスフォーメーション.	168
分類子トランスフォーメーションの概要.	168
分類子モデル.	169
分類子アルゴリズム.	169
分類子トランスフォーメーションのオプション.	169
分類子ストラテジ.	170
分類子トランスフォーメーションの詳細プロパティ.	170
分類子ストラテジの設定.	171
分類子分析の例.	171
分類子マッピングの作成.	172
入力データサンプル.	173
データソースの設定.	173
分類子トランスフォーメーションの設定.	173
Router トランスフォーメーションの設定.	173
データターゲットの設定.	175
分類子マッピングの結果.	175
非ネイティブ環境での分類子トランスフォーメーション.	176
 第 10 章 : 比較トランスフォーメーション.	177
比較トランスフォーメーションの概要.	177
フィールド一致ストラテジ.	177
バイグラム.	178
ハミング距離.	178
エディット距離.	178
Jaro 距離.	179
ハミング距離の反転.	179
ID マッチングストラテジ.	180
比較ストラテジの設定.	180
比較トランスフォーメーションの詳細プロパティ.	181
非ネイティブ環境での比較トランスフォーメーション.	181

第 11 章 : 統合トランスフォーメーション.....	182
統合トランスフォーメーションの概要.....	182
統合マッピング.....	183
Consolidation トランスフォーメーションのポート.....	183
統合トランスフォーメーションのビュー.....	183
統合トランスフォーメーションの [ストラテジ] ビュー.....	184
統合トランスフォーメーションの詳細プロパティ.....	184
キャッシュファイルサイズ.....	185
単純ストラテジ.....	186
行ベースストラテジ.....	187
詳細ストラテジ.....	187
単純な統合関数.....	188
CONSOL_AVG.....	188
CONSOL_LONGEST.....	188
CONSOL_MAX.....	189
CONSOL_MIN.....	190
CONSOL_MOSTFREQ.....	190
CONSOL_MOSTFREQ_NB.....	191
CONSOL_SHORTEST.....	191
行ベースの統合関数.....	192
CONSOL_GETROWFIELD.....	192
CONSOL_MODELEXACT.....	193
CONSOL_MOSTDATA.....	193
CONSOL_MOSTFILLED.....	194
統合マッピングの例.....	195
入力.....	195
キージェネレータトランスフォーメーション.....	196
統合トランスフォーメーション.....	196
統合マッピング出力.....	196
統合トランスフォーメーションの設定.....	196
非ネイティブ環境での統合トランスフォーメーション.....	197
Blaze エンジンでの統合トランスフォーメーション.....	197
Spark エンジンでの統合トランスフォーメーション.....	197
Databricks Spark エンジンでの統合トランスフォーメーション.....	197
第 12 章 : 1028 データマスキングトランスフォーメーション.....	198
データマスキングトランスフォーメーションの概要.....	198
マスキング方法.....	199
ランダムマスキング.....	199
式マスキング.....	201
キーマスキング.....	203
置換マスキング.....	204

依存マスキング.....	207
トークン化のマスキング.....	209
マスキングルール.....	209
マスク形式.....	210
ソース文字列の文字.....	211
結果文字列の置換文字.....	211
範囲.....	212
ブラー.....	212
特殊マスク形式.....	213
クレジットカード番号のマスキング.....	214
電子メールアドレスマスキング.....	214
高度な電子メールマスキング.....	214
IP アドレスマスキング.....	215
電話番号マスキング.....	216
社会保障番号（SSN）のマスキング.....	216
URL アドレスマスキング.....	216
社会保険番号のマスキング.....	217
デフォルト値ファイル.....	217
データマスキングトランスフォーメーションの設定.....	218
データ統合サービスの設定.....	218
データマスキングトランスフォーメーションの作成.....	218
ポートの定義.....	219
各ポートのデータマスキングの設定.....	219
マスクされたデータのプレビュー.....	219
データマスキングトランスフォーメーションランタイムプロパティ.....	220
データマスキングの例.....	221
Read_Customer Data.....	221
カスタマデータマスキングトランスフォーメーション.....	222
カスタマテストデータ結果.....	222
データマスキングトランスフォーメーションの詳細プロパティ.....	223
非ネイティブ環境でのデータマスキングトランスフォーメーション.....	223
Blaze エンジンでのデータマスキングトランスフォーメーション.....	223
Spark エンジンでのデータマスキングトランスフォーメーション.....	224
第 13 章 : データプロセッサトランスフォーメーション.....	226
データプロセッサトランスフォーメーションの概要.....	226
データプロセッサトランスフォーメーションのビュー.....	227
データプロセッサトランスフォーメーションのポート.....	228
データプロセッサトランスフォーメーションの入力ポート.....	228
データプロセッサトランスフォーメーションの出力ポート.....	229
パススルーポート.....	230
スタートアップコンポーネント.....	230
参照.....	231

データプロセッサトランスフォーメーションの設定.	232
文字エンコード.	232
文字エンコードのルールおよびガイドライン.	234
出力設定.	234
処理設定.	236
XMap 設定.	237
XML 出力設定.	237
イベント.	239
イベントタイプ.	239
データプロセッサイベントビュー.	240
ログ.	240
設計時イベントログ.	241
ランタイムイベントログ.	241
データプロセッサイベントビューでのイベントログの表示.	241
ユーザーログ.	242
データプロセッサトランスフォーメーションの開発.	242
データプロセッサトランスフォーメーションの作成.	243
スキーマオブジェクトの選択.	243
空のデータプロセッサトランスフォーメーションへのオブジェクトの作成.	244
ポートの作成.	246
トランスフォーメーションのテスト.	246
データプロセッサトランスフォーメーションのインポートとエクスポート.	247
サービスとしてのデータプロセッサトランスフォーメーションのエクスポート.	247
複数の Data Transformation サービスのインポート.	248
Data Transformation サービスのインポート.	248
データプロセッサトランスフォーメーションでマッピングを PowerCenter にエクスポート.	248
非ネイティブ環境でのデータプロセッサトランスフォーメーション.	249
第 14 章: ディシジョントランスフォーメーション.	250
ディシジョントランスフォーメーションの概要.	250
ディシジョントランスフォーメーションの関数.	251
ディシジョントランスフォーメーションの条件文.	253
ディシジョントランスフォーメーションの演算子.	254
ディシジョントランスフォーメーションの NULL 処理.	255
ディシジョンストラテジの設定.	255
ディシジョントランスフォーメーションの詳細プロパティ.	256
非ネイティブ環境でのディシジョントランスフォーメーション.	256
第 15 章: 重複レコードの例外トランスフォーメーション.	257
重複レコードの例外トランスフォーメーションの概要.	257
重複レコードの例外プロセスフロー.	258
重複レコードの例外.	258
重複レコードの例外の [設定] ビュー.	259

重複レコードテーブルの生成.	260
ポート.	260
重複レコードの例外トランスフォーメーションの入力ポート.	261
重複レコードの例外トランスフォーメーションの出力ポート.	262
ポートの作成.	262
重複レコードの例外トランスフォーメーションの詳細プロパティ.	263
重複レコードの例外マッピングの例.	264
重複レコードの例外マッピング.	264
一致トランスフォーメーション.	265
重複レコードの例外の入力グループ.	265
重複レコードの例外の [設定] ビューの例.	266
標準出力テーブルレコード.	267
クラスタ出力.	268
重複レコードの例外トランスフォーメーションの作成.	269

第 16 章 : 式トランスフォーメーション. 271

式トランスフォーメーションの概要.	271
式トランスフォーメーションのポート.	272
テスト式.	273
サンプルデータの日付形式の文字列.	274
式のテスト.	274
ポートセクタ.	274
ポートセクタの設定.	275
選択ルール.	275
ポートセクタの作成.	276
ウィンドウ化.	277
ウィンドウ化構成.	278
動的式.	281
出力ポート設定.	282
動的式の作成.	284
式トランスフォーメーションの詳細プロパティ.	285
非ネイティブ環境での式トランスフォーメーション.	286
Blaze エンジンでの式トランスフォーメーション.	286
Spark エンジンでの式トランスフォーメーション.	286
Databricks Spark エンジンでの式トランスフォーメーション.	287

第 17 章 : フィルタトランスフォーメーション. 288

フィルタトランスフォーメーションの概要.	288
動的マッピングでのフィルタトランスフォーメーション.	289
フィルタ条件.	290
フィルタ条件のパラメータ化.	290
NULL 値を含む行のフィルタ.	292
フィルタトランスフォーメーションの詳細プロパティ.	292

フィルタトランスフォーメーションのパフォーマンスのヒント.	292
非ネイティブ環境でのフィルタトランスフォーメーション.	292
Blaze エンジンでのフィルタトランスフォーメーション.	293

第 18 章 : 階層型からリレーショナルへのトランスフォーメーション. . 294

階層型からリレーショナルへのトランスフォーメーションの概要.	294
例 - 階層型からリレーショナルへのトランスフォーメーション.	295
出力リレーショナルポートと [概要] ビュー.	296
階層型からリレーショナルへのトランスフォーメーションのポート.	297
スキーマ参照.	298
ポート設定.	298
階層型からリレーショナルへのトランスフォーメーションの開発.	299
階層型からリレーショナルへのトランスフォーメーションの作成.	299
ポートとマッピングの設定.	299
トランスフォーメーションのテスト.	300

第 19 章 : Java トランスフォーメーション. 301

Java トランスフォーメーションの概要.	301
再利用可能および再利用不可能な Java トランスフォーメーション.	302
アクティブ Java トランスフォーメーションとパッシブ Java トランスフォーメーション.	302
データ型の変換.	302
Spark エンジンでの複合データ型の変換.	303
Java トランスフォーメーションの設計.	305
Java トランスフォーメーションのポート.	305
ポートの作成.	305
デフォルトポート値の設定.	306
Java トランスフォーメーションの詳細プロパティ.	306
Developer ツールクライアントのクラスパスの設定.	308
データ統合サービスのクラスパスの設定.	309
Java コードの開発.	309
Java コードスニペットの作成.	311
Java パッケージのインポート.	312
Helper コードの定義.	312
Java トランスフォーメーションの Java のプロパティ.	313
[インポート] タブ.	314
[ヘルパ] タブ.	314
[入力時] タブ.	314
[最後] タブ.	315
[関数] タブ.	315
[コード全体] タブ.	316
Java トランスフォーメーションによるフィルタの最適化.	316
Java トランスフォーメーションによる初期選択の最適化.	316
Java トランスフォーメーションによるプッシュイン最適化.	317

Java トランスフォーメーションの作成.	318
再利用可能な Java トランスフォーメーションの作成.	319
再利用不可能な Java トランスフォーメーションの作成.	319
Java トランスフォーメーションのコンパイル.	320
Java トランスフォーメーションのトラブルシューティング.	320
コンパイルエラーのソースの検出.	321
コンパイルエラーの原因の特定.	321
Struct データへの変換例.	322
非ネイティブ環境での Java トランスフォーメーション.	325
Blaze エンジンでの Java トランスフォーメーション.	325
Spark エンジンでの Java トランスフォーメーション.	325
第 20 章 : Java トランスフォーメーション API のリファレンス.	328
Java トランスフォーメーション API メソッドの概要.	328
defineJExpression.	329
failSession.	330
generateRow.	330
getInRowType.	331
getMetadata.	332
incrementErrorCount.	332
invokeJExpression.	333
isNull.	334
logError.	334
logInfo.	335
resetNotification.	335
setNull.	336
storeMetadata.	337
第 21 章 : Java 式.	338
Java 式の概要.	338
式の関数タイプ.	339
[関数の定義] ダイアログボックスを使用した式の定義.	339
手順 1. 関数の設定.	340
手順 2. 式の作成と検証.	340
手順 3. 式の Java コードの生成.	340
[関数の定義] ダイアログボックスを使用した式の作成と Java コードの生成.	340
Java 式のテンプレート.	341
単純なインタフェースに関する作業.	341
invokeJExpression.	341
単純なインタフェースの例.	342
高度なインタフェースに関する作業.	343
高度なインタフェースを使用した式の呼び出し.	343
高度なインタフェースに関する作業のルールとガイドライン.	343

EDatatype クラス.	344
JExprParamMetadata クラス.	344
defineJExpression.	345
JExpression クラス.	346
高度なインタフェースの例.	346
JExpression クラス API リファレンス.	347
getBytes.	347
getDouble.	347
getInt.	348
getLong.	348
getResultDataType.	348
getResultMetadata.	348
getStringBuffer.	349
呼び出し.	349
isResultNull.	349
第 22 章 : ジョイナトランスフォーメーション.	350
ジョイナトランスフォーメーションの概要.	350
ジョイナトランスフォーメーションの詳細プロパティ.	351
ジョイナキャッシュ.	352
ジョイナトランスフォーメーションポート.	353
動的マッピングでのジョイナトランスフォーメーション.	354
ジョイナトランスフォーメーションのポートセクタ.	354
選択ルール.	355
ポートセクタの作成.	356
結合条件の定義.	357
単純条件タイプ.	358
詳細条件タイプ.	358
結合条件のポートセクタ.	359
結合条件の動的ポート.	360
式パラメータ.	360
結合タイプ.	361
ノーマル結合.	361
マスタ外部結合.	362
明細外部結合.	362
完全外部結合.	362
ジョイナトランスフォーメーションでのソート済み入力.	363
ソート順の設定.	363
マッピングへのトランスフォーメーションの追加.	364
結合条件のルールとガイドライン.	364
結合条件とソート順の例.	365
同じソースのデータの結合.	367
同じパイプラインの 2 つのブランチの結合.	367

同じソースの2つのインスタンスの結合.....	368
同じソースのデータ結合のガイドライン.....	369
ソースパイプラインのブロック.....	369
未ソートジョイナトランスフォーメーション.....	370
ソート済みジョイナトランスフォーメーション.....	370
ジョイナトランスフォーメーションのパフォーマンスのヒント.....	370
ジョイナトランスフォーメーションのルールとガイドライン.....	371
非ネイティブ環境でのジョイナトランスフォーメーション.....	371
Blaze エンジンでのジョイナトランスフォーメーション.....	372
Spark エンジンでのジョイナトランスフォーメーション.....	372
Databricks Spark エンジンでのジョイナトランスフォーメーション.....	372
第 23 章: キージェネレータトランスフォーメーション.....	373
キージェネレータトランスフォーメーションの概要.....	373
Soundex ストラテジ.....	374
Soundex ストラテジのプロパティ.....	374
文字列ストラテジ.....	374
文字列ストラテジのプロパティ.....	375
NYSIIS ストラテジ.....	375
キージェネレータの出力ポート.....	375
グループ化ストラテジの設定.....	376
キー作成のプロパティ.....	376
キージェネレータトランスフォーメーションの詳細プロパティ.....	377
非ネイティブ環境でのキージェネレータトランスフォーメーション.....	377
第 24 章: ラベラトランスフォーメーション.....	378
ラベラトランスフォーメーションの概要.....	378
ラベラトランスフォーメーションを使用する状況.....	379
ラベラトランスフォーメーションでの参照データの使用.....	380
文字セット.....	380
確率モデル.....	381
参照テーブル.....	381
正規表現.....	381
トークンセット.....	381
ラベラトランスフォーメーションのストラテジ.....	382
文字ラベル適用操作.....	382
トークンラベル適用操作.....	382
ラベラトランスフォーメーションのポート.....	382
文字ラベル適用のプロパティ.....	383
全般プロパティ.....	383
参照テーブルのプロパティ.....	383
文字セットのプロパティ.....	384
フィルタのプロパティ.....	384

トークンラベル適用のプロパティ.....	385
全般プロパティ.....	385
トークンセットのプロパティ.....	386
カスタムラベルプロパティ.....	386
確率的な一致のプロパティ.....	387
参照テーブルのプロパティ.....	387
文字ラベル適用ストラテジの設定.....	388
トークンラベル適用ストラテジの設定.....	388
ラベラトランスフォーメーションの詳細プロパティ.....	389
非ネイティブ環境でのラベラトランスフォーメーション.....	389
第 25 章: ルックアップトランスフォーメーション.....	390
ルックアップトランスフォーメーションの概要.....	390
接続されたルックアップと接続されていないルックアップ.....	391
接続されたルックアップ.....	393
接続されていないルックアップ.....	393
ルックアップトランスフォーメーションの開発.....	394
ルックアップクエリ.....	394
デフォルトのルックアップクエリ.....	394
ルックアップクエリの SQL オーバーライド.....	395
SQL オーバーライドクエリのパラメータ.....	395
予約語.....	396
ルックアップクエリのオーバーライドのガイドライン.....	396
ルックアップクエリの上書き.....	397
ルックアップソースフィルタ.....	397
ルックアップでのソース行のフィルタリング.....	397
ルックアップ条件.....	398
ルックアップ条件の設定.....	399
ルックアップトランスフォーメーションの条件のルールとガイドライン.....	400
ルックアップキャッシュ.....	401
クエリのプロパティ.....	401
動的マッピングでのルックアップトランスフォーメーション.....	402
動的ポートの定義.....	402
ルックアップソースの変更.....	403
ルックアップソースのパラメータ化.....	404
パラメータを含むルックアップソース.....	405
重複データオブジェクトでのパラメータの設定.....	405
ポートセクタ.....	407
ポートセクタの設定.....	408
選択ルール.....	409
ルックアップ条件のパラメータ化.....	410
ポートセクタの作成.....	411
[ランタイム] プロパティ.....	413

詳細プロパティ.....	414
再利用可能なルックアップトランスフォーメーションの作成.....	415
再利用不可能なルックアップトランスフォーメーションの作成.....	416
接続されていないルックアップトランスレーションの作成.....	417
接続されていないルックアップの例.....	418
非ネイティブ環境でのルックアップトランスフォーメーション.....	420
Blaze エンジンでのルックアップトランスフォーメーション.....	420
Spark エンジンでのルックアップトランスフォーメーション.....	420
Databricks Spark エンジンでのルックアップトランスフォーメーション.....	422

第 26 章: ルックアップキャッシュ..... 423

ルックアップキャッシュの概要.....	423
ルックアップキャッシュのタイプ.....	424
キャッシュを使用しないルックアップ.....	425
静的ルックアップキャッシュ.....	425
永続ルックアップキャッシュ.....	426
永続ルックアップキャッシュの再構築.....	426
動的ルックアップキャッシュ.....	427
共有ルックアップキャッシュ.....	427
ルックアップキャッシュの共有のルールおよびガイドライン.....	427
キャッシュの比較.....	429
ルックアップのキャッシュのパーティション化.....	429

第 27 章: 動的ルックアップキャッシュ..... 430

動的ルックアップキャッシュの概要.....	430
動的ルックアップキャッシュへの使用.....	431
動的ルックアップキャッシュプロパティ.....	431
動的ルックアップキャッシュおよび出力値.....	433
ルックアップトランスフォーメーションの値.....	433
ルックアップトランスフォーメーションの値の例.....	434
SQL オーバーライドおよび動的ルックアップキャッシュ.....	436
動的ルックアップキャッシュのマッピング設定.....	437
挿入でなければ更新.....	437
更新でなければ挿入.....	438
動的ルックアップキャッシュおよびターゲットの同期.....	439
条件付き動的ルックアップキャッシュの更新.....	439
条件付きの動的ルックアップキャッシュの処理.....	440
条件付き動的ルックアップキャッシュの設定.....	440
式の結果を使用した動的キャッシュの更新.....	440
式の値が NULL.....	440
式の処理.....	441
動的キャッシュの更新のための式の設定.....	441
動的ルックアップキャッシュの例.....	441

動的ルックアップキャッシュのルールとガイドライン.	442
-----------------------------------	-----

第 28 章: 一致トランスフォーメーション. 444

一致トランスフォーメーションの概要.	444
照合分析.	445
カラム分析.	445
単一ソースでの分析とデュアルソースでの分析.	446
フィールド一致分析と ID 照合分析.	446
照合分析でのグループ化.	447
一致ペアとクラスタ.	448
マッチ率の計算.	448
加重スコア.	449
NULL のマッチ率.	449
クラスタ出力オプション.	449
クラスタ分析でのドライバスコアとリンクスコア.	451
マスターデータの分析.	452
マッピングの再利用.	453
ID 照合分析と永続インデックスデータ.	453
永続インデックスデータに関するルールとガイドライン.	453
照合マッピングのパフォーマンス.	454
一致クラスタ分析のデータの表示.	455
照合パフォーマンス分析のデータの表示.	455
ID 照合分析での照合パフォーマンス.	456
ID インデックスデータ用のデータストアの作成.	457
単一ソース分析でのインデックスデータストアの使用.	458
一致トランスフォーメーションのビュー.	459
一致トランスフォーメーションのポート.	460
一致トランスフォーメーションの入力ポート.	460
一致トランスフォーメーションの出力ポート.	461
持続ステータスコードと持続ステータス記述.	462
出力ポートと照合出力の選択.	464
一致マッピング.	465
一致マッピングの作成.	465
一致マッピングの使用.	466
照合分析操作の設定.	466
非ネイティブ環境での一致トランスフォーメーション.	467
Blaze エンジンでの一致トランスフォーメーション.	467
Spark エンジンでの一致トランスフォーメーション.	467

第 29 章: フィールド分析での一致トランスフォーメーション. 468

フィールド一致分析.	468
フィールド一致分析の処理フロー.	468
フィールド照合タイプのオプション.	469

フィールド一致ストラテジ	469
フィールド一致のアルゴリズム	470
フィールド一致ストラテジのプロパティ	472
フィールド一致の出力オプション	472
照合出力のタイプ	472
照合出力のプロパティ	473
フィールド一致の詳細プロパティ	474
フィールド一致分析の例	475
マッピングの作成	475
入力データサンプル	476
キージェネレータトランスフォーメーションの設定	476
一致トランスフォーメーションの設定	477
データビューアの実行	479
まとめ	479
第 30 章 : ID 分析での一致トランスフォーメーション	480
ID 照合分析	480
ID 照合分析の処理フロー	481
ID 一致タイプのプロパティ	481
インデックスディレクトリとキャッシュディレクトリのプロパティ	484
永続方法パラメータ	485
ID 照合ストラテジ	485
ID 照合のアルゴリズム	485
ID 照合ストラテジのプロパティ	487
ID 照合の出力オプション	487
照合出力のタイプ	487
照合出力のプロパティ	488
ID 照合の詳細プロパティ	490
永続インデックスのケーススタディ	491
ID 照合分析の例	493
マッピングの作成	494
入力データサンプル	495
式トランスフォーメーションの設定	495
一致トランスフォーメーションの設定	495
データビューアの実行	499
まとめ	500
第 31 章 : ノーマライザトランスフォーメーション	501
ノーマライザトランスフォーメーションの概要	501
複数出現フィールド	502
生成カラム ID	502
複数出現レコード	503
入力階層の定義	504

ノーマライザトランスフォーメーションの入力ポート.....	505
フィールドのマージ.....	506
フィールドのフラット化.....	507
ノーマライザトランスフォーメーションの出力グループとポート.....	512
出力グループの作成.....	513
出力グループの更新.....	514
出力グループのキーの生成.....	515
ノーマライザトランスフォーメーションの詳細プロパティ.....	515
第 1 レベルの出力グループの生成.....	516
ノーマライザトランスフォーメーションの作成.....	516
アップストリームのソースからのノーマライザトランスフォーメーションの作成.....	517
ノーマライザマッピングの例.....	517
ノーマライザのマッピング例.....	518
ノーマライザの定義例.....	518
ノーマライザの入力グループおよび出力グループ例.....	519
ノーマライザのマッピング出力例.....	520
非ネイティブ環境でのノーマライザトランスフォーメーション.....	521
第 32 章: マージトランスフォーメーション.....	522
マージトランスフォーメーションの概要.....	522
マージストラテジの設定.....	522
マージトランスフォーメーションの詳細プロパティ.....	523
非ネイティブ環境でのマージトランスフォーメーション.....	523
第 33 章: パーサートランスフォーメーション.....	524
パーサートランスフォーメーションの概要.....	524
パーサートランスフォーメーションのモード.....	525
パーサートランスフォーメーションを使用するとき.....	525
パーサートランスフォーメーションでの参照データの使用.....	526
パターンセット.....	527
確率モデル.....	527
参照テーブル.....	528
正規表現.....	528
トークンセット.....	528
トークン解析操作.....	528
トークン解析ポート.....	529
トークン解析のプロパティ.....	529
全般プロパティ.....	530
確率モデルのプロパティ.....	530
参照テーブルのプロパティ.....	531
トークンセットのプロパティ.....	531
パターンベースの解析モード.....	532
パターンベースの解析ポート.....	532

トークン解析ストラテジの設定.	533
パターン解析ストラテジの設定.	533
パーサートランスフォーメーションの詳細プロパティ.	534
非ネイティブ環境でのパーサートランスフォーメーション.	534

第 34 章: Python トランスフォーメーション. 536

第 35 章: ランクトランスフォーメーション. 537

ランクトランスフォーメーションの概要.	537
文字列値のランク付け.	538
ランクトランスフォーメーションのプロパティ.	538
動的マッピングでのランクトランスフォーメーション.	538
ランクトランスフォーメーションのポート.	538
ランクインデックス.	539
ランクポート.	540
グループ化ポートの定義.	540
グループ化パラメータ.	541
ランクキャッシュ.	541
ランクトランスフォーメーションの詳細プロパティ.	542
非ネイティブ環境でのランクトランスフォーメーション.	543
Blaze エンジンでのランクトランスフォーメーション.	543
Spark エンジンでのランクトランスフォーメーション.	543
Databricks Spark エンジンでのランクトランスフォーメーション.	544

第 36 章: 読み取りトランスフォーメーション. 545

読み取りトランスフォーメーションの概要.	545
読み取りトランスフォーメーションのプロパティ.	546
全般プロパティ.	547
データオブジェクトのプロパティ.	547
クエリのプロパティ.	547
ランタイムプロパティ.	548
ソースのプロパティ.	548
詳細プロパティ.	548
リレーショナルデータオブジェクトの同期.	549
ソースデータオブジェクトの変更.	549
読み取りトランスフォーメーションのパラメータ化.	551
読み取りトランスフォーメーションのパラメータ.	551
制約.	552
読み取りトランスフォーメーションの作成.	553
マッピングエディタでの読み取りトランスフォーメーションの作成.	553

第 37 章: リレーショナルから階層型へのトランスフォーメーション. . 555

リレーショナルから階層型へのトランスフォーメーションの概要.	555
--	-----

例 - リレーショナルから階層型へのトランスフォーメーション	556
入力リレーショナルポートと「概要」ビュー	558
リレーショナルから階層型へのトランスフォーメーションのポート	558
スキーマ参照	559
リレーショナルから階層型へのトランスフォーメーションの開発	559
リレーショナルから階層型へのトランスフォーメーションの作成	559
ポートの作成	559

第 38 章: REST Web サービスコンシューマトランスフォーメーション 561

REST Web サービスコンシューマトランスフォーメーションの概要	561
REST Web サービスコンシューマトランスフォーメーションの処理	563
REST Web サービスコンシューマトランスフォーメーションの設定	563
メッセージの設定	563
リソースの識別	564
HTTP メソッド	565
HTTP Get メソッド	565
HTTP Post メソッド	566
HTTP Put メソッド	566
HTTP Delete メソッド	567
REST Web サービスコンシューマトランスフォーメーションのポート	568
入力ポート	568
出力ポート	568
パススルーポート	568
引数ポート	569
URL ポート	569
HTTP ヘッダポート	569
クッキーポート	570
出力 XML ポート	570
応答コードポート	570
REST Web サービスコンシューマトランスフォーメーションの入力マッピング	570
入力ポートを要素にマップするためのルールとガイドライン	571
メソッド入力への入力ポートのマッピング	571
REST Web サービスコンシューマトランスフォーメーションの出力マッピング	573
要素を出力ポートにマップするためのルールとガイドライン	573
ビューのカスタマイズのオプション	574
出力ポートへのメソッド出力のマッピング	574
REST Web サービスコンシューマトランスフォーメーションの詳細プロパティ	575
REST Web サービスコンシューマトランスフォーメーションの作成	576
REST Web サービスコンシューマトランスフォーメーションの作成	576
配列を含む JSON 応答メッセージの解析	577
JSON 応答メッセージの例	577
応答メッセージ内の名前なし配列	577

第 39 章 : ルータトランスフォーメーション.....	579
ルータトランスフォーメーションの概要.....	579
動的マッピングでのルータトランスフォーメーション.....	580
グループに関する作業.....	581
入力グループ.....	581
出力グループ.....	581
グループフィルタ条件の使用.....	581
グループフィルタ条件の動的ポート.....	583
グループフィルタのパラメータ化.....	583
グループの追加.....	584
ポートに関する作業.....	584
マッピング内のルータトランスフォーメーションの接続.....	585
ルータトランスフォーメーションの詳細プロパティ.....	585
非ネイティブ環境でのルータトランスフォーメーション.....	586
 第 40 章 : シーケンスジェネレータトランスフォーメーション.....	 587
シーケンスジェネレータトランスフォーメーションの概要.....	587
シーケンスジェネレータのポート.....	587
パススルーポート.....	588
NEXTVAL ポート.....	588
シーケンスジェネレータトランスフォーメーションのプロパティ.....	590
開始値.....	590
終了値.....	590
増分値.....	591
値の範囲内でのサイクル.....	591
リセット.....	591
行順序の保持.....	591
シーケンスデータオブジェクト.....	592
シーケンスデータオブジェクトの作成.....	592
シーケンスジェネレータトランスフォーメーションの作成.....	595
FAQ（よくある質問）.....	596
シーケンスジェネレータトランスフォーメーション非ネイティブ環境で.....	597
Blaze エンジンでのシーケンスジェネレータトランスフォーメーション.....	597
Spark エンジンでのシーケンスジェネレータトランスフォーメーション.....	597
 第 41 章 : ソータートランスフォーメーション.....	 598
ソータートランスフォーメーションの概要.....	598
動的マッピングでのソータートランスフォーメーション.....	599
ソータートランスフォーメーションの開発.....	599
ソータートランスフォーメーションのポート.....	600
[ソート] タブ.....	600
ソートキーの設定.....	600

ソートキーのパラメータ化.....	601
ソータトランスフォーメーションの詳細プロパティ.....	603
ソーターキャッシュ.....	603
ソーターキャッシュの最適化.....	604
ソータートランスフォーメーションの作成.....	604
再利用可能なソータートランスフォーメーションの作成.....	604
再利用不可能なソータートランスフォーメーションの作成.....	605
ソータートランスフォーメーションの例.....	605
非ネイティブ環境でのソータートランスフォーメーション.....	606
Blaze エンジンでのソータートランスフォーメーション.....	606
Spark エンジンでのソータートランスフォーメーション.....	607
Databricks Spark エンジンでのソータートランスフォーメーション.....	608
第 42 章 : SQL トランスフォーメーション.....	610
SQL トランスフォーメーションの概要.....	610
SQL トランスフォーメーションのポート.....	611
入力ポート.....	611
出力ポート.....	612
パススルーポート.....	613
SQLException ポート.....	614
影響を受けた行の数.....	615
SQL トランスフォーメーションの詳細プロパティ.....	615
SQL トランスフォーメーションクエリ.....	617
SQL クエリの定義.....	618
入力行と出力行のカーディナリティ.....	619
クエリ文の処理.....	620
ポート設定.....	620
最大出力行数.....	621
エラー行.....	621
SQL エラー時に続行.....	622
SQL トランスフォーメーションによるフィルタの最適化.....	623
SQL トランスフォーメーションを使用した初期選択の最適化.....	623
SQL トランスフォーメーションによるプッシュイン最適化.....	623
SQL クエリを使った SQL トランスフォーメーションの例.....	624
論理データオブジェクトマッピング.....	625
Salary テーブル.....	625
Employee テーブル.....	625
SQL トランスフォーメーション.....	626
出力.....	627
ストアドプロシージャ.....	628
ストアドプロシージャのための SQL トランスフォーメーションのポート.....	629
ストアドプロシージャの結果セット.....	630
ストアドプロシージャの例.....	632

SQL トランスフォーメーションの接続.	633
接続名パラメータの作成.	633
SQL トランスフォーメーションの手動による作成.	634
ストアドプロシージャから SQL トランスフォーメーションの作成.	634
第 43 章 : 標準化トランスフォーメーション.	636
標準化トランスフォーメーションの概要.	636
標準化ストラテジ.	636
標準化のプロパティ.	637
標準化ストラテジの設定.	638
標準化トランスフォーメーションの詳細プロパティ.	638
非ネイティブ環境での標準化トランスフォーメーション.	639
第 44 章 : 共有体トランスフォーメーション.	640
共有体トランスフォーメーションの概要.	640
グループおよびポート.	641
共有体トランスフォーメーションの詳細プロパティ.	642
共有体トランスフォーメーションの処理.	642
共有体トランスフォーメーションの作成.	642
再利用可能な共有体トランスフォーメーションの作成.	642
再利用不可能な共有体トランスフォーメーションの作成.	643
非ネイティブ環境での共有体トランスフォーメーション.	643
ストリーミングマッピングでの共有体トランスフォーメーション.	643
Databricks Spark エンジンでの共有体トランスフォーメーション.	643
第 45 章 : アップデートストラテジトランスフォーメーション.	644
アップデートストラテジトランスフォーメーションの概要.	644
アップデートストラテジの設定.	645
動的のマッピングでのアップデートストラテジトランスフォーメーション.	645
マッピング内の行のフラグ設定.	645
アップデートストラテジ式.	646
アップデートストラテジトランスフォーメーションの詳細プロパティ.	646
アグリゲータトランスフォーメーションとアップデートストラテジトランスフォーメーション.	647
個々のターゲットに対する更新オプションの指定.	647
非ネイティブ環境でのアップデートストラテジトランスフォーメーション.	648
Blaze エンジンでのアップデートストラテジトランスフォーメーション.	648
Spark エンジンでのアップデートストラテジトランスフォーメーション.	649
第 46 章 : Web サービスコンシューマトランスフォーメーション.	651
Web サービスコンシューマトランスフォーメーションの概要.	651
SOAP メッセージ.	652
WSDL ファイル.	652
操作.	653

Web サービスのセキュリティ.....	653
WSDL の選択.....	654
Web サービスコンシューマトランスフォーメーションのポート.....	655
HTTP ヘッダー入力ポート.....	656
その他の入力ポート.....	656
Web サービスコンシューマトランスフォーメーションの入力マッピング.....	657
入力ポートをノードにマップするためのルールとガイドライン.....	658
[ビューのカスタマイズ] のオプション.....	658
操作入力への入力ポートのマッピング.....	658
Web サービスコンシューマトランスフォーメーションの出力マッピング.....	660
ノードを出力ポートにマップするためのルールとガイドライン.....	661
SOAP メッセージを XML としてマップ.....	661
[ビューのカスタマイズ] のオプション.....	661
出力ポートへの操作出力のマッピング.....	662
Web サービスコンシューマトランスフォーメーションの詳細プロパティ.....	663
Web サービスのエラー処理.....	665
メッセージの圧縮.....	666
並行処理.....	667
フィルタの最適化.....	667
Web サービスコンシューマトランスフォーメーションでの初期選択の最適化の有効化.....	668
Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化.....	668
Web サービスコンシューマトランスフォーメーションの作成.....	669
Web サービスコンシューマトランスフォーメーションの例.....	671
入力ファイル.....	671
論理データオブジェクトモデル.....	671
論理データオブジェクトマッピング.....	672
Web サービスコンシューマトランスフォーメーション.....	673

第 47 章 : Web サービス SOAP メッセージの解析..... 675

Web サービス SOAP メッセージの解析の概要.....	675
トランスフォーメーションのユーザーインターフェース.....	676
複数出現出力設定.....	677
正規化したリレーショナル出力.....	677
生成キー.....	677
非正規化したリレーショナル出力.....	678
ピボット化したリレーショナル出力.....	679
anyType 要素の解析.....	679
派生型の解析.....	680
QName 要素の解析.....	681
代替グループの解析.....	681
SOAP メッセージ内の XML 構造の解析.....	682
choice 要素.....	682
list 要素.....	682

union 要素.	682
第 48 章 : Web サービス SOAP メッセージの生成.	683
Web サービス SOAP メッセージの生成の概要.	683
トランスフォーメーションのユーザーインターフェース.	684
[入力ポート] 領域.	684
操作領域.	685
ポートと階層レベルのリレーション.	685
キー.	686
ポートのマッピング.	687
ポートのマッピング.	688
グループのマッピング.	689
複数のポートのマッピング.	689
複数出現ポートのピボット化.	689
非正規化データのマッピング.	691
派生型および要素の置き換え.	692
派生型の生成.	692
anyType 要素および属性の生成.	693
置き換えグループの生成.	693
SOAP メッセージ内の XML 構造の生成.	693
choice 要素.	694
list 要素.	694
union 要素.	695
第 49 章 : 加重平均トランスフォーメーション.	696
加重平均トランスフォーメーションの概要.	696
加重平均トランスフォーメーションの構成.	696
加重マッチ率の例.	697
加重平均トランスフォーメーションの詳細プロパティ.	697
非ネイティブ環境での加重平均トランスフォーメーション.	698
第 50 章 : ウィンドウトランスフォーメーション.	699
第 51 章 : 書き込みトランスフォーメーション.	700
書き込みトランスフォーメーションの概要.	700
書き込みトランスフォーメーションのプロパティ.	700
全般プロパティ.	701
データオブジェクトのプロパティ.	702
ポートのプロパティ.	702
ランタイムプロパティ.	703
ランタイムリンクプロパティ.	703
詳細プロパティ.	704
書き込みトランスフォーメーションの作成.	706

データオブジェクトからの書き込みトランスフォーメーションの作成.	707
マッピングフローからの書き込みトランスフォーメーションの作成.	707
パラメータからの書き込みトランスフォーメーションの作成.	708
既存のトランスフォーメーションからの書き込みトランスフォーメーションの作成.	709
付録 A: トランスフォーメーションの区切り文字.	711
トランスフォーメーションの区切り文字の概要.	711
索引.	712

序文

『*Informatica® Developer* トランスフォーメーションガイド』を使用して、Informatica トランスフォーメーションの設定、ガイドライン、使用方法、ランタイム動作について学びます。環境とランタイムエンジンに従って、適切なユースケースでトランスフォーメーションを使用する方法を理解します。マッピングの実行を予定している場所および方法に従って、各トランスフォーメーションのサポートを確認します。

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 からグローバルサポートセンターに連絡できます。

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 でオンラインサポートリソースを見つけるには、<https://network.informatica.com> にアクセスし、eSupport オプションを選択します。

第 1 章

トランスフォーメーションについて

この章では、以下の項目について説明します。

- [トランスフォーメーションについての概要, 32 ページ](#)
- [ネイティブ環境および非ネイティブ環境でのトランスフォーメーション, 37 ページ](#)
- [トランスフォーメーションデータ型の処理, 39 ページ](#)
- [トランスフォーメーションの開発, 41 ページ](#)
- [複数グループのトランスフォーメーション, 42 ページ](#)
- [トランスフォーメーションの式, 43 ページ](#)
- [ローカル変数, 47 ページ](#)
- [ポートのデフォルト値, 50 ページ](#)
- [トレースレベル, 56 ページ](#)
- [再利用可能なトランスフォーメーション, 57 ページ](#)
- [再利用不可能なトランスフォーメーション, 58 ページ](#)
- [トランスフォーメーションの作成, 58 ページ](#)

トランスフォーメーションについての概要

トランスフォーメーションは、データの生成、変更、または受け渡しを行うオブジェクトです。

Informatica Developer には、特定の関数を実行する一連のトランスフォーメーションが用意されています。例えば、アグリゲータトランスフォーメーションはデータのグループに対して計算を実行します。

マッピング内のトランスフォーメーションは、データ統合サービスがデータに対して実行する処理を表します。データは、マッピングまたはマップレット内のリンクされたトランスフォーメーションポートを通過します。

トランスフォーメーションはアクティブであるかパッシブであるかのいずれかです。トランスフォーメーションはデータフローに接続されているか、接続されていないかのいずれかです。

アクティブなトランスフォーメーション

アクティブなトランスフォーメーションは、トランスフォーメーションの前後で行数を変更します。または、行タイプを変更します。

例えば、フィルタトランスフォーメーションはフィルタ条件を満たさない行を削除するため、アクティブなトランスフォーメーションです。また、アップデートストラテジトランスフォーメーションは挿入、削除、更新、または拒否のフラグを行に設定するため、アクティブなトランスフォーメーションです。

複数のアクティブなトランスフォーメーション、または1つのアクティブなトランスフォーメーションと1つのパッシブなトランスフォーメーションを同じダウストリームトランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。Data Integration Service はアクティブなトランスフォーメーションから渡される行を連結できない可能性があります。

例えば、マッピング内の1つのブランチに、行に削除のフラグを付けるアップデートストラテジトランスフォーメーションが含まれているとします。別のブランチには、行に挿入のフラグを付けるアップデートストラテジトランスフォーメーションが含まれています。これらのトランスフォーメーションを1つのトランスフォーメーション入力グループに接続した場合、Data Integration Service は行の削除操作と挿入操作を結合できません。

パッシブトランスフォーメーション

パッシブトランスフォーメーションでは、トランスフォーメーションを通過する行数は変更されませんが、トランザクション境界と行タイプが維持されます。

アップストリームブランチのすべてのトランスフォーメーションがパッシブの場合、複数のトランスフォーメーションを同じダウストリームトランスフォーメーションまたは同じトランスフォーメーション入力グループに接続できます。ブランチを発生させるトランスフォーメーションはアクティブである場合とパッシブである場合があります。

接続されていないトランスフォーメーション

トランスフォーメーションはデータフローに接続されているか、接続されていないかのいずれかです。コネクトされていないトランスフォーメーションとは、マッピング内の他のトランスフォーメーションに接続されていないトランスフォーメーションのことです。コネクトされていないトランスフォーメーションは他のトランスフォーメーション内で呼び出され、そのトランスフォーメーションに値を返します。

複数ストラテジのトランスフォーメーション

ストラテジは、トランスフォーメーションがデータに実行できる1つ以上の操作のセットです。トランスフォーメーションのストラテジごとに異なる入力ポートと出力ポートのセットを割り当てることができます。定義したストラテジは、1つのトランスフォーメーションオブジェクトに格納されます。

【依存関係】 ビューを使用して、各ストラテジが使用するポートを表示します。

以下のトランスフォーメーションで、複数のトランスフォーメーションストラテジを定義することができます。

- 大文字小文字変換プログラム
- 分類子
- ディシジョン
- キージェネレータ
- ラベラ
- 一致

- マージ
- パーサー
パーサートランスフォーメーションで複数のストラテジを使用するには、トークンを解析するようにトランスフォーメーションを設定します。
- 標準化

トランスフォーメーションの説明

Developer tool には、一般的なトランスフォーメーションとデータ品質トランスフォーメーションが含まれています。共通のトランスフォーメーションは、Informatica Data Quality と Informatica Data Services で使用できます。データ品質トランスフォーメーションは、Informatica Data Quality で使用できます。

次の表に、各トランスフォーメーションを示します。

トランスフォーメーション	タイプ	説明
アドレスバリデータ	アクティブまたはパッシブ/ 接続済	郵便アドレスレコードの精度を検証して改善し、ユーザーがメールの受取人を選択したりメールを送付するのに役立つ情報を追加します。
関連付け	アクティブ/ 接続済	一致トランスフォーメーションによって異なるクラスタに割り当てられる重複レコード間のリンクを作成します。
アグリゲータ	アクティブ/ 接続済	集計の計算を実行します。
不良レコードの例外	アクティブ/ 接続済	データエラーが含まれている可能性のあるレコードを特定し、そのレコードを Analyst ツールのユーザーが確認して更新できるテーブルにロードします。
大文字小文字変換プログラム	パッシブ/ 接続済	文字列の大文字小文字を標準化します。
分類子	パッシブ/ 接続済	入力ポートフィールド内の情報をまとめたラベルを書き込みます。フィールドに膨大な量のテキストが含まれている場合に使用します。
比較	パッシブ/ 接続済	入力文字列のペア間の類似度を示す数値スコアを生成します。
統合	アクティブ/ 接続済	一致トランスフォーメーションによって重複として特定されたレコードから、統合されたレコードを作成します。
データマスキング	パッシブ/ コネクタされている、またはコネクタされていない	非プロダクション環境で、センシティブなプロダクションデータを現実的なテストデータに置き換えます。
データプロセッサ	アクティブ/ 接続済	マッピングで非構造化および半構造化のファイル形式を処理します。

トランスフォーメーション	タイプ	説明
ディシジョン	パッシブ/ 接続済	入力データの条件を評価し、それらの条件の結果に基づいて出力を作成します。
重複レコードの例外	アクティブ/ 接続済	重複した情報が含まれている可能性のあるレコードを特定し、そのレコードを <i>Analyst</i> ツールのユーザーが確認して更新できるテーブルにロードします。
式	パッシブ/ 接続済	値を計算します。
フィルタ	アクティブ/ 接続済	データをフィルタリングします。
階層型からリレーショナル	アクティブ/ 接続済	階層入力を処理して、リレーショナル出力に変換します。
Java	アクティブまたは パッシブ/ 接続済	Java で書かれたユーザコードを実行します。リポジトリにはユーザーロジックのバイトコードが格納されます。
ジョイナ	アクティブ/ 接続済	異なるデータベースまたはフラットファイルシステムから得たデータを結合します。
キージェネレータ	アクティブ/ 接続済	選択したカラム内のデータ値に基づいて、レコードをグループに割り当てます。
ラベラ	パッシブ/ 接続済	入力ポートフィールド内の文字または文字列を記述したラベルを書き込みます。
ルックアップ	アクティブまたは パッシブ/ コネクタされている、またはコネクタされていない	フラットファイル、論理データオブジェクト、参照テーブル、リレーショナルテーブル、ビュー、または同義語からのデータをルックアップして返します。
一致	アクティブ/ 接続済	入力レコード間の類似度を示すスコアを生成します。
マージ	パッシブ/ 接続済	複数の入力カラムからデータ値を読み取り、単一の出力カラムを作成します。
ノーマライザ	アクティブ/ 接続済	複数出現するデータを含むソース行を処理し、複数出現のデータの各インスタンスに対してターゲット行を返します。
出力	パッシブ/ 接続済	マップレット出力行を定義します。
パーサー	パッシブ/ 接続済	入力ポートの値を解析し、値に含まれている情報のタイプに基づいて別々の出力ポートに送ります。

トランスフォーメーション	タイプ	説明
ランク	アクティブ/ 接続済	レコードのランキング処理を行います。
読み取り	パッシブ/ 接続済	ソースからデータを読み取ります。
リレーショナル から階層型	アクティブ/ 接続済	リレーショナル入力を処理し、階層出力に変換します。
REST Web サービス コンシューマ	アクティブ/ 接続済	Web サービスクライアントとして REST Web サービスに接続し、データへのアクセスまたはデータの変換を行います。
ルータ	アクティブ/ 接続済	グループ条件に基づいて、複数のトランスフォーメーションにデータをルーティングします。
シーケンスジェネレータ	パッシブ/ 接続済	数値のシーケンスを生成します。
ソータ	アクティブ/ 接続済	ソートキーに基づいてデータをソートします。
SQL	アクティブまたは パッシブ/ 接続済	データベースに対して SQL クエリーを実行します。
標準化	パッシブ/ 接続済	入力文字列の標準化版を生成します。
共有体	アクティブ/ 接続済	異なるデータベースまたはフラットファイルシステムから得たデータを結合します。
アップデートストラテジ	アクティブ/ 接続済	行を挿入、削除、更新、または拒否するかどうかを決定します。
Web サービス コンシューマ	アクティブ/ 接続済	Web サービスクライアントとして Web サービスに接続し、データへのアクセスまたはデータの変換を行います。
加重平均	パッシブ/ 接続済	一致トランスフォーメーションがデータセット内のレコードに対して生成する一致スコアを読み取り、レコードのペアごとに平均スコアを計算します。レコードの各ペアに対してトランスフォーメーションが生成するスコアに、異なる重みを適用することができます。
書き込み	パッシブ/ 接続済	マッピングがデータを書き込むターゲットを表します。

ネイティブ環境および非ネイティブ環境でのトランスフォーメーション

非ネイティブ環境で実行したマッピングは、ネイティブ環境で実行したマッピングと異なる結果を返すことがあります。

処理上、次のような相違があることを考慮してください。

- 非ネイティブ環境では分散処理を使用して、さまざまなノードのデータを処理します。各ノードが、他のノードで処理されているデータにアクセスすることはありません。その結果、ランタイムエンジンではデータが生成された順序を判断できない場合があります。よって、マッピングを非ネイティブ環境で実行し、同じマッピングをネイティブ環境で実行すると、両方のマッピングで正しい結果を返しても、結果が同一でない場合があります。
- 非ネイティブ環境では実行時エンジンごとにマッピングロジックの処理を変えることができます。非ネイティブ環境では、Informatica トランスフォーメーションがフルサポートされることも、制約付きでサポートされることも、サポートされないこともあります。同様に、ネイティブ環境では、一部の Informatica トランスフォーメーションとトランスフォーメーション動作がサポートされないことがあります。

次の表は、各トランスフォーメーションと、非ネイティブ環境の各エンジンに対するサポートを一覧表示しています。

トランスフォーメーション	サポートされるエンジン
この表に一覧表示されていないトランスフォーメーションは非ネイティブ環境ではサポートされません。	
アドレスバリデータ	- Blaze - Spark
アグリゲータ	- Blaze - Spark* - Databricks Spark
大文字小文字変換プログラム	- Blaze - Spark
分類子	- Blaze - Spark
比較	- Blaze - Spark
統合	- Blaze - Spark
データマスキング	- Blaze - Spark*
データプロセッサ	- Blaze
ディシジョン	- Blaze - Spark
式	- Blaze - Spark* - Databricks Spark

トランスフォーメーション	サポートされるエンジン
フィルタ	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
Java	<ul style="list-style-type: none"> - Blaze - Spark*
ジョイナ	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
キージェネレータ	<ul style="list-style-type: none"> - Blaze - Spark
ラベラ	<ul style="list-style-type: none"> - Blaze - Spark
ルックアップ	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
一致	<ul style="list-style-type: none"> - Blaze - Spark
マージ	<ul style="list-style-type: none"> - Blaze - Spark
ノーマライザ	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
パーサー	<ul style="list-style-type: none"> - Blaze - Spark
Python	<ul style="list-style-type: none"> - Spark* - Databricks Spark
ランク	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
ルーター	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
シーケンスジェネレータ	<ul style="list-style-type: none"> - Blaze - Spark
ソーター	<ul style="list-style-type: none"> - Blaze - Spark* - Databricks Spark
標準化	<ul style="list-style-type: none"> - Blaze - Spark

トランスフォーメーション	サポートされるエンジン
共有体	- Blaze - Spark* - Databricks Spark
アップデートストラテジ	- Blaze - Spark
加重平均	- Blaze - Spark
Window	- Spark**
<p>* バッチマッピングおよびストリーミングマッピングの両方でサポートされます。</p> <p>** ストリーミングマッピングのみでサポートされます。詳細については、『Data Engineering Streaming ユーザーガイド』を参照してください。</p>	

トランスフォーメーションデータ型の処理

トランスフォーメーションでは、データ型固有の関数を処理したり、データを処理せずに通過させることができます。データ統合サービスは、Decimal、Timestamp with Timezone、Timestamp with Local Timezone などの一部のデータ型をトランスフォーメーションに基づいて処理します。

Decimal データ型

Decimal データ型を使用すると、フラットファイルおよびサポートされているデータベース（Oracle、Microsoft SQL Server、IBM DB2、ODBC など）に対してデータの読み取り/書き込みを行うことができます。

最大 38 桁の精度をサポートするトランスフォーメーションでは、精度が 1~38 桁、スケールが 0~38 になります。

Decimal データ型をサポートするトランスフォーメーション

以下のトランスフォーメーションは、最大 38 桁の精度の Decimal データ型をサポートし、そのデータの計算を実行できます。

- アグリゲータ
- データマスキング
- 式
- フィルタ
- Java
- ジョイナ
- ルックアップ
- ノーマライザ
- ランク
- ルータ

- シーケンスジェネレータ
- ソータ
- 共有体
- アップデートストラテジ

Decimal データ型のパススルーサポートを持つトランスフォーメーション

一部のトランスフォーメーションは、トランスフォーメーションを介して最大 38 桁の精度の 10 進数データのみを渡すことができます。トランスフォーメーションはデータに対して計算を実行できません。トランスフォーメーションで計算を実行するために最大 38 桁の精度の Decimal データ型を使用する場合、データ統合サービスはデータ型を Double として処理します。

以下のトランスフォーメーションには、最大 38 桁の精度の Decimal データ型に対するパススルーのサポートがあります。

- データプロセッサ
- 階層型からリレーショナル
- REST Web サービスコンシューマトランスフォーメーション
- SQL
- Web サービスコンシューマ

Decimal データ型をサポートしないトランスフォーメーション

一部のトランスフォーメーション（データ品質トランスフォーメーションなど）は Decimal データ型をサポートしません。

次の該当リストのデータ品質トランスフォーメーションは、最大 38 桁の精度の Decimal データ型をサポートしません。

- アドレスバリデータ
- 関連付け
- 大文字小文字変換プログラム
- 分類子
- 比較
- 統合
- ディシジョン
- キージェネレータ
- ラベラ
- 一致
- マージ
- パーサー
- 標準化
- 加重平均

Decimal (38) データ型をサポートしないトランスフォーメーションで、Decimal データ型の精度が 28 桁より大きい場合、データ統合サービスは Decimal 値を高精度モードの Double に変換します。

Timestamp with Time Zone

Timestamp with Time Zone は、タイムゾーンのオフセットまたはタイムゾーンの地域が含まれる、Timestamp データ型のバリエーションです。

Timestamp with Time Zone データ型をサポートしているトランスフォーメーションを以下に示します。

- アグリゲータ
- 式
- フィルタ
- Java
- ジョイナ
- ルックアップ
- ノーマライザ
- ランク
- ルータ
- シーケンスジェネレータ
- ソータ
- 共有体
- アップデートストラテジ

パススルーがサポートされている場合は、データをトランスフォーメーション経由で渡すことはできるが Timestamp with Time Zone データ型に対して関数を実行することはできないことを意味します。

Timestamp with Time Zone データ型のパススルーをサポートしているトランスフォーメーションを以下に示します。

- データマスキング
- データプロセッサ
- 階層型からリレーショナル
- SQL

Timestamp with Local Time Zone

機能が Timestamp と同等であるため、Timestamp with Local Time Zone は暗黙的にトランスフォーメーションでサポートされます。

トランスフォーメーションの開発

マッピングを作成する場合は、トランスフォーメーションを追加し、そのトランスフォーメーションに業務目的に応じたデータ処理方法を設定します。

トランスフォーメーションを開発してマッピングに組み込むには、以下の作業を行います。

1. 再利用できないトランスフォーメーションをマッピングまたはマップレットに追加します。または、複数のマッピングまたはマップレットに追加できる再利用可能なトランスフォーメーションを作成します。

2. トランスフォーメーションを設定します。各タイプのトランスフォーメーションには、設定可能な固有のオプションのセットがあります。
3. 再利用可能なトランスフォーメーションの場合は、そのトランスフォーメーションをマッピングまたはマプレットに追加します。
4. マッピングまたはマプレット内の他のオブジェクトにトランスフォーメーションをリンクします。

上流のオブジェクトからこのトランスフォーメーションの入力ポートに、ポートをドラッグします。このトランスフォーメーションから下流のオブジェクトのポートに、出力ポートをドラッグします。トランスフォーメーションによっては、選択できる事前定義済みポートを使用します。

注: 再利用可能なトランスフォーメーションを作成するときは、このトランスフォーメーションを他のオブジェクトにリンクする前に、必要となる入力ポートと出力ポートを追加します。マプレットまたはマッピングキャンバス上のトランスフォーメーションインスタンスにポートを追加することはできません。再利用可能なトランスフォーメーションのポートを更新するには、リポジトリプロジェクトからトランスフォーメーションオブジェクトを開いてからポートを追加します。

複数グループのトランスフォーメーション

トランスフォーメーションには、複数の入力グループと出力グループを含めることができます。グループは、入力データまたは出力データの行を定義するポートのセットです。

グループは、リレーショナルソースやターゲット定義のテーブルと類似しています。ほとんどのトランスフォーメーションは、1つの入力グループおよび1つの出力グループを持っています。ただし、いくつかのトランスフォーメーションは、複数の入力グループ、複数の出力グループ、またはその両方を持っています。グループは、トランスフォーメーションに入力されるデータ、またはトランスフォーメーションから出力されるデータの行を表すものです。

複数グループのトランスフォーメーションはすべてアクティブなトランスフォーメーションです。複数のアクティブなトランスフォーメーションまたはアクティブなトランスフォーメーションとパッシブなトランスフォーメーションを同じ後続トランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。

複数の入力グループを持つトランスフォーメーションの場合、Integration Service がある入力グループからの行を待つ間、Integration Service で別の入力グループの行をブロックする必要がある場合があります。ブロッキングトランスフォーメーションは、複数入力グループを持つトランスフォーメーションであり、入力データをブロックします。以下のトランスフォーメーションは、ブロッキングトランスフォーメーションです。

- [入力ブロック] プロパティが有効なカスタムトランスフォーメーション
- 未ソート入力に対して設定されたジョイナトランスフォーメーション

マッピングを保存または検証する場合、アクティブなトランスフォーメーションまたはブロッキングトランスフォーメーションを含むマッピングが有効にならないことがあります。

複数グループトランスフォーメーションのルールとガイドライン

マッピングでトランスフォーメーションを接続する場合は、複数グループトランスフォーメーションのルールとガイドラインを考慮する必要があります。

複数グループトランスフォーメーションに関して、次のルールとガイドラインを考慮します。

- 1つのグループは、1つのトランスフォーメーションまたはターゲットに接続できます。
- グループの1つ以上の出力ポートは、複数のトランスフォーメーションまたはターゲットに接続できます。

- トランスフォーメーションの複数の出力グループから別のトランスフォーメーションの同じ入力グループにフィールドを接続することはできません。
- ソースとトランスフォーメーションの間の各トランスフォーメーションがパッシブトランスフォーメーションではない場合、異なるトランスフォーメーションの複数の出力グループから別のトランスフォーメーションの同じ入力グループにフィールドを接続することはできません。
- 相手のトランスフォーメーションがブロッキングトランスフォーメーションではない場合、トランスフォーメーションの複数の出力グループから別のトランスフォーメーションの同じ入力グループにフィールドを接続することはできません。
- グループがノーマライズトランスフォーメーションにない場合、出力フィールドを同じ入力グループの複数の入力フィールドに接続することはできません。

トランスフォーメーションの式

一部のトランスフォーメーションでは、**式エディタ**で式を入力できます。式はデータを変更します。または、データが条件に一致するかテストします。

トランスフォーメーション言語関数を使用する式を作成します。トランスフォーメーション言語関数は、データを変換する SQL に似た関数です。

入力または入出力ポートから得たデータの値を使用するポートに、式を入力します。例えば、全従業員の給与が含まれる入力ポート IN_SALARY を持つトランスフォーメーションがあるとします。IN_SALARY カラムの値を後でマッピングに使用できます。このトランスフォーメーションを使用するテンプレート、給与の合計や平均を計算することもできます。Developer tool では各計算値について個別の出力ポートを作成する必要があります。

以下の表に、式を入力できるトランスフォーメーションを示します。

トランスフォーメーション	式	戻り値
アグリゲータ	トランスフォーメーションを通過するすべてのデータに基づいて、集計を行います。また、集計するレコードに対してフィルタを指定して、特定のレコードを排除できます。たとえば、このトランスフォーメーションを用いて、事務所の全従業員の総数と平均給与を算出できます。	ポートの集計結果。
式	単一の行内の値に基づいて計算を行います。たとえば、特定品目の価格や数量に基づいて、注文におけるその品目の購入価格合計値の算出ができます。	ポートの行レベルの計算結果。
フィルタ	このトランスフォーメーションを通過する行のフィルタリング条件を指定します。たとえば、未払い残高のある顧客についての BAD_DEBT テーブルに顧客データを書き込みたい場合には、Filter トランスフォーメーションを用いて顧客データのフィルタリングができます。	TRUE または FALSE。指定条件を行が満たしているかどうかによって決まります。TRUE を返す行はこのトランスフォーメーションを通過し、通過した各行にこの値が適用されます。

トランスフォーメーション	式	戻り値
ジョイナ	未ソートソースデータの結合に使用される詳細な条件を指定します。例えば、名と姓のマスターポートを連結し、それをフルネーム明細ポートと一致させることができます。	TRUE または FALSE。指定条件が行が満たしているかどうかによって決まります。選択した結合タイプに応じて、データ統合サービスは行を結果セットに追加するか、または行を破棄します。
ランク	ランクに含める行の条件を設定します。例えば、現在組織に所属している販売員の上位 10 名をランク付けできます。	ポートの条件適用結果または計算結果。
ルータ	グループ式に基づいて、複数のトランスフォーメーションにデータをルーティングします。たとえば、このトランスフォーメーションを使用して、3 つの異なる給与レベルに属する従業員の給与を比較します。これは、Router トランスフォーメーションに 3 つのグループを作成することによって行うことができます。たとえば、各給与範囲に対して 1 つのグループ式を作成します。	TRUE または FALSE。指定されたグループ式が行が満たしているかどうかによって決まります。TRUE を返す行は、このトランスフォーメーションの各ユーザー定義グループを通過します。FALSE を返す行は、デフォルトグループを通過します。
アップデートストラテジ	行に更新、挿入、削除、またはリジェクトのフラグを設定します。一定の条件に基づいてターゲットの更新を管理する場合に、このトランスフォーメーションを使用します。例えば、アップデートストラテジトランスフォーメーションを用いて、全顧客の行に対して、メールアドレスが変更された際に更新フラグを設定したり、全従業員の行に対して、社員でなくなった者については拒否フラグを設定したりできます。	更新、挿入、削除、またはリジェクトに対応する数値コード。このトランスフォーメーションは該当値を各取得行に適用します。

式エディタ

式エディタは、SQL に似た文を作成するために使用します。

式を手動で入力することも、ポイントアンドクリック機能を使用することもできます。ポイントアンドクリックインタフェースを使用して関数、ポート、変数、および演算子を選択することで、式を作成するときのエラーを減らすことができます。式に使用できる文字は最大 32,767 文字です。

式内のポート名

式にはトランスフォーメーションのポート名を入力することができます。

接続されているトランスフォーメーションでは、式でポート名を使用した場合、トランスフォーメーション内のポート名を変更すると Developer ツールによって式が更新されます。例えば、2 つの日付 Date_Promised と Date_Delivered の間の差を求める式を作成したとします。この場合、Date_Promised ポートの名前を Due_Date に変更すると、式内の Date_Promised ポート名は Due_Date に変更されます。

注: ポート名 Due_Date は、マッピングでこのポートに依存する他の再利用不可能なトランスフォーメーションにプロパゲートできます。

ポートへの式の追加

出力ポートに式を追加することができます。

1. トランスフォーメーションで、ポートを選択し、**式エディタ**を開きます。
2. 式を入力します。[関数] タブおよび [ポート] タブ、および演算子キーを使用します。
注: 式内でエスケープ文字を使用することはできません。式にエスケープ文字を含めた場合、Developer tool が解析エラーを表示する場合があります。
3. 必要に応じて、式にコメントを追加します。
コメントインジケータの「--」または「//」を使用します。
4. [検証] ボタンをクリックして、式を検証します。
5. [OK] をクリックします。
6. 式が有効でない場合、検証エラーを修正して再度検証します。
7. 式が有効になったら、[OK] をクリックして**式エディタ**を閉じます。

式内のコメント

コメントを式に追加して式に関する説明を記述したり、式に関連するビジネス文書にアクセスするための有効な URL を指定したりできます。

式にコメントを追加するには、コメントインジケータ「--」または「//」を使用します。

式の検証

マッピングを実行したりマプレットの出力をプレビューしたりするには、式を検証する必要があります。

式を検証するには、**式エディタ**の [検証] ボタンを使用します。自分で式の検証を行わずに**式エディタ**を閉じると、Developer ツールによって式が検証されます。式が無効な場合、Developer ツールは警告を表示します。無効な式を保存または変更することができます。

テスト式

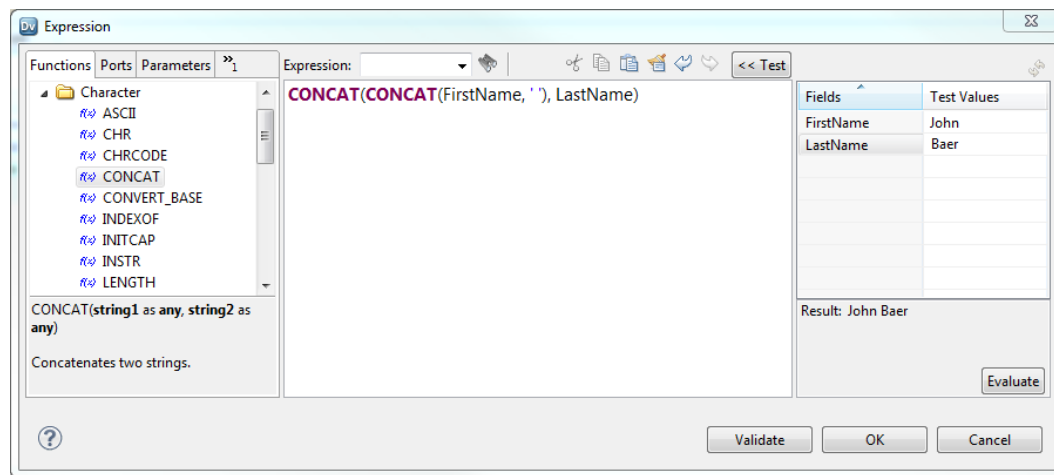
式エディタで設定したいいくつかの式をテストできます。式をテストする場合、サンプルデータを入力してから式を評価します。

式を設定すると、次の方法で式をテストできます。

- 式トランスフォーメーションの出力または変数ポートでテスト
- トランスフォーメーションをマッピングに追加後、式トランスフォーメーションの [マッピング出力] ビューでテスト

例えば、名前、スペース、姓を連結する式を設定後、ポートのサンプルデータを入力し、式を評価して結果を確認できます。

次の図は、名前と姓の例を連結する式の結果を示しています。



サンプルデータの日付形式の文字列

Date/Time または Timestamp with Time Zone データ型のポートを使用する式をテストする場合、必要な日付形式の文字列を使用してポートのサンプルデータを入力する必要があります。

Date/Time データ型のポートのサンプルデータを入力するには、「MM/DD/YYYY HH24:MI:SS」形式を使用します。式の評価を行うと、式エディタには式で指定した形式を使用した結果が表示されます。式の形式文字列を省略すると、式エディタには同じ「MM/DD/YYYY HH24:MI:SS」形式を使用した結果が表示されます。

Timestamp with Time Zone データ型のポートのサンプルデータを入力するには、「MM/DD/YYYY HH24:MI:SS TZR」形式を使用します。式の評価を行うと、式エディタには「YYYY-MM-DD HH24:MI:SS.NS TZR」形式を使用した結果が表示されます。

データ型の変換

複数の式の関数および集計関数では、入力データと異なるデータ型のデータが生成されることがあります。

例えば、18 桁の精度を持つ 2 つの 10 進数を乗算した場合、結果のデータ型は 28 桁の精度を持つ 10 進数となることがあります。

入力データ型が 38 桁の精度を持つ 10 進数の場合、特定の操作の結果として結果のデータ型に適合しないデータが生成されることがあります。そのため、ユーザーがオーバーフロー例外を受け取ることがあります。

以下の関数では、入力データ型と比較してデータサイズが拡大することに対応するために、データ型の変換が必要となることがあります。

- avg
- cume
- divide
- median
- movingavg
- movingsum
- multiply
- Percentile
- Sum

例えば、入力データが Integer データ型で、乗算操作を使用する場合、結果のデータ型は bigint データ型となることがあります。同様に、入力データ型が 18 桁の精度を持つ Decimal データ型の場合、乗算操作の結果は大きくなり、28 桁の精度を持つ Decimal データ型の値となることがあります。

ローカル変数

パフォーマンスを向上させるには、アグリゲータトランスフォーメーション、式トランスフォーメーション、およびランクトランスフォーメーションでローカル変数を使用します。式中の変数を参照したり、変数を使ってデータの一時的格納ができます。

変数を使用して、以下の作業を実行する場合があります。

- データの一時的格納。
- 複雑な式の簡素化。
- 前の行からの値の格納。
- ストアドプロシージャからの複数の戻り値の取得。
- 値の比較。
- コネクトされていないルックアップトランスフォーメーションの結果の格納。

データの一時的な格納と複雑な式の簡素化

同じトランスフォーメーションに複数の関連式を入力する際に、変数を使用するとパフォーマンスが向上します。トランスフォーメーションで同じ式のコンポーネントを複数回解析して検証する代わりに、コンポーネントを変数として定義できます。

例えば、アグリゲータトランスフォーメーションが合計と平均を計算する前に同じフィルタ条件を使用する場合、この条件を変数として定義してから両方の集計計算で再利用できます。

複雑な式を単純化できます。アグリゲータで複数の式に同じ計算が含まれる場合は、変数を作成して計算結果を格納することによってパフォーマンスを向上できます。

例えば、次の式を作成し、同じデータを使用して平均給与と給与合計を算出できます。

```
AVG( SALARY, ( ( JOB_STATUS = 'Full-time' ) AND (OFFICE_ID = 1000 ) ) )  
SUM( SALARY, ( ( JOB_STATUS = 'Full-time' ) AND (OFFICE_ID = 1000 ) ) )
```

両方の計算に同じ引数を入力する代わりに、この計算の各条件に変数ポートを作成してから変数を使用するように式を変更できます。

以下の表に、変数を使用して複雑な式を簡素化し、データを一時的に格納する方法を示します。

ポート	値
V_CONDITION1	JOB_STATUS = ヤ Full-time コ
V_CONDITION2	OFFICE_ID = 1000
AVG_SALARY	AVG (SALARY, (V_CONDITION1 AND V_CONDITION2))
SUM_SALARY	SUM (SALARY, (V_CONDITION1 AND V_CONDITION2))

複数行の値の格納

トランスフォーメーション内で、ソース行からのデータを格納するように変数を設定できます。変数は、トランスフォーメーション式の中に使用できます。

例えば、ソースファイルに次の行が含まれるとします。

```
California
California
California
Hawaii
Hawaii
New Mexico
New Mexico
New Mexico
```

各行には、州が含まれます。行数をカウントし、州ごとに行カウント数を返す必要があります。

```
California,3
Hawaii      ,2
New Mexico,3
```

この場合、アグリゲータトランスフォーメーションを設定して、ソース行が州別にグループ化され、各グループ内の行数がカウントされるようにすることができます。アグリゲータトランスフォーメーション内の変数は、行カウント数が格納されるように設定してください。別の変数を、前の行から州の名前が格納されるように定義します。

アグリゲータトランスフォーメーションは、次のポートを備えています。

ポート	ポートタイプ	式	説明
州	パススルー	n/a	州の名前。ソース行は、州の名前ごとにグループ化されています。Aggregator トランスフォーメーションによって、州ごとに 1 行が返されます。
State_Count	変数	IIF (PREVIOUS_STATE = STATE, STATE_COUNT +1, 1)	現在の州の行カウント。現在の [州] カラムの値が Previous_State カラムと同じ場合、State_Count が増分されます。それ以外の場合、State_Count が 1 にリセットされます。
Previous_State	変数	State	前の行の [州] カラムの値。行の処理時には、州の値が Previous_State に移動されます。
State_Counter	出力	State_Count	Aggregator トランスフォーメーションで、州ごとに処理される行数。Integration Service は州ごとに State_Counter を返します。

ストアドプロシージャからの値の取得

変数は、ストアドプロシージャから複数カラムの戻り値を取得する手段となります。

変数ポートの設定のガイドライン

トランスフォーメーション内に変数ポートを設定するときは、以下の項目について考慮する必要があります。

- **ポートの評価順。** Integration Service は依存関係によってポートを評価します。トランスフォーメーション内のポートの順序は、評価の順序と一致させる必要があります。入力ポート、変数ポート、出力ポートの順になります。

- **データタイプ。** 選択するデータタイプには、入力した式の戻り値が反映されます。
- **変数の初期化。** Integration Service は、カウンタを作成することが可能な変数ポートに初期値を設定します。

ポートの評価順

統合サービスは、最初に入力ポートを評価します。統合サービスは、次に変数ポートを評価し、最後に出力ポートを評価します。

Integration Service は以下の順序でポートを評価します。

1. **入力ポート。** Integration Service では最初に入力ポートがすべて評価されます。これは、入力ポートが他のポートに影響されないという理由からです。したがって、入力ポートは任意の順序で作成できます。入力ポートは他のポートを参照しないので、統合サービスは入力ポートを順序付けしません。
2. **変数ポート。** 変数ポートは入力ポートと変数ポートを参照できますが、出力ポートの参照はできません。変数ポートは入力ポートの参照ができるので、Integration Service は入力ポートの次に変数ポートの評価を行います。変数は他の変数を参照できるので、変数ポートの表示順は、統合サービスがそれぞれの変数を評価する順序と同じです。
たとえば、建物の取得価格を計算し、ついで原価償却による調整を行う場合には、取得価格計算を変数ポートとして作成できます。この変数は、原価償却による調整ポートの前に配置される必要があります。
3. **出力ポート。** 出力ポートは入力ポートと変数ポートを参照できるので、統合サービスは出力ポートを最後に評価します。出力ポートは他の出力ポートを参照できないので、出力ポートの表示順は関係ありません。出力ポートは、必ずポートリストの最下部に表示してください。

データ型

ポートを変数として設定すると、任意の式または条件を入力できます。このポートに対して選択するデータ型は、入力する式の戻り値に反映されます。変数ポートを使用して条件を指定すると、数値データ型は TRUE（非ゼロ）値と FALSE（ゼロ）値を返します。

変数の初期化

統合サービスは、変数の初期値を NULL に設定しません。

統合サービスは、次のガイドラインを使用して変数の初期値を設定します。

- 数値ポートについてはゼロ
- 文字列ポートについては、空の文字列
- Date/Time ポートについては 01/01/0001

そのため、初期値を必要とするカウンタとして変数を使用します。たとえば、以下の式では数値変数が作成できます。

`VAR1 + 1`

この式では、VAR1 ポートで行数をカウントします。変数の初期値を NULL に設定した場合には、この式は必ず NULL と評価されることになります。そのため、初期値はゼロに設定してあります。

ポートのデフォルト値

すべてのトランスフォーメーションは、統合サービスが入力 NULL 値と出力トランスフォーメーションエラーを処理する方法を決定するデフォルト値を使用します。

入力、出力、および入出力の各ポートには、必要に応じてユーザー定義デフォルト値によってオーバーライドできるシステムデフォルト値が指定されています。デフォルト値は、ポートの種類によって機能が異なります。

- **入力ポート。** NULL 入力ポートのシステムデフォルト値は NULL です。トランスフォーメーションでは、デフォルト値は空白として表示されます。入力値が NULL である場合、Integration Service では NULL のままになります。
- **出力ポート。** 出力トランスフォーメーションエラーのシステムデフォルト値は、ERROR です。トランスフォーメーションでは、デフォルト値は ERROR('transformation error')として表示されます。トランスフォーメーションエラーが発生すると、Integration Service ではその行がスキップされます。統合サービスは、ERROR 関数がスキップするすべての入力行をログファイルに記録します。

次に、トランスフォーメーションエラーを示します。

- データ変換エラー（たとえば、日付関数に数値を渡した場合など）。
- ゼロ除算などの式評価エラー。
- ERROR 関数の呼び出し。

- **パススルーポート** NULL 入力のシステムデフォルト値は入力ポート同様 NULL です。トランスフォーメーションでは、システムのデフォルト値は空白として表示されます。出力トランスフォーメーションエラーのデフォルト値は、出力ポートと同じです。出力トランスフォーメーションエラーのデフォルト値はトランスフォーメーションに表示されません。

注: Java トランスフォーメーションは、Java トランスフォーメーションのポートタイプに基づいて、PowerCenter®のデータ型を Java データ型に変換します。NULL 入力のデフォルト値は、Java データタイプによって異なります。

以下の表に、接続されたトランスフォーメーションのポートのシステムデフォルト値を示します。

ポートタイプ	デフォルト値	統合サービスの動作	サポートされているユーザー定義のデフォルト値
入力、パススルー	NULL	統合サービスは、すべての入力 NULL 値を NULL として渡します。	入力、入出力
出力、パススルー	ERROR	統合サービスは、出力ポートでのトランスフォーメーションのエラーに対して ERROR 関数を呼び出します。統合サービスは、エラーが発生した行をスキップし、入力データとエラーメッセージをログファイルに書き込みます。	アウトプット

変数ポートはデフォルト値をサポートしていません。変数ポートはそのデータタイプに応じて Integration Service で初期化されます。

一部のデフォルト値を上書きして、NULL の入力値および出力トランスフォーメーションのエラーに対する Integration Service の動作を変更できます。

ユーザー定義のデフォルト値

接続されているトランスフォーメーションでサポートされている入力、パススルー、および出力の各ポートのシステムデフォルト値は、ユーザー定義のデフォルト値でオーバーライドできます。

ポートのシステムデフォルト値をオーバーライドする場合は、次のルールとガイドラインを使用します。

- **入力ポート。**統合サービスで NULL 値を NULL として扱わないようにする場合は、入力ポートにユーザー定義のデフォルト値を入力します。NULL が入力ポートに渡されると、統合サービスでは NULL をデフォルト値に置き換えます。
- **出力ポート。**統合サービスで行をスキップしない場合、またはスキップされた行とともに特定のメッセージをログに書き込む場合は、出力ポートにユーザー定義のデフォルト値を入力します。出力ポートにデフォルト値を定義する場合、統合サービスでは、出力ポートにトランスフォーメーションエラーが発生していれば、その行をデフォルト値に置き換えます。
- **パススルーポート。**統合サービスで NULL 値を NULL として扱わない場合は、パススルーポートにユーザー定義のデフォルト値を入力します。出力トランスフォーメーションエラーのユーザー定義のデフォルト値をパススルーポートに入力することはできません。

注: 接続されていないトランスフォーメーションのユーザー定義のデフォルト値は、Integration Service によって無視されます。例えば、式を使用してルックアップトランスフォーメーションまたはストアドプロシージャトランスフォーメーションを呼び出す場合、統合サービスはすべてのユーザー定義のデフォルト値を無視してシステムデフォルト値を適用します。

ユーザー定義デフォルト値を入力する場合、以下のオプションを使用します。

- **定数値。**NULL を含めて、任意の定数（数値またはテキスト）が使用できます。
- **定数式。**定数パラメータをもつトランスフォーメーション関数を含めることができます。
- **ERROR。**トランスフォーメーションエラーを生成します。マッピングログまたは行エラーログに行とメッセージを書き込みます。
- **ABORT。**マッピングを強制終了します。

定数値

デフォルト値には、任意の定数値を入力できます。定数値はポートのデータ型と一致する必要があります。

たとえば、数値ポートのデフォルト値は数値定数でなければなりません。定数値には、以下のものがあります。

```
0
9999
NULL
'Unknown Value'
'Null input data'
```

定数式

定数式とは、定数式を記述するトランスフォーメーション関数（集計関数以外）を使う式です。定数式には、入力、入出力、または変数ポートからの値は使用できません。

有効な定数式には以下のものがあります。

```
500 * 1.75
TO_DATE('January 1, 1998, 12:05 AM','MONTH DD, YYYY, HH:MI AM')
ERROR('Null not allowed')
ABORT('Null not allowed')
SESSSTARTTIME
```

統合サービスはマッピングの初期化の際にマッピング全体にデフォルト値を割り当てるので、式内でポートからの値を使用することはできません。

次の例は、ポートからの値を使用するので無効です。

```
AVG(IN_SALARY)
IN_PRICE * IN_QUANTITY
:LKP(LKP_DATES, DATE_SHIPPED)
```

注: デフォルト値の式からストアプロシージャまたはルックアップテーブルを呼び出すことはできません。

ERROR 関数と ABORT 関数

ERROR 関数および ABORT 関数は、入力ポートおよび出力ポートのデフォルト値、および入出力ポートの入力値で使います。統合サービスは、ERROR 関数に遭遇するとその行をスキップします。ABORT 関数に遭遇すると、マッピングを強制終了します。

ユーザー定義のデフォルト入力値

統合サービスが NULL 値を NULL として扱わないようにするには、ユーザー定義のデフォルト入力値を入力します。

NULL 値をオーバーライドするには、次のいずれかのタスクを完了します。

- NULL 値を定数値または定数式によって置き換える。
- ERROR 関数によって、NULL 値をスキップする。
- ABORT 関数によって、マッピングを強制終了する。

以下の表に、Integration Service が入力および入出力ポートの NULL 入力を処理する方法をまとめます。

デフォルト値	デフォルト値のタイプ	説明
NULL (空白を表示)	System	統合サービスは NULL を渡します。
定数または定数式	ユーザー定義	統合サービスは、NULL 値を定数または定数式の値に置き換えます。
エラー (ERROR)	ユーザー定義	統合サービスは、これをトランスフォーメーションエラーとして処理します。 <ul style="list-style-type: none">- トランスフォーメーションエラーカウントを 1 つ増やします。- その行をスキップし、ログファイルまたは行エラーログにエラーメッセージを書き込みます。 統合サービスはその行を拒否ファイルに書き込みません。
ABORT	ユーザー定義	統合サービスが NULL 入力値を検出すると、マッピングが強制終了します。統合サービスはエラーカウントを増やさず、行を拒否ファイルに書き込みません。

NULL 値の置換

定数値または定数式を使用して、ポートの NULL 値を指定した値で置き換えます。

例えば、入力文字列ポートの名前が DEPT_NAME で、NULL 値を文字列「UNKNOWN DEPT」で置き換える場合は、デフォルト値を「UNKNOWN DEPT」に設定します。トランスフォーメーションタイプに応じて、Integration Service は「UNKNOWN DEPT」をトランスフォーメーション内の式か変数、またはデータフロー内の次のトランスフォーメーションに渡します。

例えば、Integration Service ではポート内の NULL 値がすべて、文字列「UNKNOWN DEPT」に置き換えられます。

DEPT_NAME	REPLACED VALUE
Housewares	Housewares
NULL	UNKNOWN DEPT
Produce	Produce

NULL レコードのスキップ

NULL 値をトランスフォーメーションに渡さないようにするには、ERROR 関数をデフォルト値として使用します。たとえば、DEPT_NAME の入力値が NULL の場合に、行をスキップするとします。以下の式をデフォルト値として使用することができます。

```
ERROR('Error. DEPT is NULL')
```

ERROR 関数をデフォルト値として使用すると、Integration Service では NULL 値を含む行がスキップされます。統合サービスは、ERROR 関数がスキップしたすべての行をログファイルに書き込みます。これらの行は、拒否ファイルには書き込まれません。

DEPT_NAME	RETURN VALUE
Housewares	Housewares
NULL	'Error. DEPT is NULL' (Row is skipped)
Produce	Produce

次のログに、統合サービスが NULL 値を含む行をスキップする場所を示します。

```
TE_11019 Port [DEPT_NAME]: Default value is: ERROR(<<Transformation Error>> [error]: Error. DEPT is NULL
... error('Error. DEPT is NULL')
).
CMN_1053 EXPTRANS: : ERROR: NULL input column DEPT_NAME: Current Input data:
CMN_1053 Input row from SRCTrans: Rowdata: ( RowType=4 Src Rowid=2 Targ Rowid=2
  DEPT_ID (DEPT_ID:Int:): "2"
  DEPT_NAME (DEPT_NAME:Char.25:): "NULL"
  MANAGER_ID (MANAGER_ID:Int:): "1"
)
```

マッピングの強制終了

統合サービスが NULL 入力値に遭遇した場合にマッピングを強制終了するには、ABORT 関数を使用します。

デフォルト値の検証

Developer ツールは、入力時にデフォルト値を検証します。

Developer ツールは、マッピングの保存時にデフォルト値を検証します。無効なデフォルト値を入力すると、Developer ツールはマッピングを無効としてマークします。

ユーザー定義のデフォルト出力値

出力ポートのシステムデフォルト値をオーバーライドする場合は、ユーザー定義デフォルト値を作成できます。

統合サービスがエラーを含む行をスキップしないようにする場合、または統合サービスがログにスキップされた行とともに特定のメッセージを書き込むようにする場合は、出力ポートにユーザー定義デフォルト値を入力できます。統合サービスで出力トランスフォーメーションエラーが発生した場合は、デフォルト値を入力して次の操作を実行します。

- エラーを定数値または定数式によって置き換える。Integration Service は行をスキップしません。
- ABORT 関数によって、マッピングを強制終了する。

- トランスフォーメーションエラーに対する特定のメッセージをログに書き込む。

出力ポートには、ユーザー定義デフォルト出力値を入力できません。

以下の表に、Integration Service による出力ポートトランスフォーメーションエラーの処理方法と、トランスフォーメーションのデフォルト値を示します。

デフォルト値	デフォルト値のタイプ	説明
トランスフォーメーションエラー	System	デフォルト値を上書きしていない場合にトランスフォーメーションエラーが発生すると、統合サービスは次のタスクを実行します。 <ul style="list-style-type: none"> - トランスフォーメーションエラーカウントを 1 つ増やします。 - その行をスキップし、セッション設定に従って、セッションログファイルまたは行エラーログにエラーおよび入力行を書き込みます。 統合サービスはその行を拒否ファイルに書き込みません。
定数または定数式	ユーザー定義	統合サービスは、エラーをデフォルト値で置き換えます。 統合サービスは、エラーカウントを増やさず、ログにメッセージを書き込みません。
ABORT	ユーザー定義	マッピングは強制終了し、統合サービスはメッセージをログに書き込みます。 統合サービスはエラーカウントを増やさず、行を拒否ファイルに書き込みません。

エラーの置換

トランスフォーメーションエラーが発生した場合に統合サービスが行をスキップしないように設定するには、出力ポートのデフォルト値として定数または定数式を使用します。

例えば、NET_SALARY という数値出力ポートがあり、トランスフォーメーションエラーが発生した場合に定数値「9999」を使用する場合、NET_SALARY ポートにデフォルト値 9999 を割り当てます。NET_SALARY の値の計算中にゼロ除算などのトランスフォーメーションエラーが発生した場合、統合サービスはデフォルト値 9999 を使用します。

マッピングの強制終了

Integration Service が NULL 入力値に遭遇した場合にセッションを強制終了するには、ABORT 関数を使用します。

マッピングログへのメッセージの書き込み

統合サービスがスキップした行に対して特定のメッセージをマッピングログに書き込むようにする場合は、出力ポートのユーザー定義デフォルト値を設定できます。システムのデフォルト値は ERROR ('transformation error') なので、統合サービスはスキップされた行とともに「transformation error」というメッセージをログに書き込みます。別のメッセージを書き込む場合は、'transformation error'を置き換えることができます。

出力ポート式の ERROR 関数

ERROR 関数を使用する式を入力すると、出力ポートのユーザー定義デフォルト値によって式の ERROR 関数が上書きされる場合があります。

例えば、エラーが発生したときに Integration Service が 'Negative Sale' という値を使用するように指示する以下の式を入力するとします。

```
IIF( TOTAL_SALES>0, TOTAL_SALES, ERROR ('Negative Sale'))
```

以下の例は、ユーザー定義デフォルト値が式の ERROR 関数をどのように上書きするのかを示しています。

- **定数値または定数式。** 定数値または定数式は、出力ポート式の ERROR 関数を上書きします。

例えば、「0」をデフォルト値として入力すると、Integration Service は出力ポート式の ERROR 関数を上書きします。エラーが発生すると、値 0 が渡されます。行をスキップしたり、ログに「Negative Sale」と書き込むことは行いません。

- **ABORT.** ABORT 関数は、出力ポート式の ERROR 関数を上書きします。

ABORT 関数をデフォルト値として使用すると、統合サービスはトランスフォーメーションエラーの発生時にセッションを強制終了します。ABORT 関数は、出力ポート式の ERROR 関数を上書きします。

- **ERROR.** ERROR 関数をデフォルト値として使用すると、統合サービスは次の情報をログに記録します。

- デフォルト値からのエラーメッセージ
- 出力ポート式の ERROR 関数に示されるエラーメッセージ
- スキップされた行

たとえば、以下の ERROR 関数でデフォルト値を上書きすることができます。

```
ERROR('No default value')
```

Integration Service は行をスキップし、両方のエラーメッセージをログに書き込みます。

```
TE_7007 Transformation Evaluation Error; current row skipped...
TE_7007 [<<Transformation Error>> [error]: Negative Sale
... error('Negative Sale')
]
Sun Sep 20 13:57:28 1998
TE_11019 Port [OUT_SALES]: Default value is: ERROR(<<Transformation Error>> [error]: No default value
... error('No default value')
```

デフォルト値の一般ルール

デフォルト値を作成する場合には、次の規則およびガイドラインに従ってください。

- デフォルト値は、NULL、定数値、定数式、ERROR 関数、または ABORT 関数のいずれかでなければなりません。
- 入出力ポートの場合、Integration Service はデフォルト値を使用して NULL 入力値を処理します。入出力ポートの出力デフォルト値は常に ERROR (Transformation Error) です。
- 変数ポートはデフォルト値を使用しません。
- アグリゲータトランスフォーメーションおよびランクトランスフォーメーションでは、Group By ポートにデフォルト値を指定できます。
- トランスフォーメーションのすべてのポートの種類について、ユーザー定義のデフォルト値を指定できるとは限りません。ポートがユーザー定義デフォルト値を使用できない場合は、デフォルト値フィールドは無効になります。
- すべてのトランスフォーメーションでユーザー定義デフォルト値を使用できるわけではありません。
- トランスフォーメーションがマッピングデータフローに接続されていない場合、Integration Service はユーザー定義値を無視します。
- 入力ポートのいずれかが未接続の場合、Integration Service は値が NULL であると仮定し、この入力ポートについてはデフォルト値を使用します。
- 入力ポートのデフォルト値に ABORT 関数が含まれ、入力値が NULL の場合、統合サービスはマッピングを直ちに停止します。ABORT 関数をデフォルト値として使用することにより、NULL 入力値の制限ができます。入力ポートで最初に NULL 値が見つかった時点でマッピングは終了します。

- 出力ポートのデフォルト値に ABORT 関数が含まれ、そのポートでトランスフォーメーションエラーが発生すると、マッピングは直ちに停止します。トランスフォーメーションエラーに対して厳密なルールを適用する場合、ABORT 関数をデフォルト値として使用します。このポートで最初のトランスフォーメーションエラーが発生した時点でマッピングは終了します。
- ABORT 関数、定数値、および定数式は、出力ポート式に設定されている ERROR 関数をオーバーライドします。

デフォルト値の検証

Developer ツールは、入力時にデフォルト値を検証します。

Developer ツールは、マッピングの保存時にデフォルト値を検証します。無効なデフォルト値を入力すると、Developer ツールはマッピングを無効としてマークします。

トレースレベル

トランスフォーメーションを設定するとき、Data Integration Service がログに書き込む情報の詳細度を指定できます。

デフォルトでは、すべてのトランスフォーメーションのトレースレベルは [Normal] です。正常に動作しないトランスフォーメーションをトラブルシュートする必要がある場合は、トレースレベルをいずれかの [Verbose] 設定に変更します。ログに最小限の情報を表示する場合は、トレースレベルを [Terse] に設定します。

[詳細] タブでは、以下のプロパティを設定します。

トレースレベル

トランスフォーメーションのログに表示される情報の詳細度。

以下の表に、トレースレベルを示します。

トレースレベル	説明
Terse	初期化情報とエラーメッセージ、およびリジェクトされたデータに関する通知を記録します。
Normal	初期化情報とステータス情報、発生したエラー、トランスフォーメーション行エラーの発生時にスキップした行を、ログに記録します。マッピング結果のまとめを行います。個別行のレベルでのまとめは行いません。 デフォルトは [Normal] です。
Verbose Initialization	Normal トレースで記録される情報に加えて、初期化の詳細、インデックス名と使用されたデータファイル名、詳細なトランスフォーメーション統計をログに記録します。
Verbose Data	Verbose Initialization トレースで記録される情報に加えて、マッピングに渡された各行をログに記録します。また、カラムの精度に合わせて文字列データを切り詰めた場所と、詳細なトランスフォーメーション統計情報も記録します。 このトレースレベルを設定した場合、トランスフォーメーションが処理されるときに、ブロック内のすべての行の行データがログに書き込まれます。

再利用可能なトランスフォーメーション

再利用可能なトランスフォーメーションは、複数のマッピングまたはマプレットで使用するトランスフォーメーションです。

例えば、カナダでの販売付加価値税を計算する式トランスフォーメーションを作成すれば、カナダにおける事業コストを分析できます。いつも同じ作業を行うのであれば、再利用可能なトランスフォーメーションを作成できます。当該のトランスフォーメーションをマッピングに組み込む必要がある場合には、そのインスタンスをマッピングに追加してください。トランスフォーメーションの定義を変更した場合には、そのインスタンスはすべてこの変更を受け継ぎます。

Developer ツールは再利用可能なトランスフォーメーションを、それを使用するマッピングまたはマプレットとは別のメタデータとして格納します。再利用可能なトランスフォーメーションは、プロジェクトまたはフォルダに格納されます。

再利用可能なトランスフォーメーションのインスタンスをマッピングに追加した場合、トランスフォーメーションに変更を加えると、マッピングが無効になったり、予期しないデータが生成されたりすることがあります。

再利用可能なトランスフォーメーションのインスタンスと継承される変更

再利用可能なトランスフォーメーションをマッピングまたはマプレットに追加する場合には、このトランスフォーメーションのインスタンスを追加します。トランスフォーメーションの定義はマッピングまたはマプレット外にあるのに対して、トランスフォーメーションのインスタンスはマッピングまたはマプレット内に表示されます。

トランスフォーメーションを変更すると、トランスフォーメーションのインスタンスにそれらの変更が反映されます。同じトランスフォーメーションを使用するマッピングでそれぞれトランスフォーメーションを更新する代わりに、再利用可能なトランスフォーメーションを一度更新すると、そのトランスフォーメーションのインスタンスすべてに変更が反映されます。インスタンスには、トランスフォーメーションのポート、式、プロパティ、および名前に対する変更が継承されます。

再利用可能なトランスフォーメーションの編集

再利用可能なトランスフォーメーションを編集すると、そのトランスフォーメーションのすべてのインスタンスに変更が継承されます。変更によっては、再利用可能なトランスフォーメーションを使用したマッピングが無効となる場合があります。

再利用可能なトランスフォーメーションは、エディタで開いて編集することができます。マッピング内でトランスフォーメーションのインスタンスを編集することはできません。ただし、トランスフォーメーションのランタイムプロパティは編集が可能です。

再利用可能なトランスフォーメーションに以下のいずれかの変更を行うと、そのインスタンスを用いるマッピングが無効となる場合があります。

- トランスフォーメーションの 1 つ以上のポートを削除すると、マッピングの一部またはすべてのデータフローからインスタンスが切り離されます。
- ポートのデータ型を変更すると、そのポートから互換性のないデータ型を使用する別のポートへマッピングできなくなります。
- ポート名を変更すると、そのポートを参照する式は無効となります。
- 再利用可能なトランスフォーメーションに無効な式を入力すると、そのトランスフォーメーションを使用するマッピングは無効となります。Data Integration Service では無効なマッピングを実行できません。

再利用可能なトランスフォーメーションのエディタのビュー

エディタ内のビューで、再利用可能な Java トランスフォーメーションのプロパティを定義したり、Java コードを作成したりします。

再利用可能なトランスフォーメーションでは、次のビューを使用できます。

概要

トランスフォーメーションの名前と説明を入力し、入出力ポートを作成および設定します。

詳細

トランスフォーメーションの詳細プロパティを設定します。

再利用不可能なトランスフォーメーション

再利用不可能なトランスフォーメーションは、特定のマッピングで作成するトランスフォーメーションです。このトランスフォーメーションを他のマッピングで使用することはできません。

例えば、複数のトランスフォーメーションを含むマッピングを作成するとします。各トランスフォーメーションは、ソースデータに対して計算を実行します。マッピングの最後に結果を処理する再利用不可能なアグリゲータトランスフォーメーションを作成します。再利用不可能なトランスフォーメーションを作成する際は、マッピングのトランスフォーメーション間でポートをドラッグして入力ポートを作成できます。

Developer ツールは、再利用不可能なアグリゲータトランスフォーメーションをメタデータとしてマッピングとともに保存します。

再利用不可能なトランスフォーメーション用のエディタのビュー

再利用不可能なトランスフォーメーション用のエディタのビューでプロパティを定義します。

再利用不可能なトランスフォーメーションには、次のビューが表示されます。

全般

トランスフォーメーションの名前および説明を入力します。

ポート

入力ポートと出力ポートを作成および設定します。

詳細

トランスフォーメーションの詳細プロパティを設定します。

トランスフォーメーションの作成

複数のマッピングまたはマブレットで再利用する再利用可能なトランスフォーメーションを作成できます。また、マッピングまたはマブレットで 1 回だけ使用する再利用不可のトランスフォーメーションも作成できます。

再利用可能なトランスフォーメーションを作成するには、**【ファイル】 > 【新規】 > 【トランスフォーメーション】** をクリックし、ウィザードの手順に従います。

マッピングまたはマブレットで再利用不可のトランスフォーメーションを作成するには、トランスフォーメーションパレットからトランスフォーメーションを選択し、エディタにドラッグします。

トランスフォーメーションによっては、作成時にモードの選択などの追加の設定が必要になるものがあります。例えば、ルックアップトランスフォーメーションを作成する場合は、ルックアップソースとして使用するデータオブジェクトを選択する必要があります。

作成したトランスフォーメーションはエディタに表示されます。トランスフォーメーションによっては、あらかじめ定義されたポートやグループが含まれていることがあります。それ以外のトランスフォーメーションは空です。

第 2 章

トランスフォーメーションポート

この章では、以下の項目について説明します。

- [トランスフォーメーションポートの概要, 60 ページ](#)
- [ポートの作成, 60 ページ](#)
- [ポートの設定, 61 ページ](#)
- [ポートのリンク, 61 ページ](#)
- [ポート属性のプロパゲート, 64 ページ](#)
- [Excel からのポートのコピー, 67 ページ](#)

トランスフォーメーションポートの概要

トランスフォーメーションポートは、マッピングまたはマップレットで接続されるデータの個別のカラムです。トランスフォーメーションは、入力ポートからデータを受信し、出力ポートを使用してデータを送信します。入出力ポート。データを受信し、変更せずに渡します。

各入力オブジェクト、出力オブジェクト、マップレット、およびトランスフォーメーションには、ポートの集まりが含まれます。入力オブジェクトはデータを提供するため、出力ポートのみを含みます。出力オブジェクトはデータを受け取るため、入力ポートのみを含みます。マップレットは入力ポートと出力ポートのみを含みます。トランスフォーメーションは、その種類と用途に応じて、入力ポート、出力ポート、および入出力ポートの組み合わせを含みます。

動的ポートは、アップストリームトランスフォーメーションから 1 つ以上のカラムを受信できます。また、動的ポートはデータフローに基づいて新しいカラムまたは変更されたカラムを受信できます。

マッピング内でトランスフォーメーションを作成したら、ポートを作成し、ポートプロパティを定義します。ポートを介してマッピングをターゲットとその他のトランスフォーメーションにリンクし、マッピングを完成させます。変更された属性をマッピング全体のポートに渡すには、ポート属性をプロパゲートします。

ポートの作成

トランスフォーメーションを作成する場合に、すべてのポートを手動で作成する必要はありません。例えば、ルックアップトランスフォーメーションを作成し、ルックアップテーブルを参照するとします。トランスフォーメーションポートを表示すると、トランスフォーメーションには、参照するテーブルの各カラムに対して出力ポートがあることがわかります。これらのポートを定義する必要はありません。

以下の方法でポートを作成します。

- **別のトランスフォーメーションからポートをドラッグする。**別のトランスフォーメーションからポートをドラッグすると、Designer は同じプロパティを持つポートを作成し、2 つのポートをリンクします。
- **【ポート】 タブの【追加】 ボタンをクリックする。**Designer は、設定可能な空のポートを作成します。

ポートの設定

トランスフォーメーションポートを定義する場合は、ポートのプロパティを定義します。ポートのプロパティには、ポート名、データ型、ポートタイプ、およびデフォルト値が含まれます。

次のポートのプロパティを設定します。

- **ポート名。**ポートの名前。ポートに名前を付けるときは、次の規則に従います。
 - 1 バイトまたは 2 バイトの文字か、1 バイトまたは 2 バイトのアンダスコア (_) で始めます。
 - 1 バイトまたは 2 バイトの文字のうち数値、アンダスコア (_)、\$、#、または@はいずれも、ポート名に含めることが可能です。
- **データ型、精度、およびスケール。**式または条件を入力する場合は、データ型が式の戻り値に一致することを確認します。
- **ポートタイプ。**トランスフォーメーションには、入力、出力、入出力、および変数のポートの種類の組み合わせが含まれる場合があります。
- **デフォルト値。**NULL 値または出力トランスフォーメーションエラーを含むポートのデフォルト値を割り当てます。いくつかのポートのデフォルト値を上書きすることができます。
- **説明。**ポートの説明。
- **その他のプロパティ。**トランスフォーメーションによっては、式プロパティやグループ化プロパティなどのトランスフォーメーション固有のプロパティを持つものがあります。

ポートのリンク

入力、出力、トランスフォーメーション、およびマプレットオブジェクトをマッピングで追加したら、マッピングオブジェクト間のポートをリンクしてマッピングを完成させます。

Developer ツールは、接続がリンクの検証および連結条件を満たした場合のみ、接続を作成します。

ポートは未接続のままにすることができます。Data Integration Service では、未接続のポートは無視されません。

入力オブジェクト、トランスフォーメーション、マプレット、および出力オブジェクト間でポートをリンクする場合、以下のタイプのリンクを作成できます。

- 1 対 1
- 1 対多

ポートは手動でリンクすることも自動的にリンクすることもできます。

1 対 1 のリンク

入力オブジェクトまたはトランスフォーメーションの 1 つのポートを、出力オブジェクトまたはトランスフォーメーションの 1 つのポートにリンクします。

1 対多のリンク

同一のデータを異なる目的で使用する場合には、このデータを提供するポートを、マッピング内の複数のポートにリンクすることができます。

以下の方法で、1 対多のリンクを作成できます。

- 1 つのポートを複数のトランスフォーメーションまたは出力オブジェクトにリンクする。
- 1 つのトランスフォーメーションの複数のポートを複数のトランスフォーメーションまたは出力オブジェクトにリンクする。

例えば、アグリゲータトランスフォーメーションを介して、銀行支店の給与情報を使用して平均給与を計算したい場合、各従業員の毎月の給与計算用に設定した式トランスフォーメーション内の同じ情報を使用できます。

ポートの手動リンク

1 つのポートまたは複数のポートを手動でリンクできます。

入力オブジェクトまたはトランスフォーメーションのポートから出力オブジェクトまたはトランスフォーメーションのポートにドラッグします。

Ctrl キーまたは Shift キーを使用して、別のトランスフォーメーションまたは出力オブジェクトにリンクする複数のポートを選択します。Developer ツールは、最上段のペアから順にポートをリンクします。検証条件を満たしているポートがすべてリンクされます。

空のポートにポートをドラッグした場合、Developer ツールはポートをコピーしてリンクを作成します。

ポートの自動リンク

ポートを自動的にリンクする場合は、位置または名前によってリンクを作成できます。

名前によってポートをリンクする場合は、接頭語または接尾語を指定してポートをリンクできます。接頭語または接尾語は、マッピング内のどこにポートがあるかを示すために使用します。

名前によるポートのリンク

名前でポートをリンクする場合、Developer ツールは同じ名前を持つ入力ポートおよび出力ポート間にリンクを追加します。トランスフォーメーション間で同じポート名を使用している場合に、名前によるリンクを行います。

ユーザーが定義したプレフィックスおよびサフィックスに基づき、ポートをリンクできます。接頭語または接尾語は、マッピング内のどこにポートがあるかを示すために使用します。ポート名でプレフィックスまたはサフィックスを使用してマッピングまたはマップレット内の出現位置を区別している場合、名前およびプレフィックス/サフィックスに基づいてリンクします。

名前によるリンクでは大文字と小文字は区別されません。

1. **[マッピング]** > **[オートリンク]** をクリックします。
[オートリンク] ダイアログボックスが表示されます。
2. **[リンク元]** ウィンドウで、リンク元のオブジェクトを選択します。

3. **【リンク先】** ウィンドウで、リンク先のオブジェクトを選択します。
4. **【名前】** を選択します。
5. 必要に応じて、**【詳細の表示】** をクリックして、プレフィックスまたはサフィックスに基づいてポートをリンクします。
6. **【OK】** をクリックします。

位置によるポートのリンク

位置によるリンクを実行すると、Developer ツールは各出力ポートに対応する入力ポートにリンクします。例えば、1 番目の出力ポートは 1 番目の入力ポートに、2 番目の出力ポートは 2 番目の入力ポートにリンクされます。位置によるリンクは、関連するポートが同じ順で設定されているトランスフォーメーションを作成する場合に使用します。

1. **【マッピング】 > 【オートリンク】** をクリックします。
【オートリンク】 ダイアログボックスが表示されます。
2. **【リンク元】** ウィンドウで、リンク元のオブジェクトを選択します。
3. **【リンク先】** ウィンドウで、リンク先のオブジェクトを選択します。
4. **【位置】** を選択し、**【OK】** をクリックします。
Developer ツールは各出力ポートに対応する入力ポートにリンクします。例えば、1 番目の出力ポートは 1 番目の入力ポートに、2 番目の出力ポートは 2 番目の入力ポートにリンクされます。

ポートのリンクに関するルールおよびガイドライン

ポートをリンクする際に適用されるルールとガイドラインがあります。

マッピングオブジェクトを接続する場合は、以下の規則とガイドラインに注意してください。

- 2 つのマッピングオブジェクト間でポートをリンクしようとした時にエラーが検出されると、ポートをリンクできないことを示す記号が表示されます。
- マッピングのデータフローのロジックに従います。以下のタイプのポートをリンクできます。
 - 受け取る側のポートは入力または入出力ポートでなければなりません。
 - 送る側のポートは、出力または入出力ポートでなければなりません。
 - 入力ポートを入力ポートにリンクしたり、出力ポートを出力ポートにリンクすることはできません。
- 少なくとも 1 つの入力グループのポートを先行するトランスフォーメーションにリンクする必要があります。
- 少なくとも 1 つの出力グループのポートを後続のトランスフォーメーションにリンクする必要があります。
- 1 つのアクティブなトランスフォーメーション、またはアクティブなトランスフォーメーションの 1 つの出力グループから、別のトランスフォーメーションの入力グループにリンクできます。
- アクティブなトランスフォーメーションとパッシブなトランスフォーメーションを同じ後続トランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。
- 2 つ以上のアクティブなトランスフォーメーションを同じ後続トランスフォーメーションまたはトランスフォーメーション入力グループに接続することはできません。
- 任意の数のパッシブなトランスフォーメーションは、同じ後続トランスフォーメーション、トランスフォーメーション入力グループ、またはターゲットに接続できます。
- 両方の出力グループのデータがソートされている場合、同一のトランスフォーメーション内の 2 つの出力グループから、ソート済みデータに設定されている 1 つのジョイナトランスフォーメーションにポートをリンクできます。

- 互換性のあるデータ型を持つポートのみをリンクできます。Developer ツールは 2 つのデータ型間でマッピングが可能であることを検査してから、リンクします。Data Integration Service は、データ型に互換性がないポート間ではデータを変換することはできません。
- マッピングがデータフロー検証に違反する場合、Developer ツールによって無効とマークされます。

ポート属性のプロパゲート

変更された属性をマッピング全体のポートに渡すには、ポート属性をプロパゲートします。

1. エディタでトランスフォーメーションのポートを選択します。
2. **【マッピング】 > 【属性のプロパゲート】** をクリックします。
【属性のプロパゲート】 ダイアログボックスが表示されます。
3. 属性をプロパゲートする方向を選択します。
4. プロパゲートする属性を選択します。
5. 必要に応じて、結果をプレビューします。
6. **【適用】** をクリックします。
ポート属性がプロパゲートされます。

依存関係のタイプ

ポート属性をプロパゲートすると、Developer ツールは依存関係を更新します。

Developer ツールは以下の依存関係を更新できます。

- リンクパスの依存関係
- 暗黙の依存関係

リンクパスの依存関係

リンクパスの依存関係とは、プロパゲートされたポートおよびリンクパス内のポートの間の依存関係です。

リンクパスの依存関係をプロパゲートすると、Developer ツールは、前方リンクパスのすべての入力および出力ポート、後方リンクパスのすべての出力および入出力ポートを更新します。Developer ツールは以下の更新を実行します。

- プロパゲートされたポートのリンクパスにおけるすべてのポートで、ポート名、データ型、精度、位取り、および説明を更新します。
- 変更されたポート名を持つプロパゲートされたポートを参照するすべての式または条件を更新します。
- 関連ポート名が変更された場合、動的 Lookup トランスフォーメーション内の関連ポートのプロパティを更新します。

暗黙の依存関係

暗黙の依存関係とは、式または条件に基づく 2 つのポートの間のトランスフォーメーション内にある依存関係です。

暗黙の依存関係を持つポートに対して、データ型、精度、位取り、および説明をプロパゲートできます。条件と式を解析して、プロパゲートされるポートのデータ型の暗黙の依存関係を特定できます。暗黙の依存関係を持つポートはすべて、出力または入出力ポートです。

条件を含めると、Developer ツールは以下の依存関係を更新します。

- リンクパスの依存関係
- プロパゲートされたポートと同じルックアップ条件で使用される出力ポート
- プロパゲートされたポートに関連付けられた、動的ルックアップトランスフォーメーションの関連ポート
- 明細ポートと同じ結合条件で使用されるマスタポート

式を含めると、Developer ツールは以下の依存関係を更新します。

- リンクパスの依存関係
- プロパゲートされたポートを使用する式を含む出力ポート

Developer ツールは、同一のトランスフォーメーション内の暗黙の依存関係にはプロパゲートしません。変更した属性は別のトランスフォーメーションからプロパゲートする必要があります。例えば、ルックアップ条件で使用されるポートのデータ型を変更して、その変更をルックアップトランスフォーメーションからプロパゲートすると、Developer ツールは、同じルックアップトランスフォーメーション内の条件に依存する他のポートにその変更をプロパゲートしません。

トランスフォーメーションでプロパゲートされるポート属性

Developer ツールは、各トランスフォーメーションの依存関係と属性をプロパゲートします。

以下の表に、Developer ツールが各トランスフォーメーションに対してプロパゲートする依存関係および属性を示します。

トランスフォーメーション	依存関係	プロパゲートされる属性
アドレスバリデータ	なし。	None このトランスフォーメーションは定義済みのポート名とデータ型を持ちます。
アグリゲータ	<ul style="list-style-type: none">- リンクパス内のポート- 式- 暗黙の依存性	<ul style="list-style-type: none">- ポート名、データ型、精度、位取り、説明- ポート名- データ型、精度、スケール
関連付け	<ul style="list-style-type: none">- リンクパス内のポート	<ul style="list-style-type: none">- ポート名、データ型、精度、位取り、説明
大文字小文字変換プログラム	<ul style="list-style-type: none">- リンクパス内のポート	<ul style="list-style-type: none">- ポート名、データ型、精度、位取り、説明
分類子	<ul style="list-style-type: none">- リンクパス内のポート	<ul style="list-style-type: none">- ポート名、データ型、精度、位取り、説明
比較	<ul style="list-style-type: none">- リンクパス内のポート	<ul style="list-style-type: none">- ポート名、データ型、精度、位取り、説明
コンソリデータ	なし。	None このトランスフォーメーションは定義済みのポート名とデータ型を持ちます。

トランスフォーメーション	依存関係	プロパゲートされる属性
データマスキング	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
データプロセッサ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ディシジョン	- リンクパスのダウストリームポート	- ポート名、データ型、精度、位取り、説明
式	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - ポート名 - データ型、精度、スケール
フィルタ	- リンクパス内のポート - 条件	- ポート名、データ型、精度、位取り、説明 - ポート名
階層型からリレーショナル	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ジョイナ	- リンクパス内のポート - 条件 - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - ポート名 - データ型、精度、スケール
キージェネレータ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ラベラ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ルックアップ	- リンクパス内のポート - 条件 - 関連ポート（動的ルックアップ） - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - ポート名 - ポート名 - データ型、精度、スケール
一致	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
マージ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ノーマライザ	- リンクパス内のポート	- ポート名
パーサー	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ランク	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - ポート名 - データ型、精度、スケール
読み取り		
REST Web サービス コンシューマ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
ルータ	- リンクパス内のポート - 条件	- ポート名、データ型、精度、位取り、説明 - ポート名
シーケンスジェネレータ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明

トランスフォーメーション	依存関係	プロパゲートされる属性
ソータ	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
SQL	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
標準化	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
共有体	- リンクパス内のポート - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - データ型、精度、スケール
アップデートストラテジ	- リンクパス内のポート - 式 - 暗黙の依存性	- ポート名、データ型、精度、位取り、説明 - ポート名 - データ型、精度、スケール
加重平均	- リンクパス内のポート	- ポート名、データ型、精度、位取り、説明
書き込み		

Excel からのポートのコピー

Excel でポートとポートプロパティを設定し、Developer tool でトランスフォーメーションポートにコピーすることができます。ポートプロパティには、カラム名、データ型、精度、スケールなどがあります。多くのポートを持つトランスフォーメーションを作成または編集する必要がある場合に、このような作業をお勧めします。

メタデータを次のトランスフォーメーションのタイプにコピーできます。

- アグリゲータ
- 式
- フィルタ
- Java
- ジョイナ
- ルックアップ
- ノーマライザ
- ランク
- 読み取り
- ルーター
- シーケンス
- ソーター
- SQL
- 共有体
- アップデートストラテジ
- Web サービスコンシューマ

- ウィンドウ
- 書き込み

Excel でのトランスフォーメーションの編集

トランスフォーメーションの大部分を編集しなければならない場合、Developer tool ですべての値を変更する必要はありません。代わりに、トランスフォーメーションポートを Excel にコピーし、オートフィル機能を使用してすべての値を同時に変更してから、トランスフォーメーションポートをもう一度 Developer tool に【貼り付け（置換）】できます。

1. Developer tool のマッピングエディタで、ポートのコピー元のトランスフォーメーションを選択します。
2. Developer tool から元のトランスフォーメーションをコピーするには、[ポート] 内で右クリックし、【すべて選択】をクリックします。
3. ポートを Excel スプレッドシートにコピーします。
4. Excel スプレッドシート内で変更します。メタデータの大部分を変更する場合、Excel のオートフィル機能を使用できます。これにより、フィルハンドルをドラッグし、連続するセルに基づいてデータを入力できます。詳細については「[例: Excel でのトランスフォーメーションの編集](#)（ページ 69）」を参照してください。
5. Excel からメタデータをコピーします。
6. 変更した内容でトランスフォーメーションを更新するには、[ポート] 内で右クリックし、【貼り付け（置換）】をクリックします。

Developer Tool へのメタデータのコピー

Excel でトランスフォーメーションポートを作成し、Developer tool にコピーできます。

1. Developer tool で必要なトランスフォーメーションを使用してマッピングを作成します。
2. Excel でトランスフォーメーションのメタデータを定義します。
3. Excel からメタデータをコピーします。
4. Developer tool でメタデータをトランスフォーメーションに移動するには、[ポート] 内で右クリックし、【貼り付け（置換）】をクリックします。

次の図は、Excel テーブルの例と、メタデータを Developer tool にコピーした後に作成されるトランスフォーメーションを示しています。

	A	B	C	D
1	Name	Type	Precision	Scale
2	EMPNO	decimal	10	0
3	ENAME	string	6	0
4	JOB	string	9	0
5	MGR	decimal	4	0
6	HIREDATE	string	19	0
7	SAL	decimal	7	0
8	COMM	string	7	0
9	DEPTNO	decimal	2	0

→

Properties Data Viewer Alerts									
General									
Ports									
	Name	Type	Precision	Scale	Input	Output	Variable	Expression	
Port Selectors	1	EMPNO	decimal	10	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		EMPNO
Dependencies	2	ENAME	string	6	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		ENAME
Run-time Linking	3	JOB	string	9	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		JOB
Advanced	4	MGR	decimal	4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		MGR
Mapping Outputs	5	HIREDATE	string	19	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		HIREDATE
	6	SAL	decimal	7	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		SAL
	7	COMM	string	7	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		COMM
	8	DEPTNO	decimal	2	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		DEPTNO

Sample Excel table

After metadata is copied to the Developer tool

注: 各セルの値をトランスフォーメーションにコピーする前に、その値が有効であることを確認する必要があります。例えば、文字列タイプには「0」以外のスケール値を指定できません。精度値には文字を指定できず、タイプ値には数値を指定できません。メタデータが正しくない場合、エラーメッセージが表示されます。

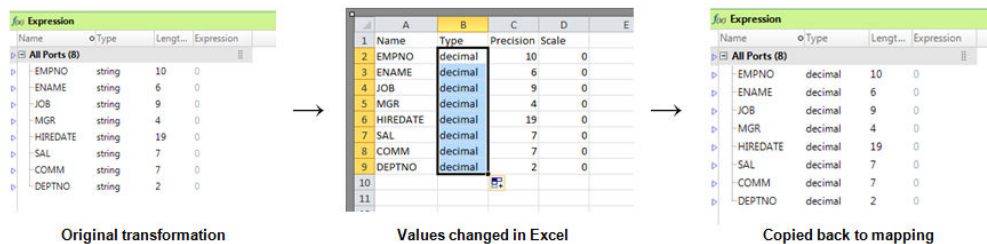
例: Excel でのトランスフォーメーションの編集

トランスフォーメーションの作成中で、すべての文字列データ型を 10 進型に変更する必要があるとします。各フィールドを個別に変更するのではなく、Excel でグローバルに変更を行い、フィールドを Developer tool にコピーします。

1. [ポート] 内で右クリックし、[すべて選択] をクリックして、メタデータを Excel に貼り付けます。
2. 最初のデータ型の値を「string」から「decimal」に変更し、フィルハンドルを使用してカラム内の残りのセルを自動的に変更します。
3. 変更した内容でトランスフォーメーションを更新するには、Excel からメタデータをコピーし、[ポート] 内で右クリックして [貼り付け (置換)] をクリックします。

Excel を使用すると、各フィールドを個別に変更する必要がなくなります。

次の図は、トランスフォーメーションを Excel に移動し、オートフィル機能を使用して特定の値を変更してから、トランスフォーメーションをもう一度 Developer tool にコピーするプロセスを示しています。



Excel からのコピーに関するルールおよびガイドライン

メタデータを Excel から Developer tool にコピーするときは、次のルールとガイドラインを考慮してください。

- 各セルの値をトランスフォーメーションにコピーする前に、その値が有効であることを確認する必要があります。例えば、文字列タイプには「0」以外のスケール値を指定できません。精度値には文字を指定できず、タイプ値には数値を指定できません。メタデータが正しくない場合、エラーメッセージが表示されます。
- メタデータのコピー先の場所は、更新するトランスフォーメーションのタイプによって異なります。例えば、[貼り付け (置換)] オプションは、トランスフォーメーションの [プロパティ] ビューで右クリックすると使用できない可能性があります。ただしこのオプションは、エディタでトランスフォーメーションポート内を直接右クリックすると引き続き使用できます。

第 3 章

トランスフォーメーションキャッシュ

この章では、以下の項目について説明します。

- [トランスフォーメーションキャッシュの概要, 70 ページ](#)
- [キャッシュタイプ, 71 ページ](#)
- [キャッシュファイル, 71 ページ](#)
- [キャッシュサイズ, 72 ページ](#)
- [データ統合サービスによるキャッシュサイズの増加, 74 ページ](#)
- [パーティション化されたキャッシュのキャッシュサイズ, 75 ページ](#)
- [キャッシュサイズの最適化, 75 ページ](#)

トランスフォーメーションキャッシュの概要

データ統合サービスでは、マッピング時にアグリゲータ、ジョイナ、ルックアップ、ランク、およびソーターの各トランスフォーメーションに対して、キャッシュメモリが割り当てられます。データ統合サービスは、アグリゲータ、ジョイナ、ルックアップ、ランクの各トランスフォーメーションに対してはインデックスキャッシュとキャッシュメモリを作成し、ソータートランスフォーメーションに対してはキャッシュを 1 つ作成します。

これらのトランスフォーメーションに対してキャッシュサイズを設定できます。キャッシュサイズによって、データ統合サービスがマッピングの実行開始時に各トランスフォーメーションに割り当てるメモリ量が決まります。

キャッシュサイズがマシン上で使用可能なメモリよりも大きい場合は、データ統合サービスが十分なメモリを割り当てることができず、マッピングの実行が失敗します。

キャッシュサイズがトランスフォーメーションの実行に必要なメモリ量よりも小さい場合、データ統合サービスはトランスフォーメーションの一部をメモリで処理して、オーバーフローしたデータをキャッシュファイルに格納します。データ統合サービスがキャッシュファイルをディスクにページングすると、処理時間が長くなります。最適なパフォーマンスのため、データ統合サービスがメモリですべてのトランスフォーメーションを処理できるようにキャッシュサイズを設定します。

デフォルトでは、データ統合サービスは割り当て可能なメモリの最大量に基づいて、実行時に必要なメモリを自動的に計算します。自動キャッシュモードでマッピングを実行すると、トランスフォーメーションのキャッシュサイズを調整できます。マッピングログでトランスフォーメーション統計を分析し、最適なパフォーマンスに必要なキャッシュサイズを決定して、トランスフォーメーションに固有のキャッシュサイズを設定します。

キャッシュタイプ

アグリゲータ、ジョイナ、ルックアップ、ランクの各トランスフォーメーションでは、インデックスキャッシュとデータキャッシュが必要です。データ統合サービスではインデックスキャッシュにキー値が格納され、データキャッシュに出力値が格納されます。ソータートランスフォーメーションにはキャッシュが1つ必要です。データ統合サービスはソートキーとソートされるデータをソーターキャッシュに格納します。

以下の表に、データ統合サービスで各キャッシュに保存される情報のタイプを示します。

マッピングオブジェクト	キャッシュタイプと説明
アグリゲータ	<ul style="list-style-type: none">- インデックス。GroupBy ポートの設定に従ってグループ値を格納します。- データ。GroupBy ポートに基づいて計算結果を格納します。
ジョイナ	<ul style="list-style-type: none">- インデックス。結合条件に、一意キーを持つすべてのマスター行を格納します。- データ。マスターソースの行を格納します。
ルックアップ	<ul style="list-style-type: none">- インデックス。ルックアップ条件の情報を格納します。- データ。インデックスキャッシュに格納されないルックアップデータを格納します。
ランク	<ul style="list-style-type: none">- インデックス。GroupBy ポートの設定に従ってグループ値を格納します。- データ。GroupBy ポートに基づいてランク情報を格納します。
ソータ	<ul style="list-style-type: none">- ソータ。ソートキーおよびデータを格納します。

キャッシュファイル

マッピングを実行すると、アグリゲータ、ジョイナ、ルックアップ、ランク、およびソーターの各トランスフォーメーションに対して、少なくとも1つのキャッシュファイルが作成されます。データ統合サービスがメモリでトランスフォーメーションを実行できない場合、オーバーフローしたデータがキャッシュファイルに書き込まれます。

以下の表に、データ統合サービスでさまざまなマッピングオブジェクトに対して作成されるキャッシュファイルのタイプを示します。

マッピングオブジェクト	キャッシュファイル
アグリゲータ、ジョイナ、ルックアップ、およびランクの各トランスフォーメーション	データ統合サービスでは、以下のタイプのキャッシュファイルが作成されます。 <ul style="list-style-type: none">- インデックスキャッシュおよびデータキャッシュごとに1つのヘッダファイル- インデックスキャッシュおよびデータキャッシュごとに1つのデータファイル
ソータートランスフォーメーション	データ統合サービスでは、1つのソーターキャッシュファイルが作成されます。

マッピングを実行すると、データ統合サービスは、マッピングログにキャッシュファイル名とトランスフォーメーション名を示すメッセージを書き込みます。マッピングが完了すると、データ統合サービスはキャッシュ

メモリを解放し、通常はキャッシュファイルを削除します。以下の状況では、インデックスキャッシュファイルおよびデータキャッシュファイルがキャッシュディレクトリに保持される場合があります。

- 永続キャッシュを使用してルックアップトランスフォーメーションを設定した場合。
- マッピングが正常終了しなかった場合。次回マッピングを実行すると、データ統合サービスによって既存のキャッシュファイルが削除され、新しいキャッシュファイルが作成されます。

キャッシュファイルへの書き込みによってマッピングのパフォーマンスが低下する可能性があるため、メモリでトランスフォーメーションが実行されるようにキャッシュサイズを設定します。

キャッシュファイルディレクトリ

アグリゲータ、ジョイナ、ルックアップ、およびランクの各トランスフォーメーションの場合、データ統合サービスは「キャッシュディレクトリ」プロパティで指定されたディレクトリ内にキャッシュファイルを作成します。ソータートランスフォーメーションの場合、「作業ディレクトリ」プロパティで指定されたディレクトリ内にキャッシュファイルが作成されます。

データ統合サービスプロセスでディレクトリが見つからない場合、マッピングは失敗し、キャッシュファイルを作成することまたは開くことができなかったことを示すメッセージがマッピングログに書き込まれます。データ統合サービスで、複数のキャッシュファイルが作成される場合もあります。キャッシュファイルの数は、キャッシュディレクトリのディスク利用可能なディスクスペース容量によって制限されます。

キャッシュサイズ

キャッシュサイズによって、データ統合サービスがマッピングの実行開始時に各トランスフォーメーションキャッシュに割り当てるメモリ量が決まります。自動キャッシュモードを使用するか、または固有の値を使用するようにトランスフォーメーションキャッシュサイズを設定できます。

自動キャッシュサイズ

デフォルトでは、トランスフォーメーションキャッシュサイズは「自動」に設定されています。データ統合サービスでは実行時のキャッシュメモリ要件が自動的に計算されます。サービスで割り当てることができるメモリの最大量を定義できます。

データ統合サービスでは、次のガイドラインに従ってメモリが自動的に割り当てられます。

処理時間がより長いトランスフォーメーションにより多くのメモリを割り当てる。

データ統合サービスでは、一般的に処理時間がより長いトランスフォーメーションにより多くのメモリが割り当てられます。例えば、データ統合サービスではソータートランスフォーメーションにより多くのメモリが割り当てられますが、これは、ソータートランスフォーメーションの実行により長い時間がかかるためです。

インデックスキャッシュよりも多くのメモリをデータキャッシュに割り当てる。

アグリゲータ、ジョイナ、ルックアップ、ランクの各トランスフォーメーションでは、インデックスキャッシュとデータキャッシュが必要です。データ統合サービスは、トランスフォーメーションに割り当てられたメモリをインデックスキャッシュとデータキャッシュ間で分割する際に、データキャッシュにより多くのメモリを割り当てます。

ソータートランスフォーメーションにはキャッシュが1つ必要です。ソータートランスフォーメーションに割り当てられたすべてのメモリがソーターキャッシュに割り当てられます。

自動キャッシュサイズの最大メモリ

Administrator ツールを使って、データ統合サービスモジュールの「要求ごとの最大メモリ」プロパティで、データ統合サービスがトランスフォーメーションキャッシュに割り当てることができるメモリの最大量を定義します。

各モジュールは、それぞれメモリ要件が異なるさまざまなタイプの要求を実行します。例えば、マッピング要求とプロファイル要求は一般的に、SQL サービス要求や Web サービス要求よりも多くのキャッシュメモリを必要とします。以下のデータ統合サービスモジュールに対して「要求ごとの最大メモリ」プロパティを設定できます。

- マッピングサービスモジュール
- プロファイリングサービスモジュール
- SQL サービスモジュール
- Web サービスモジュール

注: マッピングサービスモジュール要求には、マッピングとワークフロー内のマッピングタスクから実行されるマッピングが含まれます。

プロファイリングサービスモジュールの場合、「要求ごとの最大メモリ」は、単一のプロファイル要求に対する各マッピング実行にデータ統合サービスが割り当てることができるメモリの最大量を定義します。

残りモジュールでは、「要求ごとの最大メモリ」の動作はデータ統合サービスの設定に依存します。動作は、データ統合サービスの「ジョブオブションの開始」プロパティおよび「最大メモリサイズ」プロパティによって異なります。

以下の表に、データ統合サービスの設定に基づくマッピング、SQL サービス、および Web サービスの各モジュールの「要求ごとの最大メモリ」の動作を示します。

データ統合サービスの設定	「要求ごとの最大メモリ」の動作
ローカルまたはリモートの別々のシステムプロセスでジョブを実行する場合、または「最大メモリサイズ」が 0（デフォルト）の場合	<p>データ統合サービスが自動キャッシュモードを使用しているすべてのトランスフォーメーションに 1 つの要求で割り当てることができるメモリの最大量（バイト）。</p> <p>「要求ごとの最大メモリ」に定義した値は、自動キャッシュモードを使用するトランスフォーメーションにのみ影響します。データ統合サービスは、固有のキャッシュサイズを設定したトランスフォーメーションに、個別にメモリを割り当てます。要求によって使用されるメモリ n 合計は、要求ごとの最大メモリの値を超えることができます。</p> <p>例えば、「要求ごとの最大メモリ」が 800MB に設定されているとします。マッピングにキャッシングを必要とする 3 つのトランスフォーメーションがある場合に、2 つのトランスフォーメーションを自動キャッシュモードを使用するように設定し、3 つめのトランスフォーメーションを合計 500MB のキャッシュサイズを使用するように設定すると、データ統合サービスはすべてのトランスフォーメーションキャッシュに対し、合計 1,300MB のメモリを割り当てます。</p>
データ統合サービスプロセスでジョブを実行し、「最大メモリサイズ」が 0 より大きい場合	<p>データ統合サービスで 1 つの要求に割り当てることができるメモリの最大量（バイト）。</p> <p>「要求ごとの最大メモリ」プロパティに定義した値は、すべてのトランスフォーメーションに影響します。要求によって使用されるメモリ合計は、要求ごとの最大メモリの値を超えることはできません。</p>

自動キャッシュモードで使われるメモリの最大量を増やす場合は、モジュールに対するすべての要求に使用できる最大キャッシュサイズを増やします。メモリの最大量を増やして、キャッシュファイルがディスクにページングされないようにすることができます。ただし、この値はすべての要求で使われるため、データ統合サービスが一部の要求に対して必要以上のメモリを割り当てることがあります。

固有のキャッシュサイズ

トランスフォーメーションに固有のキャッシュサイズを設定できます。データ統合サービスは、マッピングの実行開始時に指定されたメモリ量をトランスフォーメーションキャッシュに割り当てます。キャッシュサイズを調整する場合は、固有の値をバイト単位で設定します。

初めてキャッシュサイズを設定する場合は、自動キャッシュモードを使用します。マッピングを実行した後、マッピングログでトランスフォーメーション統計を分析して、メモリでトランスフォーメーションを実行するために必要なキャッシュサイズを決定します。マッピングログで指定された値を使用するようにキャッシュサイズを設定すると、割り当てられたメモリが無駄になることを防げます。ただし、最適なキャッシュサイズはソースデータのサイズによって異なります。その後もマッピングの実行後にマッピングログを確認して、キャッシュサイズの変化を監視します。再利用可能なトランスフォーメーションに固有のキャッシュサイズを設定する場合は、マッピングでのトランスフォーメーションのそれぞれの使用に対してキャッシュサイズが最適かどうかを検証します。

固有のキャッシュサイズを定義するには、Developer tool のトランスフォーメーションプロパティでキャッシュサイズを設定します。

データ統合サービスによるキャッシュサイズの増加

データ統合サービスでは、設定したキャッシュサイズに基づいて、各メモリキャッシュが作成されます。より多くのメモリを必要とする場合には、データ統合サービスが設定したキャッシュサイズを増やす場合があります。

データ統合サービスでは、以下のいずれかの理由により、設定したキャッシュサイズを増やすことがあります。

設定されたキャッシュサイズが演算の処理に必要な最小キャッシュサイズよりも小さい場合。

データ統合サービスには、各マッピングを初期化するための最小メモリ容量が必要です。設定したキャッシュサイズが必要な最小キャッシュサイズよりも小さい場合は、最小要件を満たすように、データ統合サービスによって設定済みのキャッシュサイズが増加されます。データ統合サービスが必要な最小メモリを割り当てることができない場合、マッピングは失敗します。

設定されたキャッシュサイズがキャッシュページサイズの倍数になっていない場合。

データ統合サービスでは、キャッシュされたデータはキャッシュページに格納されます。キャッシュされたページは、キャッシュに均等に格納する必要があります。例えば、キャッシュサイズを 10MB (1,048,576 バイト) に設定し、キャッシュページサイズが 10,000 バイトである場合、データ統合サービスでは、設定したキャッシュサイズが 10,000 バイトのページサイズの倍数になるように 1,050,000 バイトまで増やされます。

データ統合サービスは、設定したキャッシュサイズを増やすと、マッピングの実行を続行して次のメッセージをマッピングログに書き込みます。

```
INFO: MAPPING, TE_7212, Increasing [Index Cache] size for transformation <transformation name> from  
<configured cache size> to <new cache size>.  
INFO: MAPPING, TE_7212, Increasing [Data Cache] size for transformation <transformation name> from  
<configured cache size> to <new cache size>.
```


パーティション化されたキャッシュのキャッシュサイズ

パーティション化オプションがある場合、キャッシュのパーティション化によって、アグリゲータ、ジョイナ、ランク、ルックアップ、またはソーターの各トランスフォーメーションを実行するパーティションごとに個別のキャッシュが作成されます。キャッシュのパーティション化の実行中に、各パーティションは異なるデータを個別のキャッシュに保存します。データ統合サービスはこれらのトランスフォーメーションに対してキャッシュのパーティション化を使用する場合、割り当てられたキャッシュサイズをパーティション全体に分割します。

例えば、トランスフォーメーションキャッシュサイズを 100MB に設定したとします。データ統合サービスは 4 つのパーティションを使用してトランスフォーメーションを実行します。サービスは、各パーティションが最大 25MB をキャッシュサイズとして使用するよう、キャッシュサイズ値を分割します。

キャッシュサイズの最適化

マッピングのパフォーマンスを最適化するため、データ統合サービスによってメモリですべてのトランスフォーメーションを実行できるようにキャッシュサイズが設定されます。

最適なキャッシュサイズを設定するには、以下のタスクを実行します。

1. トレースレベルを [詳細 - 初期化] に設定する。
2. 自動キャッシュモードでマッピングを実行する。
3. マッピングログでキャッシングパフォーマンスを分析する。
4. キャッシュサイズに固有の値を設定する。

手順 1. トレースレベルを [詳細 - 初期化] に設定する

Developer tool でトレースレベルを [詳細 - 初期化] に設定して、データ統合サービスがトランスフォーメーション統計をマッピングログに書き込めるようにします。トランスフォーメーション統計では、最適なパフォーマンスに必要なキャッシュサイズが一覧表示されます。デフォルトでは、トレースレベルは [ノーマル] に設定されています。

以下のいずれかの方法で、トレースレベルを [詳細 - 初期化] に設定します。

- キャッシュを使用する各トランスフォーメーションの詳細プロパティを変更します。
- Developer tool から初めてマッピングを実行する予定がある場合は、デフォルトのマッピング設定のプロパティを変更します。詳細については、『*Informatica Developer Tool ガイド*』を参照してください。
- コマンドラインから初めてデプロイ済みマッピングを実行する予定がある場合は、そのマッピングが含まれるアプリケーションの詳細プロパティを変更します。詳細については、『*Informatica Developer Tool ガイド*』を参照してください。

手順 2. 自動キャッシュモードでマッピングを実行する

初めてマッピングを実行する際には、トランスフォーメーションのキャッシュサイズに自動キャッシュモードを使用します。

マッピングは Developer tool から実行できます。または、アプリケーションにマッピングを追加してから、そのアプリケーションをデータ統合サービスにデプロイすることで、コマンドラインからマッピングを実行することもできます。

手順 3. キャッシングパフォーマンスを分析する

自動キャッシュモードでマッピングを実行したら、マッピングログのトランスフォーメーション統計を分析して、最適なマッピングパフォーマンスに必要なキャッシュサイズを決定します。

アグリゲータ、ジョイナ、ルックアップ、またはランクの各トランスフォーメーションがディスクにページングする場合、メモリでトランスフォーメーションを実行するために必要なインデックスキャッシュサイズとデータキャッシュサイズがマッピングログで指定されます。例えば、AGG_TRANS というアグリゲータトランスフォーメーションを実行すると、マッピングログには、以下のテキストが含まれます。

```
CMN_1791, The index cache size that would hold [1098] aggregate groups of input rows for [AGG_TRANS], in
memory, is [286720] bytes
CMN_1790, The data cache size that would hold [1098] aggregate groups of input rows for [AGG_TRANS], in
memory, is [1774368] bytes
```

このログは、ディスクにページングしないでメモリでトランスフォーメーションを実行するには、インデックスキャッシュには 286,720 バイト、データキャッシュには 1,774,368 バイトが必要なことを示しています。

ソータトランスフォーメーションがディスクにページングする場合、マッピングログにデータ統合サービスがソースデータ上に複数のパスを作成したことが示されます。ソートを完了するためにディスクにページングする必要がある場合、データ統合サービスによってデータに複数のパスが作成されます。データ統合サービスでデータが 1 回読み込まれ、ディスクにページングしないでメモリでソートが実行される場合に、このメッセージには単一パスに必要なバイト数が表示されます。

例えば、SRT_TRANS というソータトランスフォーメーションを実行すると、マッピングログには、以下のテキストが含まれます。

```
SORT_40427, Sorter Transformation [SRT_TRANS] required 2-pass sort (1-pass temp I/O: 13126221824 bytes).
You may try to set the cache size to 14128 MB or higher for 1-pass in-memory sort.
```

このログは、データ統合サービスがデータに 1 つのパスを作成するには、ソーターキャッシュに 14,128MB が必要なことを示しています。

手順 4. 固有のキャッシュサイズを設定する

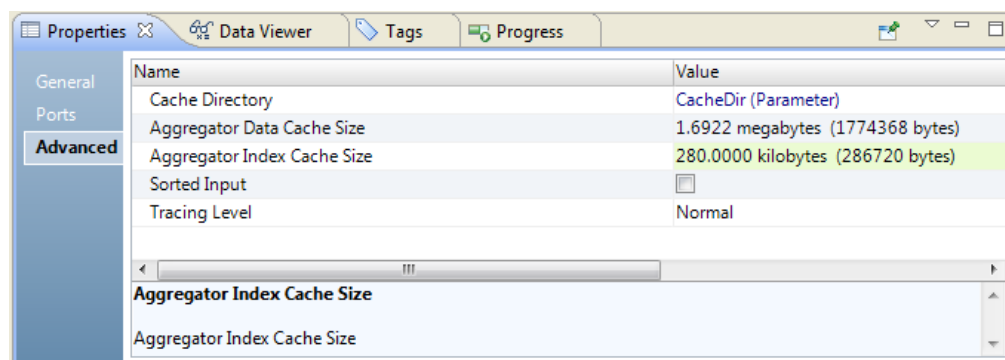
最適なパフォーマンスのため、マッピングログで指定された値を使用するようにトランスフォーメーションのキャッシュサイズを設定します。Developer tool でトランスフォーメーションのインデックスキャッシュサイズとデータキャッシュサイズのプロパティを更新します。

- Developer tool で、再利用可能なトランスフォーメーションまたは再利用不可能なトランスフォーメーションを開きます。
- 以下のトランスフォーメーションタイプに応じて、キャッシュサイズのプロパティを特定します。

オプション	説明
アグリゲータ、ジョイナ、ランク、またはソーターの再利用可能なトランスフォーメーション	【詳細】 ビューをクリックします。
アグリゲータ、ジョイナ、ランク、またはソーターの再利用不可能なトランスフォーメーション	【プロパティ】 ビューで 【詳細】 タブをクリックします。
再利用可能なルックアップトランスフォーメーション	【ランタイム】 ビューをクリックします。
再利用不可能なルックアップトランスフォーメーション	【プロパティ】 ビューで 【ランタイム】 タブをクリックします。

- マッピングログで推奨されているインデックスキャッシュサイズとデータキャッシュサイズの値をバイト単位で入力します。

以下の図は、インデックスキャッシュサイズとデータキャッシュサイズに固有の値が設定されている再利用不可能なアグリゲータ変換を示しています。



4. [ファイル] > [保存] をクリックします。

第 4 章

アドレスバリデータトランスフォーメーション

この章では、以下の項目について説明します。

- [アドレスバリデータトランスフォーメーションの概要, 79 ページ](#)
- [アドレス参照データ, 79 ページ](#)
- [モードとテンプレート, 81 ページ](#)
- [ポートグループとポートの選択, 81 ページ](#)
- [アドレスバリデータトランスフォーメーションの入力ポートグループ, 82 ページ](#)
- [アドレスバリデータトランスフォーメーションの出力ポートグループ, 83 ページ](#)
- [複数インスタンスのポート, 85 ページ](#)
- [アドレス検証プロジェクト, 86 ページ](#)
- [フォーマットされたアドレスと郵便事業者の住所表記基準, 87 ページ](#)
- [部分的アドレスの入力, 88 ページ](#)
- [アドレスバリデータのステータスポート, 89 ページ](#)
- [アドレスバリデータトランスフォーメーションの全般設定, 100 ページ](#)
- [\[設定\] ウィンドウのアドレス検証のプロパティ, 101 ページ](#)
- [アドレス検証の詳細プロパティ, 104 ページ](#)
- [認証レポート, 124 ページ](#)
- [アドレスバリデータトランスフォーメーションの設定, 127 ページ](#)
- [アドレスバリデータトランスフォーメーションへのポートの追加, 127 ページ](#)
- [ユーザー定義テンプレートの作成, 128 ページ](#)
- [アドレスバリデータのモデルの定義, 128 ページ](#)
- [認証レポートの定義, 129 ページ](#)
- [非ネイティブ環境でのアドレスバリデータトランスフォーメーション, 129 ページ](#)

アドレスバリデータトランスフォーメーションの概要

アドレスバリデータトランスフォーメーションは、入力アドレスデータをアドレス参照データと比較する複数グループトランスフォーメーションです。このトランスフォーメーションは、アドレスの精度を特定し、アドレスのエラーを修正します。トランスフォーメーションにより、データコンテンツと構造の郵便配達基準を満たすレコードが作成されます。また、トランスフォーメーションにより、各住所にステータス情報が追加されます。

アドレスバリデータトランスフォーメーションによりアドレスデータに次の操作が実行されます。

- トランスフォーメーションは、ソースデータのアドレスレコードをアドレス参照データのアドレス定義と比較します。
- 各入力アドレスの有効性、配達可能ステータス、誤りの性質、それに含まれる曖昧さに関する詳しいステータスレポートを生成します。
- エラーを修正し、部分的なアドレスレコードを補完します。アドレスを修正するには、参照データのアドレスとの明確な一致を探す必要があります。トランスフォーメーションにより、アドレス参照データからアドレスレコードに必要なデータ要素がコピーされます。
- 配布ポイントの情報やジオコーディング情報など、標準アドレスには表示されないが郵便配達を支援する情報が追加されます。
- データプロジェクトと郵政事業者が必要とする形式で出力アドレスが書き込まれます。形式はトランスフォーメーションの出力ポートを選択するときに定義します。

アドレス参照データ

アドレス参照データセットは、各国の郵便事業者が国で認識するアドレスを記述します。アドレスバリデータトランスフォーメーションでアドレス検証を実行する前に、ドメインの Informatica サービスマシンにアドレス参照データをインストールします。Informatica からアドレス参照データを購入し、ダウンロードします。

ソースのアドレスデータが識別する国別のアドレス参照データファイルをインストールします。人口が多い国ではファイルが複数必要になることがあります。また、アドレスデータを追加補足するデータファイルをインストールできます。郵便事業者はデータ補足を行って、住所の精度を保証し、郵便配達時間を短縮することができます。

アドレス検証を実行すると、アドレスバリデータトランスフォーメーションによってそれぞれの入力レコードとアドレス参照データが比較されます。トランスフォーメーションによりアドレス参照データの中に入力アドレスが検出された場合、トランスフォーメーションでレコードを正しく完全なアドレスデータに更新できます。追加の参照データセットを購入した場合は、トランスフォーメーションでアドレスデータを補足することもできます。

Developer ツールの【設定】ウィンドウを使用し、ドメインの Informatica サービスマシンのアドレス参照データファイルに関する情報を表示します。

アドレス参照データの種類

選択した検証モードによって、トランスフォーメーションが入力アドレスとアドレス参照データを比較する方法が決まります。

アドレスバリデータトランスフォーメーションは次のアドレス参照データの種類を読み取ることができます。

アドレスコードルックアップデータ

アドレスコードルックアップデータをインストールして、入力ポートのコード値から部分的なアドレスまたはアドレス全体を取得します。アドレスの完全性は、アドレスが属する国のアドレスコードサポートのレベルによって決まります。入力アドレスからアドレスコードを読み取るには、[個別] ポートグループで国固有のポートを選択します。

次の国のポートを選択できます。

- オーストリア建物レベルで住所を返します。
- ドイツ。市区町村、地方自治体、または番地レベルで住所を返します。
- 日本。一意のメールボックスレベルで住所を返します。
- 南アフリカ。番地レベルで住所を返します。
- 韓国。一意のメールボックスレベルで住所を返します。
- セルビア。番地レベルで住所を返します。
- 英国。一意のメールボックスレベルで住所を返します。

アドレスコードルックアップモードで実行されるようにトランスフォーメーションを設定した場合、アドレスバリデータトランスフォーメーションはアドレスコードルックアップデータを読み取ります。

バッチデータと対話データ

バッチデータと対話データをインストールし、アドレスレコードのセットに対してアドレス検証を実行します。バッチデータと対話データを使用し、国の郵政事業者の現行の郵便データに基づき、入力アドレスが配送可能であり、完全であることを確認します。

バッチモードで実行するようにトランスフォーメーションを設定した場合、アドレスバリデータトランスフォーメーションは入力アドレスごとに1つのアドレスを返します。対話モードで実行するようにトランスフォーメーションを設定した場合、アドレスバリデータトランスフォーメーションは入力アドレスごとに1つ以上のアドレスを返します。

CAMEO データ

CAMEO データをインストールし、カスタマセグメント化データを居住地レコードに追加します。カスタマセグメント化データは、住所ごとに居住者の推定収入レベルと希望の生活様式を示しています。

トランスフォーメーションをバッチモードまたは認証モードで実行するように設定した場合、アドレスバリデータトランスフォーメーションはCAMEO データを読み込みます。

認証データ

認証データをインストールし、郵政事業者が定義する認証基準にアドレスレコードが一致することを確認します。配送ポイントデータ要素など、一意のメールボックスを識別できるデータ要素を含む場合、アドレスは認証基準を満たします。アドレスが認証基準を満たすとき、郵政事業者は配送料を割引きます。

次の国は認証基準を定義しています。

- オーストラリア AMAS (Address Matching Approval System/住所照合承認システム) 基準で郵便を認証します。
- カナダ SERP (Software Evaluation And Recognition Program/評価と認識のソフトウェアプログラム) 基準で郵便を認証します。
- フランス SNA (Service National de l'Adresse/国の住所サービス) 基準で郵便を認証します。
- ニュージーランド SendRight 基準で郵便を認証します。
- 米国。CASS (Coding Accuracy Support System/符号化精度支援システム) 基準で郵便を認証します。

トランスフォーメーションを認証モードで実行するように設定した場合、アドレスバリデータトランスフォーメーションは認証データを読み込みます。

ジオコードデータ

ジオコードデータをインストールし、アドレスレコードにジオコードを追加します。ジオコードは緯度と経度の座標です。

トランスフォーメーションをバッチモードまたは認証モードで実行するように設定した場合、アドレスバリデータトランスフォーメーションは geocode データを読み込みます。

提案リストデータ

提案リストをインストールし、部分的なアドレスレコードの有効な代替バージョンを見つけます。アドレスレコードをリアルタイムで1つずつ処理するようにアドレス検証マッピングを設定する場合に、提案リストのデータを使用します。アドレスバリデータトランスフォーメーションでは、部分的なアドレスのデータ要素を使用し、提案リストデータで重複チェックを実行します。トランスフォーメーションは、部分的なアドレス内の情報を含む有効なアドレスを返します。

トランスフォーメーションを提案リストモードで実行するように設定した場合、アドレスバリデータトランスフォーメーションは提案リストデータを読み込みます。

補足データ

補足データをインストールし、郵政事業者の郵便配達を支援できるアドレスレコードにデータを追加します。補足データを使用して、住所を含む地理的領域または郵便区域についての情報を追加します。国によっては、補足データで郵便システム内のメールボックスに対する一意の識別子を提供できます。

トランスフォーメーションをバッチモードまたは認証モードで実行するように設定した場合、アドレスバリデータトランスフォーメーションは補足データを読み込みます。

注: トランスフォーメーションは、アドレス参照データを国認識モードや解析モードで読み取りません。

関連項目：

- [「アドレスバリデータトランスフォーメーションの全般設定」 \(ページ 100\)](#)

モードとテンプレート

アドレスバリデータトランスフォーメーションを設定する場合は、トランスフォーメーションが実行する検証のタイプを選択できます。トランスフォーメーションでは、検証タイプをモードとして定義します。**[全般設定]** タブでモードを選択するか、トランスフォーメーションの詳細プロパティとしてモードを選択します。

ポートのテンプレートをトランスフォーメーションに作成できます。テンプレートは、1つ以上のポートグループからのポートのサブセットです。テンプレートを使用して、プロジェクトで使用するポートを編成できます。

ポートグループとポートの選択

アドレスバリデータトランスフォーメーションには、使用可能な入力ポートと出力ポートを含む定義済みのポートグループが含まれています。アドレスバリデータトランスフォーメーションを設定する場合、グループを参照して必要なポートを選択します。

アドレス入力データに対応する入力ポートを選択します。プロジェクトに必要なアドレスデータを含む出力ポートを選択します。

入力ポートと出力ポートはトランスフォーメーションに直接選択できます。また、入力ポートと出力ポートを含むデフォルトモデルを作成することもできます。ポートをトランスフォーメーションに直接選択すると、選択したポートはそのトランスフォーメーションにのみ適用されます。ポートをデフォルトモデルに追加すると、そのポートは、それ以降に作成するアドレスバリデータトランスフォーメーションに適用されます。

アドレスバリデータトランスフォーメーションで処理しないカラムについては、パススルーポートをトランスフォーメーションに追加できます。

アドレスバリデータトランスフォーメーションの入力ポートグループ

アドレスデータをトランスフォーメーションの入力ポートに接続する前に、入力グループを参照して、入力データの構造および内容に対応するポートを選択する必要があります。出力グループを参照して、データの要件を満たすポートを選択します。

アドレスバリデータトランスフォーメーションには、基本モデルと詳細モデルのポートグループが表示されます。ほとんどのアドレスは、基本モデルのポートグループを使用して定義できます。詳細モデルで利用可能なその他のポートは、アドレスが非常に複雑な場合に使用します。

注: 複数のポートを選択するときは、すべて同じ入力ポートグループから選択します。

以下の入力ポートグループがあります。

個別

番地、通り名、郵便番号など、単一のデータ要素に関する完全な情報を含むデータカラムを読み取るには、[個別] ポートを使用します。基本モデルと詳細モデルで [個別] グループを検索します。

混合

1 つ以上のデータ要素に関する情報を含むデータカラムを読み取るには、[混合] ポートを使用します。[混合] グループは、[個別] グループと [複数行] グループのそれぞれのポートを組み合わせたものです。郵便事業者に登録可能な住所レコードを作成する場合は、[混合] ポートを使用します。[混合] ポートは、郵便事業者の住所表記標準に合わせて住所を構成し、各行のデータタイプを識別します。基本モデルと詳細モデルで [混合] グループを検索します。

複数行

複数のデータ要素を含むデータカラムを読み取るには、[複数行] ポートを使用します。各入力カラムは、住所内の各行に対応します。最適な結果を得るために、郵便事業者が求める形式で入力データを定義してください。印刷可能な住所レコードのセットを作成するには、[複数行] ポートを選択します。

各 [複数行] ポートは、次のような、印刷可能な住所内の各行を表します。

123 Main Street, Apartment 2

[複数行] ポートは、各住所行に表示されるデータのタイプを指定しません。基本モデルと詳細モデルで [複数行] グループを検索します。

単一行

州レベルまでの住所要素をすべて含み、要素間に区切り記号が含まれない、単一のデータカラムを読み取るには、[単一行] ポートを使用します。住所要素を送信するには、このポートグループの [住所の正式表記] ポートを使用します。このポートグループには、住所の国情報を読み取るために使用できる [国] ポートも含まれています。基本モデルと詳細モデルで [単一行] グループを検索します。

アドレスバリデータトランスフォーメーションの出力ポートグループ

アドレスバリデータトランスフォーメーションを他のトランスフォーメーションまたはデータオブジェクトに接続する前に、必要なデータの種類と、出力アドレスの構造を決定します。

出力グループを参照して、データの要件を満たすポートを選択します。

注: ポートは複数の出力グループから選択することが可能で、共通の機能を持つポートを選択することができません。

以下の定義済みの出力グループがあります。

アドレス要素

住居番号、部屋番号、町名など、住所データの各要素を個別のポートに書き出します。基本モデルと詳細モデルでアドレス要素グループを検索します。

AT 補足

建物レベルの郵便番号データなど、郵便配達に役立つデータをオーストリアの住所に書き込みます。基本モデルで AT 補足グループを検索します。

AU 補足

オーストラリア統計局が住所に割り当てる地理的地域を識別するデータをオーストラリアの住所に書き込みます。基本モデルで AU 補足グループを検索します。

オーストラリア特有

オーストラリア郵便公社の Address Matching Approval System (AMAS) 基準に準拠した住所表記のデータをオーストラリアの住所に書き込みます。基本モデルと詳細モデルでオーストラリア特有グループを検索します。

BE 補足

郵便配達に役立つデータをベルギーの住所に書き込みます。このデータは、ベルギーの統計局によって定義されている、市区町村および地域の識別コードを含みます。基本モデルで BE 補足グループを検索します。

BR 補足

地理統計院 (IBGE) により提供された地域識別コードなど、郵便配達に役立つデータをブラジルの住所に書き込みます。基本モデルで BR 補足グループを検索します。

CAMEO

カスタマセグメント化分析で利用できる人口統計および所得の概要データを生成します。基本モデルで CAMEO グループを検索します。

カナダ特有

カナダ郵便公社の Software Evaluation and Recognition Program (SERP) 基準に準拠した住所表記のデータをカナダの住所に書き込みます。基本モデルでカナダ特有グループを検索します。

CH 補足

拡張郵便番号データなど、郵便配達に役立つデータをスイスの住所に書き込みます。基本モデルで CH 補足グループを検索します。

CZ 補足

拡張郵便番号データなど、郵便配達に役立つデータをチェコ共和国の住所に書き込みます。基本モデルで CZ 補足グループを検索します。

担当者要素

名前、敬称、役職名など、個人または担当者のデータを書き出します。詳細モデルで担当者要素グループを検索します。

国

国際標準化機構 (ISO) で定義されている国名や国コードを記述します。基本モデルと詳細モデルで国グループを検索します。

DE 補足

地方自治体や地域コードデータなど、郵便配達に役立つデータをドイツの住所に書き込みます。基本モデルで DE 補足グループを検索します。

ES 補足

郵便配達に役立つデータをスペインの住所に書き込みます。基本モデルで ES 補足グループを検索します。

フォーマットされたアドレス行

印刷用または郵送用にフォーマットされた住所を書き出します。基本モデルと詳細モデルでフォーマットされたアドレス行グループを検索します。

FR 補足

フランス国立統計経済研究所 (INSEE) により提供された識別コードなど、郵便配達に役立つデータをフランスの住所に書き込みます。基本モデルで FR 補足グループを検索します。

フランス特有

フランス郵政公社の National Address Management Service (SNA) 基準に準拠した住所表記のデータをフランスの住所に書き込みます。基本モデルでフランス特有グループを検索します。

ジオコーディング

住所のジオコードデータ（緯度座標と経度座標など）を生成します。基本モデルでジオコーディンググループを検索します。

ID 要素

レコード ID とトランザクションキーを書き出します。詳細モデルで ID 要素グループを検索します。

IT 補足

郵便配達に役立つデータをイタリアの住所に書き込みます。基本モデルで IT 補足グループを検索します。

JP 補足

町名字コードなど、郵便配達に役立つデータを日本の住所に書き込みます。基本モデルで日本補足グループを検索します。

KR 補足

指定された住所の現在のバージョンと古いバージョンを指定する一意の識別子など、郵便配達に役立つデータを韓国の住所に書き込みます。基本モデルで KR 補足グループを検索します。

最終行の要素

国内で使われる住所の最終行に表示するデータを書き出します。基本モデルと詳細モデルで最終行の要素グループを検索します。

ニュージーランド固有

ニュージーランド郵政公社の SendRight 基準に準拠した住所表記のデータをニュージーランドの住所に書き込みます。基本モデルでニュージーランド固有グループを検索します。

PL 補足

Territorial Division (TERYT) データなど郵便配達に役立つデータをポーランドの住所に書き込みます。基本モデルで PL 補足グループを検索します。

その他

トランスフォーメーションで解析できないデータ要素を他のポートに書き出します。基本モデルと詳細モデルでその他グループを検索します。

RS 補足

郵便番号サフィックスデータなど、郵便配達に役立つデータをセルビアの住所に書き込みます。基本モデルで RS 補足グループを検索します。

RU 補足

住所の Federal Information Addressing System ID など、郵便配達に役立つデータをロシアの住所に書き込みます。基本モデルで RU 補足グループを検索します。

ステータス情報

入力および出力された住所ごとに品質に関する詳細情報を出力します。基本モデルでステータス情報グループを検索します。

英国補足

配布ポイントデータや陸地測量データなど、郵便配達に役立つデータを英国の住所に書き込みます。基本モデルで UK 補足グループを検索します。

米国特有

米国郵政公社の Coding Accuracy Support System (CASS) 基準に準拠した住所表記のデータを米国の住所に書き込みます。基本モデルで米国特有グループを検索します。

米国補足

アメリカ合衆国の連邦情報処理標準(FIPS)コードのような地理的あるいは人口統計学のデータを記述します。基本モデルで米国補足グループを検索します。

XML

住所レコードデータを Address Verification ソフトウェアライブラリが定義する XML 構造で書き込みます。詳細モデルで XML グループを検索します。

ZA 補足

全国住所データベースデータなど、郵便配達に役立つデータを南アフリカの住所に書き込みます。基本モデルで ZA 補足グループを検索します。

複数インスタンスのポート

住所データタイプの多くは、住所内に複数回出現します。複数回出現するデータ要素が住所に含まれている場合には、複数インスタンスのポートを選択することができます。

複数インスタンスのポートには、最大 6 個のインスタンスを含めることができます。多くの住所は、含まれている各データ要素ごとに 1 個のポートインスタンスを使用します。一部の住所は、2 個目ポートインスタンスを使用します。ごく一部の住所は、複数のポートインスタンスを使用します。

多くの場合、ポートの最初のインスタンスは、ポートが識別するプライマリ名または最大の地域です。選択したすべてのポートについて、ポートのインスタンス間の関係を確認する必要があります。

[町名の正式表記] ポートの例

英国の住所レコードには、1 つの町がより広い町の一部になっている場合に 2 つの町名が含まれる場合があります。

以下の表には、2 つの [町名の正式表記] ポートを使用する住所が含まれます。

ポート	データ
番地の正式表記 1	1A
町名の正式表記 1	THE PHYGTLE
町名の正式表記 2	SOUTH STREET
市区町村名 1	NORFOLK
郵便番号 1	NR25 7QE

この例では、[町名の正式表記 1] ポートの町名データは、[町名の正式表記 2] ポートの町名データに依存しています。[番地の正式表記 1] ポートのデータは、[町名の正式表記 1] のデータに関連しています。

注: [町名の正式表記 1] はメールボックスの場所を指定していますが、[町名の正式表記 2] はより広範な町を表している可能性があります。

[担当者] ポートの例

住所レコードには、各担当者が同じ世帯に属している場合に複数の担当者を含めることができます。

以下の表には、2 つの [担当者名] ポートを使用する住所が含まれます。

ポート	データ
担当者名 1	MR. JOHN DOE
担当者名 2	MS. JANE DOE
フォーマットされたアドレス行 1	2 MCGRATH PLACE EAST
フォーマットされたアドレス行 2	ST. JOHN'S NL A1B 3V4
フォーマットされたアドレス行 3	CANADA

この例では、[担当者名 1] または [担当者名 2] に適用する優先順位は組織が指定できます。アドレスバリデータトランスフォーメーションは、担当者データの優先順位付けを行いません。

印刷出力用に住所の形式を整える場合、[フォーマットされたアドレス行] ポートの複数のインスタンスを使用することがあります。[フォーマットされたアドレス行] ポートは、最大 12 個選択できます。

アドレス検証プロジェクト

アドレスバリデータトランスフォーメーションは、さまざまなタイプのプロジェクトで使用できます。プロジェクトタイプごとに、各種ポートを使用して住所テンプレートを作成します。

アドレス検証プロジェクトは、以下の目的で定義することができます。

郵便事業者の標準に準拠するようにフォーマットされた住所の作成

郵便でのキャンペーン用に、大きな住所レコードセットを準備することができます。郵便事業者の希望するフォーマットで住所を作成すると、郵送料が大幅に下がります。郵送用の住所を準備する場合は、フォーマットされた住所の各行を単一ポートに書き出す出力ポートを選択します。担当者名、番地住所の行、および市区町村と郵便番号の行に、別々のポートを選択できます。

収入と生活様式をインジケータとする住所の整理

カスタマセグメント化データを居住地住所レコードに追加することができます。カスタマセグメント化データは、住所ごとに居住者の推定収入レベルと希望の生活様式を示しています。CAMEO 出力グループからポートを選択して、カスタマセグメント化データを住所レコードに追加します。カスタマセグメント化データは、複数のカスタマ市場をターゲットにするメールキャンペーンで使用することができます。

郵便事業者から認証される住所の作成

Australia Post（オーストラリア郵政公社）、Canada Post（カナダ郵政公社）、または USPS（米国郵政公社）向けのレコードセットを準備する際に、各住所の配達可能性を確認するデータを追加することができます。

アドレスバリデータトランスフォーメーションでは、各郵便事業者のデータ標準に対して住所レコードが完全かつ正確であることを証明するレポートを生成できます。

規制上の要件を満たす住所の作成

組織が保持している住所レコードが、業界または政府の規制に正確に従っていることを検証できます。住所データの各要素を別々のフィールドに書き込む出力ポートを選択します。さらに、出力データの正確性と完全性について詳細な情報を提供するアドレス検証ステータスポートを選択します。

部分的なアドレスの入力

部分的なアドレスを入力し、参照データ内の部分的なアドレスと一致する有効かつ完全なアドレスを取得できます。部分的なアドレスを入力するには、提案リストモードまたは対話モードで実行されるようにトランスフォーメーションを設定します。〔住所の正式表記〕ポートに入力アドレスを単一行として入力できます。

住所のデータ品質の向上

他のデータプロジェクトと並行して、住所データセットの構造および全般的なデータ品質を高めることができます。例えば、データセットに不要なカラムが含まれていたり、同じタイプのデータが複数のカラムに含まれていたりする場合があります。データセット内のカラム数を減らすと共に、さまざまなタイプのデータに使用するカラムを簡略化することができます。

フォーマットされたアドレスと郵便事業者の住所表記基準

DM によるキャンペーン用に住所レコードを準備する場合は、印刷する住所の構造を郵便事業者の住所表記基準に合わせる必要があります。

例えば、USPS の米国国内の住所表記は次のような形式になっています。

Line 1	Person/Contact Data	JOHN DOE
Line 2	Street Number, Street, Sub-Building	123 MAIN ST NW STE 12
Line 3	Locality, State, ZIP Code	ANYTOWN NY 12345

印刷する住所を、行ごとに1つのポートへ書き出すように設定することができます。その場合、それぞれの行のデータタイプに適したポートを使用できますが、各行のデータタイプに関係なく住所の構造を受け付けるポートを使用することもできます。

以下の表に、米国の住所を印刷するときの住所の2種類のフォーマット方法を示します。

印刷する住所	使用ポートの一例	使用ポートのもう一つの例
JOHN DOE	受取人行 1	フォーマットされたアドレス行 1
123 MAIN ST NW STE 12	送付先住所 1	フォーマットされたアドレス行 2
ANYTOWN NY 12345	国特有の最終行 1	フォーマットされたアドレス行 3

「フォーマットされたアドレス行」ポートを使用するのは、データセットにタイプの異なる住所（勤務先住所と自宅住所など）が混在している場合です。勤務先住所の場合、担当者と組織の情報で3行必要になることもあります。アドレスバリデータトランスフォーメーションは、必要に応じて「フォーマットされたアドレス行」ポートのみを使用して、勤務先住所も自宅住所も適切にフォーマットします。ただし、「フォーマットされたアドレス行」ポートは、ポートに含まれるデータタイプを指定しません。

「受取人行」、「送付先住所」、「国特有の最終行」などのポートを使用するのは、すべての住所が同じ形式でフォーマットされている場合です。これらのポートにより、住所のデータ要素がデータタイプ別に分類されるので、データセットを把握しやすくなります。

注: この例の住所は、他のポートを使用して処理することもできます。上記の例は、印刷と配達用に住所をフォーマットするポートの説明を目的としたものです。

人口統計データと地理的データ

DM によるキャンペーンでレコードセットを作成する場合、さまざまなデータタイプを追加して住所表記に挿入することができます。このようなデータを使用することで、郵便物の配達状況を人口統計的に、地理的に把握することができます。

例えば、米国内のある住所がどの下院選挙区に属しているのかを特定することができます。また、郵便物の仕向国の郵便システムで、アドレス参照データに地理座標が使われている場合は、緯度と経度の座標を生成することもできます。

部分的アドレスの入力

提案リストモードまたは対話モードを使用する場合は、不完全なアドレスを入力して、参照データから有効で完全なアドレスを取得できます。

アドレスが確実ではない場合に、有効な住所の候補リストを表示する場合は、提案リストモードを選択します。アドレスが確実な場合に、完全形式を確認するには、対話モードを選択します。いずれの場合も、アドレスバリデータトランスフォーメーションはアドレス参照データを検索し、入力データを含むすべての住所を返します。

提案リストモードまたは対話モードで実行されるようにトランスフォーメーションを設定する場合は、次のルールおよびガイドラインを考慮してください。

- 複数のポートで入力アドレスを定義することも、「住所の正式表記」入力ポートですべてのアドレス要素を入力することもできます。

- 提案リストモードでトランスフォーメーションを設定する場合は、[個別] 入力グループからポートを選択します。あるいは、[住所の正式表記] ポートを選択し、必要に応じて [複数行] グループで [国名] ポートを選択します。
- 提案リストモードおよび対話モードでは、入力アドレスごとに複数のアドレスが返されることがあります。[最大結果カウント] プロパティは、返されたアドレス数の上限を指定します。一致するアドレスの数が [最大結果カウント] 値より大きい場合、[カウントオーバーフロー] ポートは追加アドレスの数を返します。
- Informatica Address Verification では、提案リストモードは高速完了モードと呼ばれます。

アドレスバリデータのステータスポート

アドレスバリデータトランスフォーメーションは、入力ポートや出力ポートで読み書きするアドレス要素に関するステータス情報を書き込みます。[ステータス情報] ポートを使用してステータス情報を表示します。

以下のステータスポートを選択できます。

アドレス解決コード

アドレス内の無効なアドレス要素について説明します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

アドレスタイプ

郵便事業者が複数の形式の住所を認識する場合に使用するアドレスタイプを示します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

要素入力ステータス

入力アドレス要素と参照データ間の類似性を示します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

要素の関連性

郵便事業者が住所のメールボックスを識別するために必要なアドレス要素を識別します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

要素の結果ステータス

アドレス検証が入力アドレスに対して行う更新を示します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

拡張要素の結果ステータス

参照データにアドレスの追加データがあるかどうかを示します。ポートには、アドレス検証がアドレスに対して行う更新についての詳細情報を含めることができます。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

ジオコーディングのステータス

アドレス検証がアドレスに返したジオコーディングデータのタイプを説明します。[基本モデル] の [ジオコーディング] ポートグループからポートを選択します。

郵送可能スコア

郵便事業者が郵送項目をその住所に配達できる可能性を示します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

照合コード

入力された住所の住所検証および住所修正の結果を示します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

結果の割合

入力アドレスと対応する出力アドレスの類似性を割合で表します。[基本モデル] の [ステータス情報] ポートグループからポートを選択します。

要素のステータスコードの定義

[要素入力のステータス]、[要素の関連性]、[要素の結果ステータス] および [拡張要素の結果ステータス] ポートは、入力と出力のデータ要素の有効性に関するステータス情報を表します。アドレス検証操作の結果を確認するには、該当する要素のポートを選択します。

コードには以下の情報が含まれます。

- [要素入力のステータス] のコードは、入力された住所のデータとアドレス参照データとの比較で見つかった一致の品質を表します。
- [要素の関連性] のコードは、配達先の国の住所に必要なアドレス要素を識別します。
- [要素の結果ステータス] のコードは、入力データに対して処理中に加えられた変更に関する説明です。
- [拡張要素の結果ステータス] のコードは、アドレス参照データにアドレス要素に関する追加情報が含まれることを示します。

各ポートは 20 文字のコードを返し、コード内の文字は、それぞれが異なる住所データ要素を表します。要素ポート上の出力コードを読み取る場合、それぞれの文字が表す要素を把握しておく必要があります。この 20 文字は、2 文字 1 組のペア 10 個で構成されます。2 文字 1 組のコードは、ペアごとに 1 つの住所情報タイプを表します。例えば、各リターンコードの先頭の位置は、基本の郵便番号情報を表しています。

注: [アドレス解決コード] ポートは、[要素のステータス] ポートと同じアドレス要素に基づいて 20 桁の文字列を返します。

以下の表に、それぞれの位置の値が識別するアドレス要素を示します。

位置	アドレス要素	説明	アドレス要素の例
1	郵便番号レベル 0	5 桁の ZIP コードなど、基本的な郵便コード情報。	5 桁の ZIP コード 10118
2	郵便番号レベル 1	ZIP+4 コードの下 4 桁など、追加の郵便コード情報。	ZIP+4 コード 10118-0110 の「0110」
3	市区町村レベル 0	市や町など、プライマリの場所情報。	イギリスの「London」
4	市区町村レベル 1	市区町村に従属する集落などの名前。	ロンドンの「Islington」
5	都道府県レベル 0	その国におけるプライマリの地域名。米国、カナダ、スイスにおける州名など。	New York State
6	都道府県レベル 1	米国における郡名。	ニューヨーク州の「Queens County」
7	町名レベル 0	プライマリの町名情報。	South Great George's Street

位置	アドレス要素	説明	アドレス要素の例
8	町名レベル 1	町名および番地等に従属する情報。	South Great George's Street の「George's Arcade」
9	番地レベル 0	主な町名に関連付けられている建物番号または住居番号。	South Great George's Street の「460」
10	番地レベル 1	従属する町名および番地等に関連付けられている建物番号または住居番号。	George's Arcade の「81」
11	配達サービスレベル 0	私書箱の記述子および番号。	PO Box 111
12	配達サービスレベル 1	配達を担当する郵便局のコード	MAIN STN
13	建物レベル 0	建物の名前または番号。 住居番号は特定しません。	Alice Tully Hall
14	建物レベル 1	補足的な建物名または番号。	Alice Tully Hall の「Starr Theater」
15	棟レベル 0	アパートメント、スイート、フロアの名前または番号。	350 5th Avenue, Floor 80 の「80」
16	棟レベル 1	棟レベル 0 の情報と組み合わされている場合は、アパートメント、スイート、またはフロアの情報。	80-18 (80 がフロア番号で、18 がスイート番号)
17	組織レベル 0	会社名。	AddressDoctor(R) GmbH
18	組織レベル 1	親会社などの追加の企業情報。	Informatica Corporation
19	国レベル 0	国名。	United States of America
20	国レベル 1	自治領。	United States Virgin Islands

ポート名に数字のサフィックスが付く場合、レベル 0 はポート番号 1 上のデータを表し、レベル 1 は、ポート 2 から 6 上のデータを表します。

印刷された住所では、レベル 0 情報をレベル 1 の前または後ろのどちらにでも配置できます。例えば、郵便番号レベル 0 の後に郵便番号レベル 1 が続き、市区町村レベル 1 は市区町村レベル 0 の前に配置されます。

[アドレス解決コード] 出力ポートの値

[アドレス解決コード] は 20 文字から成る文字列であり、個々の文字は異なる入力アドレス要素を表します。文字の値は、アドレスの対応する位置にある無効なアドレス要素を表します。

以下の表で、[アドレス解決コード] ポートの値について説明します。

コード	説明
2	アドレス要素が配達に必要ですが、入力アドレスに含まれていません。アドレス参照データに、不足しているアドレス要素が含まれています。 出力コード 2 は、アドレス要素がないため配達先の住所として無効であることを示します。
3	アドレス要素は、アドレスの有効な範囲外の住居番号または番地です。例えば、指定された番地に存在しない住居番号がアドレス要素に含まれています。提案リストモードが、代替の住所を返します。
4	入力アドレスに要素のインスタンスが複数含まれるため、アドレス検証がアドレス要素を検証または訂正できません。
5	アドレス要素が現在のアドレスでは曖昧で、アドレス参照データに代替が含まれています。アドレス検証では入力要素が出力アドレスにコピーされます。 例えば、アドレス要素が有効な郵便番号で、アドレス内の有効な市区町村に一致しない場合です。
6	アドレス要素が住所内の別の要素と矛盾します。アドレス検証は、この住所の正しい要素を判別できません。出力アドレスは入力アドレスをコピーします。
7	住所に対して複数の変更を行わないとアドレス要素を訂正できません。アドレス検証は住所を訂正できますが、変更箇所の数から、住所が信頼できないものであることが示されます。
8	データは郵便事業者の検証ルールには従っていません。

[要素入力ステータス] 出力ポートの値

[要素入力ステータス] は 20 文字から成る文字列であり、個々の文字は異なる入力アドレス要素を表します。それぞれの文字の値は、アドレス要素に対して実行された処理のタイプを表します。

このポートはステータス情報ポートグループにあります。

以下の表で、バッチモード、認証済みモード、提案リストモードで [要素入力ステータス] から出力文字列内の各位置に返されるコードについて説明します。

コード	説明
0	入力アドレスは、現在の位置にデータを含みません。
1	参照データは、現在の位置にデータを含みません。
2	参照データがないので、データをチェックできません。

コード	説明
3	現在の位置のデータは正しくありません。参照データベースは、配達サービス値の数値が参照データが想定している範囲外にあることを示します。 バッチモードと認証モードでは、トランスフォーメーションは現在の位置の入力データを出力として修正せずに渡します。
4	現在の位置のデータは参照データと一致しますが、エラーが含まれます。
5	現在の位置のデータは参照データと一致しますが、トランスフォーメーションはデータを修正または標準化しました。
6	現在の位置のデータはエラーなしに参照データと一致します。

以下の表で、解析モード時に「要素入力ステータス」から出力文字列内の各位置に返されるコードについて説明します。

コード	説明
0	入力アドレスは、現在の位置にデータを含みません。
1	トランスフォーメーションは、現在の位置の要素を出力住所の別の位置に移動しました。
2	現在の位置の要素は参照データ値に一致しましたが、トランスフォーメーションは出力住所の要素を標準化しました。
3	現在の位置のデータは正しいです。

「要素の関連性」出力ポートの値

「要素の関連性」値はアドレス要素が郵便配達に必要なかどうかを示します。このポートはステータス情報ポートグループにあります。

「要素の関連性」値は 20 文字から成る文字列で、それぞれの文字が異なるタイプの住所データを表します。アドレス検証マッピングを実行したら、ポートの出力を確認して各アドレスに必要なアドレス要素を特定します。この結果を使用して、住所データの出力ポートとして正しいポートを選択したかを検証します。関連する住所データ要素の出力ポートを選択しなかった場合、その住所の出力は無効になります。

以下の表に、「要素の関連性」から出力文字列内の各位置に返されるコードを示します。

コード	説明
0	この住所への配達には関係ありません。
1	この住所への配達に関係があります。 各国の郵便事業者は、出力文字列内のこの位置にあるデータが提供されない場合、その住所に配達できません。

注: 「要素の関連性」の値は、「照合コード」の値がバッチモードの場合は Cx または Vx のアドレス、対話モードの場合は Cx、Vx、I3、または I4 のアドレスで使用できます。「要素入力ステータス」、「要素の結果ステータス」、「拡張要素の結果ステータス」、「アドレス解決コード」などの他の評価コードは、「照合コード」の値に関係なく値を返します。

[要素の結果ステータス] 出力ポートの値

[要素の結果ステータス] は 20 文字から成る文字列であり、個々の文字は異なる入力アドレス要素を表します。それぞれの文字の値は、アドレス要素に対して検証プロセスが行った更新を示します。

このポートはステータス情報ポートグループにあります。

以下の表で、[要素の結果ステータス] ポートの値について説明します。

コード	説明
0	出力アドレスは、現在の位置にデータを含みません。
1	トランスフォーメーションは、参照データの現在の位置にデータを見つけることができません。トランスフォーメーションは、入力データを出力データにコピーします。
2	現在の位置のデータは、チェックされずに標準化されます。
3	現在の位置のデータはチェックされますが、参照データに一致しません。参照データは、数字データが有効範囲外にあることを示しています。トランスフォーメーションは、入力データを出力ポートにコピーします。 バッチモードで適用します。
4	参照データがないので、トランスフォーメーションは入力データを出力データにコピーします。
5	現在の位置にあるデータは検証済みですが、参照データに複数の一致が見つかったので変更されません。 バッチモードで適用します。
6	データ検証により、現在の位置の入力値が削除されました。
7	現在の位置のデータは検証済みですが、入力データにスペルエラーが含まれます。検証により、参照データの値を使用してエラーが修正されました。
8	現在の位置のデータは検証済みで、参照データの値を使用して更新されています。 値 8 は、参照データベースに入力要素の追加データが含まれることも意味する場合があります。例えば、検証によって町名または建物名に完全に一致する候補が見つかった場合は建物番号または棟番号を追加できます。
9	現在の位置のデータは検証済みですが変更されておらず、配達ステータスは不明です。例えば、DPV 値が間違っている場合などです。
C	現在の位置のデータは検証済みで確認済みですが、名前データが期限切れです。検証により名前データが変更されました。
D	現在の位置のデータは検証済みで確認済みですが、外名から正式名に変更されています。

コード	説明
E	現在の位置のデータは検証済みで確認済みです。ただし、アドレス検証によって大文字/小文字や言語が標準化されています。 アドレスの検証により、値が完全に代替の言語に一致する場合、言語を変更できます。たとえば、アドレスの検証で、「Brussels」がベルギーのアドレスの「Bruxelles」に変更されます。
F	現在の位置のデータは検証済みかつ確認済みで、参照データと完全に一致しているので変更されていません。

出力文字列の 19 および 20 の位置は、国に関するデータです。

次の表に、19 および 20 の位置のデータに対する検証で返される値を示します。

コード	説明
0	出力アドレスは、現在の位置にデータを含みません。
1	アドレス検証は国データを認識しません。
4	アドレス検証では、アドレスバリデータトランスフォーメーションのデフォルトの国の値によって国を識別します。
5	参照データに複数の一致が含まれるので、アドレス検証は国を識別できません。
6	アドレス検証は、スクリプトから国を識別します。
7	アドレス検証は、住所形式から国を識別します。
8	アドレス検証は、主要都市のデータから国を識別します。
9	アドレス検証は、地方データから国を識別します。
C	アドレス検証は、地域データから国を識別します。
D	アドレス検証は、国名から国を識別しますが、名前にエラーが含まれます。
E	アドレス検証は、ISO コードまたは国名などのアドレスデータから国を識別します。
F	アドレス検証は、アドレスバリデータトランスフォーメーションで設定される [国を強制的に適用] の値によって国を識別します。

〔拡張要素の結果ステータス〕 出力ポートの値

〔拡張要素の結果ステータス〕は 20 文字から成る文字列であり、個々の文字は異なる入力アドレス要素を表します。このポートの出力コードは〔要素入力のステータス〕ポートおよび〔要素の結果ステータス〕ポートのステータスデータを補完するものです。このポートの出力コードは、参照データ内のアドレス要素について追加情報の有無を示すこともできます。

このポートはステータス情報ポートグループにあります。

以下の表で、[拡張要素の結果ステータス] ポートの値について説明します。

コード	説明
1	アドレス参照データは、アドレス要素に関する追加情報を含みます。アドレス検証は追加情報を必要としません。
2	アドレス検証はデータエラーまたは形式エラーを解消するためにアドレス要素を更新しました。アドレス検証はアドレス要素を検証しませんでした。
3	アドレス検証はデータエラーまたは形式エラーを解消するためにアドレス要素を更新しました。アドレス検証はアドレス要素の数値データを検証しました。
4	アドレス検証は形式エラーを解消するためにアドレス要素を別のフィールドに移動しました。
5	アドレス参照データは、優先される市区町村名など、アドレス要素の代替バージョンを含みます。
6	アドレス検証は、アドレス要素のすべての部分を検証しませんでした。アドレス要素に、アドレス検証が検証できないデータが含まれています。
7	アドレス検証は、有効なアドレス要素を住所の間違った位置で検出しました。アドレス検証はアドレス要素を正しい位置に移動しました。
8	アドレス検証は間違ったデータフィールドで有効なアドレス要素を検出しました。アドレス検証はアドレス要素を正しいフィールドに移動しました。
9	アドレス検証は郵便事業者の検証ルールに従って、出力要素を生成しました。
A	アドレス検証は現在の位置に適した、別のアドレスタイプのアドレス要素を検出しました。アドレス検証は宛先の国の郵便事業者のルールに適合する出力アドレス要素を選択しました。
B	アドレス検証は、要素の関連性を特定できません。アドレス検証は住所が指定する国のデフォルト値を返します。
C	提案リストモード。アドレス検証はアドレス要素の追加の候補を返すことができます。追加の候補を返すには、アドレスバリデータトランスフォーメーションの[最大結果カウント] プロパティを更新します。
D	アドレス検証はアドレス要素に数値データを追加しました。
E	アドレス検証はアドレス要素を優先される言語で返すことができません。アドレス検証は要素をデフォルトの言語で返します。
F	アドレスコードルックアップモード。入力アドレスが期限切れです。

[郵送可能スコア] の出力ポートの値

[郵送可能スコア] の値は、出力住所の配達可能性を概算したものです。郵送可能スコアは、住所の配達可能性の一般的指標として使用します。このポートはステータス情報ポートグループにあります。

アドレスバリデータトランスフォーメーションでは、郵送可能スコアを計算する際に多数の因子が考慮されます。このトランスフォーメーションでの計算は原則的に、住所については[照合コード] の値および[要素の結果ステータス] の値に基づきます。この他に郵送可能スコアに影響する因子として、国については住所値の郵便での関連性および参照データの粒度があります。

〔郵送可能スコア〕のポート値は、住所の配達可能性を概算したものです。このスコアは、住所の配達可能性の指標として正確なものでも決定的なものでもありません。

以下の表に、〔郵送可能スコア〕の出力コードを示します。

値	サマリ	説明
5	確実	アドレス検証で入力住所のすべての関連要素をチェック済みで検証済みであることを示します。
4	ほぼ確実	次のいずれかのシナリオであることを示します。 <ul style="list-style-type: none">- 参照データが不完全であるため、1つ以上の関連住所要素をチェックできません。その他の住所要素は検証済みです。- アドレス検証で、確実度が非常に高い1つ以上の関連要素を修正しました。これは、アドレス検証で入力住所と参照データの間で一致が1件見つかり、バリエーションの度合いが非常に低いときに発生します。
3	問題なし	アドレス検証で入力住所の1つ以上の関連要素を修正したことを示します。アドレス検証で入力住所と参照データの間で一致が1件見つかり、バリエーションの度合いが許容可能です。
2	見込みあり	アドレス検証で次のいずれかの理由により住所を修正または検証できないことを示します。 <ul style="list-style-type: none">- アドレス検証で、参照データ内で十分な確実度のある一致候補を識別できません。- アドレス検証で、同程度の確実度のある一致候補が複数見つかりました。 郵便事業者はこの住所に配達できる可能性があります。
1	危険	アドレス検証で入力住所に対して参照データの部分一致が見つかったことを示します。
0	配達不能	アドレス検証で参照データ内に住所の一致が見つからなかったことを示します。入力住所に非常に多くの要素が不足しているか、またはアドレス検証で住所の要素の大半を検証できませんでした。

〔照合コード〕の出力ポートの値

〔照合コード〕の値は、入力された住所と参照データとを比較した結果を要約したものです。このコードは、トランスフォーメーションにより住所に対して何らかの修正が行われた場合にそれを要約したのもでもあります。このポートはステータス情報ポートグループにあります。

以下の表に、〔照合コード〕の出力ポートの値を示します。

コード	説明
A1	アドレスコードルックアップにより、入力コード内に部分的なアドレスまたは完全なアドレスが見つかりました。
A0	アドレスコードにより、入力コード内にアドレスが見つかりませんでした。
C4	修正済み。郵送に関連する要素すべてが確認されています。
C3	修正済み。いくつかの要素が確認できません。
C2	修正済みですが、参照データが存在しないため、配達ステータスが不明です。

コード	説明
C1	修正済みですが、ユーザーの標準化によって誤りが生じたため配達ステータスは不明です。
I4	データを完全に修正できませんが、参照データに1つの一致があります。
I3	データを完全に修正できませんが、参照データに複数の一致があります。
I2	データを修正できません。バッチモードで部分的に提案されたアドレスが返されます。
I1	データを修正できません。バッチモードでアドレスを提案できません。
N7	検証エラーです。単一行の検証がロック解除されていないため、検証されませんでした。
N6	検証エラーです。宛先の国で単一行の検証がサポートされていないため、検証されませんでした。
N5	検証エラーです。参照データベースが古すぎるため、検証を行えませんでした。
N4	検証エラーです。参照データが破損しているか不正なフォーマットのため、検証を行えませんでした。
N3	検証エラーです。国データのロック解除ができないため、検証を行えませんでした。
N2	検証エラーです。要求された参照データベースが利用できないため、検証を行えませんでした。
N1	検証エラーです。国が認識できなかったかサポートされていないため、検証を行えませんでした。
Q3	提案リストモード。アドレス検証では、入力アドレスに対応するアドレス参照データから完全なアドレスを1つ以上取得できます。
Q2	提案リストモード。アドレス検証では、入力アドレスの要素とアドレス参照データ内の要素を組み合わせ、完全なアドレスを作成できます。
Q1	提案リストモード。アドレスの検証は、完全なアドレスを提案できません。完全なアドレス提案を生成するには、入力アドレスにデータを追加します。
Q0	提案リストモード。提案を生成するための十分な入力データがありません。
RB	略称によって国が認識されました。ISOの2文字とISOの3文字の国コードを認識します。ドイツを表す「GER」のような一般的な略称も認識できます。
RA	トランスフォーメーション内の国を強制的に適用設定によって国が認識されました。
R9	トランスフォーメーション内のデフォルトの国設定によって国が認識されました。
R8	国名によって国が認識されました。

コード	説明
R7	国名によって国が認識されましたが、トランスフォーメーションによって国データ内にエラーが識別されました。
R6	地域データによって国が認識されました。
R5	都道府県のデータによって国が認識されました。
R4	主要な都市データによって国が認識されました。
R3	住所フォーマットによって国が認識されました。
R2	スクリプトから国が認識されました。
R1	使用できる一致が複数あるため、国が認識されませんでした。
R0	国が認識されません。
S4	解析モード。住所は完全に解析されました。
S3	解析モード。住所は解析され、複数の結果が返されました。
S1	解析モード。入力フォーマットの不一致のため解析エラーが発生しました。
V4	検証済み。入力されたデータは正確です。アドレス検証が郵送に関連する要素すべてを確認した結果、入力は完全に一致しました。
V3	検証済み。入力データは正しいですが、一部あるいは全部の要素が標準化されていたか、入力データに古い名前あるいは外名が含まれています。
V2	検証済み。入力データは正確ですが、参照データが完全ではないため、一部の要素を検証できません。
V1	検証済み。入力されたデータは正確ですが、ユーザーによる標準化が配達可能性を低下させています。たとえば、ポストコードの長さが短すぎます。

[ジオコーディングのステータス] の出力ポートの値

以下の表に、[ジオコーディングのステータス] の出力ポートの値を示します。ジオコーディングポートグループのポートを探します。

このポートは、入力アドレスの国に対応するジオコーディング参照データをインストールしている場合に選択します。

値	説明
EGC0	この住所に使用できるジオコードがないので、ジオコードを入力住所に追加できません。
EGC1-3	将来の使用のために予約済み。
EGC4	ジオコードは郵便番号レベルに対して部分的に正確です。
EGC5	ジオコードは郵便番号レベルに対して正確です。

値	説明
EGC6	ジオコードは市区町村レベルに対して正確です。
EGC7	ジオコードは番地レベルに対して正確です。
EGC8	ジオコードは住居番号レベルに対して正確です。ジオコードは、メールボックスがある通りの側に住居番号の場所を推定してオフセットを含めます。
EGC9	ジオコードは到着点またはルーフトップに対して正確です。
EGCA	ジオコードは区画の中心に対して正確です。
EGCC	ジオコードデータベースが破損しています。
EGCN	ジオコードデータベースが見つかりません。
EGCU	ジオコードデータベースがロック解除されていません。

注: Informatica では、区画の中心およびルーフトップのジオコーディング用に参照データを発行しなくなりました。

アドレスバリデータトランスフォーメーションの全般設定

アドレス検証に必要なパラメータを設定するための全般設定を設定します。

【全般設定】 ビューでは、次のプロパティを設定できます。

デフォルトの国

入力アドレスの宛先の国が特定できない場合にトランスフォーメーションが使用するアドレス参照データセットを指定します。データに国の情報が含まれている場合は「なし」を選択します。

またトランスフォーメーションで、詳細プロパティとしてデフォルトの国を設定することもできます。

実施国

オプションのプロパティ。入力アドレスの国名または略式表記を、デフォルトの国名または略式表記に置き換えます。入力アドレスで国が特定されていない場合は、トランスフォーメーションによってデフォルトの国データがアドレスに付加されます。

行セパレータ

1 行のアドレス内のデータフィールドを区切る区切り文字記号を指定します。

またトランスフォーメーションで、詳細プロパティとして行セパレータを指定することもできます。

大文字小文字表記

出力データの大文字小文字の表記を設定します。先頭の文字を大文字にする場合は、「Mixed（混在）」を選択して、アドレス参照データの標準に従います。アドレス参照データが使用する大文字小文字表記でアドレスを記述する場合は、「維持」オプションを選択します。

またトランスフォーメーションで、詳細プロパティとして大文字小文字表記を設定することもできます。

モード

トランスフォーメーションで実行する検証のタイプを決定します。

次のいずれかのオプションを選択します。

モードの種類	説明
アドレスコードのルックアップ	入力としてアドレスコードを提供すると、参照データから部分的なアドレスまたは完全なアドレスを返します。いくつかの国は、アドレスの番地、建物、または一意のメールボックスを表すアドレスコードをサポートします。
バッチ	データセットのレコードにアドレス検証を実行します。バッチ検証では、アドレスの完全性と配達可能性に焦点を合わせます。バッチモードは、品質が低いアドレスの提案を返しません。バッチはデフォルトのモードです。
認証	データセットのレコードにアドレス検証を実行します。認証基準では、各アドレスが一意のメールボックスを識別する必要があります。認証済みのアドレス検証は、オーストラリア、フランス、ニュージーランド、イギリス、および米国で実行できます。
国認識	郵便アドレスの宛先の国を特定します。トランスフォーメーションは、国認識モードでアドレス検証を実行しません。
対話型	不完全な有効アドレスを完成します。不完全な入力アドレスが参照データの複数のアドレスに一致すると、トランスフォーメーションは「最大結果カウント」に指定されている制限に到達するまで、すべての有効なアドレスを返します。
解析	アドレスのフィールドにデータを解析します。トランスフォーメーションは、解析モードでアドレス検証を実行しません。
提案リスト	入力アドレスに断片的な情報が含まれる場合に、参照データから有効なアドレスのリストを返します。アドレスの断片が参照データの複数のアドレスに一致すると、トランスフォーメーションは「最大結果カウント」に指定されている制限に到達するまで、すべての有効なアドレスを返します。

またトランスフォーメーションで、詳細プロパティとしてモードを設定することもできます。

[設定] ウィンドウのアドレス検証のプロパティ

アドレス検証エンジンのプロパティと、エンジンが Developer tool で読み取るアドレス参照データファイルを表示できます。Developer tool は、データ統合サービスがアドレス検証マッピングを実行するために使用するエンジンのプロパティを表示します。Developer tool は、アドレス検証操作を制御するコンテンツ管理サービスのプロパティを表示します。

Developer tool の **[環境設定]** ウィンドウを使用して、プロパティを確認します。**[設定]** ウィンドウの **[コンテンツステータス]** オプションを選択して、現在のデータ統合サービスが使用するコンテンツ管理サービスを特定します。プロパティを表示するには、ローカルコンテンツ管理サービスを選択します。

次のプロパティを表示できます。

アドレス検証データ

アドレス検証データのプロパティには、現在のコンテンツ管理サービスがデータ統合サービスに提供できる参照データの種類の種類が表示されます。このプロパティは、参照データが適用される国も示します。

アドレス検証エンジン

アドレス検証エンジンのプロパティには、現在のエンジンバージョン、認証コンポーネントがもっとも最近更新されたエンジン、およびデータの事前ロード方法が含まれます。

アドレス検証ライセンス

アドレス検証ライセンスのプロパティには、現在のコンテンツ管理サービスがデータ統合サービスに提供できる参照データのライセンス情報が含まれます。

アドレス検証データのプロパティ

アドレス検証データのプロパティには、現在のコンテンツ管理サービスがデータ統合サービスに提供できる参照データの種類が表示されます。プロパティには、参照データが適用される国も含まれます。

次の表は、**【コンテンツステータス】** ビューでコンテンツ管理サービスを選択したときに表示されるデータプロパティを示しています。

プロパティ	説明
国の ISO	アドレス参照データファイルが適用される国。このプロパティには、国名を表す ISO の 3 文字のコードが表示されます。
有効期限	現在のファイルの有効期限。Informatica はこの日に新しいファイルをリリースします。有効期限後もアドレス参照データファイルは使用できますが、ファイル内のデータは正確でない可能性があります。
国のタイプ	データに対して実行可能なアドレス検証のタイプ。 処理のタイプは 【全般設定】 タブの 【モード】 オプションで選択します。選択したモードがドメイン上のアドレスデータファイルと対応しない場合、アドレス検証マッピングは失敗します。
有効期限のロック解除	ライセンスの有効期限。有効期限のロック解除後はいずれのバージョンのファイルも使用できません。 [アドレス検証ライセンスのプロパティ] ビューの [有効期限のロック解除] プロパティと [有効期限] プロパティは、同じ情報を表します。
開始日のロック解除	[国の種類] プロパティのモードと、[国の ISO] プロパティの国に対してライセンスが有効になる日付。開始日のロックを解除前は、いずれのバージョンのファイルも使用することはできません。

アドレス検証ライセンスのプロパティ

アドレス検証ライセンスのプロパティには、現在のコンテンツ管理サービスがデータ統合サービスに提供できる参照データのライセンス情報が含まれます。

次の表は、**【コンテンツステータス】** ビューでコンテンツ管理サービスを選択したときに表示されるライセンスプロパティを示しています。

プロパティ	説明
コードのロック解除	[コードタイプ] プロパティに指定されたモードの参照データのロックを解除するライセンスコード。コードの最初の 4 文字が表示され、残りの文字はマスクされます。
コードタイプ	ライセンスに指定されたデータに対して実行可能なアドレス検証のモード。Informatica は、各モードに 1 つのライセンスコードを発行します。ライセンスコードは、1 つ以上の国に適用できます。 処理のタイプは 【全般設定】 タブの 【モード】 オプションで選択します。選択したモードがドメイン上のアドレスデータファイルと対応しない場合、アドレス検証マッピングは失敗します。
国リスト	ロック解除コードでロックが解除される参照データの国。 [国リスト] プロパティには、国ごとに ISO の 3 文字コードが 1 つ以上含まれています。
ステータス	ライセンスコードのステータス。ライセンスファイルが有効な場合、このプロパティは OK を返します。
有効期限	ライセンスの有効期限。 [アドレス検証データのプロパティ] ビューの [有効期限] プロパティと [有効期限のロック解除] プロパティは、同じ情報を表します。

アドレス検証エンジンのプロパティ

アドレス検証エンジンのプロパティには、現在のエンジンバージョン、認証コンポーネントがもっとも最近更新されたエンジン、およびデータの事前ロード方法が含まれます。

次の表は、**【コンテンツステータス】** ビューでコンテンツ管理サービスを選択したときに表示されるエンジンのプロパティを示しています。

プロパティ	値
エンジンバージョン	データ統合サービスが実行するアドレス検証エンジンのバージョン。
CASS バージョン	Informatica がもっとも最近 CASS 認証コンポーネントを更新したアドレス検証エンジンのバージョン。このプロパティを使用して、CASS 認証レポートのエンジンバージョンを特定します。 このプロパティには、エンジンがサポートする CASS 認証サイクルも含まれています。たとえば、エンジンが認定サイクル N をサポートしている場合があります。
AMAS バージョン	Informatica がもっとも最近 AMAS 認証コンポーネントを更新したアドレス検証エンジンのバージョン。このプロパティを使用して、AMAS 認証レポートのエンジンバージョンを特定します。

プロパティ	値
SendRight バージョン	Informatica がもっとも最近 SendRight 認証コンポーネントを更新したアドレス検証エンジンのバージョン。このプロパティを使用して、SendRight 認証レポートのエンジンバージョンを特定します。
SERP バージョン	Informatica がもっとも最近 SERP 認証コンポーネントを更新したアドレス検証エンジンのバージョン。このプロパティを使用して、SERP 認証レポートのエンジンバージョンを特定します。
SNA バージョン	Informatica がもっとも最近 SNA 認証コンポーネントを更新したアドレス検証エンジンのバージョン。このプロパティを使用して、SNA 認証レポートのエンジンバージョンを特定します。
事前にロードする方法	参照データベースをメモリに事前ロードするためにデータ統合サービスが使用する手法。コンテンツ管理サービスのプロパティは、データ統合サービスが事前ロードする参照データの国を指定します。指定可能な値は MAP と LOAD です。デフォルト値は MAP です。 MAP と LOAD どちらの方法も、メモリのブロックを割り当て、参照データをそのブロックに読み込みます。ただし、MAP では複数のプロセス間で参照データを共有できます。
キャッシュサイズ	事前ロードしない参照データに対してデータ統合サービスで使用するデータキャッシュのサイズ。指定可能な値は、NONE、SMALL、および LARGE です。デフォルト値は LARGE です。
最大メモリ使用量	アドレス検証エンジンが割り当てることができるメモリ (MB)。デフォルト値は 4096 です。
最大アドレスオブジェクト数	データ統合サービスが同時に実行できるアドレス検証インスタンスの最大数。デフォルト値は 3 です。
最大スレッド数	アドレス検証で使用できる最大スレッド数。デフォルト値は 2 です。
最大結果数	提案リストモードでマッピングを実行する場合に、アドレス検証が返すことができるアドレスの最大数。デフォルトは 20 です。このプロパティの上限は 100 です。
現在の日付	現在の日付。Developer tool は、現在の日付に適用されるプロパティ値を返します。
XML BOM の書き込み	データ統合サービスが GetConfig.xml ファイルにバイトオーダーマークを書き込むかどうかを指定します。指定可能な値は、ALWAYS、IF_NECESSARY、および NEVER です。デフォルト値は IF_NECESSARY です。
XML エンコード	アドレス検証エンジンがデータの読み取りおよび書き込みに使用する XML エンコードです。

アドレス検証の詳細プロパティ

データ統合サービスでのアドレスバリデータトランスフォーメーションのデータの処理方法を指定するには、詳細プロパティを設定します。

エイリアスの市区町村

アドレス検証で有効な市区町村のエイリアスを公式な市区町村名に置き換えるかどうかを決定します。

市区町村のエイリアスは、米国郵政公社が配達可能なアドレスの要素として認識する、市区町村の代替名です。このプロパティは、アドレスバリ Data Transformation を設定して米国のアドレスレコードを認証モードで検証する場合に使用できます。

以下の表に、エイリアスの市区町村のオプションを示します。

オプション	説明
オフ	[エイリアスの市区町村] プロパティを無効にします。
公式	任意の代替市区町村名または市区町村のエイリアスを公式な市区町村名に置き換えます。デフォルトのオプションです。
維持	有効な市区町村の代替名または市区町村のエイリアスを維持します。入力した市区町村名が有効でない場合、アドレス検証はその名前を公式名に置き換えます。

エイリアス番地

アドレス検証で通りのエイリアスを通りの公式名に置き換えるかどうかを決定します。

通りのエイリアスは、米国郵政公社が配達可能なアドレスの要素として認識する、通りの代替名です。このプロパティは、アドレスバリ Data Transformation を設定して米国のアドレスレコードを認証モードで検証する場合に使用できます。

以下の表に、エイリアス番地のオプションを示します。

オプション	説明
オフ	プロパティを適用しません。
公式	通りの代替名または通りのエイリアスを通りの公式名に置き換えます。デフォルトのオプションです。
維持	有効な通りの代替名または通りのエイリアスを維持します。入力した通りの名前が有効でない場合、アドレス検証はその名前を公式名に置き換えます。

大文字小文字表記

トランスフォーメーションによって出力アドレスデータに適用される大文字小文字の表記を指定します。

以下の表に、大文字小文字表記のオプションを示します。

オプション	説明
パラメータの割り当て	定義したパラメータを使用して、大文字小文字表記を設定します。
小文字	出力住所を小文字で書き込みます。

オプション	説明
混在	可能な場合、宛先の国で使用されている大文字小文字表記を使用します。
維持	住所参照データで使用されている大文字小文字表記を適用します。デフォルトのオプションです。
変更なし	大文字小文字表記は住所に適用されません。 注: 【変更なし】 オプションは、出力住所が入力住所の大文字小文字に一致することを保証しません。アドレス検証によりアドレス要素が参照データの要素で置き換えられた場合、アドレス要素は参照データに使用される大文字小文字に従います。
大文字	出力住所を大文字で書き込みます。

大文字小文字表記は **【全般設定】** タブで設定することもできます。

パラメータの使用

以下のいずれかのパラメータを使用して、大文字小文字表記を指定できます。

- LOWER. 出力住所を小文字で書き込みます。
- MIXED. 可能な場合、宛先の国で使用されている大文字小文字表記を使用します。
- NATIVE. 住所参照データで使用されている大文字小文字表記を適用します。デフォルトのオプションです。
【維持】 オプションに一致します。
- NOCHANGE. 大文字小文字表記は住所に適用されません。
- UPPER. 出力住所を大文字で書き込みます。

パラメータ値を大文字で入力します。

国

アドレスレコードが郵送される国を識別します。

リストから国を選択します。このプロパティは、デフォルトでは空になっています。

国のタイプ

完全なアドレスまたはフォーマットされたアドレス行のポート出力データの国名または略語の形式を決定します。トランスフォーメーションは、選択した国の標準の形式で国名や略語を表記します。

以下の表に、国のタイプのオプションを示します。

オプション	国
ISO2	ISO の 2 文字の国コード
ISO3	ISO の 3 文字の国コード
ISO #	ISO の 3 桁の国コード
略語	(将来の使用のために予約されています)

オプション	国
CN	カナダ
DA	(将来の使用のために予約されています)
DE	ドイツ
EN	英国 (デフォルト)
ES	スペイン
FI	フィンランド
FR	フランス
GR	ギリシア
IT	イタリア
JP	日本
HU	ハンガリー
KR	韓国
NL	オランダ
PL	ポーランド
PT	ポルトガル
RU	ロシア
SA	サウジアラビア
SE	スウェーデン

デフォルトの国

アドレスレコードで宛先の国が識別されないときにトランスフォーメーションで使用するアドレス参照データセットを指定します。

リストから国を選択します。住所レコードに国情報が含まれる場合は、デフォルトオプションを使用します。デフォルトは「なし」です。

デフォルトの国は、**【全般設定】** タブで設定することもできます。

パラメータの使用

パラメータを使用してデフォルトの国を指定することができます。パラメータを作成するときは、国の ISO 3166-1 alpha-3 コードをパラメータ値として入力します。パラメータ値は大文字で入力します。例えば、すべての住所レコードに国情報が含まれる場合は、NONE を入力します。

住所重複時の優先順位

検証するアドレスの種類を決定します。入力アドレスレコードに複数の種類の有効なアドレスデータが含まれるときにこのプロパティを設定します。

例えば、アドレスレコードに私書箱要素と番地要素の両方が含まれるときにこのプロパティを使用します。アドレス検証で、指定したアドレスデータの種類の含むデータ要素が読み込まれます。アドレス検証では、アドレスに含まれる互換性のないデータが無視されます。

次の表は、[住所重複時の優先順位] プロパティのオプションについて説明したものです。

オプション	説明
配送サービス	私書箱要素など、アドレスに含まれる配送サービスデータ要素を検証します。
郵政	ローカルの郵政事業者が必要とするアドレス要素を検証します。デフォルトのオプションです。
番地	住居番号や通りの名前など、アドレスに含まれる番地のデータ要素を検証します。

要素の略式表記

トランスフォーメーションでアドレス要素の略式表記を返すかどうかを決定します。アドレス参照データに略式表記が含まれる場合は、略式表記を返すようにトランスフォーメーションを設定できます。

例えば、United States Postal Service（米国郵政公社）は、多くの通り名や市区町村名の長い形式と短い形式を保持しています。HUNTSVILLE BROWNSFERRY RD の短い形式は HSV BROWNS FRY RD です。通り名や市区町村名が、米国郵政公社の定めるフィールド長の上限を上回る場合に、[要素の略式表記] プロパティを選択できます。

このオプションはデフォルトで選択されていません。略式表記のアドレス値を返すには、このプロパティを ON に設定します。トランスフォーメーションをバッチモードで使用すると、このプロパティは略式表記の市区町村名と市区町村コードを返します。トランスフォーメーションを認証モードで使用すると、このプロパティは略式表記の通り名、市区町村名、市区町村コードを返します。

実行インスタンス

データ統合サービスが、現在のトランスフォーメーションで実行時に作成するスレッドの数を指定します。データ統合サービスは、トランスフォーメーションを含むマッピングの [最大並行処理] ランタイムプロパティをオーバーライドする場合に、この実行インスタンス数の値を考慮します。デフォルトの実行インスタンス数の値は 1 です。

データ統合サービスでは、複数の要因を考慮して、トランスフォーメーションに割り当てるスレッド数を決定します。主要な要因は、実行インスタンス数の値と、マッピングおよびドメイン内の関連アプリケーションサービスの各値です。

データ統合サービスは、トランスフォーメーションに使用するスレッド数を計算する際に次の値を読み取ります。

- データ統合サービスの [最大並行処理] 値。デフォルトは 1 です。
- マッピングレベルで設定された [最大並行処理] 値。デフォルトは [自動] です。
- トランスフォーメーションの [実行インスタンス] 値。デフォルトは 1 です。

マッピングレベルの [最大並行処理] 値をオーバーライドする場合、データ統合サービスは、上記の各プロパティの最低値を使用してスレッド数を決定します。

マッピングレベルでデフォルトの「最大並行処理」値を使用している場合、データ統合サービスは実行インスタンス数の値を無視します。

データ統合サービスは、作成するスレッドの数を計算する際に、コンテンツ管理サービスの「最大アドレスオブジェクト数」プロパティも考慮します。「最大アドレスオブジェクト数」プロパティは、マッピング内で同時に実行できるアドレス検証インスタンスの最大数を決定します。「最大アドレスオブジェクト数」プロパティの値は、データ統合サービスの「最大並行処理」の値以上である必要があります。

「実行インスタンス」プロパティのルールおよびガイドライン

実行インスタンスの数を設定する際は、以下のルールとガイドラインを考慮します。

- 複数のユーザーがデータ統合サービスに同時にマッピングを実行することがあります。正しいスレッド数を計算するには、データ統合サービスがアクセスできる CPU 数を同時実行マッピング数で除算します。
- PowerCenter では、*AD50.cfg* 構成ファイルで、マッピング内で同時に実行できるアドレス検証インスタンスの最大数を指定します。
- デフォルトの「実行インスタンス」値およびデフォルトの「最大並行処理」値を使用する場合、トランスフォーメーション操作はパーティション化できません。
- 1 より大きい実行インスタンス値を設定する場合は、アドレスバリデータトランスフォーメーションをパッシブトランスフォーメーションからアクティブトランスフォーメーションに変更します。

フレキシブル範囲拡大

「拡大する範囲」プロパティを設定したときにアドレスバリデータトランスフォーメーションが返すアドレスの数に実際の制限を与えます。提案リストモードで実行するようにトランスフォーメーションを設定するときに「拡大する範囲」プロパティと「フレキシブル範囲拡大」プロパティを設定できます。

「拡大する範囲」プロパティにより、入力アドレスに住居番号データが含まれないときにトランスフォーメーションがアドレス提案を返す方法が定義されます。入力アドレスに、完全な郵便番号など、コンテキストデータが含まれない場合、「拡大する範囲」プロパティにより非常に似ているアドレスを大量に生成できます。「フレキシブル範囲拡大」プロパティは、「拡大する範囲」プロパティが1つのアドレスに生成するアドレスの数を制限します。「拡大する範囲」プロパティを「すべて」に設定するとき、「フレキシブル範囲拡大」プロパティを「オン」に設定します。

次の表は、「フレキシブル範囲拡大」プロパティのオプションをまとめたものです。

オプション	説明
オン	アドレス検証は、「拡大する範囲」プロパティが提案リストに追加するアドレスの数を制限します。デフォルトのオプションです。
オフ	アドレス検証は、「拡大する範囲」プロパティが提案リストに追加するアドレスの数を制限しません。

注: アドレスバリデータトランスフォーメーションは、それが提案リストに返すアドレスごとに異なる方法で「フレキシブル範囲拡大」プロパティを適用します。トランスフォーメーションは、リストの拡大アドレスの数に固定の制限を適用しません。トランスフォーメーションはまた、リストに含める拡大アドレスの数を計算するとき、「最大結果カウント」プロパティを考慮します。

Geocode データ型

アドレスバリデータトランスフォーメーションがアドレスの Geocode データを計算する方法を決定します。ジオコードは緯度と経度の座標です。

インストールする地理的コーディング参照データに応じてトランスフォーメーションが返す地理的コーディングの結果です。地理的コーディング参照データの詳細については、Informatica お問い合わせください。

次のいずれかのジオコードオプションを選択します。

到着点

建物または区画の入り口の緯度と経度の座標を返します。デフォルトのオプションです。

以下の国の住所では、到着点オプションを選択できます。

オーストラリア、オーストリア、カナダ、クロアチア、デンマーク、エストニア、フィンランド、フランス、ドイツ、ハンガリー、イタリア、ラトビア、リヒテンシュタイン、リトアニア、ルクセンブルグ、メキシコ、モナコ、オランダ、ノルウェー、ポーランド、スロバキア、スロベニア、スウェーデン、スイス、および米国。

到着点ジオコードを指定し、アドレスバリデータトランスフォーメーションが住所のジオコードを返せない場合、トランスフォーメーションは挿入ジオコードを返します。

標準

建物または区画の入り口の推定される緯度と経度の座標を返します。推定されるジオコードは挿入ジオコードとも呼ばれます。

アドレスバリデータトランスフォーメーションは、参照データに登録されている最寄りのジオコードを使用して、住所のジオコードを推定します。

注: Informatica では、区画の中心またはルーフトップのジオコーディング用に参照データを発行しなくなりました。

パラメータの使用

パラメータを使用して geocode 型を指定できます。ARRIVAL_POINT または NONE を入力します。標準ジオコードを返すには、NONE を入力します。

パラメータ値を大文字で入力します。

グローバル最大フィールド長

アドレスの行の最大文字数を決定します。指定した以上の文字を含む出力アドレス行を書き込む場合、アドレスバリデータトランスフォーメーションは行のアドレス要素を短縮します。

このプロパティを使用し、アドレスの行の長さを制御します。例えば、SNA 基準の場合、アドレスの行に 38 文字を超える文字を含めることはできません。SNA 基準でアドレスを生成する場合、[グローバル最大フィールド長] を 38 に設定します。

デフォルトは 1024 です。

パラメータの使用

パラメータを使用してアドレスの最大数を指定することができます。パラメータ値を設定するには、0 から 1024 の整数を入力します。

グローバル優先記述子

アドレスバリデータトランスフォーメーションが出力データに書き込む建物の記述子、棟の記述子、および番地の記述子の形式を決定します。宛先の国のアドレス参照データに 1 つ以上のデータ要素に対する一連の記述子が含まれる場合は記述子を選択します。

以下の表に、このプロパティのオプションを示します。

オプション	説明
データベース	参照データベースがアドレスの要素に指定する記述子を返します。データベースによってアドレスの記述子が指定されていない場合、トランスフォーメーションは入力値を出力アドレスにコピーします。 Database がデフォルト値です。
完全表記	Street などのように、記述子の完全な形式を返します。
入力の維持	入力アドレスから出力アドレスに記述子をコピーします。 入力記述子が有効なバージョンではない場合、トランスフォーメーションは参照データベースから同等の有効な記述子を返します。
略式表記	St などのように、記述子の略式表記を返します。

入力形式の種類

フィールド化されていない入力データに含まれる最も一般的な情報の種類を示します。[入力形式の種類] プロパティは、入力データを住所の正式表記やフォーマットされたアドレス行ポートに接続する場合に使用します。マッピングソリューションでの情報を最も適切に表すオプションを選択します。

次のいずれかのオプションを選択します。

- すべて
- 住所
- 組織
- 担当者
- 組織/担当者
アドレスには組織情報と担当者情報が含まれます。
- 組織/部署
アドレスには組織情報と部署情報が含まれます。

デフォルトは [すべて] です。

国を含む入力形式

入力に国のデータが含まれるかどうかを指定します。このプロパティは、入力データを [住所の正式表記] または [フォーマットされたアドレス行] ポートに接続する場合で、そのデータが国情報を含む場合に選択します。

このオプションはデフォルトで選択されていません。

行セパレータ

フォーマットされたアドレスの改行を示す区切り記号を指定します。

次のいずれかのオプションを選択します。

- パラメータを割り当てて行セパレータを識別する
- キャリッジリターン (CR)
- カンマ
- ラインフィード (LF)
- なし
- セミコロン
- タブ
- Windows 改行 (CRLF)

デフォルトは [セミコロン] です。

行セパレータは、**【全般設定】** タブで設定することもできます。

パラメータの使用

パラメータを使用して行セパレータを指定することができます。パラメータ値は大文字と小文字が区別されません。パラメータ値は大文字で入力します。

次のいずれかの値を入力します。

- CR
- CRLF
- COMMA
- LF
- PIPE
- SEMICOLON
- SPACE
- TAB

一致する代替用語

アドレス検証が入力アドレスにある代替の場所の名前（例えば同義語や歴史名など）を認識するかどうかを決定します。このプロパティは、通り、市区町村、および地方のデータに適用されます。

注: [一致する代替用語] プロパティは、検証されたアドレスにある代替名を維持しません。

以下の表に、一致する代替用語のオプションを示します。

オプション	説明
すべて	通りおよび場所のすべての既知の代替名を認識します。デフォルトのオプションです。
アーカイブのみ	歴史名のみ認識します。例えば、「Constantinople」を「Istanbul」の古い呼称として検証します。

オプション	説明
なし	通りや場所の代替名を認識しません。
同義語のみ	同義語およびエクソニム（外名）のみ認識します。例えば、「Londres」を「London」のエクソニム（外名）として検証します。

拡張アーカイブのマッチング

アドレス検証で、古くなった日本のアドレスに対する一意のデリバリポイントコードが返されるかどうかを決定します。

日本のアドレス参照データファイルには、対応するメールボックスの現在のアドレスと共に、古くなった住所または使用されなくなった住所のデータが格納されています。[拡張アーカイブのマッチング] プロパティを選択した場合は、アドレス検証で現在バージョンの各住所のデリバリポイントコードが返されます。また、入力した住所が古くなっていることを示す値も、アドレス検証によって [拡張要素の結果ステータス] ポートに書き込まれます。

アドレス参照データから現在のアドレスを取得するには、入力要素としてアドレスコードを入力します。

次の表に、拡張アーカイブのマッチングのオプションを示します。

オプション	説明
オフ	プロパティを適用しません。
オン	古くなった日本のアドレスの現在バージョンのアドレスコードを返します。

日本の場合、[拡張アーカイブのマッチング] プロパティには補足データおよびアドレスコードルックアップデータが使用されます。アドレス検証でこのプロパティを適用するには、アドレスコードルックアップモードで実行されるようにトランスフォーメーションを設定します。

一致するスコープ

トランスフォーメーションがアドレス検証中にアドレス参照データに対して一致させるデータ量を決定します。

以下の表に、一致するスコープのオプションを示します。

オプション	説明
すべて	すべての選択されたポートを検証します。デフォルトのオプションです。
納入場所	[番地] オプションで検証されるデータに加え、建物および棟のデータを検証します。
市区町村	都道府県、市区町村、および郵便番号のデータを検証します。
番地	[市区町村] オプションで検証されるデータに加え、番地アドレスデータを検証します。

最大結果カウント

アドレス検証が提案リストモードで返すことができるアドレスの最大数を決定します。

1 から 100 の範囲で最大数を設定できます。デフォルトは 20 です。

注: 提案リストモードは、アドレス参照データに対してアドレスチェックを実行し、入力アドレスに一致する可能性のあるアドレスのリストを返します。提案リストモードのアドレスを検証すると、アドレス検証は最も一致するものを最初に返します。

パラメータの使用

パラメータを使用してアドレスの最大数を指定することができます。パラメータ値を設定するには、0 から 100 の整数を入力します。

モード

トランスフォーメーションが実行するアドレス分析の種類を決定します。トランスフォーメーションの【全般設定】タブでモードを設定することもできます。

関連項目：

- [「アドレスバリデータトランスフォーメーションの全般設定」 \(ページ 100\)](#)

最適化レベル

トランスフォーメーションが入力アドレスデータとアドレス参照データを照合する方法を決定します。このプロパティは、トランスフォーメーションが住所レコードを更新する前に、入力データと参照データの間で行う照合のタイプを指定します。

以下の表に、最適化レベルのオプションを示します。

オプション	説明
低	トランスフォーメーションは、検証を行う前に番地情報から建物番号または住居番号を解析します。あるいは、入力ポート構造に従って入力アドレス要素を厳密に検証します。低オプションは、アドレス検証を最速で実行しますが、他のオプションよりも精度が低下する場合があります。
標準	トランスフォーメーションは検証を行う前に、入力データから複数の種類の住所情報を解析します。標準オプションを選択すると、トランスフォーメーションは複数の入力値と参照データを一致させることができる場合にアドレスを更新します。 デフォルトは【標準】です。
高	トランスフォーメーションは標準解析設定を使用し、入力データ全体に対して追加の解析処理を実行します。高オプションを選択すると、トランスフォーメーションは少なくとも 1 つの入力値と参照データを一致させることができる場合にアドレスを更新します。高オプションではマッピングの実行時間が増加します。

パラメータの使用

パラメータを使用して最適化レベルを指定することができます。NARROW、STANDARD、または WIDE を入力します。パラメータ値を大文字で入力します。

出力形式の種類

完全アドレスまたはフォーマットされたアドレス行の出力ポートにトランスフォーメーションで書き込む情報の最も一般的な種類を示します。出力ポートに書き込む情報を示すオプションを選択します。

次のいずれかのオプションを選択します。

- すべて
- 住所
- 組織
- 担当者
- 組織/担当者
アドレスには組織情報と担当者情報が含まれます。
- 組織/部署
アドレスには組織情報と部署情報が含まれます。

デフォルトは「すべて」です。

国を含む出力形式

トランスフォーメーションで完全アドレスまたはフォーマットされたアドレス行の出力ポートに国識別データを書き込むかどうかを決定します。

このオプションはデフォルトで選択されていません。

優先される言語

参照データセットにデータが複数の言語で含まれている場合、アドレスバリデータトランスフォーメーションがアドレス要素を返す際の言語を決定します。ベルギー、カナダ、中国、フィンランド、香港、アイルランド、イスラエル、マカオ、スイス、台湾の住所には、優先言語を設定することができます。

アドレスバリデータトランスフォーメーションは、次の言語でアドレスデータを返すことができます。

- アドレス参照データの住所のデフォルト言語。デフォルト言語は、各住所が属する地域で話されている主要言語です。
- 住所に対してアドレス参照データでサポートされている他の言語。例えば、ベルギーの参照データには、フラマン語、フランス語、およびドイツ語のアドレス要素が含まれます。

アドレス参照データは、1つのアドレス要素のデータや住所の正式表記のデータが複数の言語で記述されている場合があります。例えば、アドレス検証はアイルランドのすべてのアドレス要素を英語で返すことや、番地、市区町村、および県の情報をアイルランド語で返すこともできます。さらに、アドレス参照データは、1つの国のさまざまな地域の住所に対して異なるデフォルト言語を指定する場合があります。例えば、スイスの参照データのデフォルト言語は地域ごとに異なり、フランス語、ドイツ語、イタリア語に分かれます。

次の表に、[優先される言語] プロパティで選択できるオプションを示します。

オプション	説明
データベース	アドレス参照データで指定される言語で各住所を返します。アドレス参照データでは、国の中の地域によって住所に異なる言語が指定されていることがあります。 [データベース] はデフォルトのオプションです。
代替 1、 代替 2、 代替 3	代替言語のアドレス要素を参照データから返します。代替言語は、住所が属する国によって異なります。
英語	参照データに英語のデータが含まれる場合、アドレス要素を英語で返します。他の住所要素は、住所が属する地域のデフォルト言語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

注: 各アドレス参照データセットには、一部の住所要素だけがデフォルトではない言語で記録されていることがあります。プロパティで指定されている言語で記録された要素を見つけることができない場合、トランスフォーメーションはデフォルト言語で記録された要素を返します。

優先される言語オプションを設定したら、[優先されるスクリプト] プロパティで指定した文字セットが想定の実出力アドレスデータと互換性があることを確認してください。

ベルギーの住所の多言語サポート

次の表に、ベルギーの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。住所は、住所が属する地域の主要言語で返します。言語は、フラマン語、フランス語、およびドイツ語の可能性がありま。
英語	アドレス参照データに英語による情報が含まれている場合、州、地域、および番地の情報を英語で返します。 他の住所要素は、住所が属する地域の主要言語で返します。
代替 1	州、地域、および番地の情報をフラマン語で返します。
代替 2	州、地域、および番地の情報をフランス語で返します。
代替 3	州、地域、および番地の情報をドイツ語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

カナダの住所の多言語サポート

次の表に、カナダの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。ケベック州を除くすべての州について、英語で住所を返します。ケベック州の住所をフランス語で返します。
英語	すべての住所を英語で返します。
代替 1	すべての住所を英語で返します。
代替 2	ケベック州の住所をフランス語で返します。 ケベック州以外の州では、トランスフォーメーションは、番地記述子、方位情報、および州名をフランス語で返し、他の住所要素を英語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

中国の住所の多言語サポート

次の表に、中国の住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報を中国語で返します。
英語	番地記述子と方位情報を英語バージョンで返します。他のすべての住所情報を中国語で返します。 英語の住所要素では、市（shi）などの文字変換要素が省略されます。
代替 1	すべての住所情報をデータベース言語で返します。
代替 2	すべての住所情報をデータベース言語で返します。
代替 3	すべての住所情報をデータベース言語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

優先される言語を選択する場合、次のルールとガイドラインを検討します。

- 住所を中国語で返すには、[データベース]、[代替 1]、[代替 2]、または [代替 3] を選択します。
住所を中国語の文字セットで返すには、[優先されるスクリプト] プロパティを [データベース] に設定します。

- 番地記述子と方位情報を英語で返すには、[英語] を選択します。
住所をラテン文字セットまたは ASCII 文字セットで返すには、[優先されるスクリプト] プロパティを [ラテン] または [ASCII] の値に設定します。
- 優先されるスクリプトとして [ラテン] または [ASCII] の値を選択し、優先される言語として [データベース] を選択した場合、アドレス検証は住所データをピンイン式で返します。

フィンランドの住所の多言語サポート

次の表に、フィンランドの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報をフィンランド語で返します。
代替 1	すべての住所情報をデータベース言語で返します。
代替 2	番地、地域、および県の情報をスウェーデン語で返します。他のすべての情報をフィンランド語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

香港の住所の多言語サポート

次の表に、香港の住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報を中国語で返します。
英語	すべての住所情報を英語で返します。
代替 1	すべての住所情報をデータベース言語で返します。
代替 2	すべての住所情報を英語で返します。
代替 3	すべての住所情報をデータベース言語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

香港の優先される言語を選択する場合、次のルールとガイドラインを検討します。

- 住所を中国語の文字セットで返すには、[優先されるスクリプト] プロパティを [データベース] に設定します。
- 住所をラテン文字セットまたは ASCII 文字セットで返すには、[優先されるスクリプト] プロパティを [ラテン] または [ASCII] の値に設定します。

- 入力データの言語は、香港の住所に対する [入力の維持] オプションの操作に影響することがあります。入力データで 7 ビット ASCII 文字を使用していて、英語の記述子が含まれている場合、アドレス検証は入力言語を英語として識別します。

アイルランドの住所の多言語サポート

次の表に、アイルランドの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報を英語で返します。
英語	すべての住所情報を英語で返します。
代替 1	すべての住所情報を英語で返します。
代替 2	番地、地域、および郡の情報をアイルランド語で返します。他のすべての住所情報を英語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

イスラエルの住所の多言語サポート

次の表に、イスラエルの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報をヘブライ語で返します。
英語	すべての住所情報を英語で返します。
代替 1	すべての住所情報をヘブライ語で返します。
代替 2	すべての住所情報を英語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

優先される言語を選択する場合、次のルールとガイドラインを検討します。

- 住所をヘブライ語の文字セットで返すには、[優先されるスクリプト] プロパティを [データベース] に設定します。
- 住所をラテン文字セットまたは ASCII 文字セットで返すには、[優先されるスクリプト] プロパティを [ラテン] または [ASCII] の値に設定します。

- 優先されるスクリプトとしてラテン文字セットを選択し、優先される言語としてヘブライ語を選択した場合、アドレス検証によってヘブライ語の住所がラテン文字に変換されます。ラテン文字セットで最適な結果を得るには、優先される言語として英語を選択してください。

マカオの住所の多言語サポート

次の表に、マカオの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報を中国語で返します。
代替 1	すべての住所情報をデータベース言語で返します。
代替 2	すべての住所情報をポルトガル語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

- 住所を中国語の文字セットで返すには、[優先されるスクリプト] プロパティを [データベース] に設定します。
- 住所をラテン文字セットまたは ASCII 文字セットで返すには、[優先されるスクリプト] プロパティを [ラテン] または [ASCII] の値に設定します。
- 入力データの言語は、マカオの住所に対する [入力の維持] オプションの操作に影響することがあります。入力データで 7 ビット ASCII 文字を使用していて、ポルトガル語の記述子が含まれている場合、アドレス検証は入力言語をポルトガル語として識別します。

スイスの多言語サポート

次の表に、スイスの住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。住所は、住所が属する地域の主要言語で返します。 たとえば、チューリッヒの住所はドイツ語で、ジュネーブの住所はフランス語で返されます。
英語	参照アドレスデータベースに英語による情報が含まれている場合、地域および州の情報を英語で返します。 他の住所要素は、住所が属する地域の主要言語で返します。 ジュネーブやチューリッヒなどの一部の地域については市区町村情報を英語で返します。
代替 1	州および地域の情報をドイツ語で返します。
代替 2	州および地域の情報をフランス語で返します。

オプション	説明
代替 3	州および地域の情報をイタリア語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

注: ビール/ビエンヌの住所の通り情報は、設定した代替言語でも返されます。

台湾の多言語サポート

次の表に、台湾の住所に指定できる言語を示します。

オプション	説明
データベース	デフォルト値。すべての住所情報を中国語で返します。
英語	すべての住所情報を英語で返します。
入力の維持	住所情報を入力言語で返します。参照データに入力言語による住所情報が含まれる場合、アドレス検証は、その言語を保持します。 アドレス検証が入力住所でサポートされる言語を複数検出した場合は、住所をデータベース言語で返します。アドレス検証が入力された言語で要素を返すことができない場合は、要素をデータベース言語で返します。

優先される言語を選択する場合、次のルールとガイドラインを検討します。

- 住所を中国語の文字セットで返すには、[優先されるスクリプト] パラメータを [データベース] に設定します。
- 住所をラテン文字セットまたは ASCII 文字セットで返すには、[優先されるスクリプト] パラメータを [ラテン] または [ASCII] の値に設定します。
- 入力データの言語は、台湾の住所に対する [入力の維持] オプションの操作に影響することがあります。入力データが 7 ビット ASCII 文字を使用し、英語の記述子を含んでいる場合、アドレス検証は入力言語を英語として識別します。

優先されるスクリプト

アドレスバリデータトランスフォーメーションが出力データに使用する文字セットを決定します。

以下の表に、このプロパティのオプションを示します。

オプション	説明
ASCII (簡易)	住所を ASCII 文字で返します。
ASCII (拡張)	住所を ASCII 文字で返し、住所内の特殊文字を展開します。例えば、Ö は OE に置き換えられます。

オプション	説明
データベース	アドレス参照データがデフォルト言語に使用する文字セットで住所を返します。 デフォルト値は「データベース」です。
ラテン	ラテン文字セットで住所を返します。
ラテン（代替）	代替のラテン文字セットで住所を返します。 たとえば、文化観光部の翻字で韓国の住所を返すには、「ラテン」を指定します。より古い ISO/TR 11941 方式の翻字で韓国の住所を返すには、「ラテン（代替）」を指定します。
郵政	住所が所属する地域の郵政で優先される文字で住所を返します。
郵政（代替）	住所が所属する地域の郵政が代替文字として承認している文字で住所を返します。
入力の維持	入力アドレスで使用される文字セットでアドレスデータを返します。

トランスフォーメーションは、複数の言語および文字セットのデータを含むデータソースを処理できます。トランスフォーメーションはすべての入力データを Unicode UCS-2 文字セットに変換し、データを UCS-2 形式で処理します。データを処理した後、トランスフォーメーションは各住所レコードのデータをプロパティで指定した文字セットに変換します。このプロセスは「翻字」と呼ばれます。

文字を変換する際、翻字では、文字セットの各文字に対応する数値表現（コード）を使用できます。また、対応する数値表現がない文字の場合は、音写変換することもできます。アドレスバリデータトランスフォーメーションが文字を UCS-2 にマップできない場合、文字はスペースに変換されます。

注: トランスフォーメーションの優先言語または優先文字を更新する場合は、選択した言語と文字コードが互換性があることを確認してください。

拡大する範囲

住居番号を指定しない番地に対してアドレスバリデータトランスフォーメーションで提案住所を返すかどうかを決定します。トランスフォーメーションを提案リストモードで実行するときにこのプロパティを使用します。

アドレスバリデータトランスフォーメーションは、提案リストモードで部分的または不完全な番地を読み込みます。トランスフォーメーションはその住所とアドレス参照データを比較し、類似したすべてのデータをエンドユーザーに返します。入力アドレスに住居番号が含まれない場合、トランスフォーメーションは番地に対して 1 つまたは複数の住居番号提案を返すことができます。[拡大する範囲] プロパティにより、トランスフォーメーションがアドレスを返す方法が決定されます。

トランスフォーメーションは 1 つのアドレスで有効な住居番号の範囲を返すことができます。あるいは、有効な住居番号ごとにアドレスを返すことができます。トランスフォーメーションは、番地の最小の住居番号から最高の住居番号までの範囲にある番号ごとにアドレスを返すこともできます。

以下の表に、このプロパティのオプションを示します。

オプション	説明
すべて	アドレス検証で、番地の考えられる住居番号の範囲で、すべての住居番号に対して提案アドレスを返します。
なし	アドレス検証は、番地の有効範囲にある最小の住居番号と最高の住居番号を識別する 1 つのアドレスを返します。
有効な項目があるもののみ	アドレス検証は、アドレス参照データが配達可能として識別する住居番号ごとに提案アドレスを返します。

注: 提案リストモードでは、アドレスの他の要素を使用し、番地の有効範囲を指定できます。例えば、ZIP コードにより、アドレスメールボックスを含む市のブロックを識別できる場合があります。アドレスバリデータトランスフォーメーションでは ZIP コードを使用し、ブロックの最小および最高の住居番号を識別できます。

トランスフォーメーションで実際の限度内で住居番号範囲を決定できない場合、提案アドレスの番号は使用できないサイズまで増加できます。[拡大する範囲] プロパティで生成されるアドレスの数を制限するには、[フレキシブル範囲拡大] プロパティを [オン] に設定します。

無効なアドレスの標準化

アドレス検証プロセスで配送不可能なアドレスのデータ値を標準化するかどうかを決定します。このプロパティは、I1~I4 の範囲で照合コードのステータスを返すアドレスレコードに適用します。

データを標準化すると、後続のデータ処理で正確な結果が返される可能性が高くなります。例えば、2 つの住所レコードが同じ形式で共通の住所要素を表している場合、重複分析マッピングの一致スコアが高くなる可能性があります。

アドレス検証では、次のアドレス要素を標準化できます。

- road や boulevard など、通りのサフィックス要素。
- north、south、east、west など、前後の方角要素。
- 私書箱などの配達サービス要素。
- apartment、floor、suite など、棟要素。
- 都道府県の名前。標準化により、名前の短縮形が返されます。

以下の表に、このプロパティのオプションを示します。

オプション	説明
オフ	アドレス検証はデータエラーを修正しません。デフォルトのオプションです。
オン	アドレス検証はデータエラーを修正します。

パラメータの使用

データエラーの標準化ポリシーを指定するパラメータを割り当てることができます。パラメータ値として OFF または ON を入力します。値は大文字で入力します。

トレースレベル

ログに含まれる詳細の量を設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

認証レポート

認証済みモードの検証のために送信する住所のステータスを記載したレポートを作成できます。レポートでは、郵便事業者が定義する認証基準を住所セットが満たしているかを検証します。

アドレスバリデータトランスフォーメーションは、次の標準のレポートを生成します。

Address Machine Approval System (AMAS)

オーストラリア郵政公社が AMAS 認証基準を定義します。

Coding Accuracy Support System (CASS)

USPS が CASS 認証基準を定義します。

SendRight

ニュージーランド郵政公社が SendRight 認証基準を定義します。

Software Evaluation and Recognition Program (SERP)

カナダ郵政省が SERP 認証基準を定義します。

注: アドレスバリデータトランスフォーメーションを使用し、フランスのアドレスレコードを Service National de l'Adresse (SNA) 認証基準と照合して認証できます。しかし、アドレスバリデータトランスフォーメーションで SNA アドレス認証のレポートを作成することはありません。

認証基準を満たしている郵送項目を郵便事業者に提供する際に、アドレスセットを含む認証レポートを送信します。レポートには組織に関するデータが含まれます。このデータはアドレスバリデータトランスフォーメーションの設定時に入力します。トランスフォーメーションにより組織データがレポートファイルに追加されます。

AMAS レポートのフィールド

AMAS レポートを設定するとき、オーストラリア郵政公社に認定アドレスレコードセットを提出する組織の情報を提供します。レポートを保存または印刷し、オーストラリア郵政公社に提出するアドレスレコードにレポートを含めます。

【レポート】 ビューを利用し、情報を入力します。

次の表に、入力する情報を示します。

フィールド	説明
レポートファイル名	アドレス検証で作成されるレポートの名前と場所。デフォルトでは、アドレスバリデータにより、データ統合サービスマシンの bin ディレクトリにレポートが作成されます。 データ統合サービスマシンの別の場所にレポートファイルを書き込むには、ファイルパスとファイル名を入力します。パスは完全修飾パスでも相対パスでもかまいません。相対パスでは、ルートディレクトリとして bin ディレクトリが使用されます。指定するディレクトリは、アドレス検証マッピングの実行前に作成しておく必要があります。
住所リスト名	オーストラリア郵政公社に提出するアドレスレコードセットの名前。
プロセッサ名を一覧表示	アドレスレコードセットを提出する組織の名前。
リストの管理者/所有者の名前	組織内のアドレスデータの管理者または所有者の名前。
電話番号	アドレスレコードセットを提出する組織の問い合わせ電話番号。
住所	アドレスレコードセットを提出する組織の住所。

関連項目：

- [「認証レポートの定義」 \(ページ 129\)](#)

CASS レポートのフィールド

CASS レポートを設定するとき、USPS に認定アドレスレコードセットを提出する組織の情報を提供します。レポートを保存または印刷し、USPS に提出するアドレスレコードにレポートを含めます。

【レポート】 ビューを利用し、情報を入力します。

次の表に、入力する情報を示します。

フィールド	説明
レポートファイル名	アドレス検証で作成されるレポートの名前と場所。デフォルトでは、アドレスバリデータにより、データ統合サービスマシンの bin ディレクトリにレポートを作成します。 データ統合サービスマシンの別の場所にレポートファイルを書き込むには、ファイルパスとファイル名を入力します。完全修飾パスまたは相対パスを入力できます。相対パスでは、ルートディレクトリとして bin ディレクトリが使用されます。指定するディレクトリは、アドレス検証マッピングの実行前に作成しておく必要があります。
リスト名/ID	郵便運送業者に提出する住所リストの名前または ID 番号。
プロセッサ名を一覧表示	アドレス検証を実行する組織の名前。
名前/住所	アドレス検証を実行する組織の名前と住所。

注: アドレスバリデータトランスフォーメーションは、米国のバッチおよび対話型参照データファイルから CASS 参照データファイルに関するメタデータを読み取ります。CASS レポートを生成するには、トランスフォ

一メーションは現在の CASS 参照データファイルと現在のバッチおよび対話型参照データファイルを読み取る必要があります。

関連項目：

- [「認証レポートの定義」 \(ページ 129\)](#)

SendRight レポート

SendRight レポートを設定するとき、ニュージーランド郵政公社に認定アドレスレコードセットを提出する組織の情報を提供します。レポートを保存または印刷し、ニュージーランド郵政公社に提出するアドレスレコードにレポートを含めます。

【レポート】 ビューを利用し、情報を入力します。

次の表に、入力する情報を示します。

フィールド	説明
顧客名	アドレスレコードセットを提出する組織の名前。
顧客 NZP 番号	アドレスレコードセットを提出する組織のニュージーランド郵政公社のアカウント番号。メールハウスが組織の代わりにレコードを提出する場合、メールハウスの輸送 ID (TPID) 番号を入力します。
顧客データベース	アドレスレコードセットを含むファイルの名前。 アドレスバリデータトランスフォーメーションは、コンテンツ管理サービスで指定した場所にレポートを作成します。Administrator ツールを使用し、場所を設定します。
顧客住所	アドレスレコードセットを提出する組織の住所。

関連項目：

- [「認証レポートの定義」 \(ページ 129\)](#)

SERP レポートのフィールド

SERP レポートを設定するとき、カナダ郵政省に認定アドレスレコードセットを提出する組織の情報を提供します。レポートを保存または印刷し、カナダ郵政省に提出するアドレスレコードにレポートを含めます。

【レポート】 ビューを利用し、情報を入力します。

次の表に、入力する情報を示します。

フィールド	説明
レポートファイル名	アドレス検証で作成されるレポートの名前と場所。デフォルトでは、アドレスバリデータにより、データ統合サービスマシンの bin ディレクトリにレポートを作成します。 データ統合サービスマシンの別の場所にレポートファイルを書き込むには、ファイルパスとファイル名を入力します。完全修飾パスまたは相対パスを入力できます。相対パスでは、ルートディレクトリとして bin ディレクトリが使用されます。 指定するディレクトリは、アドレス検証マッピングの実行前に作成しておく必要があります。
顧客 CPC 番号	アドレス検証を実行する組織のカナダ郵政省発行の顧客番号。
顧客名/アドレス	アドレス検証を実行する組織の名前と住所。

関連項目：

- [「認証レポートの定義」 \(ページ 129\)](#)

アドレスバリデータトランスフォーメーションの設定

アドレスバリデータトランスフォーメーションは、郵便アドレスデータの品質を検証して改善するために使用します。

アドレスバリデータトランスフォーメーションでは、アドレス参照データを読み取ります。必要なアドレス参照データファイルに Developer ツールからアクセスできることを確認してください。

1. トランスフォーメーションを開きます。
2. **【全般設定】** ビューをクリックし、全般プロパティを設定します。
3. **【テンプレート】** ビューをクリックし、入力ポートと出力ポートを追加します。
4. **【レポート】** ビューをクリックし、郵政機関のアドレス認証用のレポートを生成します。
5. **【詳細】** ビューをクリックし、アドレス検証の詳細プロパティを設定します。
6. 入力ポートおよび出力ポートを接続します。

注: アドレスバリデータトランスフォーメーションで検証しない入力ポートは、**【パススルー】** 入力ポートグループに接続します。

アドレスバリデータトランスフォーメーションへのポートの追加

アドレスバリデータトランスフォーメーションにポートを追加するには、**【テンプレート】** ビューを使用します。

1. **【テンプレート】** ビューをクリックします。

2. テンプレートを展開します。
 - 一般的なアドレスフィールドを追加する場合は、**【基本モデル】** テンプレートを選択します。
 - 特殊なアドレスフィールドを追加する場合は、**【詳細モデル】** テンプレートを選択します。
3. 入力データの形式に対応する入力ポートグループを展開します。入力ポートグループは、**【個別】**、**【複数行】**、および **【混合】** です。
4. 入力ポートを選択します。

ヒント: 複数のポートを選択するには、Ctrl キーを押しながらクリックします。
5. ポートを右クリックし、**【トランスフォーメーションにポートを追加】** を選択します。
6. 必要なフィールドを含む出力ポートグループを展開します。
7. ポートを右クリックし、**【トランスフォーメーションにポートを追加】** を選択します。
8. 検証しないカラム用のパススルーポートを追加するには、**【ポート】** タブをクリックし、**【パススルー】** 入力ポートグループを選択して、**【新規】** をクリックします。

ユーザー定義テンプレートの作成

再利用するアドレスポートをグループ化するには、テンプレートを作成します。

カスタムテンプレートは、基本モデルおよび詳細モデルのテンプレートからポートを選択して作成します。作成したカスタムテンプレートは、以降にアドレスバリデータトランスフォーメーションを作成する際に選択することができます。

注: テンプレートはリポジトリオブジェクトではありません。作成に使用したマシンに保存されます。

1. **【テンプレート】** ビューを選択します。
2. **【新規】** をクリックします。
3. テンプレートの名前を入力します。
4. **【基本モデル】** テンプレートまたは **【詳細モデル】** テンプレートを展開し、必要なポートを選択します。
5. **【OK】** をクリックします。

アドレスバリデータのモデルの定義

アドレスバリデータのモデルとは、アドレスバリデータトランスフォーメーションのデフォルトの入力ポートと出力ポートを定義したものです。

アドレスバリデータトランスフォーメーションには、デフォルトの入力ポートと出力ポートはありません。ただし、モデルを定義することで、アドレスバリデータトランスフォーメーションで使用する入力ポートと出力ポートを指定できます。

注: モデルはリポジトリオブジェクトではありません。作成に使用したマシンに保存されます。

アドレスバリデータのモデルを定義するには、以下の手順を実行します。

1. **【テンプレート】** ビューを選択します。
2. **【基本モデル】** テンプレートまたは **【詳細モデル】** テンプレートを展開し、必要なポートを選択します。
3. **【選択したポートを使用してデフォルト AV モデルを作成】** を選択します。

4. モデルをリセットしてすべてのポートを削除するには、**[デフォルト AV モデルのクリア]** を選択します。

認証レポートの定義

アドレスバリデータトランスフォーメーションで認証レポートを定義するときは、**[全般設定]** ビューと **[レポート]** ビューでオプションを設定します。

1. **[全般設定]** ビューで、**[モード]** オプションを **[認証]** に設定します。
2. **[レポート]** ビューで、生成するレポートのタイプを選択します。以下のタイプのレポートを選択できます。

オプション	説明
AMAS レポート	オーストラリア郵政公社がレコードセットの処理に必要とする情報を含みます。
CASS レポート	USPS がレコードセットの処理に必要とする情報を含みます。
SendRight レポート	ニュージーランド郵政公社がレコードセットの処理に必要とする情報を含みます。
SERP レポート	カナダ郵政省がレコードセットの処理に必要とする情報を含みます。

3. 選択したレポートのタイプに応じて、レポートの詳細を入力します。

レポートファイルは、アドレスバリデータトランスフォーメーションで検証したアドレスレコードのリストと一緒に郵便運送業者に提出します。

関連項目：

- [「AMAS レポートのフィールド」 \(ページ 124\)](#)
- [「CASS レポートのフィールド」 \(ページ 125\)](#)
- [「SendRight レポート」 \(ページ 126\)](#)
- [「SERP レポートのフィールド」 \(ページ 126\)](#)

非ネイティブ環境でのアドレスバリデータトランスフォーメーション

非ネイティブ環境でのアドレスバリデータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのアドレスバリデータトランスフォーメーション

アドレスバリデータトランスフォーメーションのサポートには、次の制限があります。

- アドレスバリデータトランスフォーメーションでは、認証レポートは生成できません。
- アドレスバリデータトランスフォーメーションは、対話モードまたは提案リストモードで実行されるように構成されていると、検証に失敗します。

Spark エンジンでのアドレスバリデータトランスフォーメーション

アドレスバリデータトランスフォーメーションのサポートには、次の制限があります。

- アドレスバリデータトランスフォーメーションでは、認証レポートは生成できません。
- アドレスバリデータトランスフォーメーションは、対話モードまたは提案リストモードで実行されるように構成されていると、検証に失敗します。

アドレスバリデータトランスフォーメーション（ストリーミングマッピング）

ストリーミングマッピングには、Spark エンジン上のバッチマッピングと同じ処理ルールがあります。

第 5 章

アグリゲータトランスフォーメーション

この章では、以下の項目について説明します。

- [アグリゲータトランスフォーメーションの概要, 131 ページ](#)
- [動的マッピングでのアグリゲータトランスフォーメーション, 132 ページ](#)
- [アグリゲータトランスフォーメーションの開発, 132 ページ](#)
- [アグリゲータトランスフォーメーションのポート, 132 ページ](#)
- [集計式, 133 ページ](#)
- [グループ化ポート, 135 ページ](#)
- [アグリゲータキャッシュ, 137 ページ](#)
- [アグリゲータトランスフォーメーションの \[ソート済み入力\] オプション, 138 ページ](#)
- [アグリゲータトランスフォーメーションの詳細プロパティ, 140 ページ](#)
- [再利用可能なアグリゲータトランスフォーメーションの作成, 141 ページ](#)
- [再利用不可能なアグリゲータトランスフォーメーションの作成, 141 ページ](#)
- [アグリゲータトランスフォーメーションに関するヒント, 142 ページ](#)
- [アグリゲータトランスフォーメーションのトラブルシューティング, 142 ページ](#)
- [非ネイティブ環境でのアグリゲータトランスフォーメーション, 143 ページ](#)

アグリゲータトランスフォーメーションの概要

アグリゲータトランスフォーメーションを設定して、データのグループに対して平均値や合計値などの集計計算を実行します。アグリゲータトランスフォーメーションを使用して重複行を削除できます。アグリゲータトランスフォーメーションはアクティブなトランスフォーメーションです。

アグリゲータトランスフォーメーションは式トランスフォーメーションとは異なり、データのグループに対して計算を実行するように設定できます。式トランスフォーメーションは、行単位の結果を返します。

例えば、組織の各部署における従業員の平均給与を計算することができます。部署番号別にグループを設定します。平均給与を計算し、一意の部署番号ごとに結果を返すように式を設定します。

トランスフォーメーション言語を使用して集計式を作成します。

データ統合サービスは、データを読み取って集計キャッシュに保存する際に、集計計算を実行します。入力データをソートしてパフォーマンスを向上させることができます。入力データをソートする場合、データ統合サービスはキャッシュを作成しません。

動的マッピングでのアグリゲータトランスフォーメーション

動的マッピングでアグリゲータトランスフォーメーションを使用できます。アグリゲータトランスフォーメーションで動的ポートを設定し、生成されたポートを参照することができます。

以下のタスクを実行することで、動的マッピングにアグリゲータトランスフォーメーションを設定できます。

動的ポートまたは生成されたポートをグループ化カラムで参照する。

動的ポートまたは生成されたポートをグループ化カラムとして含めることができます。グループ化カラムとして動的ポートを指定する場合、動的ポートのすべての生成されたポートをグループ化カラムに指定することになります。このため、親動的ポートをグループ化カラムとして指定した場合、生成されたポートをグループ化カラムとして指定できません。生成されたポートを参照している場合に、そのポートが実行時に存在しないと、マッピングは失敗します。グループ化カラムをパラメータ化することで、実行時にグループ化するカラムを指定できます。

生成されたポートを集計式で参照する。

生成されたポートを集計式に含めることができます。実行時にポートが存在しない場合、マッピングは失敗します。動的ポートを集計式内で参照することはできません。

集計式を出力ポートに作成する。

集計式を、生成されたポートまたは動的ポートに作成することはできません。集計式を動的な式にすることはできません。

集計式で値をパラメータ化する。

集計式にパラメータを含めることができます。ただし、集計式で式パラメータまたはポートパラメータを使用することはできません。

アグリゲータトランスフォーメーションの開発

アグリゲータトランスフォーメーションを作成する場合、式を各行に対して実行するように定義します。返される結果を分類するグループ化ポートのリストを定義します。

アグリゲータトランスフォーメーションを作成するには、次の手順を実行します。

1. トランスフォーメーションを定義し、ポートを作成します。
2. 変数または出力ポートに対して集計式を設定します。
3. 返される集計結果を分類するポートグループを定義します。

アグリゲータトランスフォーメーションのポート

アグリゲータトランスフォーメーションには複数のポートタイプがあり、これらを使用してグループを設定したり、式を集計したりできます。

アグリゲータトランスフォーメーションの [ポート] ビューには、以下のフィールドがあります。

名前

ポートの名前。

タイプ

ポートのデータ型。

精度

フィールドの長さ。

スケール

数値データの小数点以下の位数。

入力

アップストリームトランスフォーメーションからのデータであることを示します。

出力

式の値をポートが返すことを示します。式には非集計式や条件句を含めることができます。複数の集計出力ポートを作成できます。

パススルー

ポートがデータを変更しないで返す入出力ポートであることを示します。

変数

式で使用する値または計算をポートに格納できることを示します。変数ポートは入力ポートまたは出力ポートであってはなりません。変数ポートはトランスフォーメーション内でのみデータを渡します。

式

行または行のグループを集計するための式。

デフォルト値

ポートの値が無効または NULL 値の場合のデフォルト値。

入力ルール

ポート名とデータ型に基づいて、トランスフォーメーションに含めるまたは除外するポートをフィルタリングする一連のルール。動的ポートを定義する場合に入力ルールを設定します。

集計式

アグリゲータトランスフォーメーションの変数ポートまたは出力ポートに、集計式を設定します。ポートが入力ポート、パススルーポート、または動的ポートの場合は、式を入力できません。

集計式には、条件節および集計関数以外の関数を含めることができます。集計関数の場合、次のように別の集計関数内にネストした集計関数を含めることもできます。

```
MAX( COUNT( ITEM ))
```

集計式の結果は、トランスフォーメーション内のグループ化ポートに基づいて変わります。例えば、次の集計式は販売した品目の合計数を計算します。

```
SUM( QUANTITY )
```

一方、同じ式で ITEM ポート別にグループ分けを指定すると、データ統合サービスは品目ごとの合計数を返します。

任意の出力ポートで集計式を作成でき、また 1 つのトランスフォーメーションで複数の集計ポートを使用できます。

集計関数

集計関数はアグリゲータトランスフォーメーション内で設定します。1つの集計関数を別の集計関数にネストすることができます。

トランスフォーメーション言語には、以下の集計関数が用意されています。

- AVG
- COUNT
- FIRST
- LAST
- MAX
- MEDIAN
- MIN
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、データ統合サービスはポート内の最後の行を使用して式を処理します。

例えば、ポート COMMISSIONS および SALARY を含むアグリゲータトランスフォーメーションを作成するとします。ポート SALARY はグループ化ポートです。

出力ポートでは次の式を使用できます。

`SUM(COMMISSIONS)`

データ統合サービスは、アグリゲータ関数を処理し、ポート COMMISSIONS の値の合計を出力ポートで返します。

この式は次の式に変更できます。

`SUM(COMMISSIONS) + COMMISSIONS`

この式を処理するため、データ統合サービスはポート COMMISSIONS の値の合計を返し、ポート COMMISSIONS の最後の行の値を出力ポートの戻り値に追加します。

別の出力ポートでは、次の式を使用できます。

`SUM(COMMISSIONS) + SALARY`

この式を処理するため、データ統合サービスはポート COMMISSIONS の値の合計を返し、ポート SALARY の最後の行の値を出力ポートの戻り値に追加します。SALARY ポートはグループ化ポートであるため、ポート SALARY の各行の値は同じであることに注意してください。

ネストされた集計関数

アグリゲータトランスフォーメーションの異なる出力ポートに、単一レベルの関数を複数含めたり、複数レベルにネストされた関数を複数含めたりすることができます。

アグリゲータトランスフォーメーションには、単一レベルの関数とネストされた関数の両方を含めることはできません。そのため、アグリゲータトランスフォーメーションの出力ポートに単一レベルの関数が含まれる場合には、そのトランスフォーメーションの他のポートでネストされた関数を使うことはできません。単一レベルの関数とネストされた関数を同じアグリゲータトランスフォーメーションに含めると、そのマッピングまたはマッピングレットは Developer ツールによって無効とされます。単一レベルの関数とネストされた関数の両方を作成する必要がある場合は、別々のアグリゲータトランスフォーメーションを作成してください。

集計式の条件句

集計で使用する行の数を削減するには、集計式で条件句を使用します。条件句には、TRUE または FALSE に評価される任意の句を使用できます。

たとえば以下の式を使用して、四半期単位のノルマを超過して達成した従業員の歩合総額を算出します。

`SUM(COMMISSION, COMMISSION > QUOTA)`

グループ化ポート

すべての入力データに対して集計を実行する代わりに、集計対象となる行のグループを定義できます。例えば、会社の総売上高を計算したり、地域グループ別の総売上高を確認したりできます。

集計式に対してグループを定義するには、アグリゲータトランスフォーメーションで該当する入力、入出力、出力、および変数ポートを選択します。複数のグループ化ポートを選択して、一意なグループの組み合わせそれぞれについて新しいグループを作成することができます。その後、データ統合サービスではグループごとに定義済みの集計が実行されます。

値をグループ分けすると、データ統合サービスはグループごとに 1 つの行を生成します。値をグループ分けをしないと、データ統合サービスはすべての入力行に対して 1 つの行を返します。データ統合サービスは、各グループの最後の行に集計結果を返します。特定の行を返すように指定することもできます。例えば、FIRST 集計関数を使用すると、先頭行が返されます。

アグリゲータトランスフォーメーションで複数のグループ化ポートを選択すると、データ統合サービスはポートの順序に基づいてグループ分けの順序を決定します。グループ順序は集計結果に影響する場合があります。正しくグループ分けされるようにグループ化ポートを並べ替えてください。グループのポートを選択した後にポート順を変更することもできます。

例えば、Price_Out という出力ポートを作成します。Price_Out の式は、SUM (Qty * Price) です。グループ化ポートとして Store_ID と Item を定義します。トランスフォーメーションによって、各項目の合計価格が店舗別に返されます。

入力行に、次のようなデータが含まれているとします。

Store_ID	Item	Qty	Price
101	battery	3	2.99
101	battery	1	3.19
101	battery	2	2.59
101	AAA	2	2.45
201	battery	1	1.99
201	battery	4	1.59
301	battery	1	2.45

データ統合サービスは、以下の一意なグループに対して集計計算を実行します。

Store_Id	Item
101	battery
101	AAA
201	battery
301	battery

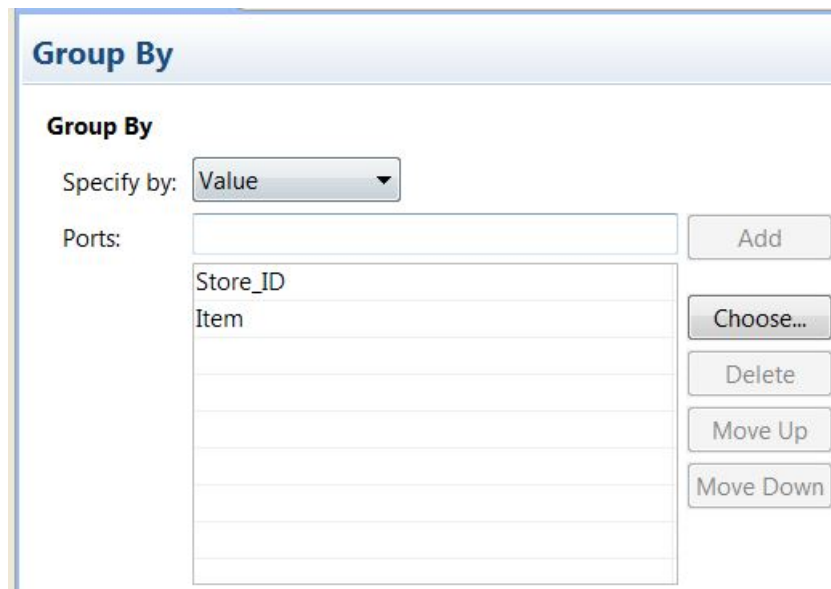
データ統合サービスは、店舗別の項目ごとに、Store_ID、Item、Qty、Price、および最後の行に（Price * Qty）の合計価格を返します。

Store_ID	Item	Qty	Price	Price_Out
101	battery	2	2.59	17.34
101	AAA	2	2.45	4.90
201	battery	4	1.59	8.35
301	battery	1	2.45	2.45

グループ化ポートの設定

トランスフォーメーションの【プロパティ】ビューの【グループ化】タブで、グループ化ポートを定義します。

次の図は【グループ化】タブを示しています。



【グループ化】タブには、以下のオプションがあります。

指定元

【値】または【パラメータ】を選択します。ポート名を使用するには、【値】を選択します。ポートリストパラメータを使用するには、【パラメータ】を選択します。

追加

手動で入力したポート名を受け入れます。【追加】をクリックする前に、有効なポート名を入力する必要があります。

選択

ポートを選択してグループに追加するには、【選択】をクリックします。Developer tool は、トランスフォーメーションからポートのリストを表示します。このリストから選択します。

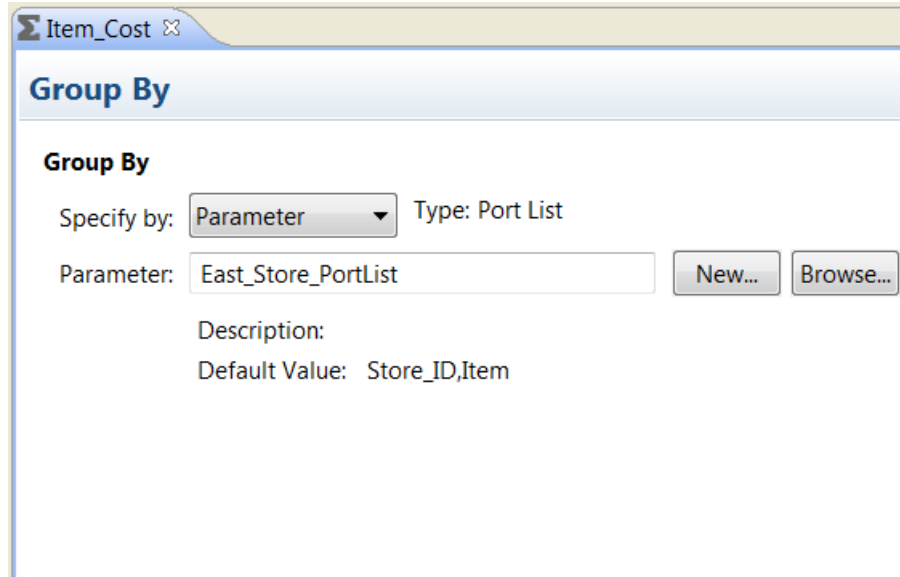
上に移動、下に移動

グループ内のポートの順序を変更できます。ポート名を選択し、この移動ボタンのどちらかをクリックして、ソート順の中でポート名を上下に移動します。

グループ化パラメータ

グループに含める 1 つ以上のポートを含んだ、ポートリストパラメータを設定することができます。 トランスフォーメーションのポートのリストからポートを選択して、ポートリストパラメータを作成します。

次の図は、パラメータを使用してグループ内のポートを特定する場合の【グループ化】タブを示しています。



ポートリストパラメータを参照するか、【新規作成】をクリックしてポートリストパラメータを作成できます。ポートリストパラメータの作成を選択すると、トランスフォーメーションのポートのリストからポートを選択できます。

グループ化ポートのデフォルト値

データ統合サービスは、グループ化ポートに NULL 値が含まれる場合、グループを作成しません。グループ内の各ポートにデフォルト値を定義して、NULL 入力値を置き換えられます。これで、データ統合サービスは集計の合計にその行を含めることができます。

非集計式

グループの変更または置き換えを実行するには、Group By ポートで非集計式を使用します。

例えば、グループ分けの前に「AAA battery」を置換する場合には、以下の式を用いて、CORRECTED_ITEM という名の Group By 出力ポートを作成できます。

```
IIF( ITEM = 'AAA battery', battery, ITEM )
```

アグリゲータキャッシュ

アグリゲータトランスフォーメーションを使用するマッピングを実行すると、データ統合サービスはメモリにインデックスキャッシュおよびデータキャッシュを作成してトランスフォーメーションを実行します。メモリ

キャッシュ内の使用可能スペースよりも多くのスペースが必要な場合、データ統合サービスはオーバーフローしたデータをキャッシュファイルに格納します。

データ統合サービスでは、アグリゲータトランスフォーメーションに対して以下のキャッシュが作成されます。

- Group By ポートの設定に従ってグループ値を格納するインデックスキャッシュ。
- Group By ポートに基づいて計算結果を格納するデータキャッシュ。

ソート済みポートでアグリゲータトランスフォーメーションを実行する場合、キャッシュメモリは使用されません。ソート済みポートを使用するアグリゲータトランスフォーメーションでは、キャッシュメモリを設定する必要はありません。

アグリゲータトランスフォーメーションの [ソート済み入力] オプション

[ソート済み入力] オプションを使用して、アグリゲータトランスフォーメーションのパフォーマンスを向上させることができます。

[ソート済み入力] オプションを使用すると、データ統合サービスではすべてのデータがグループ別にソートされているものと想定され、グループの行を読み込んでから集計計算が実行されます。必要に応じて、メモリにグループ情報が格納されます。ソート済み入力オプションを使用するには、ソート済みデータをアグリゲータトランスフォーメーションに渡す必要があります。ソート済み入力を使用する場合、アグリゲータトランスフォーメーションがソートされた出力を提供します。

[ソート済み入力] オプションを使用しない場合、データ統合サービスは読み込みと並行して集計計算を実行します。この場合、データはソートされていないため、すべての集計計算が正確に実行されるように、データ統合サービスはソース全体の読み込みが完了するまで各グループのデータを格納しておきます。

例えば、あるアグリゲータトランスフォーメーションで STORE_ID と ITEM の Group By ポートがあり、[ソート済み入力] オプションが選択されているとします。Aggregator を通して以下のデータを渡すと、データ統合サービスは 201/battery グループを検出した時点で、101/battery グループの 3 つの行を集計します。

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19
101	'battery'	2	2.59
201	'battery'	4	1.59
201	'battery'	1	1.99

[ソート済み入力] オプションを使用した場合、前もってデータを正しくソートしないと、データ統合サービスはマッピングの実行を失敗します。

[ソート済み入力] オプションの条件

特定の条件では、[ソート済み入力] オプションを使用できない場合があります。

以下のいずれかの条件が当てはまる場合には、[ソート済み入力] オプションを使用できません。

- 集計式が、ネストされた集計関数を含んでいる。

- ソースデータがデータドリブンである。

これらの条件のいずれかが満たされている場合、データ統合サービスはソート済み入力オプションが使用されていない場合と同様にトランスフォーメーションを処理します。

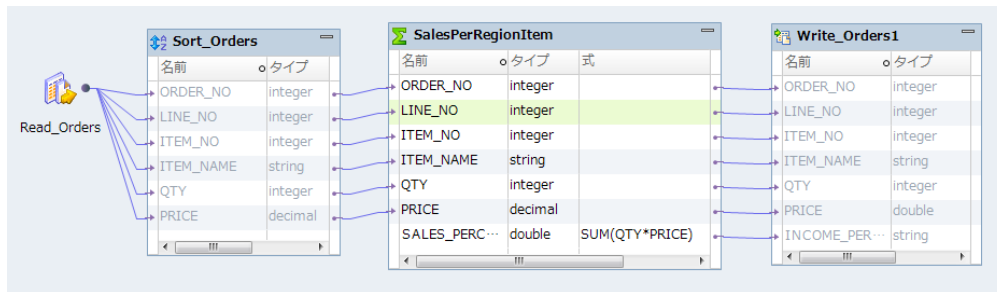
アグリゲータトランスフォーメーションでのデータのソート

[ソート済み入力] オプションを使用する場合は、ソート済みデータをアグリゲータトランスフォーメーションへ渡します。

データのソートは、アグリゲータの Group By ポートで、アグリゲータトランスフォーメーションに表示される順に行う必要があります。

リレーショナルファイル入力およびフラットファイル入力の場合、アグリゲータトランスフォーメーションにデータを渡す前に、ソータートランスフォーメーションを使用してマッピング内でデータをソートします。ソート済みデータの順序を変更するトランスフォーメーションが 1 つもない場合は、マッピング内のアグリゲータより前の任意の場所にソータートランスフォーメーションを配置できます。アグリゲータトランスフォーメーションの中の Group By カラムの順序は、ソータートランスフォーメーションと同じでなければなりません。

次のマッピングは、ソースデータを ITEM_NO で昇順にソートするよう構成されたソータートランスフォーメーションを示しています。



Sorter トランスフォーメーションは下記のようにデータをソートします。

ITEM_NO	ITEM_NAME	QTY	PRICE
345	Soup	4	2.95
345	Soup	1	2.95
345	Soup	2	3.25
546	Cereal	1	4.49
546	Cereal	2	5.25

[Sorted Input] オプションを使用した場合、Aggregator トランスフォーメーションは下記の結果を返します。

ITEM_NAME	QTY	PRICE	INCOME_PER_ITEM
Cereal	2	5.25	14.99
Soup	2	3.25	21.25

アグリゲータトランスフォーメーションの詳細プロパティ

データ統合サービスでアグリゲータトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

アグリゲータトランスフォーメーションの以下の詳細プロパティを設定します。

キャッシュディレクトリ

データ統合サービスがインデックスキャッシュファイルとデータキャッシュファイルを作成するディレクトリ。このディレクトリが存在し、キャッシュファイルを格納するのに十分なディスク容量を備えていることを確認します。

キャッシュのパーティション化時のパフォーマンスを向上させるには、セミコロンで区切って複数のディレクトリを入力します。キャッシュのパーティション化により、トランスフォーメーションを処理する各パーティションに個別のキャッシュが作成されます。

デフォルトは CacheDir システムパラメータです。このプロパティには、別のシステムパラメータまたはユーザー定義のパラメータを設定できます。

データキャッシュサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってデータキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

インデックスキャッシュサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってインデックスキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

ソート済み入力

入力データがグループで事前にソートされていることを示します。このオプションは、マッピングでアグリゲータトランスフォーメーションにソート済みデータが渡される場合にのみ選択してください。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

関連項目：

- [「キャッシュサイズ」 \(ページ 72\)](#)

再利用可能なアグリゲータトランスフォーメーションの作成

複数のマッピングまたはマプレットで使用する、再利用可能なアグリゲータトランスフォーメーションを作成します。

1. **[Object Explorer]** ビューで、プロジェクトまたはフォルダーを選択します。
2. **[ファイル]** > **[新規]** > **[トランスフォーメーション]** をクリックします。
[新規] ダイアログボックスが表示されます。
3. アグリゲータトランスフォーメーションを選択します。
4. **[次へ]** をクリックします。
5. トランスフォーメーションの名前を入力します。
6. **[完了]** をクリックします。
トランスフォーメーションがエディタに表示されます。
7. **[新規]** ボタンをクリックして、トランスフォーメーションにポートを追加します。
8. ポートを編集して、名前、データ型、および精度を設定します。
9. 各ポートのタイプ（入力、出力、パススルー、または変数）を決定します。
10. **[式]** フィールドをクリックして、出力ポートの集計式を設定します。また、ポートとパラメータを選択して集計式を定義することもできます。
11. **[詳細]** ビューをクリックし、トランスフォーメーションのプロパティを編集します。

再利用不可能なアグリゲータトランスフォーメーションの作成

マッピングまたはマプレットで再利用不可能なアグリゲータトランスフォーメーションを作成します。

1. マッピングまたはマプレットで、トランスフォーメーションパレットからエディタにアグリゲータトランスフォーメーションをドラッグします。
トランスフォーメーションがエディタに表示されます。
2. **[プロパティ]** ビューで、トランスフォーメーションの名前と説明を編集します。
3. **[ポート]** タブで、**[新規]** ボタンをクリックして、トランスフォーメーションにポートを追加します。
4. ポートを編集して、名前、データ型、および精度を設定します。
5. 各ポートのタイプ（入力、出力、パススルー、または変数）を決定します。
6. 出力ポートの集計式を設定します。
7. **[詳細]** ビューで、トランスフォーメーションのプロパティを編集します。

アグリゲータトランスフォーメーションに関するヒント

アグリゲータトランスフォーメーションをより効果的に使用するためのヒントを紹介します。

[ソート済み入力] オプションを使用し、集計キャッシュの使用量を減らしてください。

[ソート済み入力] オプションはマッピングの実行時にキャッシュに格納されるデータ量を減らし、パフォーマンスを向上させます。このオプションをソータートランスフォーメーションと共に使用して、ソート済みデータをアグリゲータトランスフォーメーションに渡します。

接続される入出力ポートまたは出力ポートを制限してください。

接続される入出力ポートまたは出力ポートの数を制限することにより、アグリゲータトランスフォーメーションがデータキャッシュに格納するデータ量を減らしてください。

集計に先立ってデータをフィルタリングします。

マッピングで Filter トランスフォーメーションを使用する場合は、そのあとにアグリゲータトランスフォーメーションを配置して、不要な集計を減らしてください。

ソートされたアグリゲータトランスフォーメーションのみがソートされた出力を提供します。

未ソートの入力を使用してソートされた出力を生成する場合は、アグリゲータトランスフォーメーションの後にソータートランスフォーメーションを使用します。

アグリゲータトランスフォーメーションのトラブルシューティング

アグリゲータトランスフォーメーションをトラブルシューティングすることができます。

[ソート済み入力] オプションを選択しましたが、マッピングの処理にかかる時間が前と変わりません。

以下のいずれかの条件が当てはまる場合には、[ソート済み入力] オプションを使用できません。

- 集計式が、ネストされた集計関数を含んでいる。
- ソースデータがデータドリブンである。

これらの条件のいずれかが満たされている場合、データ統合サービスはソート済み入力オプションが使用されていない場合と同様にトランスフォーメーションを処理します。

アグリゲータトランスフォーメーションを使用するとマッピングのパフォーマンスが低下します。

データ統合サービスは、ディスクへのページングを行う場合があります。トランスフォーメーションのプロパティでインデックスキャッシュとデータキャッシュのサイズを増やせば、パフォーマンスを向上させることができます。

非ネイティブ環境でのアグリゲータトランスフォーメーション

非ネイティブ環境でのアグリゲータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

Blaze エンジンでのアグリゲータトランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

アグリゲータトランスフォーメーションのデータキャッシュは、可変長を使用して、アグリゲータトランスフォーメーションを介して渡されるバイナリデータ型と文字列データ型を格納するように最適化されます。この最適化は、8 MB までのレコードに対して有効化されます。レコードサイズが 8 MB を超える場合、可変長の最適化は無効になります。

アグリゲータトランスフォーメーションを介して渡されるデータのデータキャッシュ内の格納に可変長が使用される場合、アグリゲータトランスフォーメーションは、ランタイムマッピング内のアグリゲータトランスフォーメーションの前に挿入されるソートされた入力とパススルーのソータートランスフォーメーションを使用するように最適化されます。

ソータートランスフォーメーションを表示するには、最適化されたマッピングを表示するか、Blaze 検証環境内の実行プランを表示します。

データキャッシュの最適化中は、アグリゲータトランスフォーメーション用のそのデータキャッシュとインデックスキャッシュは「自動」に設定されます。ソータートランスフォーメーションのソーターキャッシュは、アグリゲータトランスフォーメーションのデータキャッシュと同じサイズに設定されます。ソーターキャッシュを設定するには、アグリゲータトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

Spark エンジンでのアグリゲータトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

ストリーミングマッピングでのアグリゲータトランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

マッピングの検査

マッピング検証は、次の場合に失敗します。

- ストリーミングパイプラインに複数のアグリゲータトランスフォーメーションが含まれている。
- ストリーミングパイプラインにアグリゲータトランスフォーメーションとランクトランスフォーメーションが含まれている。
- アグリゲータトランスフォーメーションがルックアップトランスフォーメーションからのアップストリームである。
- アグリゲータトランスフォーメーションが、不等式ルックアップ条件を使用して設定されたルックアップトランスフォーメーションと同じストリーミングパイプライン内にある。

Databricks Spark エンジンでのアグリゲータトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

集計関数

アグリゲータトランスフォーメーションの式でポートを使用し、集計関数内でポートを使用しないと、ランタイムエンジンはポート内の任意の行を使用して式を処理することがあります。

ランタイムエンジンは、ポートの最後の行以外の行を使用することがあります。処理が分散されているために、ランタイムエンジンがポートの最後の行を特定できないことがあるからです。

データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

第 6 章

関連付けトランスフォーメーション

この章では、以下の項目について説明します。

- [関連付けトランスフォーメーションの概要, 146 ページ](#)
- [メモリ割り当て, 147 ページ](#)
- [関連付けトランスフォーメーションの詳細プロパティ, 148 ページ](#)

関連付けトランスフォーメーションの概要

関連付けトランスフォーメーションは、一致トランスフォーメーションの出力データを処理します。異なる一致クラスに割り当てられた重複レコードの間のリンクを作成し、それらのレコードをデータ統合やマスターデータ管理の操作で関連付けることができるようにします。

関連付けトランスフォーメーションは、関連レコードのグループ内の行ごとに **AssociationID** 値を生成し、その ID 値を出力ポートに書き込みます。

統合トランスフォーメーションは、関連付けトランスフォーメーションの出力を読み取ります。統合トランスフォーメーションを使用して、共通の関連付け ID 値を持つレコードに基づいてマスターレコードを作成します。

関連付けトランスフォーメーションは、入力ポートのデータ値として文字列と数値を受け入れます。別のデータ型の入力ポートを追加した場合、トランスフォーメーションによってポートのデータ値が文字列に変換されます。

AssociationID 出力ポートは、整数データを書き込みます。以前のバージョンの Informatica Data Quality で設定されたトランスフォーメーションは、AssociationID ポートに文字列データを書き込むことができます。

例: 一致トランスフォーメーションの出力の関連付け

次の表に、同一であると見なされる可能性がある 3 つのレコードを示します。

ID	名前	住所	市区町村	州	郵便番号	SSN
1	David Jones	100 Admiral Ave.	New York	NY	10547	987-65-4321
2	Dennis Jones	1000 Alberta Ave.	New Jersey	NY	-	987-65-4321
3	D. Jones	Admiral Ave.	New York	NY	10547-1521	-

一致トランスフォーメーションで定義された重複分析操作では、以下の理由で、これらの3つのレコードはどれも相互に重複するとは見なされません。

- 名前および住所のデータに対する重複検索を定義した場合、レコード1とレコード3は重複と見なされますが、レコード2は除外されます。
- 名前および社会保障番号のデータに対する重複検索を定義した場合、レコード1とレコード2は重複と見なされますが、レコード3は除外されます。
- 3つのすべての属性（名前、住所、および社会保障番号）に対する重複検索を定義した場合、いずれのレコードも一致とは見なされません。

関連付けトランスフォーメーションは、異なる一致クラスタのデータをリンクし、クラスタIDが同じレコードに共通の AssociationID 値が割り当てられるようにします。この例の場合、次の表に示すように、3つのすべてのレコードに同じ AssociationID が割り当てられます。

ID	名前	住所	市区町村	州	郵便番号	SSN	名前と住所のクラスタID	名前とSSNのクラスタID	関連付けID
1	David Jones	100 Admiral Ave.	New York	NY	10547	987-65-4320	1	1	1
2	Dennis Jones	1000 Alberta Ave.	New Jersey	NY	-	987-65-4320	2	1	1
3	D. Jones	Alberta Ave.	New York	NY	10547-1521	-	1	2	1

重複レコードのデータは、統合トランスフォーメーションで統合することができます。

メモリ割り当て

関連付けトランスフォーメーションで使用される最小限のキャッシュメモリ量を設定できます。デフォルトの設定は 400,000 バイトです。

この値は、[詳細] タブの [キャッシュファイルサイズ] プロパティで設定します。

デフォルト値は、トランスフォーメーションで使用される最小限のメモリ量を表します。関連付けトランスフォーメーションは、関連付けるポートの数に基づいて、デフォルト値の倍数を取得しようとします。取得するキャッシュメモリは、次の式で計算されます。

(関連付けポートの数 + 1) x デフォルトのキャッシュメモリ

例えば、関連付けポートを 7 つ設定すると、トランスフォーメーションは 320 万バイト (3.05MB) をキャッシュメモリに割り当てようとします。

デフォルトの設定を変更した場合、トランスフォーメーションで追加のメモリの取得は試行されません。

注: 入力したキャッシュメモリの値が 65536 を下回る場合、関連付けトランスフォーメーションでは値が MB 単位で読み取られます。

関連付けトランスフォーメーションの詳細プロパティ

関連付けトランスフォーメーションには、キャッシュメモリの動作とトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

キャッシュファイルディレクトリ

データ統合サービスが現在のトランスフォーメーションの一時データを書き込むディレクトリを指定します。入力データの量が利用可能なシステムメモリより大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

キャッシュファイルサイズ

トランスフォーメーションの入力データをソートするためにデータ統合サービスが使用するシステムメモリの量を決定します。デフォルト値は 400,000 バイトです。このパラメータを使用して、キャッシュファイルサイズを指定できます。

データをソートする前に、データ統合サービスは指定されているメモリ量を割り当てます。ソート操作によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュファイルディレクトリに書き込みます。ソート操作にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

注: 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

第 7 章

不良レコードの例外トランスフォーメーション

この章では、以下の項目について説明します。

- [不良レコードの例外トランスフォーメーションの概要, 149 ページ](#)
- [不良レコードの例外の出力レコードタイプ, 150 ページ](#)
- [不良レコードの例外管理プロセスフロー, 151 ページ](#)
- [不良レコードの例外マッピング, 151 ページ](#)
- [不良レコードの例外のポート, 154 ページ](#)
- [不良レコードの例外の〔設定〕ビュー, 155 ページ](#)
- [不良レコードの例外の問題の割り当て, 157 ページ](#)
- [例外トランスフォーメーションの詳細プロパティ, 158 ページ](#)
- [不良レコードの例外トランスフォーメーションの設定, 158 ページ](#)
- [不良レコードの例外マッピングの例, 159 ページ](#)

不良レコードの例外トランスフォーメーションの概要

不良レコードの例外トランスフォーメーションは、データ品質プロセスの出力を読み取り、手動確認が必要なレコードを識別するアクティブなトランスフォーメーションです。不良レコードの例外トランスフォーメーションは、複数グループトランスフォーメーションです。

レコード内のデータ品質の問題を識別するプロセスの出力を分析するように、不良レコードの例外トランスフォーメーションを設定します。データ品質の問題があり、さらに確認が必要なレコードは例外と呼ばれます。

不良レコード例外トランスフォーメーションでは、別のトランスフォーメーションや別のマッピング内のデータオブジェクトから入力を受け取ります。不良レコードトランスフォーメーションへの入力には、データ品質の問題のテキスト説明を受け取る 1 つまたは複数の品質の問題ポートが含まれている必要があります。不良レコードの例外トランスフォーメーションへの入力には、各レコードのデータ品質を決定するために使用できる数値レコードスコアを含めることもできます。例外トランスフォーメーションで上限スコアおよび下限スコアのしきい値を設定して、レコードスコアに基づいて、レコードを正常品質および不良品質として分類します。不良レコードの例外トランスフォーメーションでは、例外および関連付けられた品質の問題のテキストが不良レコードテーブルに書き込まれます。

たとえば、組織が郵便物を顧客に郵送する前に、顧客の住所を検証する必要があるとします。ラベラトランスフォーメーションで顧客の市町村、都道府県、郵便番号を参照テーブルに照らして検証するマッピングを作成

します。ラベラトランスフォーメーションでは、フィールドを検証し、結果に基づいてレコードスコアを各レコードに追加します。また、ラベラトランスフォーメーションでは、エラーのある各レコードの品質の問題を説明するテキストも追加します。ラベラトランスフォーメーションでは、市町村が無効や郵便番号コードが空白などの問題のテキストを各例外に追加します。不良レコードの例外トランスフォーメーションでは、手動確認が必要な顧客レコードが不良レコードテーブルに書き込まれます。データアナリストは、Analyst ツールで不良レコードの確認と修正を行うことができます。

不良レコードの例外の出力レコードタイプ

不良レコードの例外では、入力レコードスコアを調べて、レコードの品質を判別します。その後、レコードを複数の出力グループに返します。

例外トランスフォーメーションは、各レコードスコアに基づいて、次のレコードタイプを識別します。

正常レコード

スコアが上限しきい値以上のレコードです。正常レコードは有効で、確認は不要です。たとえば、上限しきい値を 90 に設定する場合、スコアが 90 以上のレコードには確認が不要です。

不良レコード

スコアが上限しきい値未満で、下限しきい値以上のレコードです。不良レコードは、Analyst ツールで確認する必要がある例外です。たとえば、下限しきい値が 40 の場合、スコアが 40～90 のレコードには手動確認が必要です。

却下されたレコード

スコアが下限しきい値未満のレコード。却下されたレコードは無効です。デフォルトでは、例外トランスフォーメーションは却下されたレコードをデータフローから削除します。この例では、スコアが 40 以下のレコードが却下されたレコードです。

注: 品質の問題フィールドが NULL の場合、レコードは例外ではありません。品質の問題にテキストまたは空の文字列が含まれるとき、レコードは例外です。フィールドにエラーがない場合、品質の問題ポートに NULL 値が含まれることを確認します。品質の問題ポートに NULL 値の代わりに空白が含まれる場合、例外トランスフォーメーションによりすべてのレコードに例外フラグが設定されます。Analyst ツールで問題を修正する必要があるとき、データの品質の問題で例外をフィルタ処理することはできません。

レコードのスコアが 0 未満または 100 を上回る場合、その行は有効ではありません。データ統合サービスは行が有効ではなく、レコードの処理がスキップされるというエラーメッセージを記録します。

レコードスコアを入力として例外トランスフォーメーションに接続しなかった場合、トランスフォーメーションは品質の問題が含まれるすべてのレコードを不良レコードテーブルに書き込みます。

マッピングタスクに不良レコードの例外トランスフォーメーションを含める場合、同じワークフローでヒューマンタスクを設定し、例外の手動確認を含めることができます。ヒューマンタスクはワークフロー内のマッピングタスクの終了時に開始します。ヒューマンタスクでは、Analyst ツールにアクセスし、品質の問題を解決する必要があります。ユーザーはデータを更新したり、不良レコードテーブル内の各レコードの品質のステータスを変更したりできます。

不良レコードの例外管理プロセスフロー

例外トランスフォーメーションはデータ品質トランスフォーメーションからレコードスコアを受信し、さまざまなレベルのデータ品質のレコードを含むテーブルを作成します。品質の問題を検出し、各行にレコードスコアを提供するようにデータ品質トランスフォーメーションを設定する必要があります。

複数のデータ品質トランスフォーメーションを単一のマッピングで設定することも、データ品質プロセス内のステージ別にマッピングを作成することもできます。

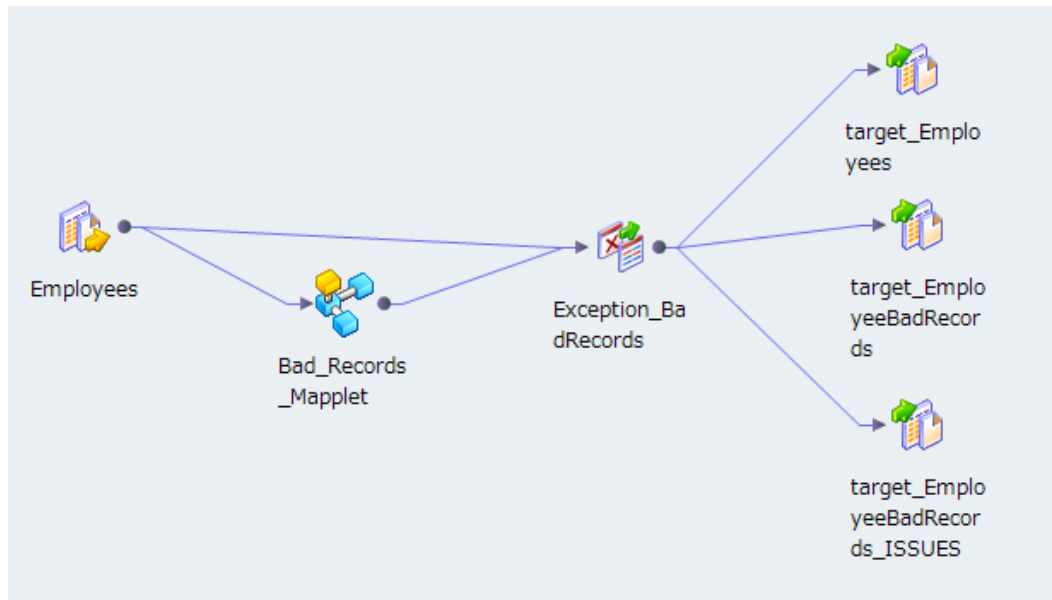
以下の不良レコードの例外管理タスクを実行します。

1. Developer ツールで、定義したデータ品質の問題に基づいてソースデータのスコア値を生成するトランスフォーメーションを定義します。ソースデータの品質を説明するテキストを返すトランスフォーメーションを定義します。たとえば、ラベラトランスフォーメーションを設定することで、ソースデータを参照テーブルに照らして確認し、比較ごとに説明ラベルを書き込むことができます。ディシジョントランスフォーメーションでデータフィールドを調べるために IF/THEN ルールを定義できます。複数のトランスフォーメーションと、複数のデータ品質操作を実行するマプレットを定義できます。
2. データ品質操作から受け取るレコードを分析するように例外トランスフォーメーションを設定します。スコア値に基づいてレコードをデータベーステーブルに書き込むには、このトランスフォーメーションを設定します。正常レコード、不良レコード、品質の問題、却下されたレコードにそれぞれテーブルを作成できます。
3. 品質の問題ポートを、不良データが含まれている可能性がある各入力ポートに割り当てます。
4. 必要に応じて、正常レコードと不良レコードのターゲットデータオブジェクトを設定します。例外トランスフォーメーション出力ポートをマッピングのターゲットデータオブジェクトに接続します。
5. 不良レコードのターゲットデータオブジェクトの作成します。不良レコードテーブルを生成し、マッピングに追加するように選択します。不良レコードテーブルの生成時に、Developer ツールは品質の問題テーブルも生成します。品質の問題テーブルをマッピングに追加します。
6. マッピングをワークフローに追加します。
7. 不良レコードの手動確認をユーザーに割り当てるようにヒューマンタスクを設定します。ユーザーは、Analyst ツールで不良レコードの確認と更新を行うことができます。

不良レコードの例外マッピング

不良レコードの例外を特定するマッピングを作成するときは、レコードのデータの品質に基づいて、1 つ以上のデータベースターゲットにレコードを書き込むようにマッピングを設定します。

下の図に、不良レコードの例外マッピング例を示します。



マッピングには次のオブジェクトが含まれます。

データソース

データ品質を分析するためのレコードが含まれている Employees データソース。

マップレット

Bad_Records_Mapplet には、品質の問題およびレコードスコアを確認し、ソースレコードに追加するトランスフォーメーションが含まれています。ルールは、データを分析し、品質の問題を見つけるトランスフォーメーションです。たとえば、ラベラトランスフォーメーションを含めて、入力データを参照テーブルと比較することができます。結果に応じて、品質の問題を列の追加カラムとして返すようにラベラトランスフォーメーションを構成できます。IF、THEN、ELSE の各ステートメントを使用して、データを調べて、品質の問題およびレコードスコアを入力データに適用するようにディシジョントランスフォーメーションを構成できます。

例外トランスフォーメーション

例外トランスフォーメーションは、不良レコードテーブルや問題テーブルなどのデータターゲットに書き込むレコードを判別します。

高品質レコードテーブル

例外トランスフォーメーションは、すべての高品質レコードを target_Employees テーブルに書き込みます。

不良レコードテーブル

例外トランスフォーメーションは、すべての低品質レコードを target_EmployeeBadRecords テーブルに書き込みます。不良レコードには手動確認が必要です。

問題テーブル

例外トランスフォーメーションは、品質の問題を target_EmployeeBadRecords_ISSUES テーブルに書き込みます。Analyst ツールで不良レコードを表示すると、ユーザーインターフェースは品質の問題を不良レコードにリンクします。

必要に応じて、例外トランスフォーメーションで却下されたレコードを却下されたレコードテーブルに書き込むことができます。トランスフォーメーションの**設定**ビューで却下されたレコード用の独立した出力グループを作成するように選択する必要があります。

不良レコードの例外の品質に関する問題

品質の問題は、低いレコードスコアの原因となったデータ品質の問題のタイプを説明するテキスト文字列です。不良レコードの例外トランスフォーメーションは、低いレコードスコアが含まれる各ソース行に関連付けられた品質の問題を受信します。品質の問題およびレコードスコアを判別する複数のタイプのトランスフォーメーションを設定できます。

たとえば、電話番号を判別するディシジョントランスフォーメーションを作成できます。ディシジョントランスフォーメーションは電話番号のレコードスコアおよび品質の問題を生成します。

以下のディシジョンストラテジでは、ディシジョントランスフォーメーションで不正な長さとした電話番号が特定されます。

```
IF LENGTH(Phone_Number) > 10 THEN
  Score:=50
  Phone_Quality_Issue:='Phone num too long'
ELSEIF LENGTH(Phone_Number) < 10 THEN
  Score:=50
  Phone_Quality_Issue:=' Phone num too short'
ELSE
  Score:=90
ENDIF
```

例外トランスフォーメーションを設定する場合、Phone_Quality_Issue を電話番号ポートに関連付ける必要があります。このポートは複数の入力グループで構成されています。

例外トランスフォーメーションレコードは、ディシジョントランスフォーメーションによって生成されたスコアを読み取り、スコアが「50」のレコードを出力ポートの不良レコードグループに割り当てます。これによって、Phone_Quality_Issue が出力ポートの問題グループに書き込まれます。

ヒューマンタスク

例外トランスフォーメーションが含まれるワークフローを設定する場合、マッピングタスクにマッピングを含めます。ヒューマンタスクを同じワークフローに追加します。ヒューマンタスクは、Analyst ツール内の例外レコードを修正するために 1 人以上のユーザーを必要とします。

マッピングタスクでは、データ品質の問題が解決されていないソースデータ内のレコードが特定されます。データアナリストは、Analyst ツールを使用して問題の解決や各レコードのデータ品質ステータスの更新を行います。

ヒューマンタスクを設定するときは、タスクインスタンスとタスクステップをそれぞれ 1 つ以上作成します。タスクインスタンスは、ユーザーが作業する必要があるデータセットです。タスクステップは、ユーザーがタスクインスタンスのレコードに対して実行する作業のタイプを表します。Analyst ツールで複数のユーザーがデータの複数の部分に対して作業できるように、複数のタスクインスタンスを作成できます。

ユーザーは、以下のいずれかの方法で Analyst ツールで不良レコードのステータスを更新できます。

- レコードが有効な場合は、レコードのデータベースへの永続的な格納を確定するために、テーブルのメタデータを更新します。
- レコードが無効な場合は、ワークフローの以降のステージでレコードをデータベースから削除するために、テーブルのメタデータを更新します。
- レコードのステータスが確定されていない場合は、マッピングタスクでの将来の処理のために、レコードがワークフローに戻ります。

ヒューマンタスクの詳細については、『*Informatica Developer ワークフローガイド*』を参照してください。

不良レコードの例外のポート

不良レコードの例外トランスフォーメーションの【ポート】タブで入力ポートと出力ポートを設定します。
不良レコードの例外トランスフォーメーションには入力ポートグループと出力ポートグループがあります。
以下の図は、【ポート】タブを示しています。

出力								
	名前	タイプ	精度	スケ...	入力	出力	デフォルト	説明
入力								
データ (6)								
1	CUST_ID	deci...	8	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	COMPANY	string	49	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	CONTACT	string	19	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	TITLE	string	35	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	ADDR1	string	47	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
6	ADDR2	string	47	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
品質の問題 (1)								
1	ADDR3	string	42	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
制御 (1)								
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
出力								
標準出力 (7)								
1	Score	double	15	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	CUST_ID	deci...	8	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	COMPANY	string	49	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	CONTACT	string	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
5	TITLE	string	35	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
6	ADDR1	string	47	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
7	ADDR2	string	47	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
不良レコード (9)								
1	Workflow...	string	64	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Row_Iden...	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

不良レコードの例外トランスフォーメーションの入力ポート

不良レコードの例外トランスフォーメーションには、データ、品質の問題、およびレコードスコアにそれぞれ入力グループがあります。

不良レコードの例外トランスフォーメーションには次の入力グループがあります。

データ

ソースデータフィールドです。

品質の問題

レコードの品質の問題を説明するポートが含まれます。品質の問題ポートには、「Excess_Characters」や「Bad_Data_Format」などの文字列が含まれます。各レコードに複数の品質の問題を含めることができます。トランスフォーメーションでは、【問題の割り当て】ビューで問題をデータポートに割り当てるまで、品質の問題グループのポートがソースデータフィールドに関連付けられません。

コントロール

レコードスコアです。例外トランスフォーメーションは、レコードスコアを分析して、入力行が例外であるかどうかを判断します。スコアポートに接続していない場合、品質の問題ポートにデータが含まれていなければ、例外トランスフォーメーションは行を例外として特定します。

不良レコードの例外トランスフォーメーションの出力

不良レコードの例外トランスフォーメーションには複数の出力グループがあります。

不良レコードの例外トランスフォーメーションには次の出力グループがあります。

標準出力

データ品質の問題がないか調べる必要のない正常なレコードです。

標準出力グループの各レコードには、レコードのデータ品質を表すスコアポートが含まれています。

不良レコード

データ品質の問題がないか調べる必要がある例外です。

不良レコードグループの各レコードには、ワークフロー ID、行識別子、およびレコードステータスポートが含まれています。

問題

不良レコードグループのレコードの品質の問題です。品質の問題は、不良レコードを確認するときに Analyst ツールによって表示されるメタデータ要素です。

問題グループの各レコードには、問題が存在している不良レコード行を特定するワークフロー ID および行識別子ポートが含まれています。

却下されたレコード

データベースから削除できるレコードが含まれるオプションのグループです。拒否されたレコードグループの各レコードには、スコアポート内のレコードスコアが含まれています。

不良レコードの例外の [設定] ビュー

[設定] ビューでは、トランスフォーメーションが正常レコードと不良レコードを識別するために使用する上限と下限のしきい値を指定します。**[設定]** ビューでは、しきい値以上またはしきい値以下のレコードのスコアと共にレコードのターゲットテーブルも指定します。

以下の図に、例外トランスフォーメーションの **[設定]** ビューを示します。

手動レビューのしきい値

下限しきい値:
 上限しきい値:

データルーティングのオプション

タイプ	標準出力	重複レコードテーブル
自動統合（上限しきい値を超える）	<input checked="" type="checkbox"/>	<input type="checkbox"/>
手動統合（しきい値間）	<input type="checkbox"/>	<input checked="" type="checkbox"/>
一意のレコード（下限しきい値未満）	<input type="checkbox"/>	<input type="checkbox"/>

☐ 一意のレコードのための独立した出力グループを作成

【設定】 ビューでは、次のプロパティを設定できます。

下限しきい値

不良レコードのスコア範囲の下限です。 トランスフォーメーションは、スコアが下限のしきい値未満のレコードを却下されたレコードとして識別します。

上限しきい値

不良レコードのスコア範囲の上限です。 トランスフォーメーションは、スコアが上限のしきい値以上のレコードを正常なレコードとして識別します。

データルーティングのオプション

出力レコードのタイプ。デフォルト設定では、トランスフォーメーションは、正常レコードを標準出力に書き込み、不良レコードを不良レコードテーブルに書き込みます。デフォルトでは、トランスフォーメーションは拒否されたレコードをデータベーステーブルに書き込みません。

標準出力

標準出力ポートに書き込まれるレコードのタイプです。デフォルトは「正常レコード」です。

不良レコードテーブル

不良レコード出力ポートに書き込まれるレコードのタイプです。デフォルトは「不良レコード」です。

却下されたレコードのための独立した出力グループを作成

却下されたレコード用の独立した出力グループを作成します。このオプションは、デフォルトでクリアされています。

不良レコードテーブルの生成

不良レコードデータを含めるデータベーステーブルを作成します。このオプションを選択すると、例外トランスフォーメーションによってデータベーステーブルが作成され、データベースオブジェクトがモデルリポジトリに追加され、オブジェクトのインスタンスがマッピングキャンバスに追加されます。マッピングで例外トランスフォーメーションインスタンスの不良レコードテーブルを生成できます。Developer ツールは、不良レコードテーブルを生成する際に、レコードに関する説明メタデータを格納する問題テーブルも作成します。

注: Developer ツールは、不良レコードテーブルの各カラム名に 12 文字のサフィックスを追加します。Oracle データベースを使用する場合、ソースカラム名は 18 文字以内にする必要があります。

不良レコードテーブルおよび問題テーブルの生成

トランスフォーメーションをマッピングに追加すると、不良レコードテーブルと問題テーブルを生成できます。Developer ツールは、テーブルをモデルリポジトリに追加します。

1. **【不良レコードテーブルの生成】** をクリックして、テーブルを生成します。
【リレーショナルデータオブジェクトの作成】 ダイアログボックスが表示されます。
2. データベース接続を参照します。テーブルを含むデータベースへの接続を選択します。
3. 不良レコードテーブルの名前を入力します。Developer ツールは、入力した名前を不良レコードテーブルと問題テーブルに適用します。

Developer ツールは、次の文字列を問題テーブル名に追加します。

_ISSUE

Oracle データベースに接続する場合、不良レコードテーブル名は 24 文字以内にする必要があります。

4. モデルリポジトリ内の不良レコードのデータオブジェクト名を入力します。
5. **【完了】** をクリックします。
Developer ツールは、マッピングキャンバスとモデルリポジトリにテーブルを追加します。

不良レコードの例外の問題の割り当て

データ品質の問題にポートおよび優先順位を割り当てる必要があります。

以下の図に、**【問題の割り当て】** ビューを示します。

問題の割り当て		
品質の問題に対するポートと優先順位の割り当て		
品質の問題	入力	問題...
EmployeeID_Quality_Issue	EmployeeID	1
Name_Quality_Issue	Employee_Name	1
Phone_Quality_Issue		1

【問題の割り当て】 ビューには以下のフィールドがあります。

品質の問題

品質の問題入力グループで定義した各品質の問題ポートは、**品質の問題** カラムに表示されます。

入力

入力 カラムには、**【問題の割り当て】** ビューで品質の問題に割り当てたデータポートが含まれています。各品質の問題ポートに☐ 入力ポートを関連付けます。不良品質データが含まれる各入力ポートには、問題のタイプを示す、対応する問題の品質ポートが少なくとも 1 つは必要です。たとえば、Phone_Quality_Issue に対して Phone_number ポートを選択できます。複数の品質の問題にポートを選択できます。

問題の優先順位

問題の優先順位によって、同じ入力ポートを複数の品質の問題に割り当てる場合に最も重要な品質の問題を決定します。1 つの入力フィールドに対して複数の品質の問題が発生する場合、データ統合サービスは優先順位が最も高い問題を適用します。1 つの入力ポートに対して複数の品質の問題が存在し、問題の優

先順位が同じ場合、データ統合サービスはリストの一番上にある品質の問題を適用します。1～99 の優先度を入力します。1 は最高の優先順位を表します。

Analyst ツールでレコードをフィルタリングするための、問題の優先順位を設定します。

品質の問題に対するポートの割り当て

品質の問題のそれぞれにポートを割り当て、関連付けます。Developer ツールによって、**【問題の割り当て】**ビューで追加する各関連付けのために出力グループにポートが作成されます。

1. 各品質の問題の**【入力】**フィールドをクリックすると、入力ポートのリストが表示されます。
2. 品質の問題に関連付ける入力ポートを選択します。
複数の問題に同じポートを選択できます。
3. **【問題】** カラムをクリックし、品質の問題の優先順位を選択します。

例外トランスフォーメーションの詳細プロパティ

データ統合サービスで例外トランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

不良レコードの例外トランスフォーメーションの設定

不良レコードの例外トランスフォーメーションを設定する場合、各ポートで発生する可能性がある入力ポートおよび品質の問題を設定します。データの品質を特定するための上限および下限のしきい値を定義します。例外および却下されたレコードを書き込むかどうかを設定します。

1. 再利用可能な、または再利用不可能な不良レコードの例外トランスフォーメーションを作成します。
 - 再利用可能なトランスフォーメーションを作成するには、**【ファイル】** > **【新規】** > **【トランスフォーメーション】** を選択して、不良レコードの例外トランスフォーメーションを選択します。
 - 再利用不可能なトランスフォーメーションを作成するには、マッピングを開き、例外トランスフォーメーションをマッピングキャンバスに追加します。ウィザードで、不良レコードの例外トランスフォーメーションを選択します。

2. **【次へ】** をクリックするか、**【完了】** をクリックします。

【次へ】 をクリックすると、トランスフォーメーションを作成する前にデフォルトのしきい値とデータルーティングのオプションを更新できます。

3. 入力ポートを設定します。
 - 再利用可能なトランスフォーメーションを作成する場合は、**【ポート】** タブを選択し、トランスフォーメーションに接続するデータのポートを追加します。
 - 再利用不可能なトランスフォーメーションを作成する場合は、他のオブジェクトをマッピングキャンバスに追加し、入力ポートをトランスフォーメーションにドラッグします。
4. **【設定】** ビューを選択します。
5. スコアの上限および下限のしきい値を設定します。
6. **【データルーティングのオプション】** セクションで、標準出力と例外テーブルのプロパティを設定して、トランスフォーメーションが各レコードタイプを書き込む場所を設定します。

正常レコード、不良レコード、および却下されたレコードを書き込むかどうかを設定します。標準出力または不良レコードテーブルに書き込むことができます。
7. **【問題の割り当て】** ビューを開きます。データ品質の問題をデータポートに割り当てます。

問題ごとに優先順位を割り当てます。ポートに複数の問題の値が含まれている場合は、優先順位の一番高い問題が表示されます。
8. 不良レコードテーブルを生成するオプションを選択します。データベース接続とテーブル名の情報を入力します。このテーブルはデフォルトのスキーマから取得される必要があります。
 - 不良レコードテーブルを生成するとき、レコードのテーブルとレコードに関連するデータ品質の問題の追加テーブルを生成します。モデルリポジトリにデータベースオブジェクトが作成されます。
9. トランスフォーメーション出力ポートを 1 つ以上のデータターゲットに接続します。出力ポートを **【設定】** ビューで設定した出力オプションに対応するデータオブジェクトに接続します。
 - 再利用可能なトランスフォーメーションを作成する場合は、トランスフォーメーションをマッピングに追加し、出力ポートを接続します。
 - 再利用不可能なトランスフォーメーションを作成する場合は、トランスフォーメーションにより、ポートが不良レコードテーブルに接続されます。出力ポートを他のいずれかのデータターゲットに接続します。

不良レコードの例外マッピングの例

組織が新しい顧客データを確認するデータプロジェクトを実施します。組織は顧客の連絡先データが有効であることを確認する必要があります。次の例は、顧客レコードのデータ品質の分析を行うマプレットからレコードを受け取る不良レコードの例外トランスフォーメーションを定義する方法を示します。

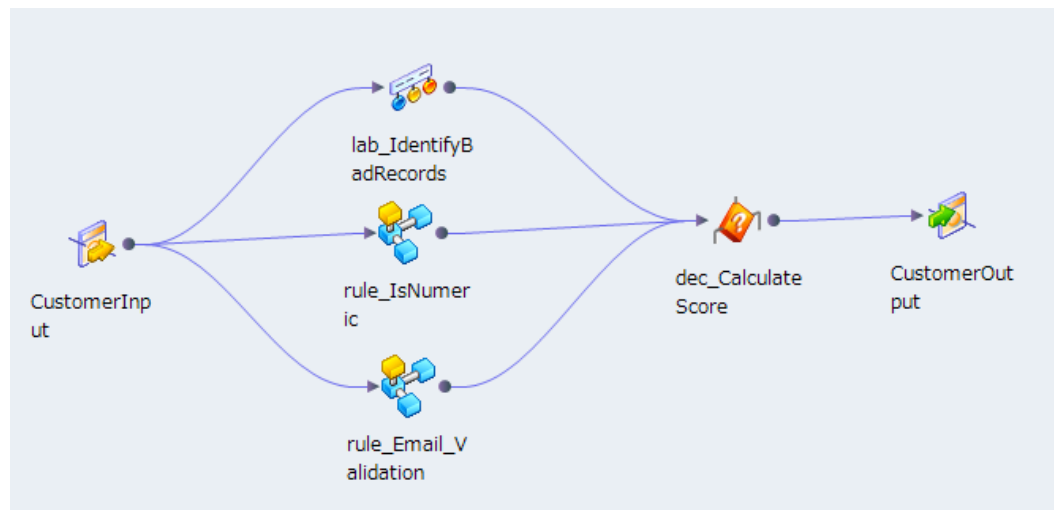
顧客データの形式や精度を評価するデータ品質トランスフォーメーションを含むマプレットを作成します。マプレットには、データ品質の分析結果に基づいてレコードスコアを生成するトランスフォーメーションが含まれています。また、トランスフォーメーションは、分析結果に基づいてデータの品質の問題も定義します。

不良レコードの例外のマプレット

データ品質トランスフォーメーションが含まれるマプレットを作成して、特定のフィールドの値を確認します。このトランスフォーメーションは参照テーブルとコンテンツセットを確認して、レコード内のフィールドが有効であるかどうかを判断します。このトランスフォーメーションは結果に基づいて、レコードスコアを各レコードに適用します。例外トランスフォーメーションはマプレットからレコードを受け取り、レコードスコアに基づいて、各レコードを適切な出力にルーティングします。

マプレットはラベラートランスフォーメーション、ディシジョントランスフォーメーション、および式トランスフォーメーションで構成されています。

以下の図に、マプレット内のオブジェクトを示します。



マプレットは、以下のタスクを実行します。

- ラベラトランスフォーメーションは入力ポートで受け取る状態、国コード、郵便番号などのデータをローカルで検証します。このトランスフォーメーションには、各ポートのストラテジが含まれています。ストラテジは、ソースデータを参照テーブルと比較し、有効でない値を識別します。
- 式トランスフォーメーションマプレットは電話番号が数値であることを確認し、10桁が含まれる数値であることを検証します。
- ラベラトランスフォーメーションおよび式トランスフォーメーションマプレットは、電子メールアドレスが有効であることを確認します。式トランスフォーメーションは電子メール文字列の構造を検証します。ラベラトランスフォーメーションはIPアドレスを国際IPアドレスサフィックスの参照テーブルに照らして確認します。
- ディジジョントランスフォーメーションはトランスフォーメーションおよびマプレットから出力を受信します。これは顧客の連絡先レコード全体のレコードスコアを計算します。

マプレットが含まれる不良レコードの例外マッピングを作成します。不良レコードの例外マッピングには、例外を不良レコードのデータベーステーブルに書き込む例外トランスフォーメーションが含まれています。データアナリストは、Analyst ツールを使用し、不良レコードテーブル内の例外レコードを調べたり更新したりします。

不良レコードの例外の入力グループの例

例外トランスフォーメーションには3つの入力グループがあります。トランスフォーメーションには、ソースデータを受け取るデータグループがあります。品質の問題グループは、データ品質トランスフォーメーションによって見つかったデータ品質の問題を受け取ります。また、行のレコードスコアが含まれるコントロールグループもあります。

以下の図に、例外トランスフォーメーションの入力グループを示します。

	名前	タイプ	精度	スケ...	入力	出力
	▢ 入力					
	▢ データ (6)					
1	CUST_ID	decimal	8	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	COMPANY	string	49	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	CONTACT	string	19	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	TITLE	string	35	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	ADDR1	string	47	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	ADDR2	string	47	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	▢ 品質の問題 (1)					
1	ADDR3	string	42	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	▢ 制御 (1)					
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>

不良レコードの例外の設定例

【設定】ビューで上限および下限のしきい値を定義します。トランスフォーメーションで正常レコード、不良レコード、および却下されたレコードが書き込まれる場所を特定します。

正常レコード、不良レコード、および問題のルーティングのデフォルト設定を受け入れます。

以下の図に、例外トランスフォーメーションの【設定】ビューを示します。

手動レビューのしきい値

下限しきい値:

上限しきい値:

データルーティングのオプション

タイプ	標準出力	重複レコードテーブル
自動統合 (上限しきい値を超える)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
手動統合 (しきい値間)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
一意のレコード (下限しきい値未満)	<input type="checkbox"/>	<input type="checkbox"/>

☐ 一意のレコードのための独立した出力グループを作成

以下の表に、構成設定を示します。

オプション	設定項目
下限しきい値	10
上限しきい値	90
正常レコード	標準出力
不良レコード	不良レコードテーブル
却下されたレコード	-

【不良レコードテーブルの生成】をクリックし、不良レコードテーブルと問題テーブルを作成します。

不良レコードの例外のマッピングの出力例

マッピングに書き込みトランスフォーメーションを追加し、標準出力ポートをそのデータオブジェクトに接続します。マッピングには【設定】ビューで作成した不良レコードのデータベースオブジェクトと問題のデータベースオブジェクトも含まれます。

不良レコードテーブル

不良レコードテーブルには、レコードスコアが下限しきい値と上限しきい値の間の例外が含まれています。

以下の図に、例外トランスフォーメーションで返される不良レコードを示します。

出力									
名前: exc_BadRecord.Bad_Output_DI									
	Workflow_ID	Row_Identif...	Record_Stat...	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2
1	DummyWor...	0	INVALID	<NULL>	1001590	E-AGENCY	OAKLAND	Federal Exp...	291 3RD ST...
2	DummyWor...	1	INVALID	<NULL>	1001599	BANK ONE	ELGIN	US Postal In...	2500 WEST...
3	DummyWor...	2	INVALID	<NULL>	1001604	KPMG PEAT...	WOODCLIF...	US Postal 2...	530 CHESN...
4	DummyWor...	3	INVALID	<NULL>	1001622	INVESTEX	NEW YORK	United Parc...	"50 BROAD...
5	DummyWor...	4	INVALID	<NULL>	7121564	"ARTHUR V...	INC."	WHITEHOU...	Federal Exp...
6	DummyWor...	5	INVALID	<NULL>	7121565	OSTERREIC...	NEW YORK	United Parc...	767 FIFTH...
7	DummyWor...	6	INVALID	<NULL>	7121566	K2 ADVISORS	NEW YORK	Courier	570 LEXING...
8	DummyWor...	7	INVALID	<NULL>	7121567	MFP INVES...	SHORT HILLS	Federal Exp...	51 JFK PAR...
9	DummyWor...	8	INVALID	<NULL>	7121568	EASTON	FORT LEE	US Postal In...	ONE PARKE...
10	DummyWor...	9	INVALID	<NULL>	7121569	RICE VOEL...	MANDEVILLE	Federal Exp...	3840 HIGH...

不良レコードテーブルにはソースレコードのすべてのフィールドが含まれています。また、不良レコードには以下のフィールドも含まれています。

Workflow_ID

例外トランスフォーメーションに含まれているワークフロー名です。ワークフローには、問題を確認する例外トランスフォーメーションマッピングタスクとヒューマンタスクが含まれています。例外トランスフォーメーションがワークフローに存在していない場合、Workflow_ID に DummyWorkflowID が含まれています。

Row_Identifier

各行を識別する一意の名前です。

Record_Status

Analyst ツールにおけるレコードステータス。不良レコードテーブル内の各レコードは無効ステータスを受け取ります。レコードステータスのメンテナンスは、Analyst ツールでレコードを更新するときに行うことができます。

問題テーブル

問題テーブルには、不良レコードテーブル内の各行に対して 1 行が含まれています。各行にはデータ品質アナリストがソースレコードの品質を行う問題が含まれています。

以下の図に、問題テーブル内のカラムを示します。

出力				
名前:	exc_BadRecord.Issues_DI			
	Workflow_ID	Row_Identif...	CUST_ID	DQAPRIORITY_CUST_ID
1	DummyWor...	0	6803 S. TU...	1
2	DummyWor...	1	6803 S. TU...	1
3	DummyWor...	2	20TH FL"	1
4	DummyWor...	3	310 BRIER...	1
5	DummyWor...	4	767 FIFTH...	1
6	DummyWor...	5	6803 S. TU...	1
7	DummyWor...	6	6803 S. TU...	1
8	DummyWor...	7	6803 S. TU...	1
9	DummyWor...	8	6803 S. TU...	1
10	DummyWor...	9	SW HOUS...	1
11	DummyWor...	10	6803 S. TU...	1
12	DummyWor...	11	6803 S. TU...	1
13	DummyWor...	12	6803 S. TU...	1
14	DummyWor...	13	6803 S. TU...	1

以下のカラムが問題テーブルに含まれています。

Workflow_ID

レコードを作成したワークフローを特定します。ワークフローには、問題を確認する例外トランスフォーメーションマッピングタスクとヒューマンタスクが含まれています。

Row_Identifier

データベーステーブル内のレコード行を特定します。行 ID は、問題テーブル内の行に対応する不良レコードテーブル内の行を特定します。

問題フィールド名

このフィールド名は、品質の問題である可能性があるフィールドの名前です。このフィールドにエラーが含まれている場合、列の値が品質の問題テキストです。上の図では、ADDR2 フィールド名に invalid_locality という品質の問題が含まれています。

DQAPriority

問題の優先順位です。同じフィールドに複数の問題が発生している場合、優先順位の高い問題が問題フィールド名に表示されます。

正常レコードテーブル

正常レコードテーブルの各レコードには、上限しきい値を上回るレコードスコアが含まれています。この例では、上限しきい値は 90 です。

以下の図に、例外トランスフォーメーションで返される正常レコードを示します。

出力							
名前: exc_BadRecord.Output_DI							
	Score	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2
1	<NULL>	<NULL>	1001658	HOUSE INF...	WASHINGT...	Federal Exp...	"2ND & D STREET
2	<NULL>	<NULL>	15951451	D E FREY &...	DENVER	Federal Exp...	1700 LINCOLN...
3	<NULL>	<NULL>	15951453	SONANGOL...	HOUSTON	United Parc...	1360 POST OAK...
4	<NULL>	<NULL>	15951457	CITADEL IN...	CHICAGO	United Parc...	225 W WASHIN...
5	<NULL>	<NULL>	15951458	IOMEGA	ROY	US Postal In...	1821 WEST IOM...
6	<NULL>	<NULL>	15951461	VICTORY S...	NEW YORK	US Postal In...	45 ROCKEFELLE...
7	<NULL>	<NULL>	15951465	WILLIAMS...	TULSA	United Parc...	7 EAST 2ND ST...
8	<NULL>	<NULL>	15951469	PAINE WEB...	DALLAS	US Postal 2...	2200 ROSS AVE...
9	<NULL>	<NULL>	15951475	PINKSHEET...	NEW YORK	United Parc...	11 PENN PLAZA
10	<NULL>	<NULL>	15951482	JAMES A LU...	DENVER	United Parc...	410 17TH ST.

正常レコードテーブルのレコードには、レコードスコアとソースデータのフィールドが含まれています。

第 8 章

大文字小文字変換プログラムトランスフォーメーション

この章では、以下の項目について説明します。

- [大文字小文字変換プログラムトランスフォーメーションの概要, 165 ページ](#)
- [大文字小文字ストラテジのプロパティ, 165 ページ](#)
- [大文字小文字変換プログラムストラテジの設定, 166 ページ](#)
- [大文字小文字変換プログラムトランスフォーメーションの詳細プロパティ, 167 ページ](#)
- [非ネイティブ環境での大文字小文字変換トランスフォーメーション, 167 ページ](#)

大文字小文字変換プログラムトランスフォーメーションの概要

大文字小文字変換プログラムトランスフォーメーションは、入力文字列の英字の大文字小文字を標準化するパッシブトランスフォーメーションです。

大文字、小文字、タイトルケース、センテンスケースなど、大文字小文字変換形式を選択できます。入力データの各文字の現在の大文字小文字を逆にすることもできます。

大文字小文字変換プログラムトランスフォーメーションでは、参照テーブルの有効な列にある値を使用して、入力文字の大文字の例を定義できます。トランスフォーメーションは、入力値と有効値の間で一致を検出すると、有効値の大文字小文字を入力値に適用します。参照テーブルは、大文字小文字変換タイプが **【タイトルの大文字/小文字の区別】** または **【文の大文字/小文字の区別】** であるときに使用できます。

大文字小文字変換プログラムトランスフォーメーションでは、複数の大文字小文字変換ストラテジを作成できます。各ストラテジは、1 つの変換タイプを使用します。

大文字小文字ストラテジのプロパティ

大文字小文字変換ストラテジのプロパティを設定できます。

【ストラテジ】 ビューで、以下の大文字小文字変換のプロパティを設定できます。

変換タイプ

ストラテジで使用される大文字小文字変換メソッドを定義します。以下の大文字小文字変換タイプを適用できます。

- **[Uppercase]**。すべての文字を大文字に変換します。
- **[Sentence Case]**。フィールドデータ文字列の最初の文字を大文字にします。
- **[Toggle Case]**。小文字を大文字に、大文字を小文字に変換します。
- **[Title Case]**。各部分文字列の最初の文字を大文字にします。
- **[Lowercase]**。すべての文字を小文字に変換します。

デフォルトの大文字小文字変換メソッドは大文字です。

大文字の単語をそのまま残す

大文字の文字列に対して選択された大文字化をオーバーライドします。

区切り文字

大文字小文字変換に対して大文字化がどのように機能するかを定義します。例えば、「smith-jones」を「Smith-Jones」に変換するには、区切り文字としてダッシュを選択します。デフォルトの区切り文字はスペース文字です。

参照テーブル

参照テーブルによって指定された大文字化形式を適用します。大文字小文字変換オプションが**【タイトル
の大文字/小文字の区別】**または**【文の大文字/小文字の区別】**である場合にのみ適用されます。**【新規】**をクリックし、参照テーブルをストラテジに追加します。

注: トークンの先頭部分が参照テーブルと一致した場合、そのトークンの次の文字が大文字に変わります。例えば、入力文字列が mcdonald で、参照テーブルに Mc というエントリがある場合、出力文字列は McDonald になります。

大文字小文字変換プログラムストラテジの設定

入力文字列の大文字小文字を変更するには、大文字小文字変換プログラムトランスフォーメーションの**【ストラテジ】**ビューで設定を構成します。

1. **【ストラテジ】** ビューを選択します。
2. **【新規】** をクリックします。
新しいストラテジウィザードが開きます。
3. 必要に応じて、ストラテジ名と説明を編集します。
4. **【入力】** フィールドと **【出力】** フィールドをクリックして、ストラテジのポートを選択します。
5. ストラテジのプロパティを設定します。デフォルトの変換ストラテジは **【Uppercase】** です。
6. **【次へ】** をクリックします。
7. 必要に応じて参照テーブルを追加し、参照テーブルのエントリに一致する入力データの大文字小文字オプションをカスタマイズします。参照テーブルの大文字小文字のカスタマイズは、タイトルの大文字小文字ストラテジと文の大文字小文字ストラテジにのみ適用されます。
8. **【完了】** をクリックします。

大文字小文字変換プログラムトランスフォーメーションの詳細プロパティ

Data Integration Service で大文字小文字変換プログラムトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

非ネイティブ環境での大文字小文字変換トランスフォーメーション

非ネイティブ環境での大文字小文字変換トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

第 9 章

分類子トランスフォーメーション

この章では、以下の項目について説明します。

- [分類子トランスフォーメーションの概要, 168 ページ](#)
- [分類子モデル, 169 ページ](#)
- [分類子アルゴリズム, 169 ページ](#)
- [分類子トランスフォーメーションのオプション, 169 ページ](#)
- [分類子ストラテジ, 170 ページ](#)
- [分類子トランスフォーメーションの詳細プロパティ, 170 ページ](#)
- [分類子ストラテジの設定, 171 ページ](#)
- [分類子分析の例, 171 ページ](#)
- [非ネイティブ環境での分類子トランスフォーメーション, 176 ページ](#)

分類子トランスフォーメーションの概要

分類子トランスフォーメーションはパッシブトランスフォーメーションであり、入力フィールドを分析して各フィールド内の情報のタイプを特定します。入力フィールドに複数のテキスト値が含まれている場合は分類子トランスフォーメーションを使用します。

分類子トランスフォーメーションを設定する場合、分類子モデルと分類子アルゴリズムを選択します。分類子モデルは、参照データセットの一種です。分類子アルゴリズムとは、文字列の中から類似する単語の数と、単語の相対的な位置を計算するための一連のルールです。このトランスフォーメーションでは、アルゴリズムによる分析結果を分類子モデルの内容と比較します。このトランスフォーメーションでは、文字列の中で多数派である情報タイプを特定するモデル分類を返します。

分類子トランスフォーメーションでは、かなりの長さの文字列を分析することができます。例えば、このトランスフォーメーションを使用することで、電子メールメッセージの本文やソーシャルメディアのメッセージ、さらにはドキュメントのテキストを分類することができます。それぞれのドキュメントまたはメッセージの内容をデータソースカラムのフィールドに渡し、そのカラムを分類子トランスフォーメーションに接続します。それぞれのケースで、分析する必要のあるドキュメントまたは文字列の内容が完全に各フィールドに含まれるようにデータソースを準備します。

分類子モデル

分類子トランスフォーメーションでは、入力データの分析に、分類子モデルと呼ばれる参照データオブジェクトを使用します。分類子トランスフォーメーションを設定する際に分類子モデルを選択します。このトランスフォーメーションは、入力データの列を分類子モデルデータと比較して、各入力フィールドの情報のタイプを説明するラベルを返します。

分類子モデルには、参照データ行とラベル値が含まれます。行は、分類子トランスフォーメーションに接続する可能性のあるポートの入力データを表します。ラベル値は、データ行に含まれる情報のタイプを説明します。分類子モデルを設定するときに、モデルの参照データの各行にラベルを割り当てます。

参照データ行を分類子モデルのラベルにリンクするには、モデルをコンパイルします。コンパイル処理で、データ行とラベル値間の一連の論理的な関連付けが生成されます。モデルを読み取るマッピングを実行すると、データ統合サービスがモデルのロジックを分類子トランスフォーメーションの入力データに適用します。データ統合サービスは、入力データの各フィールドの情報を最も正確に説明するラベルを返します。

Developer ツールで分類子モデルを作成します。モデルリポジトリは、分類子モデルオブジェクトを格納します。Developer ツールは、データ行、ラベル、およびコンパイルデータを、Informatica ディレクトリ構造内のファイルに書き込みます。

分類子アルゴリズム

分類子モデルをトランスフォーメーションストラテジに追加するときに、分類子アルゴリズムも選択します。このアルゴリズムでは、トランスフォーメーションが分類子モデルを入力データと比較する方法を決定します。

【単純ベイズ】 アルゴリズムまたは【最大エントロピー】 アルゴリズムを選択することができます。

アルゴリズムの選択時には次の要因を考慮してください。

- 最大エントロピーアルゴリズムでは、単純ベイズアルゴリズムよりさらに徹底した分析を行います。
- 同じデータの場合、単純ベイズアルゴリズムを使用するマッピングの実行速度は、最大エントロピーアルゴリズムを使用するマッピングよりも速くなります。
- Informatica が Core Accelerator に組み込んだ分類子モデルとともに最大エントロピーアルゴリズムを選択します。

分類子トランスフォーメーションのオプション

分類子トランスフォーメーションでは、設定可能なオプションが Developer ツールの一連のタブまたはビューに表示されます。

再利用可能なトランスフォーメーションを開くと、トランスフォーメーションエディタの一連のタブにオプションが表示されます。再利用できないトランスフォーメーションを開くと、マッピングエディタの一連のビューにオプションが表示されます。マッピングプロパティを選択して、再利用できないトランスフォーメーションのビューを表示します。

次に挙げるビューを選択することができます。

全般

トランスフォーメーションの名称と説明を表示し更新します。

ポート

このトランスフォーメーションの入力ポートと出力ポートを表示します。

注: 再利用可能な分類子トランスフォーメーションでは、全般ビューとポートビューが組み合わされて【概要】タブになっています。

ストラテジ

ストラテジを追加、削除、編集します。

依存関係

各ストラテジの入力ポートと出力ポートを表示します。

詳細

トランスフォーメーションでログファイルに書き込まれる詳細のレベルを設定します。

分類子ストラテジ

ストラテジとは、トランスフォーメーションが入力データに対して実行する一連のデータ分析操作です。分類子トランスフォーメーションにストラテジを少なくとも 1 つ作成します。分類子ストラテジでは、単一の入力ポートを読み取ります。

ストラテジに 1 つまたは複数の操作を定義します。分類子操作は、入力ポートのデータに適用する分類子モデルと分類子アルゴリズムを特定します。各操作で異なるポートに書き込みが行われます。入力ポートを異なる方法で分析する必要がある場合は、ストラテジに複数の操作を作成します。

注: ソースデータで使用する言語を特定する必要がある場合、分類子操作で最大エントロピーアルゴリズムを選択します。

分類子トランスフォーメーションの詳細プロパティ

Data Integration Service で分類子トランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

分類子ストラテジの設定

データの情報タイプを特定するようにストラテジを設定します。各ストラテジは入力ポートを分析します。

再利用できないトランスフォーメーションでは、ストラテジを設定する前に入力ポートをトランスフォーメーションに接続しておきます。

1. トランスフォーメーションを開き、**【ストラテジ】** ビューを選択します。
2. **【新規ストラテジ】** をクリックします。
[プロファイルの作成] ウィザードが開きます。
3. ストラテジの名前と、必要に応じてその説明を入力します。
4. [入力] フィールドから入力ポートを選択します。
5. 入力ポートからすべてのフィールドが読み込まれるように、入力ポートの精度が十分高い値に設定されているか確認します。これは、入力ポートの精度を超える入力データは切り捨てられてしまうためです。
6. オプションを選択またはクリアし、スコアデータをストラテジ出力に追加します。
7. **【次へ】** をクリックします。
8. 分類子の操作タイプを確認して、**【次へ】** をクリックします。
9. 分類子アルゴリズムを選択します。次のアルゴリズムを選択することができます。
 - 単純ベイズ
 - 最大エントロピー**注:** ソースデータで使用されている言語を特定するには、最大エントロピーアルゴリズムを選択します。
10. 出力ポートを検証します。
このトランスフォーメーションでは、ストラテジの各操作に対して単一のポートを作成します。ポート名と精度は編集することができます。
11. 分類子モデルを選択します。
ウィザードに、モデルリポジトリ内の分類子モデルオブジェクトが表示されます。
12. **【次へ】** をクリックして、別の操作をストラテジに追加します。それ以外の場合は **【完了】** をクリックします。

分類子分析の例

あなたは、スマートフォン向けの新しいアプリケーションをリリースしたソフトウェア会社のデータスチュワードです。この会社は、このアプリケーションが人々にどう受け止められているのか、そしてメディアの取り上げ方について知る必要性を感じています。このため、会社はあなたとあなたのチームに対して、アプリケーションに関するソーシャルメディアでの発言を分析するように指示します。

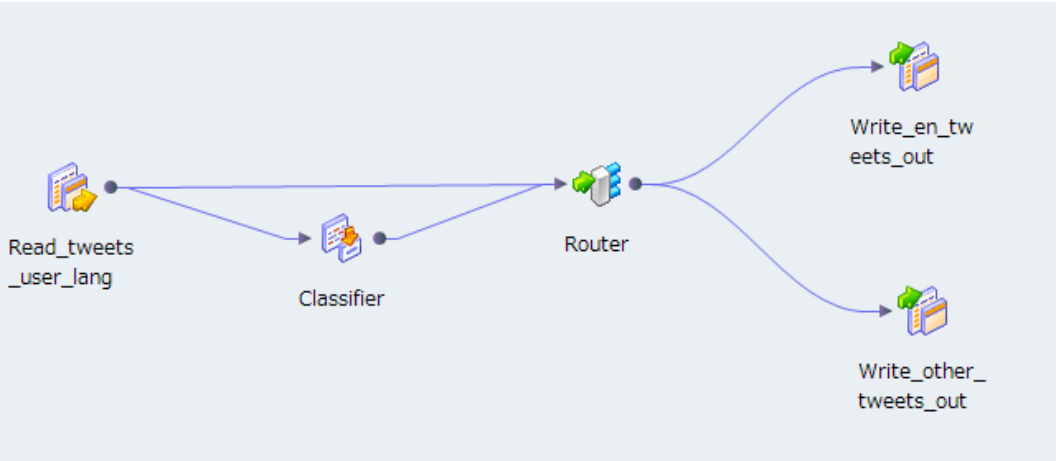
そこであなたは、スマートフォンの話をしているツイッターフィードからデータを取り込むことにします。ツイッターアプリケーションのプログラミングインターフェイスを使ってツイッターのデータストリームをフィルタリングします。そして、データソースを作成し、分析する必要のあるツイッターデータをそこに格納します。

ツイッターフィードには複数の言語で書かれたメッセージが含まれているため、各メッセージで使用されている言語を特定する必要があります。そこで、分類子トランスフォーメーションを使用して言語を分析することに決めます。ソースデータの言語を特定するマッピングを作成し、ツイッターメッセージを英語データターゲットと英語以外のデータターゲットに書き込みます。

分類子マッピングの作成

作成するマッピングでは、まずデータソースを読み込み、データの言語を分類し、含まれる言語に基づいてデータをターゲットに書き込みます。

次の図に、Developer ツールのマッピングを示します。



作成するマッピングには、以下のオブジェクトが含まれます。

オブジェクト名	説明
Read_tweet_user_lang	データソース。 ツイッターのメッセージが入ります。
分類子	分類子トランスフォーメーション。 ツイッターメッセージに使用されている言語を特定します。
Router	Router トランスフォーメーション。 ツイッターメッセージを、そこに含まれる言語に従ってデータターゲットオブジェクトにルーティングします。
Write_en_tweets_out	データターゲット。 英語のツイッターメッセージが入ります。
Write_other_tweets_out	データターゲット。 英語以外の言語のツイッターメッセージが入ります。

入力データサンプル

次に示すデータの抜粋は、マッピングで分析するツイッターデータのサンプルです。

Twitter Message

```
RT @GanaphoneS3: Faltan 10 minutos para la gran rifa de un iPhone 5...
RT @Clarified: How to Downgrade Your iPhone 4 From iOS 6.x to iOS 5.x (Mac)...
RT @jerseyjazz: The razor was the iPhone of the early 2000s
RT @KrissiDevine: Apple Pie that I made for Thanksgiving. http://t.com/s9ImzFx0
RT @sophieHz: Dan yang punya 2 kupon undian. Masuk dalam kotak undian yang berhadiah Samsung
RT @IsabelFreitas: o galaxy tem isso isso isso e a bateria Á melhor que do iPhone
RT @PremiusIpad: Faltan 15 minutos para la gran rifa de un iPhone 5...
RT @payyton3: I want apple cider
RT @wiesteronder: Retweet als je iets van Apple, Nike, Adidas of microsoft hebt!
```

データソースの設定

データソースには単一のポートが含まれています。このポートに各行に、単一のツイッターメッセージが入ります。

次の表に、データソースの設定を示します。

ポート名	ポートタイプ	精度
テキスト	なし	200

分類子トランスフォーメーションの設定

分類子トランスフォーメーションは、単一の入力ポートと出力ポートを使用します。トランスフォーメーション入力ポートは、データソースのテキストフィールドを読み取ります。出力ポートには、テキストフィールド内のツイッターメッセージに特定された言語が含まれます。分類子トランスフォーメーションでは、ISO 国コードを使用して言語を特定しています。

次の表に、分類子トランスフォーメーションの設定を示します。

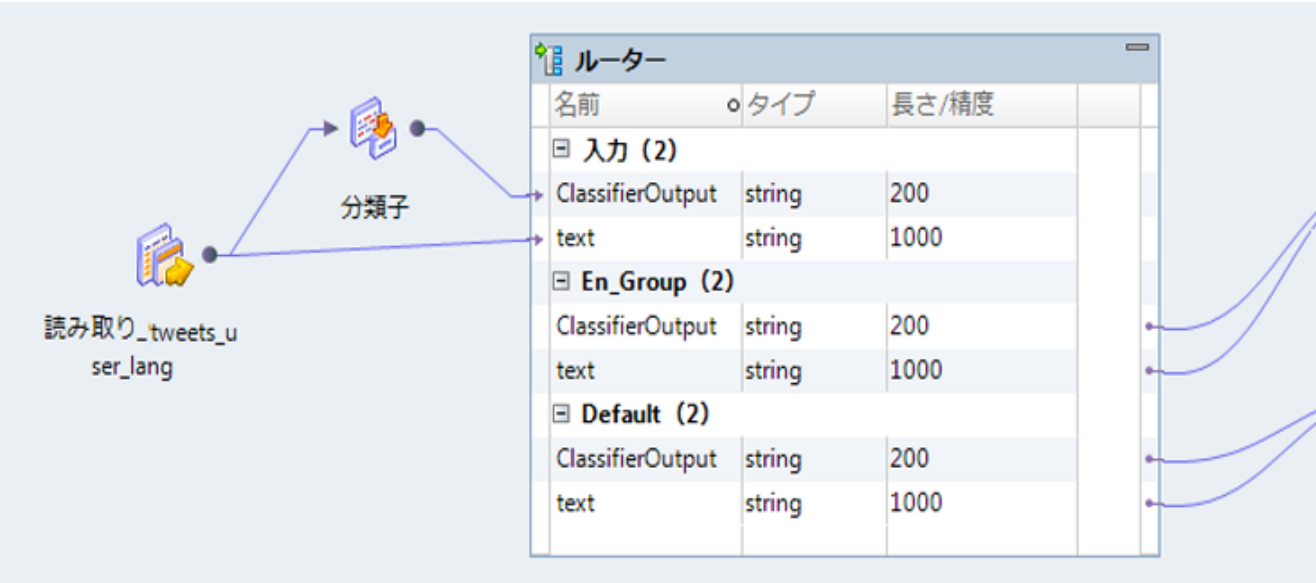
ポート名	ポートタイプ	精度	ストラテジ
text_input	入力	200	分類子 1
Classifier_Output	出力	2	分類子 1

Router トランスフォーメーションの設定

Router トランスフォーメーションでは 2 つの入力ポートを使用します。データソースからはツイッターメッセージを読み込み、分類子トランスフォーメーションからは ISO 国コードを読み取ります。Router トランスフ

オーメーションでは、入力ポートのデータを、指定した条件に基づいて異なる出力ポートにルーティングします。

次の図に、Router トランスフォーメーションのポートグループとポートの接続を示します。



次の表に、Router トランスフォーメーションの設定を示します。

ポート名	ポートタイプ	ポートグループ	精度
Classifier_Output	入力	入力	2
text	入力	入力	200
Classifier_Output	入力	デフォルト	2
text	入力	デフォルト	200
Classifier_Output	入力	En_Group	2
text	入力	En_Group	200

英語のメッセージおよび英語以外のメッセージのデータストリームを作成するためのトランスフォーメーションを設定します。データストリームを作成するには、トランスフォーメーションに出力ポートを追加します。トランスフォーメーションの【グループ】オプションを使用してポートグループを追加します。

トランスフォーメーションがデータを各データストリームにルーティングする方法を決めるには、ポートグループに対する条件を定義します。この条件でポートを特定し、そのポートで可能な値を指定します。トランスフォーメーションが条件に一致する入力ポート値を検出すると、入力データを条件を適用するポートグループにルーティングします。

En_Group に対して次の条件を定義します。

ClassifierOutput='en'

注: Router トランスフォーメーションでは、マッピング内の 2 つのオブジェクトからデータを読み込みます。このトランスフォーメーションでは、データオブジェクトで定義した行の順番が変わらないため、各出力グループ内のデータを結合することができます。

データターゲットの設定

このマッピングには、英語によるツイッターメッセージ用のデータターゲットが1つと、英語以外の言語のメッセージ用のターゲットが1つ含まれています。ポートを Router トランスフォーメーションの出力グループからデータターゲットに接続します。

次の表に、データターゲットの設定を示します。

ポート名	ポートタイプ	精度
テキスト	なし	200
Classifier_Output	なし	2

分類子マッピングの結果

このマッピングを実行すると、分類子トランスフォーメーションでツイッターメッセージの言語が特定されます。Router トランスフォーメーションでは、言語の分類に基づいてメッセージのテキストを各データターゲットに書き込みます。

次に示すデータの抜粋は、英語のターゲットデータのサンプルです。

ISO Country Code	Twitter Message
en	RT @Clarified: How to Downgrade Your iPhone 4 From iOS 6.x to iOS 5.x (Mac)...
en	RT @jerseyjazz: The razor was the iPhone of the early 2000s
en	RT @KrissiDevine: Apple Pie that I made for Thanksgiving. http://t.com/s9ImzFx0
en	RT @payyton3: I want apple cider

次に示すデータの抜粋は、英語以外の言語であると特定されたターゲットデータのサンプルです。

ISO Country Code	Twitter Message
es	RT @GanaphoneS3: Faltan 10 minutos para la gran rifa de un iPhone 5...
id	RT @sophieHz: Dan yang punya 2 kupon undian. Masuk dalam kotak undian yang berhadiah Samsung Champ.
pt	RT @IsabelFreitas: o galaxy tem isso isso isso e a bateria ã melhor que do iPhone
es	RT @PremiusIpad: Faltan 15 minutos para la gran rifa de un iPhone 5...
nl	RT @wiesteronder: Retweet als je iets van Apple, Nike, Adidas of microsoft hebt! http://t.co/Je6Ts00H

非ネイティブ環境での分類子トランスフォーメーション

非ネイティブ環境での分類子トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

第 10 章

比較トランスフォーメーション

この章では、以下の項目について説明します。

- [比較トランスフォーメーションの概要, 177 ページ](#)
- [フィールド一致ストラテジ, 177 ページ](#)
- [ID マッチングストラテジ, 180 ページ](#)
- [比較ストラテジの設定, 180 ページ](#)
- [比較トランスフォーメーションの詳細プロパティ, 181 ページ](#)
- [非ネイティブ環境での比較トランスフォーメーション, 181 ページ](#)

比較トランスフォーメーションの概要

比較トランスフォーメーションは、入力文字列のペア間の類似度を評価し、各ペアの類似度を数値スコアとして計算するパッシブなトランスフォーメーションです。

トランスフォーメーションを設定するときは、入力カラムのペアを選択し、それらに一致ストラテジを割り当てます。

比較トランスフォーメーションでは、一致スコアを 0~1 の範囲で出力します。1 が完全一致を表します。

注: 比較トランスフォーメーションで使用するストラテジは、一致トランスフォーメーションでも使用できます。一致マプレットに追加する一致比較処理を定義する場合は、比較トランスフォーメーションを使用します。マプレットに複数の比較トランスフォーメーションを追加することができます。1 つのトランスフォーメーション内で一致比較を定義する場合は、一致トランスフォーメーションを使用します。一致トランスフォーメーションに一致マプレットを組み込むことができます。

フィールド一致ストラテジ

比較トランスフォーメーションには、入力データフィールドのペアを比較する定義済みのフィールド一致ストラテジが用意されています。

バイグラム

バイグラムアルゴリズムは、郵便アドレスが1つのフィールドに入力されている場合など、長いテキスト文字列を比較する場合に使用します。

バイグラムアルゴリズムでは、2つのデータ文字列の一致スコアを、両方の文字列に含まれる連続した文字に基づいて計算します。このアルゴリズムでは、連続する文字から成る文字ペアが両方の文字列に共通して存在するかを調べます。そして相手の文字列に一致するものが存在する文字ペアの数を、両方の文字列の文字ペアの総数で割ります。

バイグラムの例

次の文字列について考えてみます。

- larder
- lerder

これらの文字列をバイグラムのグループに分けると次のようになります。

l a, a r, r d, d e, e r

l e, e r, r d, d e, e r

文字列"lerder"の2つ目の"e r"は一致と見なされません。文字列"larder"には文字列"e r"が1つしかなく、2つ目に対応するものはないからです。

バイグラムの一致スコアを計算するには、一致するペアの数（6）を両方の文字列のペアの総数（10）で割ります。この例では、文字列の類似度は60%で、一致スコアは0.60になります。

ハミング距離

電話番号、郵便番号、製品コードなどの数値フィールドやコードフィールドのように、データ文字の位置が重要な要素である場合には、ハミング距離アルゴリズムを使用します。

ハミング距離アルゴリズムでは、2つのデータ文字列の一致スコアを、データ文字列間で文字が異なる位置の数に基づいて計算します。長さが異なる文字列の場合、長い方の文字列にしかない各文字は文字列間の相違としてカウントされます。

ハミング距離の例

次の文字列について考えてみます。

- Morlow
- Marlowes

強調表示された文字は、ハミングアルゴリズムで相違と見なされる位置を示しています。

ハミングの一致スコアを計算するには、一致する文字の数（5）を長い方の文字列の文字数（8）で割ります。この例では、文字列の類似度は62.5%で、一致スコアは0.625になります。

エディット距離

エディット距離アルゴリズムは、単語や短いテキスト文字列（名前など）を比較する場合に使用します。

エディット距離アルゴリズムでは、文字列を別の文字列に変換するために文字の挿入、削除、または置き換えが必要な最小限の「コスト」を計算します。

エディット距離の例

次の文字列について考えてみます。

- Levenston

- Levenshtein

強調表示された文字は、文字列をもう一方の文字列に変換するために処理が必要な部分を示しています。

エディット距離アルゴリズムでは、変更されない文字の数（8）を長い方の文字列の文字数（11）で割ります。この例では、文字列の類似度は 72.7% で、一致スコアは 0.727 になります。

Jaro 距離

2 つの文字列を比較するときに文字列内の最初の文字の類似度を優先する場合は、Jaro 距離アルゴリズムを使用します。

Jaro 距離のマッチ率は、両方の文字列から最初の 4 文字を取り出して比較したときの類似度、および識別された文字の転置の数を反映しています。最初の 4 文字の一致の重要度に、[ペナルティ] プロパティに入力した値を使用して重みが設定されます。

Jaro 距離のプロパティ

Jaro 距離アルゴリズムを設定する場合は、次のプロパティが設定できます。

ペナルティ

比較する 2 つの文字列内の最初の 4 文字が同一でない場合の一致スコアのペナルティを指定します。最初の文字が一致しない場合は、ペナルティの値がそのまま減算されます。それ以外の文字が異なる場合は、その位置に基づいてペナルティの端数が減算されます。デフォルトのペナルティ値は 0.20 です。

大文字小文字の区別

Jaro 距離アルゴリズムで文字の比較を行うときに大文字と小文字を区別するかどうかを指定します。

Jaro 距離の例

次の文字列について考えてみます。

- 391859
- 813995

[ペナルティ] を 0.20（デフォルト値）にしてこれらの文字列を分析した場合、Jaro 距離アルゴリズムで返される一致スコアは 0.513 になります。文字列の類似度は 51.3% となります。

ハミング距離の反転

ハミング距離反転アルゴリズムは、2 つの文字列間で文字が異なる位置の割合を、文字列を右から左に読み取りながら計算する場合に使用します。

ハミング距離アルゴリズムでは、2 つのデータ文字列の一致スコアを、データ文字列間で文字が異なる位置の数に基づいて計算します。長さが異なる文字列の場合、長い方の文字列にしかない各文字は文字列間の相違としてカウントされます。

ハミング距離の反転の例

次の文字列について考えてみましょう。この文字列は、ハミング反転アルゴリズムについて説明するために右から左に文字を配置しています。

- 1-999-9999
- **011-01**-999-9991

強調表示された文字は、ハミング距離反転アルゴリズムで相違と見なされる位置を示しています。

ハミングの反転の一致スコアを計算するには、一致する文字の数（9）を長い方の文字列の文字数（15）で割ります。この例では、一致スコアは 0.6 になり、文字列の類似度は 60% となります。

ID マッチングストラテジ

比較トランスフォーメーションには、個人、住所、または法人の一致を見つけるために使用できる定義済みの ID 一致ストラテジが用意されています。

次の表に、それぞれの ID マッチングストラテジで実行される一致操作を示します。

ID 一致ストラテジ	一致操作
住所	住所の一致を特定します。
担当者	単一の場所にいる組織内の担当者を特定します。
法人	組織を正式な社名で特定します。
除算	ある住所を所在地とする組織を特定します。
家族	家族を姓と住所または電話番号で特定します。
フィールド	選択するカスタムフィールドを特定します。
世帯	同じ住所の同じ家族のメンバーを特定します。
個人	個人を名前と ID または誕生日で特定します。
組織	組織を名前で特定します。
個人名	個人を名前で特定します。
住居	ある住所にいる個人を特定します。
担当者（広域）	所在地に関係なく、組織内の担当者を特定します。
世帯（広域）	住所に関係なく、同じ家族のメンバーを特定します。

注: ID 一致ストラテジでは、**ポピュレーション**と呼ばれる参照データファイルを読み取ります。システムにインストールされているポピュレーションデータファイルについては、Informatica Administrator ユーザーに問い合わせてください。

比較ストラテジの設定

比較ストラテジを設定するには、比較トランスフォーメーションの **【ストラテジ】** ビューで設定を編集します。

1. **【ストラテジ】** ビューを選択します。
2. **【ストラテジ】** セクションから比較ストラテジを選択します。
3. **【フィールド】** セクションで、**【使用可能なフィールド】** カラムのセルをダブルクリックして入力を選択します。

注: **【入力フィールド】** カラムで入力名が太字で示されている各行について、入力を選択する必要があります。

比較トランスフォーメーションの詳細プロパティ

Data Integration Service で比較トランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

非ネイティブ環境での比較トランスフォーメーション

非ネイティブ環境での比較トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

第 11 章

統合トランスフォーメーション

この章では、以下の項目について説明します。

- [統合トランスフォーメーションの概要, 182 ページ](#)
- [統合マッピング, 183 ページ](#)
- [Consolidation トランスフォーメーションのポート, 183 ページ](#)
- [統合トランスフォーメーションのビュー, 183 ページ](#)
- [単純ストラテジ, 186 ページ](#)
- [行ベースストラテジ, 187 ページ](#)
- [詳細ストラテジ, 187 ページ](#)
- [単純な統合関数, 188 ページ](#)
- [行ベースの統合関数, 192 ページ](#)
- [統合マッピングの例, 195 ページ](#)
- [統合トランスフォーメーションの設定, 196 ページ](#)
- [非ネイティブ環境での統合トランスフォーメーション, 197 ページ](#)

統合トランスフォーメーションの概要

統合トランスフォーメーションは、関連するレコードのグループを分析し、各グループに対して統合されたレコードを作成する、アクティブなトランスフォーメーションです。統合トランスフォーメーションを使用して、キージェネレータ、一致、および関連付けなどのトランスフォーメーションによって生成されたレコードグループを統合します。

統合トランスフォーメーションは、関連するレコードのグループにストラテジを適用することによって、統合されたレコードを生成します。トランスフォーメーションには、どのレコードが統合されたレコードであるかを示す出力ポートが含まれています。統合されたレコードのみが含まれるようにトランスフォーメーション出力を制限することも選択できます。

例えば、一致トランスフォーメーションによって生成された従業員の重複レコードのグループを統合できます。統合トランスフォーメーションは、グループ内のすべてのレコードからマージされたデータが含まれる、統合されたレコードを作成できます。

統合トランスフォーメーションを設定すると、ユーザーの統合の要件に基づいてさまざまなタイプのストラテジを使用することができます。複数のレコードから統合されたレコードを作成するには、単純ストラテジを使用します。単純ストラテジを使用するときは、ポートごとにストラテジを指定します。レコードグループ内の行を分析し、いずれかの行からの値を持つ統合されたレコードを作成するには、行ベースストラテジを使用します。作成した式を適用することによって統合されたレコードを作成するには、詳細ストラテジを使用します。

統合マッピング

レコードを統合するには、関連するレコードのグループを作成するマッピングを作成します。統合トランスフォーメーションをマッピングに追加し、各レコードグループを 1 つのマスタレコードに統合するようにトランスフォーメーションを設定します。

ビジネス目標とデータ要件に従って、結合トランスフォーメーションを他のトランスフォーメーションに接続します。一致したレコードを統合する場合は、統合トランスフォーメーションを一致トランスフォーメーションに接続できます。例外レコード管理の一環としてレコードを統合するには、統合トランスフォーメーションを例外トランスフォーメーションに接続します。キージェネレータトランスフォーメーションを使用してレコードをグループ化する場合は、統合トランスフォーメーションをキージェネレータトランスフォーメーションに直接接続できます。統合トランスフォーメーションは、キージェネレータトランスフォーメーションで作成された各グループの統合レコードを作成します。

ネイティブおよび Hadoop 環境のマッピング出力

ネイティブ環境および Hadoop 環境で統合マッピングを実行する場合、統合トランスフォーメーションで異なる結果が生成される場合があります。Hadoop では複数のノードでマッピングが実行されるため、ネイティブ環境とは異なる順序で入力レコードが統合トランスフォーメーションに入力される可能性があります。このため、トランスフォーメーションで、同じ入力データセットに対して環境ごとに異なるセットの存続レコードが生成される場合があります。どちらの場合で入力された行順序であっても、トランスフォーメーションの計算と統合結果は正確です。

ネイティブ環境および Hadoop 環境で同じ存続レコードを生成するには、統合トランスフォーメーションが次の順序でレコードをソートするように設定します。

- 最初にグループ化ポートでレコードをソートします。
- 次に、入力ポートがトランスフォーメーションに表示される順序でレコードをソートします。

Consolidation トランスフォーメーションのポート

Developer ツールは、追加する入力ポートごとに出力ポートを作成します。トランスフォーメーションに手動で出力ポートを追加することはできません。統合トランスフォーメーションには、統合されたレコードを示す **IsSurvivor** 出力ポートも含まれています。

統合トランスフォーメーションに追加する入力ポートのいずれかにはグループキーが含まれている必要があります。統合トランスフォーメーションはデータセット全体ではなくレコードグループを処理するため、グループキーの情報を必要とします。

入力ポートを追加するとき、Developer ツールは、入力ポート名にサフィックス「1」を追加することによって出力ポート名を作成します。トランスフォーメーションには、レコードが統合されたレコードであるかどうかを示す **IsSurvivor** 出力ポートも含まれています。統合されたレコードの場合、統合トランスフォーメーションは、**IsSurvivor** ポートに文字列「Y」を書き込みます。入力レコードの場合、統合トランスフォーメーションは、**IsSurvivor** ポートに文字列「N」を書き込みます。

統合トランスフォーメーションのビュー

統合トランスフォーメーションには、ポート、ストラテジ、および詳細の各プロパティに関するビューが含まれています。

統合トランスフォーメーションの [ストラテジ] ビュー

[ストラテジ] ビューには、単純ストラテジ、行ベースストラテジ、および詳細ストラテジのプロパティが含まれます。

以下のリストでは、統合ストラテジのタイプについて説明します。

単純ストラテジ

単純ストラテジは、レコードグループ内のポートのすべての値を分析し、1つの値を選択します。単純ストラテジはポートごとに指定します。統合トランスフォーメーションは、すべての単純ストラテジによって選択されたポートの値を使用して、統合されたレコードを作成します。単純ストラテジの例には、ポート内の最頻値、ポート内の最長の値、ポート内の空白以外の最頻値などがあります。

行ベースストラテジ

行ベースストラテジは、レコードグループ内の行を分析し、1つの行を選択します。統合トランスフォーメーションはその行のポート値を使用して、統合されたレコードを作成します。行ベースストラテジの例には、最大の文字数、空白フィールドの最小数、最も頻度の高いフィールドの最大数などがあります。

詳細ストラテジ

詳細ストラテジは、ユーザー定義のストラテジを使用してレコードグループを分析します。詳細ストラテジは、式に統合関数を使用することによって作成します。統合トランスフォーメーションは、式の出力に基づいて統合されたレコードを作成します。また、作成する式には、ディシジョントランスフォーメーションで使用可能なすべての関数を使用することができます。

統合トランスフォーメーションの詳細プロパティ

統合トランスフォーメーションには、ソートの動作、出力モード、キャッシュメモリの動作、およびトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

ソート

トランスフォーメーションが [グループ別] ポートデータの入力行をソートするかどうかを決定します。このプロパティは、デフォルトで有効になっています。

入力行が事前にソートされていない場合は、このプロパティを選択します。

大文字小文字を区別したソート

ソート操作で大文字と小文字を区別するかどうかを決定します。このプロパティは、デフォルトで有効になっています。

出力モード

トランスフォーメーションがすべてのレコードを出力として書き込むか、または統合されたレコードを出力として書き込むかを決定します。デフォルト値は All です。

キャッシュファイルディレクトリ

データ統合サービスが現在のトランスフォーメーションの一時データを書き込むディレクトリを指定します。入力データの量が利用可能なシステムメモリより大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

キャッシュファイルサイズ

トランスフォーメーションの入力データをソートするためにデータ統合サービスが使用するシステムメモリの量を決定します。デフォルト値は 400,000 バイトです。このパラメータを使用して、キャッシュファイルサイズを指定できます。

データをソートする前に、データ統合サービスは指定されているメモリ量を割り当てます。ソート操作によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュファイルディレクトリに書き込みます。ソート操作にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

注: 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

キャッシュファイルサイズ

キャッシュファイルサイズのプロパティは、データ統合サービスがソート操作のために統合トランスフォーメーションに割り当てるシステムメモリの量を決定します。データ統合サービスのホストマシン上の RAM 容量以下の値を使用してプロパティを設定します。

最高のパフォーマンスを得るために、少なくとも 16 MB のキャッシュファイルサイズを指定します。

ソート操作を始める前に、データ統合サービスはキャッシュファイルサイズのプロパティが指定するメモリを割り当てます。データ統合サービスは、ソート操作を実行する前に、すべての入力データを統合トランスフォーメーションに渡します。

入力データのボリュームがキャッシュファイルサイズより大きい場合、データ統合サービスはデータをキャッシュファイルディレクトリに書き込みます。データ統合サービスは、キャッシュファイルディレクトリにデータを書き込む際に、入力データボリュームの少なくとも 2 倍のディスク容量を消費します。

下記の式で入力されるデータのサイズを求めます。

$$[\text{number_of_input_rows} * (\text{Sum}(\text{column_size}) + 16)]$$

以下の表に、キャッシュファイルのデータ計算に使用できるデータタイプとカラムサイズ値を示します。

データタイプ	カラムサイズ
バイナリ	精度+8。 最も近い 8 の倍数に丸められます。
日付/時刻	29
Decimal、高精度オフ（全精度）	16
Decimal、高精度（精度<=18）	24
Decimal、高精度（精度>18、<=28）	32
Decimal、高精度（精度>28）	16
Decimal、高精度（負の位取り）	16
ダブル	16

データタイプ	カラムサイズ
Real	16
Integer	16
文字列、テキスト	Unicode モード：2*（精度+5） ASCII モード：精度 +9

単純ストラテジ

単純ストラテジは、レコードグループ内のポートを分析し、1つの値を返します。単純ストラテジはポートごとに指定します。統合トランスフォーメーションは、すべての単純ストラテジによって選択されたポートの値を使用して、統合されたレコードを作成します。

トランスフォーメーションの【ストラテジ】ビューでストラテジを設定すると、統合方式として次のテキストがストラテジに表示されます。

デフォルトを使用。

デフォルトのストラテジは「最大の行 ID」です。

単純ストラテジは、以下のいずれかを選択できます。

平均

レコードグループのポートを分析し、すべての値の平均を返します。

String 型および Date/time 型のデータの場合、ストラテジは最も頻繁に出現する値を返します。

最長

レコードグループのポートを分析し、文字数が最も多い値を返します。最も多い文字数が2つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

最大

レコードグループのポートを分析し、最大値を返します。

String データ型の場合、ストラテジ是最長の文字列を返します。Date/time データ型の場合、ストラテジは最新の日付を返します。

最小

レコードグループのポートを分析し、最小値を返します。

String データ型の場合、ストラテジ是最短の文字列を返します。Date/time データ型の場合、ストラテジは最も古い日付を返します。

最も頻繁

レコードグループのポートを分析し、空白または NULL 値を含む、最も頻繁に出現する値を返します。出現の最大数を2つ以上の値が共有している場合、ストラテジは要件を満たす最初の値を返します。

空白以外の最も頻繁

レコードグループのポートを分析し、空白または NULL 値以外の最も頻繁に出現する値を返します。空白以外の出現の最大数を2つ以上の値が共有している場合、ストラテジは要件を満たす最初の値を返します。

最短

レコードグループのポートを分析し、文字数が最も少ない値を返します。最も少ない文字数が2つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

最大の行 ID

レコードグループのポートを分析し、行 ID が最も高い値を返します。

行ベースストラテジ

行ベースストラテジは、レコードグループ内の行を分析し、1つの行を選択します。統合トランスフォーマーはその行のポート値を使用して、統合されたレコードを作成します。デフォルトのストラテジは、「Most Data」です。

以下のいずれかの行ベースストラテジを選択します。

Most Data

文字数が最も多い行を選択します。最も多い文字数が2つ以上の行により共有されている場合、ストラテジは要件を満たす最後の値を返します。

Most Filled

空白以外のカラム数が最大の行を選択します。空白以外のカラムの最大数が2つ以上の行により共有されている場合、ストラテジは要件を満たす最後の値を返します。

Modal Exact

最も頻度の高い空白以外の値の数が最大の行を選択します。例えば、レコードグループ内に最も頻度の高い値を含む3つのポートを持つ行があるとします。この行の最も頻度の高い値の数は「3」です。

最も頻度の高い空白以外の値の最大数が2つ以上の行により共有されている場合、ストラテジは要件を満たす最後の値を返します。

行ベースストラテジの例

以下の表に、レコードグループの例を示します。最後のカラムでは、特定の行ベースストラテジがこのレコードグループで異なる行を選択する理由を説明しています。

製品 ID名	姓	郵便番号	ストラテジの選択
2106	Bartholomew	28516	Most Data ストラテジは、他の行よりも多くの文字が含まれているため、この行を選択します。
2236	Bart	Smith28579	Most Filled ストラテジは、他の行よりも多くの空白以外のカラムが含まれているため、この行を選択します。
2236	<空白>	Smith28516	Modal Exact ストラテジは、最も頻度の高い値の数が最大であるため、この行を選択します。

詳細ストラテジ

詳細ストラテジを使用すると、定義済み関数から統合ストラテジを作成できます。統合関数と Informatica のその他の関数を使用できます。

単純な統合関数または行ベースの統合関数を含む式を作成できます。レコードグループ内のポートの値に基づいて統合されたレコードを作成するには、単純な統合関数を使用します。レコードグループから行を選択するには、行ベースの統合関数を使用します。

統合の式には、統合トランスフォーメーションのすべての出力ポートを組み込む必要があります。統合の式がすべての出力ポートを使用しない場合、トランスフォーメーションが原因でマッピングが失敗します。

単純ストラテジまたは行ベースストラテジは、詳細ストラテジのテンプレートとして使用できます。単純ストラテジまたは行ベースストラテジを設定し、[詳細] を選択します。統合トランスフォーメーションは、ストラテジを実行する関数が含まれる式を生成します。さらに関数を追加すると、追加の要件を実装することができます。

単純な統合関数

単純な統合関数は、レコードグループのすべてのポート値から値を選択します。単純な統合関数の使用時は、ポートおよび Group By ポートを使用して関数を指定します。

CONSOL_AVG

レコードグループのポートを分析し、すべての値の平均を返します。

構文

`CONSOL_AVG(string, group by)`

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

ポートのすべての値の平均。

String および Date/time データ型の場合、関数は最も頻繁に出現する値を返します。

例

以下の式は、CONSOL_AVG 関数を使用して SalesTotal 入力ポートの平均値を検索します。

`SalesTotal1:= CONSOL_AVG(SalesTotal, GroupKey)`

この式で、CONSOL_AVG 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は SalesTotal ポートを分析し、平均値を返します。式は平均値を SalesTotal1 出力ポートに書き込みます。

CONSOL_LONGEST

レコードグループのポートを分析し、文字数が最も多い値を返します。

構文

`CONSOL_LONGEST(string, group by)`

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

文字数が最も多いポート値。

最も多い文字数が 2 つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

例

以下の式は、CONSOL_LONGEST 関数を使用して FirstName 入力ポートを分析し、文字数が最も多い値を検索します。

```
FirstName1:= CONSOL_LONGEST(FirstName, GroupKey)
```

この式で、CONSOL_LONGEST 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は FirstName ポートを分析し、最長値を返します。式はこの値を FirstName1 出力ポートに書き込みます。

CONSOL_MAX

レコードグループのポートを分析し、最大値を返します。

構文

```
CONSOL_MAX(string, group by)
```

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

最大ポート値。

String データ型の場合、関数は最長の文字列を返します。Date/time データ型の場合、関数は最新の日付を返します。

例

以下の式は、CONSOL_MAX 関数を使用して SalesTotal 入力ポートを分析し、最大値を検索します。

```
SalesTotal1:= CONSOL_MAX(SalesTotal, GroupKey)
```

この式で、CONSOL_MAX 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は SalesTotal ポートを分析し、最大値を返します。式はこの値を SalesTotal1 出力ポートに書き込みます。

CONSOL_MIN

レコードグループのポートを分析し、最小値を返します。

構文

`CONSOL_MIN(string, group by)`

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

最小ポート値。

String データ型の場合、関数は最短の文字列を返します。Date/time データ型の場合、関数は最も古い日付を返します。

例

以下の式は、CONSOL_MIN 関数を使用して SalesTotal 入力ポートを分析し、最小値を検索します。

`SalesTotal1:= CONSOL_MIN(SalesTotal, GroupKey)`

この式で、CONSOL_MIN 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は SalesTotal ポートを分析し、最小値を返します。式はこの値を SalesTotal1 出力ポートに書き込みます。

CONSOL_MOSTFREQ

レコードグループのポートを分析し、空白または NULL 値を含む、最も頻繁に出現する値を返します。

構文

`CONSOL_MOSTFREQ(string, group by)`

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

空白または NULL 値を含む最も頻繁に出現する値。

出現の最大数が 2 つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

例

以下の式は、CONSOL_MOSTFREQ 関数を使用して Company 入力ポートを分析し、最も頻繁に出現する値を検索します。

`Company1:= CONSOL_MOSTFREQ(Company, GroupKey)`

この式で、CONSOL_MOSTFREQ 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は Company ポートを分析し、最も頻繁に出現する値を返します。式はこの値を Company1 出力ポートに書き込みます。

CONSOL_MOSTFREQ_NB

レコードグループのポートを分析し、空白または NULL 値以外の最も頻繁に出現する値を返します。

構文

CONSOL_MOSTFREQ_NB(*string*, *group by*)

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

空白または NULL 値以外の最も頻繁に出現する値。

出現の最大数が 2 つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

例

以下の式は、CONSOL_MOSTFREQ_NB 関数を使用して Company 入力ポートを分析し、最も頻繁に出現する値を検索します。

Company1:= CONSOL_MOSTFREQ_NB(Company, GroupKey)

この式で、CONSOL_MOSTFREQ_NB 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は Company ポートを分析し、最も頻繁に出現する値を返します。式はこの値を Company1 出力ポートに書き込みます。

CONSOL_SHORTEST

レコードグループのポートを分析し、文字数が最も少ない値を返します。

構文

CONSOL_SHORTEST(*string*, *group by*)

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

文字数が最も少ないポート値。

最も少ない文字数が 2 つ以上の値により共有されている場合、ストラテジは要件を満たす最初の値を返します。

例

以下の式は、CONSOL_SHORTEST 関数を使用して FirstName 入力ポートを分析し、文字数が最も少ない値を検索します。

```
FirstName1:= CONSOL_SHORTEST(FirstName, GroupKey)
```

この式で、CONSOL_SHORTEST 関数は GroupKey ポートを使用してレコードグループを識別します。このレコードグループ内で、関数は FirstName ポートを分析し、最短値を返します。式はこの値を FirstName1 出力ポートに書き込みます。

行ベースの統合関数

行ベースの統合関数を使用して、レコードグループのレコードを選択します。行ベースの統合関数は、IF-THEN-ELSE 文内で使用する必要があります。

CONSOL_GETROWFIELD

行ベースの統合関数によって識別された行を読み取り、指定したポートの値を返します。ポートを指定するには、数値の引数を使用します。

CONSOL_GETROWFIELD 関数は以下のいずれかの行ベースの統合関数と組み合わせて使用する必要があります。

- CONSOL_MODELEXACT
- CONSOL_MOSTDATA
- CONSOL_MOSTFILLED

行ベースの統合関数の各入力ポートに、CONSOL_GETROWFIELD 関数のインスタンスを 1 つ使用する必要があります。

構文

CONSOL_GETROWFIELD(*value*)

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>value</i>	必須	行ベースの統合関数の入力ポートを示す数字。関数の一番左のポートを指定するには、「0」を使用します。他のポートを示すには、後続の数字を使用します。

戻り値

指定したポートの値。関数は行ベースの統合関数によって識別された行からこの値を読み取ります。

例

以下の式は、CONSOL_GETROWFIELD 関数を CONSOL_MOSTDATA 関数と組み合わせて使用しています。

```
IF (CONSOL_MOSTDATA(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
```

```
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF
```

この式で、CONSOL_MOSTDATA 関数はレコードグループの行を分析し、単一行を識別します。CONSOL_GETROWFIELD 関数は連続した数字を使用して、その行のポート値を読み取り、その値を出力ポートに書き込みます。

CONSOL_MODELEXACT

最も頻度の高い値の数が最大の行を識別します。

例えば、レコードグループ内に最も頻度の高い値を含む 3 つのポートを持つ行があるとします。この行の最も頻度の高い値の数は「3」です。

この関数は CONSOL_GETROWFIELD 関数を組み合わせて使用する必要があります。CONSOL_GETROWFIELD は、CONSOL_MODELEXACT 関数が識別した行からの値を返します。

構文

```
CONSOL_MODELEXACT(string1, [string2, ..., stringN,]
group by)
```

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

行が、最も頻度の高いフィールドの最大数を獲得する場合は TRUE、他のすべての行の場合は FALSE。

例

以下の式は、CONSOL_MODELEXACT 関数を使用して、最も頻度の高いフィールドの数が最大の行を検索します。

```
IF (CONSOL_MODELEXACT(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF
```

この式で、CONSOL_MODELEXACT 関数はレコードグループの行を分析し、単一行を識別します。CONSOL_GETROWFIELD 関数は連続した数字を使用して、その行のポート値を読み取り、その値を出力ポートに書き込みます。

CONSOL_MOSTDATA

すべてのポートで最多数の文字を含む行を識別します。

この関数は CONSOL_GETROWFIELD 関数を組み合わせて使用する必要があります。CONSOL_GETROWFIELD は、CONSOL_MOSTDATA 関数が識別した行からの値を返します。

構文

```
CONSOL_MOSTDATA(string1, [string2, ..., stringN,]
group by)
```

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

すべてのポートで最も多い文字を含む行の場合は TRUE、他のすべての行の場合は FALSE。

例

以下の式は、CONSOL_MOSTDATA 関数を使用して、最多数の文字を含む行を検索します。

```
IF (CONSOL_MOSTDATA(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF
```

この式で、CONSOL_MOSTDATA 関数はレコードグループの行を分析し、単一行を識別します。CONSOL_GETROWFIELD 関数は連続した数字を使用して、その行のポート値を読み取り、その値を出力ポートに書き込みます。

CONSOL_MOSTFILLED

空白以外のフィールド数が最大の行を識別します。

この関数は CONSOL_GETROWFIELD 関数を組み合わせて使用する必要があります。CONSOL_GETROWFIELD は、CONSOL_MOSTFILLED 関数が識別した行からの値を返します。

構文

```
CONSOL_MOSTFILLED(string1, [string2, ..., stringN],
group by)
```

以下の表に、このコマンドの引数を示します。

引数	必須/オプション	説明
<i>string</i>	必須	入力ポート名。
<i>Group By</i>	必須	グループ識別子を含む入力ポートの名前。

戻り値

空白以外のフィールド数が最大の行の場合は TRUE、他のすべての行の場合は FALSE。

例

以下の式は、CONSOL_MOSTFILLED 関数を使用して、最多数の文字を含む行を検索します。

```
IF (CONSOL_MOSTFILLED(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
```

```

ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF

```

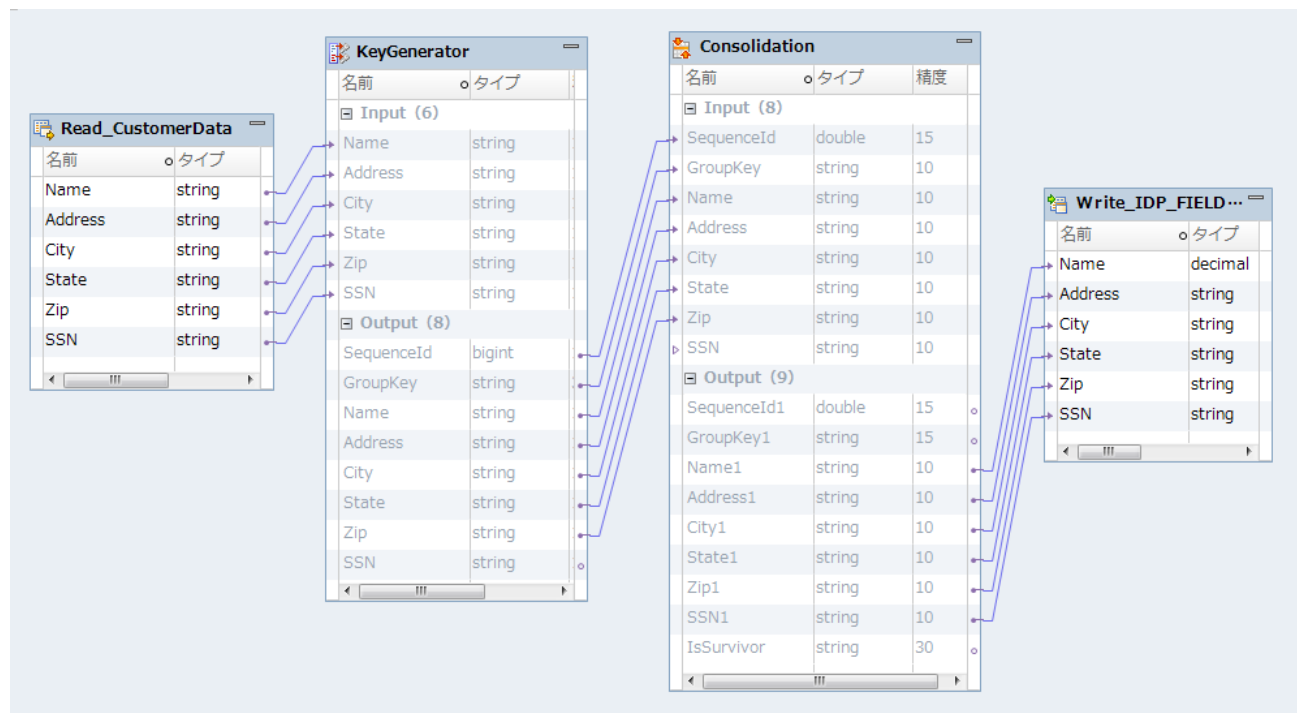
この式で、CONSOL_MOSTFILLED 関数はレコードグループの行を分析し、単一行を識別します。CONSOL_GETROWFIELD 関数は連続した数字を使用して、その行のポート値を読み取り、その値を出力ポートに書き込みます。

統合マッピングの例

あなたの会社では、重複する顧客レコードを統合することになりました。顧客レコードを統合するには、キージェネレータ変換でデータをグループ化し、統合変換を使用しレコードを統合します。

顧客レコードを含むデータソース、キージェネレータ変換、統合変換、およびデータターゲットのマッピングを作成します。このマッピングは、カスタムレコードをグループ化してそのグループを統合し、1つの統合されたレコードを書き込みます。

以下の図はマッピングを示しています。



入力

分析する入力データには、顧客情報が含まれています。

以下の表に、この例の入力データを示します。

名前	住所	市区町村	都道府県	郵便番号	SSN
Dennis Jones	100 All Saints Ave	New York	NY	10547	987-65-4320

名前	住所	市区町村	都道府県	郵便番号	SSN
Dennis Jones	1000 Alberta Rd	New York	NY	10547	987-65-4320
D Jones	100 All Saints Ave	New York	NY	10547-1521	

キージェネレータトランスフォーメーション

キージェネレータトランスフォーメーションを使用し、ZIP code ポートに基づいて入力データをグループ化します。

トランスフォーメーションは以下のデータを返します。

SequenceId	GroupKey	名前	住所	市区町村	都道府県	郵便番号	SSN
1	10547	Dennis Jones	100 All Saints Ave	New York	NY	10547	987-65-4320
2	10547	Dennis Jones	1000 Alberta Rd	New York	NY	10547	
3	10547	D Jones	100 All Saints Ave	New York	NY	10547-1521	987-65-4320

統合トランスフォーメーション

統合トランスフォーメーションを使用して、統合されたレコードを生成します。

行ベースストラテジタイプを使用するように統合トランスフォーメーションを設定します。Modal Exact ストラテジを選択し、最も頻度の高い値の数が最大の行を選択します。Modal Exact ストラテジは、その行からの値を使用して、統合されたレコードを生成します。統合されたレコードは、IsSurvivor ポートに「Y」の値が含まれているレコードです。

トランスフォーメーションは以下のデータを返します。

GroupKey	名前	住所	市区町村	都道府県	郵便番号	SSN	IsSurvivor
10547	Dennis Jones	100 All Saints Ave	New York	NY	10547	987-65-4320	N
10547	Dennis Jones	1000 Alberta Rd	New York	NY	10547		N
10547	D Jones	100 All Saints Ave	New York	NY	10547-1521	987-65-4320	N
10547	D Jones	100 All Saints Ave	New York	NY	10547-1521	987-65-4320	Y

統合マッピング出力

マッピング出力が統合されたレコードのみを含むように統合トランスフォーメーションを設定します。

この例では、Modal Exact ストラテジにより選択された最も頻度の高い値が正しいポート値であることはほぼ信頼できます。統合されたレコードのみをマッピングターゲットに書き込むには、**【詳細】** ビューを選択して出力モードを「存続のみ」に設定します。

マッピングの実行時に、マッピング出力には統合されたレコードのみが含まれます。

統合トランスフォーメーションの設定

統合トランスフォーメーションを設定する際は、ストラテジタイプ、ストラテジまたは式の記述、グループ化ポートを選択し、詳細オプションを設定します。

1. **【コンソリデーション】** ビューを選択します。
2. ストラテジタイプを選択します。
3. ストラテジを設定します。

- 単純ストラテジタイプの場合は、ポートごとにストラテジを選択します。
 - 行ベースストラテジタイプの場合は、ストラテジを選択します。
 - 詳細ストラテジタイプの場合は、統合関数を使用する式を作成します。
4. [グループ別] フィールドで、グループ識別子を含むポートを選択します。
 5. 入力データがソートされていない場合、[詳細] ビューでソートを有効にします。
 6. 統合されたレコードまたはすべてのレコードを含むように出力を設定します。

非ネイティブ環境での統合トランスフォーメーション

非ネイティブ環境でのアドレスバリデータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングで制限付きでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでの統合トランスフォーメーション

統合トランスフォーメーションのサポートには、次の制限があります。

- このトランスフォーメーションは、それぞれの環境でレコードを異なる順序で処理する場合があります。
- このトランスフォーメーションは、それぞれの環境で異なるレコードを存続レコードとして特定する場合があります。

Spark エンジンでの統合トランスフォーメーション

統合トランスフォーメーションのサポートには、次の制限があります。

- このトランスフォーメーションは、それぞれの環境でレコードを異なる順序で処理する場合があります。
- このトランスフォーメーションは、それぞれの環境で異なるレコードを存続レコードとして特定する場合があります。

Databricks Spark エンジンでの統合トランスフォーメーション

非ネイティブ環境での統合トランスフォーメーションは、そのトランスフォーメーションを実行するエンジンに依存します。

次の非ネイティブランタイムエンジンのサポートを検討します。

- Databricks Spark エンジン。サポートしません。

第 12 章

1028 データマスキングトランスフォーマーション

この章では、以下の項目について説明します。

- [データマスキングトランスフォーマーションの概要, 198 ページ](#)
- [マスキング方法, 199 ページ](#)
- [マスキングルール, 209 ページ](#)
- [特殊マスク形式, 213 ページ](#)
- [デフォルト値ファイル, 217 ページ](#)
- [データマスキングトランスフォーマーションの設定, 218 ページ](#)
- [データマスキングトランスフォーマーションランタイムプロパティ, 220 ページ](#)
- [データマスキングの例, 221 ページ](#)
- [データマスキングトランスフォーマーションの詳細プロパティ, 223 ページ](#)
- [非ネイティブ環境でのデータマスキングトランスフォーマーション, 223 ページ](#)

データマスキングトランスフォーマーションの概要

データマスキングトランスフォーマーションは、機密性が高い実稼働データを、非プロダクション環境向けの現実的なテストデータに変換します。データマスキングトランスフォーマーションは、カラムごとに設定されたマスキング方法に基づいてソースデータを変更します。

ソフトウェア開発、テスト、トレーニング、およびデータマイニング用にマスクされたデータを作成します。マスクされたデータ内のデータリレーションシップを保持し、データベーステーブル間の参照整合性を保持することができます。

データマスキングトランスフォーマーションでは、カラムに設定したソースのデータ型およびマスキング方法に基づいてマスキングルールが提供されます。文字列では、文字列内の置換する文字を制限できます。マスクで適用する文字を制限できます。数と日付の場合は、マスクされたデータの数の範囲を指定できます。範囲は、元の数に対する固定偏差またはパーセント偏差に基づいて設定できます。データ統合サービスでは、トランスフォーマーションに設定したロケールに基づいて文字を置き換えます。

マスキング方法

マスキング方法は、選択したカラムに適用するデータマスキングのタイプです。

入力カラムに、以下のいずれかのマスキング方法を選択できます。

ランダム

同じソースデータとマスキングルールに対して再現可能でないランダムな結果を生成します。マスクできるのは、日付、数値、および文字列データ型です。ランダムマスキングでは seed 値は不要です。ランダムマスキングの結果は、確定的ではありません。

式

式をソースカラムに適用して、データを作成またはマスクします。すべてのデータ型をマスクできます。

キー

ソースデータを再現可能な値に置き換えます。データマスキングトランスフォーメーションによって、同じソースデータ、マスキングルール、および seed 値に対して確定的な結果が生成されます。マスクできるのは、日付、数値、および文字列データ型です。

置換

データのカラムを、ディクショナリ内の似ているが関連のないデータに置き換えます。マスクできるのは、文字列データ型です。

依存

あるソースカラムの値を別のソースカラムの値に基づいて置き換えます。マスクできるのは、文字列データ型です。

トークン化

ソースデータを、カスタマイズしたマスク基準に基づいて生成データと置き換えます。データマスキングトランスフォーメーションは、カスタマイズしたアルゴリズムで指定されるルールを適用します。マスクできるのは、文字列データ型です。

特殊マスク形式

クレジットカード番号、電子メールアドレス、IP アドレス、電話番号、SSN、SIN、または URL に対応します。データマスキングトランスフォーメーションでは、これらの一般的なセンシティブデータをインテリジェントにマスクするビルトインルールが適用されます。

マスキングなし

データマスキングトランスフォーメーションでは、ソースデータは変更されません。

デフォルトは、[マスキングなし] です。

ランダムマスキング

ランダムマスキングでは、非決定性のマスクされたデータがランダムに生成されます。データマスキングトランスフォーメーションでは、異なる行に同じソース値が出現する場合に、異なる値が返されます。データマスキングトランスフォーメーションによって返されるデータのフォーマットに影響を与えるマスキングルールを定義できます。ランダムマスキングでは、数値、文字列値、および日付値がマスクされます。

文字列値のマスキング

ランダムマスキングを設定すると、文字列カラムにランダムな出力が生成されます。出力文字列に含まれる各文字の制限を設定するには、マスク形式を定義します。ソース文字列の文字と結果文字列の置き換え文字を設定するには、マスクするソース文字とマスクに使用する文字を定義します。

文字列ポートには、以下のマスキングルールを適用できます。

範囲

文字列長の上限と下限を設定します。データマスキングトランスフォーメーションでは、文字列長の上限と下限の範囲内でランダムに構成された文字列が返されます。

マスク形式

置き換える文字の種類を入力データの文字ごとに定義します。各文字の種類を英文字、数字、または英数字に制限できます。

ソース文字列の文字

ソース文字列内のマスク対象文字を定義します。例えば、入力データに出現する各シャープ記号（#）をマスクできます。[ソース文字列の文字] が空白の場合は、すべての入力文字がマスクされます。

結果文字列の置換文字

対象文字列内の文字を、[結果文字列の文字] で定義された文字に置き換えます。例えば、各マスクに英文字の大文字 A～Z を含めるには、以下の文字を入力します。

ABCDEFGHIJKLMNOPQRSTUVWXYZ

数値のマスキング

数値データをマスクする場合は、カラムの出力データ範囲を設定できます。データマスキングトランスフォーメーションでは、ポート精度に応じて、範囲の上限と下限の間の値が返されます。範囲を定義するには、範囲の上限と下限を設定するか、元のソース値に対する偏差に基づくブラー範囲を設定します。

数値データには、以下のマスキングパラメータを設定できます。

範囲

出力値の範囲を定義します。データマスキングトランスフォーメーションでは、値の上限と下限の間にある数値データが返されます。

ブラー範囲

ソースデータに対する固定偏差またはパーセント偏差に基づく範囲として、出力値の範囲を定義します。データマスキングトランスフォーメーションでは、ソースデータの値に近い数値データが返されます。範囲とブラー範囲を両方設定することもできます。

日付値のマスキング

日付値をランダムマスキングでマスクするには、出力日の範囲を設定するか、偏差を選択します。偏差を設定する場合は、ブラー対象となる日付部分を選択します。選択できるのは年、月、日、時、分、または秒です。データマスキングトランスフォーメーションは、設定した範囲内の日付を返します。

日時の値をマスクする場合は、以下のマスキングルールを設定できます。

範囲

選択した日時の値に対して返す値の上限と下限を設定します。

ブラー

日付の単位に適用する偏差に基づいて日付をマスクします。データマスキングトランスフォーメーションでは、偏差の範囲内の日付が返されます。ブラーできるのは、年、月、日、時、分、または秒です。適用する低偏差と高偏差を選択します。

式マスキング

式マスキングは、データを変更または新しいデータを作成するための式をポートに適用します。式マスキングを設定するときは、式エディタで式を作成します。式を構築するための入力ポートおよび出力ポート、関数、変数、演算子を選択します。

複数のポートからのデータを連結して別のポートの値を作成できます。例えば、ログイン名を作成する必要があるとします。ソースには名と姓のカラムがあります。ルックアップファイルから名と姓をマスクします。データマスキングトランスフォーメーションで、Login という別のポートを作成します。Login ポートで、名の最初の文字を姓と連結するための式を設定します。

```
SUBSTR(FIRSTNM,1,1)||LASTNM
```

ポイントアンドクリックインタフェースを使用して関数、ポート、変数、および演算子を選択することで、式を作成するときのエラーを減らすことができます。

式エディタには、式マスキングに対して設定されていない出力ポートが表示されます。ある式からの出力を別の式に対する入力として使用することはできません。出力ポート名を手動で式に追加した場合、予期しない結果が発生することがあります。

式を作成するときは、その式が、ポートのデータタイプに一致する値を返すことを確認します。式ポートのデータタイプが数値であり、式のデータタイプが同じではない場合、データマスキングトランスフォーメーションではゼロが返されます。式ポートのデータタイプが文字列であり、式のデータタイプが同じではない場合、データマスキングトランスフォーメーションでは NULL 値が返されます。

再現可能な式マスキング

複数のテーブルでソースカラムが出現し、各テーブルのカラムを同じ値でマスクする必要がある場合、再現可能な式マスキングを設定します。

再現可能な式マスキングを設定すると、データマスキングトランスフォーメーションによって式の結果が格納テーブルに保存されます。別のソーステーブルで同じカラムが出現した場合、データマスキングトランスフォーメーションは式からマスク値を返すのではなく、格納テーブルからマスク値を返します。

ディクショナリ名

再現可能な式マスキングを設定する場合、ディクショナリ名を入力する必要があります。ディクショナリ名は、複数のデータマスキングトランスフォーメーションで、同じソース値から同じマスク値を生成できるようにするキーです。各データマスキングトランスフォーメーションで同じディクショナリ名を定義します。ディクショナリ名には、任意のテキストを指定できます。

格納テーブル

ストレージテーブルには、セッション間の再現可能な式マスキングの結果が含まれます。格納テーブルの行には、ソース列およびマスクされた値ペアが含まれています。式マスキング用の格納テーブルは、置換マスキング用の格納テーブルとは別のテーブルです。

データマスキングトランスフォーメーションでは、再現可能な式で値がマスクされるたびに、ディクショナリ名、ロケール、列名、および入力値によって格納テーブルの検索が行われます。格納テーブル内に行が見つかり、格納テーブルからマスクされた値が返されます。データマスキングトランスフォーメーションで行が見つからない場合、その列に対して式からマスク値が生成されます。

ストレージ内のデータの暗号化を解除した後、同じディクショナリ名をキーとして使用する場合は、式マスキング用の格納テーブルを暗号化する必要があります。

式マスキング用の格納テーブルの暗号化

格納テーブルは、トランスフォーメーション言語エンコード機能を使用して暗号化できます。ストレージの暗号化を有効にした場合は、格納テーブルを暗号化する必要があります。

1. IDENTITY_EXPRESSION_STORAGE 格納テーブルをソースとして使用し、マッピングを作成します。
2. データマスキングトランスフォーメーションを作成します。
3. マスクされた値ポートに式マスキング方法を適用します。
4. MASKEDVALUE ポートに対して次の式を使用します。
`Enc_Base64(AES_Encrypt(MASKEDVALUE, Key))`
5. ポートをターゲットにリンクします。

例

例えば、従業員テーブルに以下のカラムが含まれているとします。

FirstName
LastName
LoginID

データマスキングトランスフォーメーションでは、FirstName と LastName を組み合わせた式を使用して LoginID がマスクされます。式マスクを再現可能に設定します。再現可能なマスキングのキーとしてディクショナリ名を入力します。

Computer_Users テーブルには LoginID が含まれていますが、FirstName 列または LastName 列は含まれていません。

Dept
LoginID
Password

Computer_Users テーブル内の LoginID を同じ LoginID で従業員としてマスクするには、LoginID カラム用の式マスキングを設定します。再現可能なマスキングを有効にし、LoginID 従業員テーブルに対して定義した同じディクショナリ名を入力します。Integration Service では、格納テーブルから LoginID 値が取得されます。

Integration Service で LoginID の格納テーブル内の行が見つからない場合に使用するデフォルトの式を作成します。Computer_Users テーブルには FirstName カラムまたは LastName カラムがないので、式によって生成される LoginID にはあまり意味がありません。

格納テーブルスクリプト

Informatica は、格納テーブルを作成するために実行できるスクリプトを提供します。スクリプトは以下の場所に格納されています。

<PowerCenter installation directory>\client\bin\Extensions\DataMasking

ディクショナリには、Sybase、Microsoft SQL Server、IBM DB2、および Oracle の各データベース用のスクリプトが含まれています。各スクリプトの名前は、<Expression_<データベースタイプ>です。

式マスキングのルールとガイドライン

式マスキングには以下のルールおよびガイドラインを使用します。

- ある式からの出力を別の式に対する入力として使用することはできません。出力ポート名を手動で式に追加した場合、予期しない結果が発生することがあります。
- 式を作成する際は、ポイントアンドクリックを使用します。ポイントアンドクリックインタフェースを使用して関数、ポート、変数、および演算子を選択することで、式を作成するときのエラーを減らすことができます。

- 再現可能なマスキング用にデータマスキングトランスフォーメーションが設定されており、格納テーブルが存在しない場合、Integration Service はソースデータをデフォルト値で置換します。

キーマスキング

キーマスキングに対して設定されたカラムは、ソース値とシード値が同じ場合に、マスクされた確定的なデータを返します。データマスキングトランスフォーメーションは、カラムに対して一意の値を返します。

キーマスキングに対してカラムを設定すると、データマスキングトランスフォーメーションによって、そのカラムのシード値が生成されます。異なるデータマスキングトランスフォーメーションで再現可能なデータを生成するようにシード値を変更できます。例えば、キーマスキングを設定して、参照整合性を強制します。あるテーブルのプライマリキーと別のテーブルの外部キーをマスクするには、同じ seed 値を使用します。

データマスキングトランスフォーメーションによって返されるデータのフォーマットに影響を与えるマスキングルールを定義できます。文字列値と数値のマスクには、キーマスキングを使用します。

文字列値のマスキング

文字列の再現可能な出力を生成するために、キーマスキングを設定できます。マスク形式を設定して、出力文字列に含まれる各文字に対する制限を定義します。マスクするソース文字を定義するソース文字列の文字を設定します。結果文字列の置換文字を設定して、マスクされたデータを特定の文字に制限します。

キーマスキング文字列には、以下のマスキングルールを設定できます。

シード

シード値を適用し、列に対して確定的なマスクされた値を生成します。1 から 1,000 までの数値を入力できます。

マスク形式

置き換える文字の種類を入力データの文字ごとに定義します。各文字の種類を英文字、数字、または英数字に制限できます。

ソース文字列の文字

ソース文字列内のマスク対象文字を定義します。例えば、入力データに出現する各シャープ記号 (#) をマスクできます。[ソース文字列の文字] が空白の場合は、すべての入力文字がマスクされます。データマスキングトランスフォーメーションは、ソース文字列の文字数が結果文字列の文字数より少ない場合、一意のデータが返されないことがあります。

結果文字列の文字

対象文字列内の文字を、[結果文字列の文字] で定義された文字に置き換えます。例えば、各マスクに英文字の大文字をすべて含めるには、以下の文字を入力します。

ABCDEFGHIJKLMNOPQRSTUVWXYZ

数値のマスキング

決定性出力が生成されるようにするには、数値ソースデータにキーマスキングを設定します。カラムに対して数値キーマスキングを設定する場合、カラムにランダムなシード値を割り当てます。データマスキングトランスフォーメーションによってソースデータがマスクされる場合、seed を必要とするマスキングアルゴリズムが適用されます。

同じソース値が別のカラムに出現した場合に再現可能な結果が生成されるようにするには、カラムの seed 値を変更します。たとえば、2 つのテーブル間でプライマリキーと外部キーのリレーションが維持されるようにします。この場合、各データマスキングトランスフォーメーションで、プライマリキーカラムの seed 値と外部キーカラムの seed 値として同じ seed 値を入力します。データマスキングトランスフォーメーション

によって、同じ数値に対して確定的な結果が生成されます。これにより、この2つのテーブル間で参照整合性が維持されるようになります。

日時の値のマスキング

日時の値に対してキーマスキングを設定できる場合、データマスキングトランスフォーメーションでシードとしてランダムな数値が必要になります。カラム間で再現可能な日時の値を返すには、別のカラムのシード値に一致するようにシードを変更できます。

データマスキングトランスフォーメーションは、キーマスキングで1753～2400の日付をマスクできます。ソース年がうるう年の場合、データマスキングトランスフォーメーションは同じくうるう年の年を返します。ソース月に31日が含まれる場合、データマスキングトランスフォーメーションは31日を含む月を返します。ソース月が2月の場合、データマスキングトランスフォーメーションは2月を返します。

データマスキングトランスフォーメーションは、常に有効な日付を生成します。

置換マスキング

置換マスキングでは、データのカラムを似ているが関連のないデータに置き換えることができます。置換マスキングを使用して、プロダクションデータを現実的なテストデータに置き換えます。置換マスキングを設定する場合は、代替値が含まれるディクショナリを定義します。

データマスキングトランスフォーメーションでは、設定したディクショナリでのルックアップが実行されます。データマスキングトランスフォーメーションでは、ソースデータがディクショナリから取得したデータに置き換えられます。ディクショナリファイルには、文字列データ、日時の値、整数、および浮動小数点数を含めることができます。日時の値は次の形式で入力します。

mm/dd/yyyy

データは、再現可能な値または再現不可能な値に置き換えることができます。再現可能な値を選択すると、データマスキングトランスフォーメーションで、同じソースデータおよびシード値に対する確定的な結果が生成されます。データを確定的な結果に置き換えるには、シード値を設定する必要があります。Integration Service では、ソースの格納テーブルと、再現可能なマスキングに使用するマスキング値が保持されます。

データの複数のカラムを同じディクショナリ行のマスキング値に置き換えることができます。1つの入力カラムに対して置換マスキングを設定します。同じディクショナリ行からマスキングデータを受け取る他のカラムに対しては、依存データマスキングを設定します。

ディクショナリ

ディクショナリは、置換データとテーブル内の各行のシリアル番号が含まれている参照テーブルです。モデルリポジトリにインポートするフラットファイルまたはリレーショナルテーブルから置換マスキングの参照テーブルを作成します。

データマスキングトランスフォーメーションでは、シリアル番号単位でディクショナリ行を取得するために番号が生成されます。データマスキングトランスフォーメーションでは、再現可能な置換マスキング用にハッシュキーが生成されます。再現不可能なマスキング用には、乱数が生成されます。再現可能な置換マスキングを設定する場合は、追加のルックアップ条件を設定することができます。

データマスキングトランスフォーメーションでは、ディクショナリを設定して、複数のポートをマスクすることができます。

データマスキングトランスフォーメーションがディクショナリから置換データを取得するとき、置換データの値が元の値と同じであるかどうかは確認されません。例えば、データマスキングトランスフォーメーションがJohnという名前をディクショナリファイルの同じ名前（John）に置き換える可能性があります。

次の例に、名と性別が含まれるディクショナリテーブルを示します。

SNO	GENDER	FIRSTNAME
1	M	Adam
2	M	Adeel
3	M	Adil
4	F	Alice
5	F	Alison

このディクショナリでは、行の最初のフィールドはシリアル番号、2 番目のフィールドは性別です。Integration Service では、常にシリアル番号でディクショナリのレコードが検索されます。再現可能なマスキングを設定する場合、ルックアップ条件として性別を追加することができます。Integration Service では、ハッシュキーを使用して行がディクショナリから取得され、ソースデータ内の性別と一致する性別の行が検出されます。

参照テーブルを作成する場合は、以下のルールおよびガイドラインに従います。

- テーブル内の各レコードには、シリアル番号が存在する必要があります。
- シリアル番号は、1 から始まる連続した整数です。シリアル番号のシーケンスには、欠番が存在してはなりません。
- シリアル番号のカラムは、テーブル行の任意の場所に配置できます。任意のラベルを設定することもできます。

フラットファイルテーブルを使用して参照テーブルを作成する場合は、以下のルールおよびガイドラインに従います。

- フラットファイルテーブルの第 1 行には、各レコードのフィールドを識別するためにカラムラベルを設定する必要があります。フィールドは、カンマで区切ります。1 行目にカラムラベルが含まれていない場合、Integration Service では最初の行のフィールド値がカラム名として処理されます。
- Windows でフラットファイルテーブルを作成し、UNIX マシンにコピーする場合は、ファイルが UNIX に適した形式であることを確認します。例えば、Windows と UNIX では、行末マーカに異なる文字を使用します。

格納テーブル

データマスキングトランスフォーメーションには、セッションごとに再現可能な置換用に格納テーブルが維持されています。格納テーブルの行には、ソースカラムおよびマスクされた値ペアが含まれています。データマスキングトランスフォーメーションでは、再現可能な代替値で値がマスクされるたびに、ディクショナリ名、ロケール、カラム名、入力値、およびシードで格納テーブルの検索が行われます。行が見つかったと、格納テーブルからマスクされた値が返されます。行が見つからない場合、データマスキングトランスフォーメーションはハッシュキーを使用してディクショナリから行を取得します。

格納テーブル内のディクショナリ名の形式は、フラットファイルディクショナリとリレーショナルディクショナリでは異なります。フラットファイルディクショナリの名前は、ファイル名によって識別されます。リレーショナルディクショナリの名前の構文は次のとおりです。

<Connection object>_<dictionary table name>

Informatica は、リレーショナル格納テーブルを作成するために実行できるスクリプトを提供します。スクリプトは以下の場所に格納されています。

<PowerCenter Client installation directory>\client\bin\Extensions\DataMasking

ディクショナリには、Sybase、Microsoft SQL Server、IBM DB2、および Oracle の各データベース用のスクリプトが含まれています。各スクリプトの名前は、Substitution_<データベースタイプ>です。SQL 文およびプライマリキー制約を設定する場合は、別のデータベースでテーブルを作成できます。

ストレージ内のデータの暗号化を解除した後、同じシード値とディレクトリを使用して同じカラムを暗号化する場合は、置換マスキング用の格納テーブルを暗号化する必要があります。

置換マスキング用の格納テーブルの暗号化

格納テーブルは、トランスフォーメーション言語エンコード機能を使用して暗号化できます。ストレージの暗号化を有効にした場合は、格納テーブルを暗号化する必要があります。

1. IDENTITY_SUBSTITUTION_STORAGE 格納テーブルをソースとして使用し、マッピングを作成します。
2. データマスキングトランスフォーメーションを作成します。
3. 入力値とマスクされた値ポートに、式マスキング方法を適用します。
4. INPUTVALUE ポートに対して次の式を使用します。
`Enc_Base64(AES_Encrypt(INPUTVALUE, Key))`
5. MASKEDVALUE ポートに対して次の式を使用します。
`Enc_Base64(AES_Encrypt(MASKEDVALUE, Key))`
6. ポートをターゲットにリンクします。

置換マスキングプロパティ

置換マスキングには、以下のマスキングルールを設定できます。

- **再現可能な出力。** セッションごとに確定的な結果を返します。データマスキングトランスフォーメーションは、マスクされた値をストレージテーブルに格納します。
- **シード値。** シード値を適用し、カラムに対して確定的なマスクされた値を生成します。1~1,000 の範囲で数値を入力してください。
- **一意の出力。** データマスキングトランスフォーメーションに一意の入力値に対して一意の出力値を作成させます。同じ出力値に対して 2 つの入力値はマスクされません。ディクショナリに、一意の出力を有効にするのに十分な一意の行が必要です。
一意の出力を無効にすると、データマスキングトランスフォーメーションが一意の出力値に対する入力値をマスクしない場合があります。ディクショナリにいくつかの行が含まれている場合があります。
- **一意のポート。** 置換マスキングの一意のレコードの特定に使用されるポート。例えば、Customer というテーブルで名をマスクするとします。名が一意のポートとして含まれるテーブルカラムを選択した場合、データマスキングトランスフォーメーションは重複する名をマスクされた同じ値に置き換えます。Customer_ID カラムを一意のポートとして選択した場合、データマスキングトランスフォーメーションはそれぞれの名を一意の値に置き換えます。
- **ディクショナリ情報。** 置換データ値が含まれる参照テーブルを設定します。【ソースの選択】をクリックして参照テーブルを選択します。
 - **ディクショナリ名。** 選択した参照テーブルの名前を表示します。
 - **出力カラム。** データマスキングトランスフォーメーションに返すカラムを選択します。
- **ルックアップ条件。** 置換マスキングで使用するディクショナリ行の適正を評価するには、ルックアップ条件を設定します。ルックアップ条件は、SQL クエリの WHERE 句に似ています。ルックアップ条件を設定する場合、ソース内のカラムとディクショナリ内のカラムを比較します。

たとえば、ファーストネームをマスクするとします。ソースデータおよびディクショナリには、ファーストネームの列と性別の列があります。それぞれの女性のファーストネームをディクショナリの女性の名前で置

き換えるという条件を追加できます。ルックアップ条件は、ソース内の性別とディクショナリ内の性別を比較します。

- **入力ポート**。ルックアップで使用するソースデータカラム。
- **ディクショナリカラム**。入力ポートと比較するディクショナリカラム。

置換マスキングのルールおよびガイドライン

置換マスキングに使用されるルールおよびガイドラインは、次のとおりです。

- 一意の再現可能な置換マスクの格納テーブルが存在しない場合、セッションは失敗します。
- ディクショナリに行が含まれていない場合は、データマスキングトランスフォーメーションがエラーメッセージを返します。
- データマスキングトランスフォーメーションで格納テーブル内にロケール、ディクショナリ、シードを持つ入力値が見つかった場合、その行がディクショナリ内にすでに存在しない場合でもマスク値が取得されます。
- 接続オブジェクトの削除またはディクショナリの変更を行った場合、格納テーブルが切り詰められます。そうでない場合、予期しない結果になる可能性があります。
- ディクショナリ内の値の数がソースデータ内の一意の値の数よりも少ない場合、データマスキングトランスフォーメーションでは一意の再現可能な値でデータをマスクすることができません。データマスキングトランスフォーメーションはエラーメッセージを返します。

依存マスキング

依存マスキングは、ソースデータの複数のカラムを同じディクショナリ行のデータに置き換えます。

データマスキングトランスフォーメーションが複数のカラムに対して置換マスキングを実行すると、マスクされたデータに現実的でないフィールドの組み合わせが含まれる場合があります。同じディクショナリ行の複数の入力カラムのデータを置き換えるために、依存マスキングを設定することができます。マスクされたデータは、「New York, New York」や「Chicago, Illinois」などの有効な組み合わせを受け取ります。

依存マスキングを設定するときは、最初に置換マスキングの入力カラムを設定します。その置換カラムに依存するように他の入力カラムを設定します。例えば、置換マスキングの郵便コードカラムを選択し、郵便番号カラムに依存する市区町村カラムと州カラムを選択します。依存マスキングにより、置き換えられた市区町村と州の値が、置き換えられた郵便番号の値に対して有効になります。

注: 最初に置換マスキングのカラムを設定せずに依存マスキングのカラムを設定することはできません。

依存マスキングのカラムを設定するときは、次のマスキングルールを設定します。

依存カラム

置換マスキングに対して設定した入力カラムの名前。データマスキングトランスフォーメーションは、そのカラムのマスキングルールを使用して、ディクショナリルールの置換データを受け取ります。置換マスキングに対して設定するカラムは、ディクショナリからマスクされたデータを受け取るためのキーカラムになります。

出力カラム

依存マスキングで設定するカラムの値が含まれるディクショナリカラムの名前。

依存マスキングの例

データマスキングディクショナリには、次の値の住所行が含まれている可能性があります。

SNO	STREET	CITY	STATE	ZIP	COUNTRY
1	32 Apple Lane	Chicago	IL	61523	US
2	776 Ash Street	Dallas	TX	75240	US
3	2229 Big Square	Atleeville	TN	38057	US
4	6698 Cowboy Street	Houston	TX	77001	US

ソースデータは、住所ディクショナリの市区町村、州、および郵便番号の有効な組み合わせでマスクする必要があります。

置換マスキングの ZIP ポートを設定します。ZIP ポートに対して次のマスキングルールを入力します。

ルール	値
辞書名	Address
シリアル番号カラム	SNO
出力カラム	ZIP

依存マスキングの City ポートを設定します。City ポートに対して次のマスキングルールを入力します。

ルール	値
依存カラム	ZIP
出力カラム	City

依存マスキングの State ポートを設定します。State ポートに対して次のマスキングルールを入力します。

ルール	値
依存カラム	ZIP
出力カラム	State

データマスキングトランスフォーメーションが郵便番号をマスクするときは、ディクショナリ行からその郵便番号に対して正しい市区町村と州を返します。

トークン化のマスキング

トークン化のマスキング技法を使用して、アルゴリズム内で指定する基準に基づいてソース文字列データをマスキングすることができます。例えば、ソースデータ内のフィールドエントリを置き換える架空の電子メールアドレスが入ったアルゴリズムを作成することができます。

トークン化のマスクを使用することで、マスクされたデータの形式を設定することができます。それを利用できるようにするには、その前にトークナイザ名をマスクアルゴリズムに割り当てる必要があります。トークナイザ名は、使用されるマスクアルゴリズム（JAR）を参照します。トークン化のマスキング技法を適用するときは、トークナイザ名を指定します。

トークン化のマスキングの設定

トークン化マスキング技法を使用する前に、以下のタスクを実行します。

1. tokenprovider というディレクトリ（<Informatica_home>\services\shared）に移動します。
2. XML ファイル（com.informatica.products.ilm.tx-tokenizerprovider.xml）を開きます。
3. 使用する各トークナイザに対して、トークナイザ名とクラスファイルの完全修飾名を追加します。トークナイザのクラスを、tokenprovider ディレクトリ内の com.informatica.products.ilm.tx-tokenprovider-<Build-Number>.jar クラスの中に実装します。各トークナイザに対して、次の例のように情報を XML ファイルに入力します。

```
<TokenizerProvider>
<Tokenizer Name="CCTokenizer"
ClassName="com.informatica.tokenprovider.CCTokenizer"/>
</TokenizerProvider>
```

ここで、

- 「Tokenizer Name」は、引用符で括った部分がユーザー定義の名前です。
- ClassName は、CLASSNAME 属性に対するユーザー定義の名前です。これを、com.informatica.products.ilm.tx-tokenprovider-<Build-Number>.jar の中から実装します。

設定後、トークン化のマスキング技法が使用できます。マッピング作成時に使用するアルゴリズムを指定する際、トークナイザ名を入力してください。

マスキングルール

マスキングルールは、マスキング方法の選択後に設定するオプションです。

ランダムまたはキーマスキング方法を選択する場合、マスク形式、ソース文字列の文字、および結果文字列の文字を設定できます。ランダムマスキングを持つ範囲またはブラーを設定できます。

以下の表で、各マスキング方法に設定できるマスキングルールについて説明します。

マスキングルール	説明	マスキング方法	ソースのデータ型
マスク形式	出力文字列内の各文字を英文字、数字、または英数字に制限するマスクです。	ランダムおよびキー	String
ソース文字列の文字	マスクするソース文字セット、またはマスク対象から除外するソース文字セットです。	ランダムおよびキー	String

マスキングルール	説明	マスキング方法	ソースのデータ型
結果文字列の置換文字	マスクに含める文字セットまたはマスクから除外する文字セットです。	ランダムおよびキー	String
範囲	出力値の範囲です。 - 数値。 データマスキングトランスフォーメーションでは、値の上限と下限の間にある数値データが返されます。 - 文字列。上限と下限の文字列長の範囲内で、ランダムな文字で構成される文字列を返します。 - 日付／時刻。日時の最大値と最小値の範囲内で日付と時間を返します。	ランダム	Numeric String 日付/時刻
ブラー	ソースデータに対する固定偏差またはパーセント偏差の範囲の出力値です。データマスキングトランスフォーメーションでは、ソースデータの値に近いデータが返されます。日時のカラムは固定偏差である必要があります。カラムは固定偏差である必要があります。	ランダム	Numeric 日付/時刻

マスク形式

出力カラム内の各文字を英文字、数字、または英数字に制限するには、マスク形式を設定します。以下の文字を使用して、マスク形式を定義します。

A, D, N, X, +, R

注: マスク形式には、大文字を使用します。マスク文字として小文字を入力すると、データマスキングトランスフォーメーションによって大文字に変換されます。

以下の表では、マスク形式文字について説明します。

文字	説明
A	英文字。例えば、ASCII 文字 a～z、A～Z です。
D	数字 0～9。データマスキングトランスフォーメーションは、数字 0～9 以外の文字に対して「X」を返します。
N	英数字。例えば、ASCII 文字 a～z、A～Z、および 0～9 です。
X	任意の文字。たとえば、英文字や記号です。
+	マスキングなし。
R	残りの文字。R は、文字列内のその他の文字に、任意の種類の文字を使用できることを示します。R は、マスクの最後の文字にする必要があります。

たとえば、部署名のフォーマットが以下のとおりであるとします。

nnn-<department_name>

最初の 3 文字を数値に限定し、部署名を英文字に限定し、ダッシュを出力に残すマスクを設定できます。この場合、マスクフォーマットを以下のように設定します。

DDD+AAAAAAAAAAAAAAAA

データマスキングトランスフォーメーションによって、最初の 3 文字が数値に置き換えられます。4 番目の文字は置き換えられません。残りの文字は、英文字に置き換えられます。

マスクフォーマットの定義がない場合、各ソース文字は任意の文字に置き換えられます。マスクフォーマットが入力文字列より長い場合は、マスクフォーマットの余分な文字が無視されます。マスクフォーマットがソース文字列より短い場合、データマスキングトランスフォーメーションは残りの文字を「R」でマスクします。

注: 範囲オプションを使用してマスクフォーマットを設定することはできません。

ソース文字列の文字

ソース文字列の文字とは、マスクするように選択した文字列、またはマスクしないように選択したソース文字のことです。ソース文字列内での文字の位置は、重要ではありません。ソース文字では、大文字と小文字が区別されます。

設定できる文字数に制限はありません。[文字] が空白の場合は、カラム内のすべてのソース文字列が置き換えられます。

ソース文字列の文字について、以下のいずれかのオプションを選択します。

指定文字のみマスク

データマスキングトランスフォーメーションでは、ソースに含まれる文字のうち、ソース文字列の文字として設定されている文字がマスクされます。たとえば、文字 A、B、および c を入力すると、ソースデータに出現した A、B、または c が別の文字に置き換えられます。A、B、または c ではないソース文字は置き換えられません。マスクでは、大文字と小文字が区別されます。

指定文字以外をすべてマスク

ソース文字列に出現したソース文字列の文字を除いて、すべての文字をマスクします。たとえば、フィルタソース文字 "-" を入力し、[指定文字以外をすべてマスク] を選択した場合は、文字 "-" がソースデータに出現しても置き換えられません。それ以外のソース文字は変更されます。

ソース文字列の例

ソースファイルに [従属] という名前のカラムがあるとします。[従属] カラムには、カンマで区切られた複数の名前が含まれています。[従属] カラムをマスクし、名前を区切るカンマをテストデータで保持する必要があります。

[従属] カラムに対して、[ソース文字列の文字] を選択します。[マスクしない] を選択し、対象外の文字として "," を入力します。引用符は入力しないでください。

データマスキングトランスフォーメーションによって、ソース文字列内のカンマ以外の文字がすべて置き換えられます。

結果文字列の置換文字

結果文字列の置換文字は、マスクされたデータに含める置き換え文字として選択した文字です。結果文字列の置換文字を設定すると、データマスキングトランスフォーメーションによって、ソース文字列内の文字が結果文字列の置換文字に置き換えられます。異なる入力値に対して同じ出力が生成されないようにするには、設定する置換文字の範囲を広くするか、マスクするソース文字の数を少なくします。文字列内での文字の位置は、重要ではありません。

結果文字列の置換文字について、以下のいずれかのオプションを選択します。

指定文字のみ使用

結果文字列の置換文字として定義した文字のみを使用してソースをマスクします。例えば、文字 A、B、および c を入力すると、ソースカラムの各文字が A、B、または c に置き換えられます。"horse" という単語は、"BAcBA" などに置き換えられます。

指定文字以外をすべて使用

結果文字列の置換文字として定義した文字以外を使用してソースをマスクします。例えば、結果文字列の置換文字として A、B、および c を入力すると、マスクされたデータに文字 A、B、または c は出現しません。

結果文字列の置換文字の例

[従属] カラムに含まれるすべてのカンマをセミコロンに置き換えるには、以下のタスクを完了します。

1. ソース文字列の文字としてカンマを設定し、[指定文字のみマスク] を選択します。
データマスキングトランスフォーメーションでは、[従属] カラムにカンマが出現した場合にのみ、カンマがマスクされます。
2. 結果文字列の置換文字としてセミコロンを設定し、[指定文字のみ使用] を選択します。
データマスキングトランスフォーメーションでは、[従属] カラムにカンマが出現するたびにカンマがセミコロンに置き換えられます。

範囲

数値、日付、または文字列データの範囲を定義します。数値または日付値の範囲を定義すると、データマスキングトランスフォーメーションによって、ソースデータが上限値と下限値の範囲内の値でマスクされます。文字列の範囲を設定する場合は、文字列長の範囲を設定します。

文字列の範囲

ランダム文字列マスキングを設定すると、データマスキングトランスフォーメーションによって、ソース文字列の長さとは異なる長さの文字列が生成されます。オプションで、文字列長の上限と下限を設定できます。文字列長の上限および下限として入力する値は、正の整数である必要があります。長さは、ポート精度以下である必要があります。

数値の範囲

数値カラムの上限値と下限値を設定します。上限値は、ポート精度以下である必要があります。デフォルトの範囲は、1 からポート精度長までです。

日付範囲

日時の値の上限値と下限値を設定します。上限と下限の各フィールドには、デフォルトの日付の上限と下限が表示されます。デフォルトの日付形式は、MM/DD/YYYY HH24:MI:SS です。上限の日付は、下限の日付より後である必要があります。

ブラー

ブラーでは、ソースデータ値に対する固定偏差またはパーセント偏差の範囲の出力値が生成されます。元の値に近いランダムな値が返されるようにする場合は、ブラーを設定します。ブラーの対象は、数値および日付値です。

数値のブラー

ソース数値のブラー方法として、固定偏差またはパーセント偏差を選択します。低ブラー値は、ソース値より小さい値に関する偏差です。高ブラー値は、ソース値より大きい値に関する偏差です。どちらの値もゼロ以上である必要があります。データマスキングトランスフォーメーションによって返されるマスクされたデータで、数値データは定義した値の範囲内の値に置き換えられます。

以下の表に、入力ソース値が 66 の場合のブラーの範囲値に応じたマスキング結果を示します。

ブラーの種類	低	高	結果
固定長	0	10	66～76
固定長	10	0	56～66
固定長	10	10	56～76
Percent	0	50	66～99
Percent	50	0	33～66
Percent	50	50	33～99

日付値のブラー

ブラーを設定すると、ソース日付に対する偏差で日付をマスクできます。偏差の適用対象の日付単位を選択します。年、月、日、または時を選択できます。ソース日付単位の上下の偏差を定義するには、上限と下限を入力します。データマスキングトランスフォーメーションは、偏差を適用し、偏差の範囲内の日付を返します。

例えば、マスクされた日付をソース日付の 2 年以内に制限するには、単位として年を選択します。高低の境界として 2 を入力します。ソースデータが 02/02/2006 の場合、データマスキングトランスフォーメーションは 02/02/2004 から 02/02/2008 の日付を返します。

デフォルトのブラー単位は年です。

特殊マスク形式

特殊マスク形式は、一般的なタイプのデータに適用できるマスクです。特殊マスク形式では、データマスキングトランスフォーメーションで、現実的な形式を持つものの有効な値ではない、マスクされた値が返されます。

例えば、SSN をマスクすると、データマスキングトランスフォーメーションにより形式は正しいものの有効ではない SSN が返されます。社会保障番号には、再現可能なマスキングを設定できます。

以下のタイプのデータの、特殊マスクを設定します。

- 社会保障番号
- クレジットカード番号
- 電話番号
- URL アドレス
- 電子メールアドレス
- IP アドレス

- 社会保険番号

ソースデータの形式またはデータ型がマスクとして無効の場合、Data Integration Service では、デフォルトのマスクがデータに適用されます。Integration Service では、デフォルト値のファイルからマスクされた値が適用されます。デフォルト値のファイルを編集してデフォルト値を変更できます。

クレジットカード番号のマスクング

データマスクングトランスフォーメーションでは、有効なクレジットカード番号をマスクする場合、論理的に有効なクレジットカード番号が生成されます。ソースクレジットカード番号の長さは 13~19 桁です。入力クレジットカード番号に、クレジットカード業界のルールに基づく有効なチェックサム値がある必要があります。

ソースクレジットカード番号には、数字、スペース、およびハイフンを使用できます。クレジットカード番号に無効な文字が含まれている場合、または長さが正しくない場合、Integration Service によって、セッションログにエラーが書き込まれます。ソースデータが無効の場合、Integration Service によって、デフォルトのクレジットカード番号マスクが適用されます。

6 桁の銀行識別番号 (BIN) はマスクされません。クレジットカード番号 4539 1596 8210 2773 は、4539 1516 0556 7067 などとしてマスクされます。データマスクングトランスフォーメーションによって生成されるマスクされた番号は、有効なチェックサムを持ちます。

電子メールアドレスマスクング

データマスクングトランスフォーメーションを使用し、文字列値が含まれる電子メールアドレスをマスクします。データマスクングトランスフォーメーションを使用すると、電子メールアドレスをランダムな ASCII 文字でマスクすることも、電子メールアドレスを現実的な電子メールアドレスに置き換えることもできます。

電子メールアドレスには次のタイプのマスクングを適用できます。

標準の電子メールアドレスマスクング

データマスクングトランスフォーメーションは、電子メールアドレスをマスクするときにランダムな ASCII 文字を返します。例えば、Georgesmith@yahoo.com が KtrlupQAPyk@vdSKh.BIC などとしてマスクされます。デフォルトは標準のマスクングです。

詳細電子メールアドレスマスクング

データマスクングトランスフォーメーションは、トランスフォーメーション出力ポートまたはディクショナリカラムから派生した他の現実的な電子メールアドレスを使用して電子メールアドレスをマスクします。

高度な電子メールアドレスマスクング

タイプが高度な電子メールアドレスマスクングであれば、電子メールを実際にありそうな別の電子メールでマスクすることができます。データマスクングトランスフォーメーションでは、辞書カラムまたはトランスフォーメーション出力ポートから電子メールアドレスを作成します。

マッピング出力ポートから電子メールアドレスのローカル部分を作成することができます。または、電子メールアドレスのローカル部分をリレーショナルテーブルまたはフラットファイルカラムから作成することができます。

データマスクングトランスフォーメーションでは、定数値から、またはドメインディクショナリのランダム値から電子メールのドメイン名を作成することができます。

次のオプションに基づいて高度な電子メールアドレスマスクングを作成することができます。

依存ポートに基づく電子メールアドレス

データマスクングトランスフォーメーション出力ポートに基づいて電子メールアドレスを作成することができます。名前カラムと苗字カラムのトランスフォーメーション出力ポートを選択します。データマスク

ングトランスフォーメーションでは、名前と苗字の長さとして指定した値に基づいて、名前か苗字、またはその両方をマスクします。

ディクショナリに基づいた電子メール

ディクショナリからカラムに基づいて電子メールアドレスを作成することができます。参照テーブルをディクショナリのソースとして選択します。

名前と苗字のためのディクショナリカラムを選択します。データマスキングトランスフォーメーションでは、名前と苗字の長さとして指定した値に基づいて、名前か苗字、またはその両方をマスクします。

詳細電子メールアドレスマスキングタイプの設定パラメータ

詳細電子メールアドレスマスキングを設定する際には、設定パラメータを指定します。

以下のパラメータを指定できます。

区切り文字

電子メールアドレス内の名と姓を区切るために、ドット、ハイフン、アンダースコアなどの区切り文字を選択できます。電子メールアドレス内の名と姓を区切らない場合は、区切り文字を空欄のままにします。

【名】 カラム

電子メールアドレス内の名をマスクするデータマスキングトランスフォーメーション出力ポートまたはディクショナリカラムを選択します。

【姓】 カラム

電子メールアドレス内の姓をマスクするデータマスキングトランスフォーメーション出力ポートまたはディクショナリカラムを選択します。

【名】 カラムまたは 【姓】 カラムの長さ

【名】 カラムと 【姓】 カラムでマスクする文字列の長さを制限します。例えば、入力データの名が Timothy で、姓が Smith であるとします。【名】 カラムの長さを 5、【姓】 カラムの長さを 1 に選択し、区切り文字にはドットを選択したとします。この設定では、データマスキングトランスフォーメーションによって次の電子メールアドレスが生成されます。

```
timot.s@<domain_name>
```

DomainName

ドメイン名には、gmail.com のような一定の値を使用できます。または、ドメイン名の一覧が含まれている別のディクショナリファイルを指定することもできます。ドメインディクショナリは、フラットファイルでもリレーショナルテーブルでもかまいません。

IP アドレスマスキング

データマスキングトランスフォーメーションでは、IP アドレスをマスクする場合、ピリオドで区切られた 4 つの数による別の IP アドレスとしてマスクされます。最初の数値はネットワークを表します。ネットワーク番号は、ネットワークの範囲内でマスクされます。

クラス A の IP アドレスはクラス A の IP アドレスにマスクされ、10.x.x.x アドレスは 10.x.x.x アドレスにマスクされます。クラスとプライベートネットワークアドレスはマスクされません。例えば、11.12.23.34 は 75.32.42.52 に、10.23.24.32 は 10.61.74.84 に、それぞれマスクされます。

注: 多数の IP アドレスをマスクした場合に、IP アドレスのクラスやプライベートネットワークがマスクされないために、データマスキングトランスフォーメーションで非固有値が戻されることがあります。

電話番号マスキング

データマスキングトランスフォーメーションでは、元の電話番号のフォーマットが変更されることなく、電話番号がマスクされます。例えば、電話番号(408)382 0658 は、(607)256 3106 などとしてマスクされます。

ソースデータには、数字、スペース、ハイフン、およびかっこを使用できます。Integration Service では、英文字または特殊文字はマスクされません。

データマスキングトランスフォーメーションは、string、integer、bigint データをマスクできます。

社会保障番号（SSN）のマスキング

データマスキングトランスフォーメーションでは、社会保障庁による最新のハイグループ履歴リストに基づいて、無効な社会保障番号が生成されます。ハイグループ履歴リストには、社会保障庁が発行した有効な番号が記載されています。

デフォルトのハイグループ履歴リストは、以下の場所にあるテキストファイルです。

```
<Installation Directory>\infa_shared\SrcFiles\highgroup.txt
```

ワークフローでハイグループ履歴リストを使用するには、このテキストファイルを Data Integration Service に対して設定するソースディレクトリにコピーします。

データマスキングトランスフォーメーションでは、ハイグループ履歴リストにない SSN 番号が生成されます。社会保障庁では、ハイグループ履歴リストを毎月更新しています。このリストの最新バージョンは、以下の場所からダウンロードします。

<http://www.socialsecurity.gov/employer/ssns/highgroup.txt>

社会保障番号（SSN）形式

データマスキングトランスフォーメーションは、9 桁の数字を含む SSN 形式を受け付けます。この数字は任意の文字で区切ることができます。例えば、データマスキングトランスフォーメーションは以下の形式を受け付けます。+54-*9944\$#789-,*()”。

地域コードの要件

データマスキングトランスフォーメーションは、ソースと同じ形式で有効でない社会保障番号を返します。SSN の最初の 3 桁では地域コードが定義されます。地域コードはマスクされません。グループ番号とシリアル番号はマスクされます。ソース SSN には有効な地域コードが含まれている必要があります。データマスキングトランスフォーメーションは、ハイグループリストから地域コードを検索し、マスクされたデータとして適用できる未使用の番号の範囲を判断します。SSN が無効な場合、ソースデータはマスクされません。

再現可能な社会保障番号のマスキング

データマスキングトランスフォーメーションでは、再現可能なマスキングが設定された確定的な社会保障番号が返されます。データマスキングトランスフォーメーションでは、社会保障庁が発行した有効な社会保障番号が返されないため、一意のすべての社会保障番号を返すことはできません。

URL アドレスマスキング

データマスキングトランスフォーメーションでは、'://'文字列を検索し、その右側の部分文字列を解析することによって URL が解析されます。ソース URL には、'://'文字列が含まれている必要があります。ソース URL には、数字と英文字を使用できます。

URL のプロトコル部分はマスクされません。URL が `http://www.yahoo.com` の場合は、`http://MgL.aHjCa.VsD/` が返されます。データマスキングトランスフォーメーションは、有効でない URL を生成する可能性があります。

注: URL の場合は、常に ASCII 文字が返されます。

社会保険番号のマスキング

このデータマスキングトランスフォーメーションでは、9 桁の社会保険番号がマスクされます。番号は任意の文字で区切ることができます。

番号に区切り文字が含まれない場合、マスクされた番号に区切り文字は含まれません。それ以外の場合、マスクされた番号は以下の形式になります。

XXX-XXX-XXX

再現可能な SIN 番号

再現可能な SIN 値を返すようにデータマスキングトランスフォーメーションを設定することができます。再現可能な SIN マスキングに対応するようポートを設定すると、データマスキングトランスフォーメーションでは、ソース SIN 値とシード値が同じ場合に、マスクされた確定的なデータが返されます。

再現可能な SIN 番号を返すには、**【再現可能な値】** を有効にしてシード番号を入力します。データマスキングトランスフォーメーションは、各 SIN に対して一意の値を返します。

SIN の開始桁

マスクされた SIN の最初の桁を定義できます。

【開始桁】 を有効にして、桁を数字で入力します。データマスキングトランスフォーメーションを実行すると、ここで入力した桁より上をマスクした SIN 番号が作成されます。

デフォルト値ファイル

ソースデータの形式またはデータ型がマスクとして無効の場合、Data Integration Service では、デフォルトのマスクがデータに適用されます。Integration Service では、デフォルト値のファイルからマスクされた値が適用されます。デフォルト値のファイルを編集してデフォルト値を変更できます。

デフォルト値ファイルは、以下の場所にある XML ファイルです。

<インストールディレクトリ>\infa_shared\SrcFiles\defaultValue.xml

ワークフローでデフォルト値ファイルを使用するには、デフォルト値ファイルを Data Integration Service に対して設定するソースディレクトリにコピーします。

defaultValue.xml ファイルには、以下の名前と値のペアが用意されています。

```
<?xml version="1.0" standalone="yes" ?>
<defaultValue
  default_char = "X"
  default_digit = "9"
  default_date = "11/11/1111 00:00:00"
  default_email = "abc@xyz.com"
  default_ip = "99.99.9.999"
  default_url = "http://www.xyz.com"
  default_phone = "999 999 999 9999"
  default_ssn = "999-99-9999"
  default_cc = "9999 9999 9999 9999"
```

```
default_sin = "999-999-999"  
default_seed = "500"/>
```

関連項目：

- [「データ統合サービスの設定」 \(ページ 218\)](#)

データマスキングトランスフォーメーションの設定

以下の手順を使用して、データマスキングトランスフォーメーションを設定します。

1. Data Integration Service の実行オプションを設定します。
2. トランスフォーメーションを作成します。
3. 入力ポートを定義します。
4. 変更する各ポートのマスキングルールを設定します。
5. データをプレビューして結果を確認します。

データ統合サービスの設定

Informatica Administrator (Administrator ツール) でデータ統合サービスの実行オプションを設定することができます。

実行オプションを設定して、次のデフォルトのディレクトリを設定します。

- ホームディレクトリ。ソースディレクトリとキャッシュディレクトリが含まれます。
- ソースディレクトリ。ワークフローのソースファイルが含まれます。例えば、ソースディレクトリには highgrp.txt および defaultvalue.xml ファイルを含めることができます。
- キャッシュディレクトリ。置換マスキングのキャッシュファイルが含まれます。

実行オプションの値を設定するには、Administrator ツールを開き、**ドメインナビゲータ**でデータ統合サービスを選択します。【プロパティ】ビューをクリックしてから、【実行オプション】セクションの【編集】をクリックします。

関連項目：

- [「デフォルト値ファイル」 \(ページ 217\)](#)

データマスキングトランスフォーメーションの作成

データマスキングトランスフォーメーションは Developer ツールで作成します。

データマスキングトランスフォーメーションを作成する前に、ソースを作成します。フラットファイルまたはリレーショナルデータベーステーブルを物理データオブジェクトとしてインポートします。

1. 【Object Explorer】ビューで、プロジェクトまたはフォルダーを選択します。
2. 【ファイル】 > 【新規】 > 【トランスフォーメーション】をクリックします。
【新規】ダイアログボックスが表示されます。
3. データマスキングトランスフォーメーションを選択します。
4. 【次へ】をクリックします。
5. トランスフォーメーションの名前を入力します。

6. **【完了】** をクリックします。
トランスフォーメーションがエディタに表示されます。

ポートの定義

【概要】 ビューで、データマスキング入力ポートを追加します。入力ポートの作成時に、Developer ツールにより対応する出力ポートがデフォルトで作成されます。出力ポートの名前は入力ポートと同じです。

1. **【概要】** ビューで、**【新規】** をクリックしてポートを追加します。
2. カラムのデータ型、精度、スケールを設定します。
カラムのマスキングルールを定義する前に、カラムのデータ型を設定する必要があります。
3. ポートのデータマスキングを設定するには、**【概要】** ビューのマスキングタイプカラムの矢印をクリックします。

各ポートのデータマスキングの設定

【データマスク】 ダイアログボックスで、ポートのマスキング方法と対応するマスキングルールを選択します。
【データマスク】 ダイアログボックスは、**【ポート】** タブの **【データマスキング】** カラムをクリックすると表示されます。

1. 選択したポートにマスキングを設定するには、**【マスキングの適用】** を有効にします。
マスキングしているポートのデータ型に基づいて、使用できるマスキング方法のリストが Developer ツールによって表示されます。
2. リストからマスキング方法を選択します。
選択するマスキング方法に基づいて、各種マスキングルールが Developer ツールによって表示されます。一部の特殊なマスク形式には、設定するマスキングルールがありません。
3. マスキングルールを設定します。
4. **【OK】** をクリックして、ポートのデータマスキング設定を適用します。
ポートのデータマスキングを定義すると、Developer ツールにより **out-<ポート名>** という出力ポートが作成されます。**<ポート名>** は入力ポートと同じ名前です。データマスキングトランスフォーメーションにより、**out-<ポート名>** ポートのマスクされたデータが返されます。

マスクされたデータのプレビュー

【データビューア】 でデータマスキングトランスフォーメーションの結果を表示する際、マスクされたデータと元のデータを比較できます。

1. データマスキングトランスフォーメーションポートとマスキングルールの設定後、物理データオブジェクトソースとデータマスキングトランスフォーメーションを含むマッピングを作成します。
2. ソースをデータマスキングトランスフォーメーションに接続します。
3. データ統合サービスがアクセス可能な共有場所に、ソースのデータがあることを確認します。
4. データマスキングトランスフォーメーションをクリックして、マッピングで選択します。
5. **【データビューア】** をクリックして、**【実行】** をクリックします。

Developer ツールで、すべてのデータマスキングトランスフォーメーション出力ポートのデータが表示されます。**【出力】** プレフィックスを持つポートに、マスクされたデータが含まれます。**【データビューア】** ビューで、マスクされたデータと元のデータを比較できます。

データマスキングトランスフォーメーションランタイムプロパティ

データマスキングトランスフォーメーションのランタイムプロパティを設定すると、パフォーマンスを向上させることができます。

次のようにランタイムプロパティを設定します。

キャッシュサイズ

メインメモリのディクショナリキャッシュのサイズ。パフォーマンスを向上させるには、メモリサイズを増やします。推奨される最小サイズは、100,000 レコードに対して 32 MB です。デフォルトは 8MB です。

キャッシュディレクトリ

ディクショナリキャッシュの場所。そのディレクトリに対する書き込み権限が必要です。デフォルトは CacheDir です。

共有ストレージテーブル

データマスキングトランスフォーメーションのインスタンス間でのストレージテーブルの共有を可能にします。共有ストレージテーブルは、データマスキングトランスフォーメーションのインスタンスで、データベース接続、シード値、およびロケールに対して同じディクショナリカラムが使用されているときに有効にします。共有ストレージテーブルは、同じデータマスキングトランスフォーメーション内の 2 つのポートで、接続、シード、およびロケールに対して同じディクショナリカラムが使用されているときにも有効にできます。データマスキングトランスフォーメーションまたはポートでディクショナリカラムが共有されていないときは、共有ストレージテーブルを無効にします。デフォルトでは無効になっています。

ストレージのコミット間隔

ストレージテーブルに一度にコミットする行数。パフォーマンスを向上させるには、この値を増やします。コミット間隔は、共有ストレージテーブルを設定しないときに設定します。デフォルトは 100,000 です。

ストレージの暗号化

IDM_SUBSTITUTION_STORAGE、IDM_EXPRESSION_STORAGE などのストレージテーブルを暗号化します。暗号化ストレージプロパティを有効にする前に、ストレージテーブル内のデータが暗号化されていることを確認してください。ストレージテーブルを暗号化しない場合は、このオプションを無効にします。デフォルトでは無効になっています。

ストレージの暗号化キー

データマスキングトランスフォーメーションは、ストレージ暗号化キーに基づいてストレージを暗号化します。同一のデータマスキングトランスフォーメーションインスタンスの各セッションの実行には、同じ暗号化キーを使用します。

置換ディクショナリのオーナー名

置換マスキングタイプを選択する場合の置換ディクショナリテーブルのオーナー名です。データベース接続で指定されたデータベースユーザーがセッションの置換ディクショナリテーブルのオーナーではない場合、テーブルのオーナーを指定する必要があります。

ストレージのオーナー名

再現可能な式マスキングタイプまたは一意の再現可能な置換マスキングタイプを選択する場合、IDM_SUBSTITUTION_STORAGE または IDM_EXPRESSION_STORAGE のテーブルオーナー名です。

データマスキングの例

開発者は、カスタマアプリケーション用のテストデータを作成する必要があります。データには、他の開発者が会社の開発環境でアクセス可能な現実的なカスタマデータを含める必要があります。

開発者は、カスタマ ID、クレジットカード番号、所得などのマスクされたカスタマデータを返すデータサービスを作成します。マッピングには、カスタマデータを変換するデータマスキングトランスフォーメーションが含まれます。

以下の図はマッピングを示しています。



マッピングには、以下のトランスフォーメーションが含まれています。

- Read_Customer_Data。カスタマのクレジットカード情報および所得情報が含まれます。
- Customer_Data_Masking トランスフォーメーション。FirstName と LastName を除くすべてのコラムをマスクします。データマスキングトランスフォーメーションによって、マスクされたコラムがターゲットに渡されます。
- Customer_TestData。マスクされたカスタマデータを受け取る出力トランスフォーメーション。

Read_Customer Data

カスタマデータには、以下のコラムが含まれます。

コラム	データ型
CustomerID	Integer
LastName	String
FirstName	String
CreditCard	String
Income	Integer
Join_Date	日時 (MM/DD/YYYY)

以下の表に、カスタマデータのサンプルを示します。

CustomerID	LastName	FirstName	CreditCard	Income	JoinDate
0095	Bergeron	Barbara	4539-1686-3069-3957	12000	12/31/1999
0102	Brosseau	Derrick	5545-4091-5232-8948	4000	03/03/2011
0105	Anderson	Lauren	1234-5678-9012-3456	5000	04/03/2009
0106	Boonstra	Pauline	4217-9981-5613-6588	2000	07/07/2007
0107	Chan	Brian	4533-3156-8865-3156	4500	06/18/1995

カスタマデータマスキングトランスフォーメーション

データマスキングトランスフォーメーションでは、最初と最後の名前以外、カスタマ行のすべてのカラムをマスクします。

データマスキングトランスフォーメーションでは、次のタイプのマスキングを実行します。

- キーマスキング
- ランダムマスキング
- クレジットカードマスキング

以下の表に、データマスキングトランスフォーメーションでの各ポートのマスキングルールを示します。

入力ポート	マスキングの種類	マスキングルール	説明
CustomerID	キー	シードは 934 です。 カスタマ ID に、マスク形式はありません。 結果文字列の置換文字は 1234567890 です。	カスタマ ID マスクは確定的です。 マスクされたカスタマ ID には数字が含まれます。
LastName	マスキングなし	-	-
FirstName	マスキングなし	-	-
CreditCard	CreditCard	-	データマスキングトランスフォーメーションでは、クレジットカード番号を有効なチェックサムを持つ他の番号でマスクします。
Income	ランダム	ブラー Percent 下限 = 1 上限 = 10	マスクされた Income は、源泉所得の 10 パーセントの範囲内になります。
JoinDate	ランダム	ブラー 単位 = 年 下限 = 5 上限 = 5	マスクされた Date は元の日付から 5 年以内になります。

カスタマテストデータ結果

Customer_TestData トランスフォーメーションは、データマスキングトランスフォーメーションから実際のカスタマデータを受け取ります。

Customer_TestData ターゲットは、以下のデータを受け取ります。

out-CustomerID	out-LastName	outFirstName	out-CreditCard	out-Income	out-JoinDate
3954	Bergeron	Barbara	4539-1625-5074-4106	11500	03/22/2001
3962	Brosseau	Derrick	5545-4042-8767-5974	4300	04/17/2007
3964	Anderson	Lauren	1234-5687-2487-9053	5433	09/13/2006
3965	Boonstra	Pauline	4217-9935-7437-4879	1820	02/03/2010
3966	Chan	Brian	4533-3143-4061-8001	4811	10/30/2000

Income は、元の所得の 10 パーセントの範囲内になります。JoinDate は元の日付から 5 年以内になります。

データマスキングトランスフォーメーションの詳細プロパティ

Data Integration Service でデータマスキングトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

非ネイティブ環境でのデータマスキングトランスフォーメーション

非ネイティブ環境でのデータマスキングトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

Blaze エンジンでのデータマスキングトランスフォーメーション

データマスキングトランスフォーメーションのサポートには、次の制限があります。

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションが、再現可能な式のマスキングに対して設定されている。
- トランスフォーメーションが、再現可能な一意の置換に対して設定されている。

このエンジンでは次のマスキング方法を使用できます。

クレジットカード

Email

式

IP アドレス

キー

電話番号

ランダム

SIN

SSN

トークン化
URL
ランダムな置換
再現可能な置換
ランダムな置換による依存
再現可能な置換による依存

Spark エンジンでのデータマスキングトランスフォーメーション

データマスキングトランスフォーメーションのサポートには、次の制限があります。

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションが、再現可能な式のマスキングに対して設定されている。
- トランスフォーメーションが、再現可能な一意の置換に対して設定されている。

このエンジンでは次のマスキング方法を使用できます。

クレジットカード
Email
式
IP アドレス
キー
電話番号
ランダム
SIN
SSN
トークン化
URL
ランダムな置換
再現可能な置換
ランダムな置換による依存
再現可能な置換による依存

データマスキングトランスフォーメーションのパフォーマンスを最適化するには、Hadoop 接続の次の Spark エンジン設定プロパティを設定します。

`spark.executor.cores`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するコアの数を示します。

`spark.executor.cores=1` に設定します。

`spark.executor.instances`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するインスタンスの数を示します。

`spark.executor.instances=1` に設定します。

`spark.executor.memory`

各実行プログラムプロセスが Spark エンジンでタスクレットを実行するために使用するメモリの量を示します。

`spark.executor.memory=3G` に設定します。

データマスキングトランスフォーメーションストリーミングマッピングでの

ストリーミングマッピングには、Spark エンジン上のバッチマッピングと同じ処理ルールがあります。

第 13 章

データプロセッサトランスフォーメーション

この章では、以下の項目について説明します。

- [データプロセッサトランスフォーメーションの概要, 226 ページ](#)
- [データプロセッサトランスフォーメーションのビュー, 227 ページ](#)
- [データプロセッサトランスフォーメーションのポート, 228 ページ](#)
- [スタートアップコンポーネント, 230 ページ](#)
- [参照, 231 ページ](#)
- [データプロセッサトランスフォーメーションの設定, 232 ページ](#)
- [イベント, 239 ページ](#)
- [ログ, 240 ページ](#)
- [データプロセッサトランスフォーメーションの開発, 242 ページ](#)
- [データプロセッサトランスフォーメーションのインポートとエクスポート, 247 ページ](#)
- [非ネイティブ環境でのデータプロセッサトランスフォーメーション, 249 ページ](#)

データプロセッサトランスフォーメーションの概要

データプロセッサトランスフォーメーションは、マッピングで非構造化ファイル形式および半構造化ファイル形式を処理します。メッセージング形式、HTML ページ、XML、JSON、および PDF ドキュメントを処理するトランスフォーメーションを設定します。また、ACORD、HIPAA、HL7、EDI-X12、EDIFACT、SWIFT などの構造化形式も変換できます。

マッピングではデータプロセッサトランスフォーメーションを使用して、ドキュメントの形式を別の形式に変更します。データプロセッサトランスフォーメーションはマッピング内の任意の形式のファイルを処理します。データプロセッサトランスフォーメーションを作成する場合は、データを変換するコンポーネントを定義します。

データプロセッサトランスフォーメーションには、データを処理する複数のコンポーネントを含めることができます。各コンポーネントにはほかのコンポーネントが含まれることがあります。

例えば、顧客請求書を Microsoft Word ファイルで受け取っているとします。各 Word ファイルのデータを解析するようにデータプロセッサトランスフォーメーションを設定します。顧客データは顧客テーブルに抽出します。また、注文情報は注文テーブルに抽出します。

データプロセッサトランスフォーメーションを作成する場合は、XMap、スクリプト、またはライブラリを定義します。XMap は入力階層ファイルを別の構造の出力階層ファイルに変換します。ライブラリは、業界標準のメッセージングタイプを階層構造を備えた XML ドキュメントに変換するか、XML から業界標準の形式に変換します。スクリプトはソースドキュメントを階層形式に解析するか、階層形式を別のファイル形式に変換するか、あるいは階層ドキュメントを別の階層形式にマッピングすることができます。

スクリプトはデータプロセッサトランスフォーメーションの IntelliScript エディタで定義します。次のタイプのスクリプトを定義できます。

- パーサー。ソースドキュメントを XML に変換します。パーサーの出力は常に XML です。入力には、テキスト、HTML、Word、PDF、HL7 など任意の形式を使用できます。
- シリアライザ。XML ファイルを任意の形式の出力ドキュメントに変換します。シリアライザの出力は、テキストドキュメント、HTML ドキュメント、PDF など、任意の形式にすることができます。
- マッパー。XML ソースドキュメントを別の XML 構造またはスキーマに変換します。XMap 内と同じ XML ドキュメントを変換できます。
- トランスフォーマ。任意の形式のデータを変更します。テキストを追加、削除、変換、または変更します。パーサー、マッパー、またはシリアライザと共にトランスフォーマを使用します。また、トランスフォーマをスタンドアロンコンポーネントとして実行することもできます。
- ストリーマ。マルチギガバイトのデータストリームなどサイズの大きな入力ドキュメントをセグメントに分割します。ストリーマは、HIPAA ファイルや EDI ファイルなど、複数のメッセージやレコードを含むドキュメントを処理します。

詳細については、『*Data Transformation ユーザーガイド*』を参照してください。

データプロセッサトランスフォーメーションのビュー

データプロセッサトランスフォーメーションには、トランスフォーメーションを設定して Developer ツールで実行する際にアクセスするビューが複数あります。

データプロセッサトランスフォーメーションの一部のビューは、デフォルトでは Developer ツールに表示されません。トランスフォーメーションのビューを変更するには、**[ウィンドウ] > [ビューの表示] > [その他] > [Informatica]** をクリックします。そして、表示するビューを選択します。

データプロセッサトランスフォーメーションには次の固定ビューがあります。

概要ビュー

ポートを設定し、スタートアップコンポーネントを定義します。

リファレンスビュー

トランスフォーメーションへのスキーマの追加、またはトランスフォーメーションからのスキーマの削除を行います。

設定ビュー

エンコード、出力コントロール、および XML 生成に関するトランスフォーメーションの設定を行います。

オブジェクトビュー

トランスフォーメーションからスクリプト、Xmap、およびライブラリオブジェクトの追加、変更または削除を行います。

データプロセッサトランスフォーメーションの次のビューにもアクセスすることができます。

データプロセッサ 16 進数ソースビュー

入力ドキュメントが 16 進形式で表示されます。

データプロセッサイベントビュー

Developer ツールでトランスフォーメーションを実行するときに発生するイベントに関する情報が表示されます。初期化イベント、実行イベント、およびサマリイベントが表示されます。

スクリプトヘルプビュー

スクリプトエディタの文脈依存ヘルプが表示されます。

データビューアビュー

サンプル入力データの表示、トランスフォーメーションの実行、および出力結果の表示を行います。

データプロセッサトランスフォーメーションのポート

データプロセッサトランスフォーメーションのポートは、トランスフォーメーションの【概要】ビューで定義します。

データプロセッサトランスフォーメーションは、ファイル、バッファ、複合ファイルリーダーのストリームバッファからの入力を読み取ることができます。フラットファイルリーダーをバッファとして使用して、ファイル全体を一度に読み取ることができます。また、データベースから入力ファイルを読み取ることもできます。

作成する出力ポートは、トランスフォーメーションから文字列、複合ファイル、またはリレーショナルデータの行を返すかどうかによって異なります。

データプロセッサトランスフォーメーションの入力ポート

データプロセッサトランスフォーメーションを作成すると、Developer ツールによってデフォルトの入力ポートが作成されます。また、スクリプトのスタートアップコンポーネントに追加の入力ポートを定義すると、Developer ツールによってトランスフォーメーションに追加の入力ポートが作成されます。

入力タイプによって、Data Integration Service がデータプロセッサトランスフォーメーションに渡すデータのタイプが決まります。入力タイプによって、入力がデータであるかソースファイルパスであるかが決まります。

以下の入力タイプのいずれかを設定します。

バッファ

データプロセッサトランスフォーメーションは、入力ポートでソースデータ行を受け取ります。フラットファイルまたは Informatica トランスフォーメーションからデータを受け取るようにトランスフォーメーションを設定する場合は、バッファ入力タイプを使用します。

ファイル

データプロセッサトランスフォーメーションは、入力ポートでソースファイルパスを受け取ります。データプロセッサのスタートアップコンポーネントによってソースファイルは開かれます。ファイル入力タイプは、Microsoft Excel ファイルまたは Microsoft Word ファイルなどのバイナリファイルを解析するのに使用します。また、バッファ入力ポートで処理すると大量のシステムメモリが必要な場合がある大きなファイルにも、ファイル入力タイプを使用することができます。

サービスパラメータ

データプロセッサトランスフォーメーションは、サービスパラメータポートで、変数に適用する値を受け取ります。入力データを受け取る変数を選択すると、Developer ツールによって変数ごとにサービスパラメータポートが作成されます。

Output_Filename

行データではなくファイル名を返すようにデフォルトの出力ポートを設定すると、Developer ツールによって Output_Filename ポートが作成されます。マッピングから Output_Filename ポートにファイル名を渡すことができます。

入力ポートを定義する際には、ポートのサンプル入力ファイルの場所を定義することができます。サンプル入力ファイルは、入力ファイルの小さなサンプルです。スクリプトの作成時にはサンプル入力ファイルを参照します。また、**【データビューア】** ビューでトランスフォーメーションをテストするときにもサンプル入力ファイルを使用します。サンプル入力ファイルは**【入力場所】** フィールドに定義します。

サービスパラメータポート

変数の値を受け取る入力ポートを作成することができます。変数には、文字列、日付、数値など任意のデータ型を格納することができます。また、ソースドキュメントの場所を格納することもできます。変数は、データプロセッサコンポーネントで参照できます。

変数の入力ポートを作成するときは、Developer ツールに変数のリストが表示され、そこから選択することができます。

サービスパラメータポートの作成

変数の値を受け取る入力ポートを作成することができます。また、作成したポートを変数から削除することもできます。

1. データプロセッサトランスフォーメーションの**【概要】** ビューを開きます。
2. **【選択】** をクリックします。
Developer ツールに変数のリストが表示され、すでにポートが存在する変数が示されます。
3. 変数を 1 つ以上選択します。
Developer ツールによって、選択した変数ごとにバッファ入力ポートが作成されます。このポートを変更することはできません。
4. 作成したポートを変数から削除するには、変数リストの選択を無効にします。選択を無効にすると、Developer ツールによって入力ポートが削除されます。

データプロセッサトランスフォーメーションの出力ポート

データプロセッサトランスフォーメーションには、出力ポートがデフォルトで 1 つ存在します。スクリプトに追加の出力ポートを定義する場合、Developer ツールによって出力ポートがデータプロセッサトランスフォーメーションに追加されます。リレーショナルデータを返すようにトランスフォーメーションを設定する場合は、ポートのグループを作成することができます。また、サービスパラメータポートとパススルーポートを作成することもできます。

デフォルトの出力ポート

データプロセッサトランスフォーメーションには、出力ポートがデフォルトで 1 つ存在します。リレーショナル出力を作成する場合は、デフォルトの出力ポートではなく、関連出力ポートのグループを定義できます。ス

クリプトのコンポーネントに追加の出力ポートを定義すると、Developer ツールによってトランスフォーメーションに出力ポートが追加されます。

デフォルトの出力ポートには、次のいずれかの出力タイプを設定します。

バッファ

データプロセッサトランスフォーメーションが出力ポートを介して XML を返します。ドキュメントを解析する場合や、データプロセッサトランスフォーメーションで XML を別の XML ドキュメントにマップする場合は、バッファファイルタイプを選択します。

ファイル

Data Integration Service がソースインスタンスまたはソース行ごとに出力ポートに出力ファイル名を返します。データプロセッサトランスフォーメーションのコンポーネントは、データプロセッサトランスフォーメーションの出力ポートを介してデータを返す代わりに出力ファイルを書き込みます。

ファイル出力ポートを選択すると、Developer ツールによって Output_Filename 入力ポートが作成されます。この出力ファイル名ポートにはファイル名を渡すことができます。データプロセッサトランスフォーメーションは、このポートで受け取った名前で作成ファイルを作成します。

出力ファイル名が空白の場合、Data Integration Service は行エラーを返します。エラーが発生すると、Data Integration Service は出力ポートに NULL 値を書き込み、行エラーを返します。

XML を PDF ファイルや Microsoft Excel ファイルなどのバイナリデータファイルに変換する場合は、出力タイプとしてファイルを選択します。

パススルーポート

データプロセッサトランスフォーメーションに対してパススルーポートを設定できます。パススルーポートは、入力データを受信し、同じデータを変更せずにマッピングに返す入出力ポートです。

パススルーポートは、マッピング内にあるデータプロセッサトランスフォーメーションインスタンスで設定できます。

パススルーポートを追加するには、ポートをマッピング内の別のトランスフォーメーションからドラッグします。また、**【プロパティ】** ビューの **【ポート】** タブでポートを追加することもできます。パススルーポートを追加するには **【新規】** をクリックします。

注: リレーショナル入力と階層出力のあるデータプロセッサトランスフォーメーションにパススルーポートを追加する場合、リレーショナル構造のルートグループにポートを追加します。

データプロセッサトランスフォーメーションには、カスタムデータタイプのパススルーポートを含めることができます。

スタートアップコンポーネント

スタートアップコンポーネントは、データプロセッサトランスフォーメーションでの処理を開始するコンポーネントを定義します。**【概要】** ビューでスタートアップコンポーネントを設定します。

データプロセッサトランスフォーメーションには、データを処理する複数のコンポーネントを含めることができます。各コンポーネントにはほかのコンポーネントが含まれることがあります。トランスフォーメーションのエントリポイントがどのコンポーネントであるのかを識別する必要があります。

データプロセッサトランスフォーメーションでスタートアップコンポーネントを構成するときは、スタートアップコンポーネントとして XMap、ライブラリ、またはスクリプトコンポーネントを選択できます。スクリプトでは次のタイプのコンポーネントの中から 1 つを選択できます。

- パーサー。ソースドキュメントを XML に変換します。入力には、テキスト、HTML、Word、PDF、HL7 など任意の形式を使用できます。
- マッパー。XML ソースドキュメントを別の XML 構造またはスキーマに変換します。
- シリアライザ。XML ファイルを任意の形式の出力ドキュメントに変換します。
- ストリーマ。マルチギガバイトのデータストリームなどサイズの大きな入力ドキュメントをセグメントに分割します。
- トランスフォーマ。任意の形式のデータを変更します。テキストを追加、削除、変換、または変更します。パーサー、マッパー、またはシリアライザと共にトランスフォーマを使用します。また、トランスフォーマをスタンドアロンコンポーネントとして実行することもできます。

注: スタートアップコンポーネントが XMap またはライブラリではない場合は、**【概要】** ビューで行なう代わりにスクリプトでスタートアップコンポーネントを構成することもできます。

参照

参照として機能させるスキーマまたはマップレットを選択することで、スキーマ参照やマップレット参照などのトランスフォーメーション参照を定義することができます。一部のデータプロセッサトランスフォーメーションでは、トランスフォーメーションの関連コンポーネントに対して入力または出力階層を定義するための階層スキーマが必要です。トランスフォーメーションでスキーマを使用するには、そのトランスフォーメーションのスキーマ参照を定義します。また、RunMapplet アクションという専用アクションを使って、データプロセッサトランスフォーメーションからマップレットを呼び出すこともできます。マップレットを呼び出すには、最初にそのトランスフォーメーションのマップレット参照を定義する必要があります。

トランスフォーメーションの**【参照】**ビューで、スキーマ参照やマップレット参照などのトランスフォーメーション参照を定義できます。

スキーマ参照

データプロセッサトランスフォーメーションは、モデルリポジトリのスキーマオブジェクトを参照します。このスキーマオブジェクトは、トランスフォーメーションを作成する前のリポジトリに存在することができます。また、スキーマをトランスフォーメーションの**【リファレンス】**ビューからインポートすることもできます。

スキーマのエンコードは、Serializer オブジェクトまたは Mapper オブジェクトの入力エンコードに一致する必要があります。スキーマのエンコードは、Parser オブジェクトまたは Mapper オブジェクトの出力エンコードに一致する必要があります。トランスフォーメーションの**【設定】**ビューで使用中のエンコードを設定してください。

スキーマは別のスキーマを参照することができます。Developer ツールに、データプロセッサトランスフォーメーションが参照する各スキーマの名前空間およびプレフィックスが表示されます。名前空間が空になっている複数のスキーマを参照した場合、トランスフォーメーションは無効になります。

マップレット参照

RunMapplet アクションで、データプロセッサトランスフォーメーションからマップレットを呼び出すことができます。RunMapplet アクションをデータプロセッサトランスフォーメーションコンポーネントに追加する前に、呼び出すマップレットへの参照を最初に定義する必要があります。

データプロセッサトランスフォーメーションの設定

コードページ、XML 処理オプション、およびログの設定は、データプロセッサトランスフォーメーションの【設定】ビューで行います。

文字エンコード

文字エンコードは、言語または言語グループから 16 進コードへの文字のマッピングです。

スクリプトを設計する場合は、入力ドキュメントのエンコードと出力ドキュメントのエンコードを定義します。使用中のエンコードを定義して、IntelliScript エディタによる文字の表示方法や、データプロセッサトランスフォーメーションによる文字の処理方法を決定します。

使用中のエンコード

使用中のエンコードとは、メモリ内のデータのコードページと、ユーザーインターフェースおよび作業ファイルに表示されるデータのコードページのことです。データプロセッサトランスフォーメーションで参照するスキーマのエンコードと互換性がある使用中のエンコードを選択する必要があります。

以下の表に、使用中のエンコードの設定を示します。

設定	説明
データプロセッサのデフォルトのコードページを使用	データプロセッサトランスフォーメーションのデフォルトのエンコードを使用します。
その他	リストからエンコードを選択します。
XML 特殊文字のエンコード	XML 特殊文字の表現を決定します。【なし】 または 【XML】 を選択できます。 - なし。 & < > " 'のままで残す XML 特殊文字のエンティティ参照は、テキストとして解釈されます。 例えば、文字>は次のように表示されます。 > デフォルトはなしです。 - XML。& < > " 'に変換する XML 特殊文字のエンティティ参照は、通常文字として解釈されます。 例えば、>は次の文字として表示されます。 >

入力エンコード

入力エンコードは、入力ドキュメント内で文字データがどのようにエンコードされるかを決定します。スクリプトで、追加入力ポートのエンコードを構成できます。

以下の表に、**【入力】** 領域でのエンコード設定を示します。

設定	説明
入力ドキュメントで指定したエンコードを使用	ソースドキュメントで定義されているコードページ（XML ドキュメントのエンコード属性など）を使用します。 ソースドキュメントでエンコードが指定されていない場合、データプロセッサトランスフォーメーションは、 【設定】 ビューのエンコード設定を使用します。
使用中のエンコードを使用	使用中のエンコードと同じエンコードを使用します。
その他	ドロップダウンリストから入力エンコードを選択します。
XML 特殊文字のエンコード	XML 特殊文字の表現を決定します。 【なし】 または 【XML】 を選択できます。 <ul style="list-style-type: none"> - なし。 &amp; &lt; &gt; &quot; &apos; のままで残す XML 特殊文字のエンティティ参照は、テキストとして解釈されます。例えば、> は次のように表示されます。 &gt; デフォルトは 【なし】 です。 - XML。 & < > " ' に変換する XML 特殊文字のエンティティ参照は、通常文字として解釈されます。例えば、&gt; は次の文字として表示されます。 >
バイトオーダー	マルチバイト文字が入力ドキュメント内でどのように表示されるかを示します。次のオプションを選択することができます。 <ul style="list-style-type: none"> - リトルエンディアン。最初に最下位バイトが表示されます。デフォルト。 - ビッグエンディアン。最初に最上位バイトが表示されます。 - バイナリ変換なし。

出力エンコード

出力エンコードは、メインの出力ドキュメント内で文字データがどのようにエンコードされるかを決定します。

以下の表に、**【出力】** 領域でのエンコード設定を示します。

設定	説明
使用中のエンコードを使用	出力エンコードは使用中のエンコードと同じです。
その他	ユーザーはリストから出力エンコードを選択します。

設定	説明
XML 特殊文字のエンコード	<p>XML 特殊文字の表現を決定します。[なし] または [XML] を選択できます。</p> <ul style="list-style-type: none"> - なし。 & < > &quot; &apos; のままで残す XML 特殊文字のエンティティ参照は、テキストとして解釈されます。例えば、> は次のように表示されます。 &gt; デフォルト。 - XML。& < > " ' に変換する XML 特殊文字のエンティティ参照は、通常文字として解釈されます。例えば、&gt; は次の文字として表示されます。 >
入力エンコードと同じ	出力エンコードは入力エンコードと同じです。
バイトオーダー	<p>マルチバイト文字が入力ドキュメント内でどのように表示されるかを示します。次のオプションを選択することができます。</p> <ul style="list-style-type: none"> - リトルエンディアン。最初に最下位バイトが表示されます。デフォルト。 - ビッグエンディアン。最初に最上位バイトが表示されます。 - バイナリ変換なし。

文字エンコードのルールおよびガイドライン

エンコードを設定する場合には、以下のルールおよびガイドラインに従ってください。

- パフォーマンスを向上させるには、使用中のエンコードを出力ドキュメントと同じエンコードに設定します。
- 入力エンコードを、入力ドキュメントと同じエンコードに設定します。
- 出力エンコードを、出力ドキュメントと同じエンコードに設定します。
- マルチバイト文字が存在する言語の場合は、使用中のエンコードを UTF-8 に設定します。入力と出力のエンコードについては、Unicode エンコード（UTF-8 など）、またはダブルバイトコードページ（Big5 や Shift_JIS など）を使用することができます。

出力設定

データプロセッサトランスフォーメーションがイベントログを作成し、出力ドキュメントを保存するかどうかを制御する出力制御設定を設定します。

データプロセッサトランスフォーメーションが設計時イベントログに書き込むメッセージのタイプを制御できます。解析された入力ドキュメントをイベントログとともに保存すると、エラーが発生したコンテキストを [イベント] ビューで見ることができます。

以下の表に、**【設計時イベント】** 領域での設定を示します。

設定	説明
設計時イベントのログ	設計時イベントログを作成するかどうかを決定します。デフォルトでは、データプロセッサトランスフォーメーションは、通知、警告、エラーを設計時イベントログに記録します。以下のイベントタイプを除外することができます。 <ul style="list-style-type: none">- 通知- 警告- エラー
解析したドキュメントを保存	データプロセッサトランスフォーメーションが解析された入力ドキュメントをいつ保存するかを決定します。次のオプションを選択することができます。 <ul style="list-style-type: none">- 常時。- 保存しない- 失敗時のみ デフォルトは「常時」です。

以下の表に、**【ランタイムイベント】** 領域での設定を示します。

設定	説明
ランタイムイベントのログ	マッピングからトランスフォーメーションを実行したときに、イベントログを作成するかどうかを決定します。 <ul style="list-style-type: none">- 作成しない。- 失敗時のみ デフォルトは「作成しない」です。

以下の表に、**【出力】** 領域での設定を示します。

設定	説明
自動出力を無効にする	データプロセッサトランスフォーメーションが標準の出力ファイルに出力を書き込むかどうかを決定します。以下の状況では標準出力を無効にします。 <ul style="list-style-type: none">- トランスフォーメーションが出力ファイルを作成する前に、パーサーの出力を別のコンポーネントの入力に渡した場合。- データを出力ポートから渡さずに、WriteValue アクションを使用してデータをスクリプトから出力に直接書き込む場合。
値の圧縮を無効にする	データプロセッサトランスフォーメーションが、メモリの使用を最適化するために値の圧縮を使用するかどうかを決定します。 重要: 値の圧縮は、Informatica グローバルカスタマサポートが無効にするようにアドバイスしない限り無効にしないでください。

以下の表に、**【バイナリ出力ポート収集モード】** 領域での設定を示します。XML、Avro、または Parquet 出力を使用したリレーショナルから階層型へのトランスフォーメーション、または Avro または Parquet 出力を使

用したデータプロセッサトランスフォーメーションパーサーでは、バイナリ出力のオプションとして次のいずれかを選択できます。

設定	説明
入力行を単一の出力にまとめる	データプロセッサトランスフォーメーションが単一バイナリ出力ポートにリレーショナル入力を蓄積するかどうかを決定します。
サイズを超えたら出力を分割	指定された最大出力サイズに基づいて、データプロセッサトランスフォーメーションが出力をチャンクに分割するかどうかを決定します。
各行の出力行（まとめない）	データプロセッサトランスフォーメーションが出力を別々の行で渡すかどうかを決定します。

処理設定

プロセス設定は、データプロセッサトランスフォーメーションが定義されたデータ型を使わずに要素を処理する方法を定義します。この設定はスクリプトに影響します。XMap が処理する要素には影響しません。

以下の表に、スクリプト内の XML 処理に影響する処理設定を示します。

設定	説明
xs:string として扱う	データプロセッサトランスフォーメーションは、タイプのない要素を文字列として扱います。[XPath を選択] ダイアログボックスで、この要素または属性は単一のノードとして表示されます。
xs:anyType として扱う	データプロセッサトランスフォーメーションは、タイプのない要素を anyType として扱います。[XPath を選択] ダイアログボックスで、この要素または属性はノードのツリーとして表示されます。1 つのノードは xs:string タイプで、すべての名前付きの複雑なデータ型はツリーノードとして表示されます。

次の表に、ストリーマ処理に影響する処理設定を示します。

設定	説明
ストリーマチャンクサイズ	この設定は、入力ファイルストリームからストリーマが毎回読み取るデータの量を定義します。データプロセッサトランスフォーメーションは、この設定をファイル入力を含むストリーマに適用します。

以下の表に、リレーショナルトランスフォーメーション処理への階層に影響を及ぼす処理設定を示します。

設定	説明
厳密な検証の適用	この設定は、データプロセッサトランスフォーメーションが階層入力の詳細な検証を実行するかどうかを決定します。厳密な検証が適用される場合、階層入力ファイルはそのスキーマに厳密に準拠する必要があります。このオプションは、データプロセッサモードが 【出力マッピング】 に設定されている場合に適用され、リレーショナル出力用の出力ポートが作成されます。このオプションは、以前のバージョンからバージョン 10.2.1 への JSON 入力によるマッピングには適用されません。
XML 入力の正規化	この設定は、データプロセッサトランスフォーメーションによって XML 入力を正規化するかどうかを決定します。デフォルトでは、トランスフォーメーションによって XML 入力の正規化が実行されます。場合によっては、パフォーマンスを上げるために、自動正規化をスキップすることを選択できます。

XMap 設定

XMap 設定は、データプロセッサトランスフォーメーションで出力要素に変換されない XMap 入力要素の処理方法を定義します。読み取られない要素は **XMap_Unread_Input_Values** という名前の専用ポートに渡されます。この設定が有効になるのは、XMap がスタートアップコンポーネントとして選択されている場合のみです。この設定は XMap で処理される要素には影響しません。

読み取られない XMap 要素を専用ポートに渡すには、**【読み取られていない要素を別の出力ポートに書き出します。】**。

XML 出力設定

XML 生成の設定は、XML 出力ドキュメントの特性を定義します。

以下の表は、**【スキーマのタイトル】** 領域の XML 生成設定を示しています。

設定	説明
スキーマの場所	メイン出力ドキュメントのルート要素の schemaLocation を定義します。
名前空間のないスキーマの場所	メイン出力ドキュメントのルート要素の xsi:noNamespaceSchemaLocation 属性を定義します。

入力 XML ドキュメント内の欠如している要素または属性をデータプロセッサトランスフォーメーションがどのように処理するかを決定するには、[XML 出力モード] 設定を指定します。以下の表は、[XML 出力モード] 領域の XML 生成設定を示しています。

設定	説明
修正しない	空の要素の追加も削除も行いません。デフォルトは [有効] です。
完全	出力スキーマで定義される必須要素とオプション要素がすべて出力に書き込まれます。コンテンツがない要素は空の要素として書き込まれます。
コンパクト	出力から空の要素を削除します。 [要素に対して追加] が有効になっていると、データプロセッサトランスフォーメーションはオプション要素だけを削除します。 [要素に対して追加] が無効になっていると、データプロセッサトランスフォーメーションはすべての空の要素を削除します。XML 出力は有効でない可能性があります。

以下の表は、[必須ノードのデフォルト値] 領域の XML 生成設定を示しています。

設定	説明
要素に対して追加	出力スキーマが必須要素にデフォルト値を定義する場合、出力にはデフォルト値とともに要素が含まれます。デフォルトは [有効] です。
属性に対して追加	出力スキーマが必須属性にデフォルト値を定義する場合、出力にはそのデフォルト値とともに属性が含まれます。デフォルトは [有効] です。
追加された値の検証	フルモード出力で追加される空の要素をデータプロセッサトランスフォーメーションが検証するかどうかを決定します。デフォルトは [無効] です。 [追加された値の検証] が有効になっており、スキーマが空の要素を許可しない場合は、XML 出力が有効にならない可能性があります。

以下の表は、[処理命令] 領域の XML 生成設定を示しています。

設定	説明
XML 処理命令の追加	出力ドキュメントの文字エンコードと XML バージョンを定義します。デフォルトではオンに設定されています。
XML バージョン	XML バージョンを定義します。[XML バージョン] 設定には次のオプションがあります。 - 1.0 - 1.1 デフォルトは 1.0 です。
エンコード	処理命令で指定される文字エンコードを定義します。[エンコード] 設定には次のオプションがあります。 - 出力エンコードと同じ。処理命令における出力エンコードは、データプロセッサトランスフォーメーションの設定で定義されている出力エンコードと同じです。 - カスタム。処理命令における出力エンコードを定義します。ユーザーがフィールド内に値を入力します。
カスタム処理方法の追加	出力ドキュメントに他の処理命令を追加します。出力ドキュメントに表示されるとおりに処理命令を入力します。デフォルトは [無効] です。

以下の表は、[XML ルート] 領域の XML 生成設定を示しています。

設定	説明
XML ルート要素の追加	出力ドキュメントにルート要素を追加します。このオプションは、出力ドキュメントに、出力スキーマで定義されているルート要素のオカレンスが複数含まれる場合に使用します。デフォルトは [無効] です。
ルート要素名	出力ドキュメントに追加するルート要素の名前を定義します。

イベント

イベントは、データプロセッサトランスフォーメーション内のコンポーネントからの処理手順の記録です。スクリプトまたはライブラリでは、各アンカー、アクション、またはトランスフォーマがイベントを生成します。XMap では、各マッピング文がイベントを生成します。

イベントは、[データプロセッサイベント] ビューで表示できます。

イベントタイプ

データプロセッサトランスフォーメーションが、ログファイルにイベントを書き込みます。各イベントには、イベントが正常に完了したか、イベントが失敗したか、エラーが発生した状態でイベントが実行されたかを示すイベントタイプが示されます。

コンポーネントは 1 つ以上のイベントを生成できます。コンポーネントの成否は、イベントが正常に完了するか失敗するかによって決まります。1 つのイベントが失敗すると、コンポーネントが失敗します。

以下の表に、データプロセッサトランスフォーメーションが生成するイベントのタイプを示します。

イベントのタイプ	説明
通知	通常の処理。
警告	データプロセッサトランスフォーメーションは実行されましたが、不測の状況が発生しました。例えば、データプロセッサトランスフォーメーションが、同じ要素に対して複数回データを書き込んだとします。この要素が上書きされるたびに、データプロセッサトランスフォーメーションは警告を生成します。
失敗	データプロセッサトランスフォーメーションが実行されましたが、コンポーネントが失敗しました。例えば、必要な入力要素が空だったときなどです。
オプションのエラー	データプロセッサトランスフォーメーションが実行されましたが、オプションのコンポーネントが失敗しました。例えば、オプションのアンカーがソースドキュメントで見つからなかったなどです。
致命的なエラー	重大なエラーのために、データプロセッサトランスフォーメーションが失敗しました。例えば、入力ドキュメントが存在しなかったなどです。

データプロセッサイベントビュー

Developer ツールからデータプロセッサトランスフォーメーションを実行すると、**【データプロセッサイベント】** ビューにイベントが表示されます。

【データプロセッサイベント】 ビューには **【ナビゲーション】** パネルと **【詳細】** パネルがあります。ナビゲーションパネルにはナビゲーションツリーが含まれています。ナビゲーションツリーには、トランスフォーメーションが実行したコンポーネントが時系列で一覧表示されます。ツリーの各ノードには、最も重大なイベントを示すアイコンがあります。**【ナビゲーション】** パネルでノードを選択すると、**【詳細】** パネルにイベントが表示されます。

ナビゲーションツリーには次の最上位ノードが含まれています。

- サービスの初期化。データプロセッサトランスフォーメーションが初期化するファイルと変数が記述されます。
- 実行。スクリプト、ライブラリ、または XMap が実行したコンポーネントが一覧表示されます。
- サマリ。処理に関する統計が表示されます。

XMap を実行すると、ナビゲーションパネルの各ノード名に角括弧で囲まれた数字（[5]など）が追加されます。ノードのイベントを生成した文を特定するには、文のグリッドを右クリックし、**【行番号に移動】** を選択します。そして、ノード番号を入力します。

スクリプトを実行し、**【ナビゲーション】** パネルまたは **【詳細】** パネルでイベントをダブルクリックすると、イベントを生成したスクリプトコンポーネントがスクリプトエディタで強調表示されます。**【データビューア】** ビューの **【入力】** パネルでは、イベントを生成したサンプルソースドキュメントの一部が強調表示されます。

ログ

ログには、データプロセッサトランスフォーメーションのレコードが含まれています。データプロセッサトランスフォーメーションが、ログにイベントを書き込みます。

データプロセッサトランスフォーメーションは、次のタイプのログを作成します。

設計時イベントログ

設計時イベントログには、**【データビューア】** ビューでデータプロセッサトランスフォーメーションを実行するときに発生するイベントが含まれます。設計時ログは、**【イベント】** ビューに表示されます。

ランタイムイベントログ

ランタイムイベントログには、マッピングにおいてデータプロセッサトランスフォーメーションを実行するときに発生するイベントが含まれます。ランタイムイベントログはテキストエディタで表示することができます。また、ランタイムイベントログを、データプロセッサトランスフォーメーションの**【イベント】** ビューにドラッグすることもできます。

ユーザーログ

ユーザーログには、スクリプトでコンポーネントに設定するイベントが含まれます。データプロセッサトランスフォーメーションは、**【データビューア】** ビューからユーザーログを実行する場合と、マッピングでユーザーログを実行する場合に、ユーザーログに書き込みを行います。ユーザーログはテキストエディタで表示できます。

設計時イベントログ

設計時イベントログには、Developer ツールの **【データビューア】** からデータプロセッサトランスフォーメーションを実行するときに発生するイベントが記録されます。

【データビューア】 ビューからデータプロセッサトランスフォーメーションを実行すると、**【データプロセッサイベント】** ビューに設計時イベントログが表示されます。設計時イベントログには、デフォルトで通知、警告、およびエラーが記録されます。トランスフォーメーションの設定で、ログから 1 つ以上のイベントタイプを除外するようにデータプロセッサトランスフォーメーションを設定することができます。

ログと共に入力ドキュメントを保存する場合は、**【データプロセッサイベント】** ビューでイベントをクリックすると、イベントを生成した入力ドキュメント内で場所を見つけることができます。データプロセッサトランスフォーメーションの設定を行う際に、入力ファイルを実行ごとに保存するか、それともエラー時にのみ保存するか、選択することができます。

設計時イベントログの名前は events.cme です。次のディレクトリに、データプロセッサトランスフォーメーションの前の実行での設計時イベントログが格納されます。

```
C:\<Installation_directory>\clients\DT\CMReports\Init\events.cme
```

データプロセッサトランスフォーメーションは、**【データビューア】** でトランスフォーメーションが実行されるたびに設計時イベントログを上書きします。トランスフォーメーションの後の実行が終了してからログを表示する場合や、別の実行のログを比較する場合は、設計時イベントログの名前を変更します。Developer ツールを閉じると、ファイルが保存されずに終了します。

ランタイムイベントログ

ランタイムイベントログは、マッピングにおいてデータプロセッサトランスフォーメーションを実行するときに発生するイベントを記録します。

データプロセッサトランスフォーメーションがエラーなく実行を完了した場合、イベントログに書き込みはしません。実行中にエラーがあった場合、データプロセッサトランスフォーメーションは 2 回目を実行し、この 2 回目の実行中にイベントログを書き込みます。このランタイムイベントログの名前は、events.cme です。

Windows マシンでは、ランタイムイベントログは次のディレクトリにあります。

```
C:\<Installation_Directory>\clients\DT\CMReports\Tmp\
```

Linux または UNIX マシンでは、ルートユーザーのランタイムイベントログは次のディレクトリにあります。

```
/root/<Installation_Directory>/clients/DT/CMReports/Tmp
```

Linux または UNIX マシンでは、非ルートユーザーのランタイムイベントログは次のディレクトリにあります。

```
/home/[UserName]/<Installation_Directory>/DT/CMReports/Tmp
```

ランタイムイベントログの場所を変更するには、設定エディタを使用します。

データプロセッサイベントビューでのイベントログの表示

設計時のイベントログや実行時のイベントログの表示には、**【データプロセッサイベント】** ビューを使用します。

Windows Explorer を開き、表示するイベントログファイルに移動します。Windows Explorer ウィンドウから **【データプロセッサイベント】** ビューにログをドラッグします。**【データプロセッサイベント】** ビューで右クリックし、**【検索】** を選択してログを検索します。

注: 最新の設計時イベントログをリロードするには、**【データプロセッサイベント】** ビューを右クリックし、**【プロジェクトイベントのリロード】** を選択します。

ユーザーログ

ユーザーログには、スクリプト内のコンポーネントのエラーについて構成できるカスタムメッセージが含まれています。

データプロセッサトランスフォーメーションは、ユーザーが【データビューア】ビューからスクリプトを実行する場合と、マッピングでスクリプトを実行する場合にユーザーログにメッセージを書き込みます。

スクリプトコンポーネントに **on_fail** プロパティが含まれる場合は、コンポーネントでエラーが発生する際にユーザーログにメッセージを書き込むように構成できます。スクリプトで、**on_fail** プロパティを、次の値の1つに設定してください。

- LogInfo
- LogWarning
- LogError

スクリプトを実行するたびに、新しいユーザーログが生成されます。ユーザーログファイル名には、トランスフォーメーション名と固有の GUID が入ります。

`<Transformation_Name>_<GUID>.log`

たとえば、`CalculateValue_Aa93a9d14-a01f-442a-b9cb-c9ba5541b538.log` のようになります。

Windows マシンでは、ユーザーログは次のディレクトリにあります。

`c:\Users\[UserName]\AppData\Roaming\Informatica\DataTransformation\UserLogs`

Linux マシンまたは UNIX マシンでは、ルートユーザーのユーザーログは次のディレクトリにあります。

`/<Installation_Directory>/DataTransformation/UserLogs`

Linux マシンまたは UNIX マシンでは、非ルートユーザーのユーザーログは次のディレクトリにあります。

`home/<Installation_Dirctory>/DataTransformation/UserLogs`

データプロセッサトランスフォーメーションの開発

新しいトランスフォーメーションウィザードを使用してデータプロセッサトランスフォーメーションを自動生成するか、空白のデータプロセッサトランスフォーメーションを作成して後で設定します。空白のデータプロセッサトランスフォーメーションを作成する場合は、トランスフォーメーションでスクリプト、XMap、ライブラリ、または検証ルールオブジェクトの作成を選択する必要があります。スクリプトはソースドキュメントを階層形式に解析するか、階層形式を別のファイル形式に変換するか、あるいは階層ドキュメントを別の階層形式にマッピングすることができます。XMap は入力階層ファイルを別の構造の出力階層ファイルに変換します。ライブラリは、業界標準のメッセージングタイプを階層構造を備えた XML ドキュメントに変換するか、XML から業界標準の形式に変換します。入力または出力階層を定義するスキーマを選択します。

1. Developer tool でトランスフォーメーションを作成します。
2. 空白のデータプロセッサトランスフォーメーションの場合は、次の追加手順を実行します。
 - a. 入力 XML 階層または出力 XML 階層を定義するスキーマ参照を追加します。
 - b. スクリプト、XMap、ライブラリ、または検証ルールオブジェクトを作成します。
3. 入力ポートおよび出力ポートを設定します。
4. トランスフォーメーションをテストします。

データプロセッサトランスフォーメーションの作成

Developer tool でデータプロセッサトランスフォーメーションを作成します。空白のデータプロセッサトランスフォーメーションを作成する場合は、トランスフォーメーションにスクリプト、XMap、ライブラリ、または検証ルールオブジェクトを作成する必要があります。または、新しいトランスフォーメーションウィザードを使用して、データプロセッサトランスフォーメーションを自動生成します。

1. Developer ツールで、**【ファイル】** > **【新規】** > **【トランスフォーメーション】** をクリックします。
2. データプロセッサトランスフォーメーションを選択し、**【次へ】** をクリックします。
3. トランスフォーメーションの名前を入力し、モデルリポジトリでトランスフォーメーションを配置する場所を参照します。
4. ウィザードを使用してデータプロセッサトランスフォーメーションを作成するか、空白のデータプロセッサトランスフォーメーションを作成するかを選択します。
5. 空のデータプロセッサトランスフォーメーションの作成を選択した場合、**【完了】** をクリックします。

Developer ツールにより、リポジトリに空のトランスフォーメーションが作成されます。Developer ツールに**【概要】** ビューが表示されます。

6. ウィザードを使用したデータプロセッサトランスフォーメーションの作成を選択した場合、次の手順を実行します。
 - a. **【次へ】** をクリックします。
 - b. 入力形式を選択します。
 - c. COBOL、JSON、または ASN.1 などの特定の入力形式に必要な場合、スキーマ、コピーブック、サンプルファイル、または仕様ファイルを参照して選択します。
 - d. 出力形式を選択します。
 - e. 出力形式に必要な場合、スキーマ、コピーブック、サンプルファイル、または仕様ファイルを参照して選択します。
 - f. **【完了】** をクリックします。ウィザードにより、リポジトリにトランスフォーメーションが作成されます。

トランスフォーメーションには、共通のコンポーネントを持つパーサー、シリアライザ、マッパー、またはオブジェクトが含まれる場合があります。スキーマ、コピーブック、サンプルファイル、または仕様ファイルを選択した場合、ウィザードはファイル内の階層と同等のスキーマもリポジトリに作成します。

スキーマオブジェクトの選択

作成する XMap またはスクリプトコンポーネントごとに、入力または出力階層を定義するスキーマオブジェクトを選択します。

参照ビューでスキーマ参照を追加することも、スクリプトまたは XMap オブジェクトを作成するときにスキーマ参照を追加することもできます。スキーマオブジェクトは、スクリプトや XMap で参照する前に、モデルリポジトリ内に存在する必要があります。

1. データプロセッサトランスフォーメーションの**【参照】** ビューで、**【追加】** をクリックします。
2. スキーマオブジェクトがモデルリポジトリ内に存在する場合は、スキーマを参照して選択します。
3. スキーマがモデルリポジトリ内に存在しない場合は、**【新規スキーマオブジェクトの作成】** をクリックし、スキーマオブジェクトを階層スキーマファイルからインポートします。
4. **【完了】** をクリックして、スキーマ参照をデータプロセッサトランスフォーメーションに追加します。

空のデータプロセッサトランスフォーメーションへのオブジェクトの作成

データプロセッサトランスフォーメーションの【オブジェクト】ビューで、スクリプト、ライブラリ、XMap、または検証ルールオブジェクトを作成します。オブジェクトを作成したら、【オブジェクト】ビューからオブジェクトを開いて設定することができます。

スクリプトの作成

スクリプトオブジェクトを作成し、作成するスクリプトコンポーネントのタイプを定義します。必要に応じて、スキーマ参照とサンプルソースファイルを定義することができます。

1. データプロセッサトランスフォーメーションの【オブジェクト】ビューで、**【新規作成】**をクリックします。
2. スクリプトに付ける名前を入力し、**【次へ】**をクリックします。
3. パーサーを作成するか、シリアルライザを作成するかを選択します。マッパーコンポーネント、トランスフォーマコンポーネント、またはストリーマコンポーネントを作成する場合は、**【その他】**を選択します。
4. コンポーネントの名前を入力します。
5. このコンポーネントが、トランスフォーメーションでデータを処理する最初のコンポーネントである場合は、**【スタートアップコンポーネントとして設定】**を有効にします。
6. このスクリプトのスキーマ参照を入力する場合は、**【次へ】**をクリックします。スキーマ参照を入力しない場合は、**【完了】**をクリックします。
7. スキーマ参照の作成を選択した場合は、**【スキーマオブジェクトへの参照の追加】**を選択し、モデルリポジトリ内でスキーマオブジェクトを参照します。**【新規スキーマオブジェクトの作成】**をクリックし、モデルリポジトリにスキーマオブジェクトを作成します。
8. **【次へ】**をクリックし、サンプルソース参照またはサンプルテキストを入力します。サンプルソースを定義しない場合は、**【完了】**をクリックします。
サンプルソースを使用してサンプルデータを定義し、スクリプトをテストします。
9. サンプルソースを選択する場合は、**【ファイル】**を選択し、サンプルファイルを参照します。
【テキスト】領域にサンプルテキストを入力することもできます。Developer ツールはこのテキストを使用してスクリプトをテストします。
10. **【完了】**をクリックします。
Developer ツールエディタに**【スクリプト】**ビューが表示されます。

XMap の作成

Data Transformation の【オブジェクト】ビューで XMap を作成します。XMap を作成する際には、入力および出力階層ドキュメントが記述されたスキーマが必要です。入力階層のルート要素であるスキーマの要素を選択します。

1. データプロセッサトランスフォーメーションの【オブジェクト】ビューで、**【新規作成】**をクリックします。
2. **【XMap】**を選択し、**【次へ】**をクリックします。
3. XMap の名前を入力します。
4. この XMap コンポーネントが、トランスフォーメーションでデータを処理する最初のコンポーネントである場合は、**【スタートアップコンポーネントとして設定】**を有効にします。
【次へ】をクリックします。

5. スキーマ参照の作成を選択した場合は、**【スキーマオブジェクトへの参照の追加】**を選択し、モデルリポジトリ内でスキーマオブジェクトを参照します。
新しいスキーマオブジェクトをインポートするには、**【新規スキーマオブジェクトの作成】**をクリックします。
6. XMap のテストに使用できるサンプル階層ファイルがある場合は、そのファイルをファイルシステムで参照して選択します。
サンプル階層ファイルは、変更することができます。
7. 入力階層のルートを選択します。
【ルート要素の選択】 ダイアログボックスで、入力階層ファイルのルート要素となっている、スキーマ内の要素を選択します。スキーマの要素は検索することができます。その場合、パターン検索を使用できます。
*string>と入力すると、文字列内の任意の文字数に一致します。?character>と入力すると、1つの文字に一致します。
8. **【完了】** をクリックします。
Developer ツールによって、作成する XMap ごとにビューが作成されます。マッピングを設定するには、ビューをクリックします。

ライブラリの作成

Data Transformation の **【オブジェクト】** ビューでライブラリオブジェクトを作成します。メッセージタイプ、コンポーネント、名前を選択しますオプションとして、ライブラリオブジェクトのテストに使用できるサンプルメッセージタイプのソースファイルを定義することもできます。

データプロセッサトランスフォーメーションにライブラリを作成する前に、コンピュータにライブラリソフトウェアパッケージをインストールします。

1. データプロセッサトランスフォーメーションの **【オブジェクト】** ビューで、**【新規作成】** をクリックします。
2. ライブラリを選択し、**【次へ】** をクリックします。
3. メッセージタイプを参照して、選択します。
4. パーサーを作成するか、シリアライザを作成するかを選択します。
ライブラリオブジェクトの入力がメッセージタイプで、出力が XML の場合は、パーサーを作成します。ライブラリオブジェクトの入力が XML で、出力がメッセージタイプの場合は、シリアライザを作成します。
5. ライブラリが、データプロセッサトランスフォーメーションでデータを処理する最初のコンポーネントである場合は、**【スタートアップコンポーネントとして設定】** を有効にします。
【次へ】 をクリックします。
6. ライブラリのテストに使用できるサンプルメッセージタイプのソースファイルがある場合は、そのファイルをファイルシステムで参照して選択します。
サンプルファイルは、変更することができます。
7. **【完了】** をクリックします。
Developer tool は、作成するメッセージタイプごとにビューを作成します。マッピングにアクセスするには、ビューをクリックします。

検証規則の作成

検証規則オブジェクトをデータプロセッサトランスフォーメーションの **【オブジェクト】** ビューに作成します。

1. データプロセッサトランスフォーメーションの **【オブジェクト】** ビューで、**【新規作成】** をクリックします。

2. 検証ルールを選択して、**[次へ]**をクリックします。
3. 検証ルールの名前を入力します。
4. 検証ルールのテストに使用できるサンプル XML ファイルがある場合は、そのファイルをファイルシステムで参照して選択します。
サンプル XML ファイルは変更することができます。
5. **[完了]** をクリックします。
Developer ツールにより、検証ルールオブジェクトが作成され、検証ルールエディタで開かれます。

サンプルソースの追加

サンプルソースを選択して、作成する予定のスクリプト、XMap、ライブラリ、または検証ルールをテストします。

スクリプト、XMap、ライブラリ、または検証ルールを作成するときに、サンプルソースを追加できます。サンプルソースはいったん選択されると、モデルリポジトリに追加されます。モデルリポジトリの制限のため、サンプルソースファイルのサイズは 5 MB に制限されます。

サンプルソースは変更可能です。

ポートの作成

【概要】 ビューで入力ポートおよび出力ポートを設定します。

スクリプトで追加の入力ポートまたは出力ポートを設定すると、デフォルトで Developer ツールによって追加の入力ポートおよび追加の出力ポートがトランスフォーメーションに追加されます。**【概要】** ビューで入力ポートを追加しないでください。

1. XML ではなく出力データの行を返す場合は、**[リレーショナル出力]** を有効にします。
リレーショナル出力を有効にすると、Developer ツールによってデフォルトの出力ポートが削除されます。
2. 入力ポートのデータ型、ポートタイプ、精度、スケールを選択します。
3. リレーショナル出力ポートを定義していない場合は、出力ポートのデータ型、ポートタイプ、精度、スケールを定義します。
4. スクリプトに追加の入力ポートがある場合は、ポートのサンプル入力ファイルの場所を定義できます。ファイルを参照するには、**[入力場所]** フィールドの **[開く]** ボタンをクリックします。
5. リレーショナル出力を有効にしてある場合は、**[出力マッピング]** をクリックして出力ポートを作成します。
6. **[ポート]** ビューで、**[階層出力]** 領域のノードを **[リレーショナルポート]** 領域のフィールドにマップします。

トランスフォーメーションのテスト

データプロセッサトランスフォーメーションを **[データビューア]** ビューでテストします。

トランスフォーメーションをテストする前に、スタートアップコンポーネントを定義してあることを確認します。スタートアップコンポーネントは、スクリプトで定義することも、**【概要】** タブで選択することもできます。また、テストに使用するサンプル入力ファイルを選択しておく必要もあります。

1. **[データビューア]** ビューを開きます。
2. **[実行]** をクリックします。

トランスフォーメーションが検証されます。エラーがない場合は、**【入力】** 領域にサンプルファイルが表示されます。出力結果は**【出力】** パネルに表示されます。

3. **【イベントの表示】** をクリックすると、**【データプロセッサイベント】** ビューが表示されます。
4. スクリプトエディタでイベントをデバッグするには、**【データプロセッサイベント】** ビューでイベントをダブルクリックします。
5. 複数のコンポーネントをテストしているときに **【エディタと同期】** をクリックすると、入力ファイルが変更されて、コンポーネントごとに異なるサンプル入力ファイルが使用されます。
サンプルファイルの内容をファイルシステムで変更すると、その変更が **【入力】** 領域に反映されます。

データプロセッサトランスフォーメーションのインポートとエクスポート

データプロセッサトランスフォーメーションをサービスとしてエクスポートし、Data Transformation リポジトリから実行することができます。また、Data Transformation サービスを Developer ツールにインポートすることもできます。Data Transformation サービスをインポートすると、Developer ツールによって、サービスからデータプロセッサトランスフォーメーションが作成されます。

注: Data Transformation サービスをモデルリポジトリにインポートすると、Developer ツールによって、関連するスキーマがリポジトリにインポートされます。リポジトリのスキーマを変更しても、トランスフォーメーションのスキーマリアレンスに変更が表示されないことがあります。その場合は、モデルリポジトリ接続または Developer ツールをいったん閉じて再度開くと、スキーマの変更がトランスフォーメーションに表示されるようになります。

サービスとしてのデータプロセッサトランスフォーメーションのエクスポート

データプロセッサトランスフォーメーションを Data Transformation サービスとしてエクスポートすることができます。このサービスを、サービスを実行させるマシンのファイルシステムリポジトリにエクスポートしてください。このサービスは、PowerCenter、ユーザー定義のアプリケーション、または Data Transformation CM_console コマンドを使用して実行することができます。

1. **【Object Explorer】** ビューで、エクスポートするデータプロセッサトランスフォーメーションを右クリックし、**【エクスポート】** を選択します。
【エクスポート】 ダイアログボックスが表示されます。
2. **【Informatica】** > **【データプロセッサトランスフォーメーションのエクスポート】** をクリックし、**【次へ】** をクリックします。
【選択】 ページが表示されます。
3. **【次へ】** をクリックします。
【サービス名とエクスポート先フォルダーの選択】 ページが表示されます。
4. エクスポート先フォルダーを選択します。
 - Developer ツールをホストするマシンにサービスをエクスポートするには、**【サービスフォルダー】** をクリックします。
 - 他のマシンにサービスをデプロイするには、**【フォルダー】** をクリックします。サービスをデプロイするマシンの `\ServiceDB` ディレクトリを参照します。
5. **【完了】** をクリックします。

複数の Data Transformation サービスのインポート

Data Transformation サービスのディレクトリは保存先のマシンからインポートできます。Data Transformation サービスを Developer モデルリポジトリにインポートすると、Developer tool によって、.cmw ファイルと共にトランスフォーメーション、スキーマ、サンプルデータがインポートされます。多くのサービスをインポートする必要がある場合は、一度に 1 つのサービスをインポートするのではなく、サービスのディレクトリをインポートします。

1. **【ファイル】 > 【インポート】** をクリックします。
【インポート】 ダイアログボックスが表示されます。
2. **【Informatica】 【Data Transformation サービス（フォルダ）のインポート】** を選択し、**【次へ】** をクリックします。
【Data Transformation サービスのインポート】 ページが表示されます。
3. インポートするディレクトリを参照します。
4. トランスフォーメーションを保存するリポジトリの場所を参照して **【完了】** をクリックします。
Developer tool によって、.cmw ファイルと共にトランスフォーメーション、スキーマ、サンプルデータがインポートされます。

Data Transformation サービスのインポート

Data Transformation サービスの .cmw ファイルをモデルリポジトリにインポートして、データプロセッサトランスフォーメーションを作成できます。Developer ツールによって、.cmw ファイルと共にトランスフォーメーション、スキーマ、サンプルデータがインポートされます。

1. **【ファイル】 > 【インポート】** をクリックします。
【インポート】 ダイアログボックスが表示されます。
2. **【Informatica】 【Data Transformation サービス（シングル）のインポート】** を選択し、**【次へ】** をクリックします。
【Data Transformation サービスのインポート】 ページが表示されます。
3. インポートするサービスの .cmw ファイルを参照します。
Developer ツールは、サービスのファイル名に従ってトランスフォーメーションの名前を付けます。名前は変更できます。
4. トランスフォーメーションを保存するリポジトリの場所を参照して **【完了】** をクリックします。
Developer ツールによって、.cmw ファイルと共にトランスフォーメーション、スキーマ、サンプルデータがインポートされます。
5. トランスフォーメーションを編集するには、**【Object Explorer】** ビューでトランスフォーメーションをダブルクリックします。

データプロセッサトランスフォーメーションでマッピングを PowerCenter にエクスポート

データプロセッサトランスフォーメーションでマッピングを PowerCenter にエクスポートする場合は、オブジェクトをローカルファイルにエクスポートしてからマッピングを PowerCenter にインポートできます。または、マッピングを PowerCenter リポジトリに直接エクスポートすることができます。

1. **Object Explorer** ビューで、エクスポートするマッピングを選択します。右クリックして **【エクスポート】** を選択します。
【エクスポート】 ダイアログボックスが表示されます。

2. **[Informatica] > [PowerCenter]** を選択します。
3. **[次へ]** をクリックします。
[エクスポート先 PowerCenter] ダイアログボックスが表示されます。
4. プロジェクトを選択します。
5. PowerCenter のリリースを選択します。
6. エクスポートの場所として、PowerCenter インポート XML ファイルまたは PowerCenter リポジトリを選択します。
7. エクスポートオプションを指定します。
8. **[次へ]** をクリックします。
Developer ツールはエクスポートするオブジェクトを選択するように求めるプロンプトを表示します。
9. エクスポートするオブジェクトを選択して、**[完了]** をクリックします。
Developer ツールはオブジェクトを選択した場所にエクスポートします。マッピングを特定の場所にエクスポートした場合、Developer ツールは指定した場所のフォルダーにマッピングのデータプロセッサトランスフォーメーションもサービスとしてエクスポートします。
10. PowerCenter リポジトリにマッピングをエクスポートした場合、サービスは%temp%\DTServiceExport2PC\ディレクトリパスにエクスポートされます。
エクスポート機能で、以下の名前のサービスそれぞれに個別のフォルダが作成されます。
<date><serviceFullName>
トランスフォーメーションにリレーショナルマッピングが含まれる場合、リレーショナルから階層へのマッピングのフォルダが作成され、階層からリレーショナルへのマッピングに別のフォルダが作成されます。
11. データプロセッサトランスフォーメーションのサービスを使用して、ファイルをエクスポートしたローカルの場所から PowerCenter ServiceDB フォルダーに 1 つ以上のフォルダーをコピーします。
12. マッピングを PowerCenter インポート XML ファイルにエクスポートした場合は、そのマッピングを PowerCenter にインポートします。PowerCenter にオブジェクトをインポートする方法の詳細については、『*PowerCenter 9.6.0 リポジトリガイド*』を参照してください。

非ネイティブ環境でのデータプロセッサトランスフォーメーション

非ネイティブ環境でのデータプロセッサトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングまたはストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

第 14 章

ディシジョントランスフォーメーション

この章では、以下の項目について説明します。

- [ディシジョントランスフォーメーションの概要, 250 ページ](#)
- [ディシジョントランスフォーメーションの関数, 251 ページ](#)
- [ディシジョントランスフォーメーションの条件文, 253 ページ](#)
- [ディシジョントランスフォーメーションの演算子, 254 ページ](#)
- [ディシジョントランスフォーメーションの NULL 処理, 255 ページ](#)
- [ディシジョンストラテジの設定, 255 ページ](#)
- [ディシジョントランスフォーメーションの詳細プロパティ, 256 ページ](#)
- [非ネイティブ環境でのディシジョントランスフォーメーション, 256 ページ](#)

ディシジョントランスフォーメーションの概要

ディシジョントランスフォーメーションは、入力データの条件を評価し、それらの条件の結果に基づいて出力を作成するパッシブなトランスフォーメーションです。

ディシジョントランスフォーメーションは、入力フィールドで見つかった値に基づいて異なる値を生成するように設定できます。例えば、顧客の収益が特定の金額を超えたら顧客名に文字列"Priority"を追加するように構成できます。

ディシジョントランスフォーメーションには、複数のディシジョンストラテジを追加することができます。各ストラテジで IF-THEN-ELSE 条件文を評価します。この文の中で、ELSEIF 条件を使用したり、別の IF-THEN-ELSE 文をネストしたりできます。

ディシジョントランスフォーメーションは式トランスフォーメーションに似ており、条件文と関数を使用してソースデータをテストすることができます。ただし、ディシジョントランスフォーメーションは、以下の点が式トランスフォーメーションと異なります。

- ディシジョントランスフォーメーションでは、条件の評価に IF-THEN-ELSE 文を使用します。式トランスフォーメーションでは、IIF 文を使用します。
- ディシジョントランスフォーメーションの関数の中には、式トランスフォーメーションでは使用できないものもあります。
- 各ディシジョンストラテジで複数の出力を生成できます。

ディシジョントランスフォーメーションの関数

ディシジョントランスフォーメーションでは、ディシジョンストラテジを定義するために使用できる定義済みの関数にアクセスできます。

ディシジョントランスフォーメーションの式エディタには「ディシジョン」というフォルダーがあります。このフォルダーに、ディシジョントランスフォーメーションに固有の関数が格納されています。エディタにはさらに、式トランスフォーメーションの関数にアクセスするためのその他のフォルダーもあります。

式エディタで関数をクリックすると、その関数による処理の説明に加え、関数の使用法とデータ型が表示されます。

注: 式トランスフォーメーションのすべての関数をディシジョントランスフォーメーションで利用できるわけではありません。ディシジョントランスフォーメーションでは、互換性がある式トランスフォーメーションの式にのみアクセスできます。

ディシジョントランスフォーメーションの関数の一覧

- ABS
- ADD_TO_DATE
- ASCII
- CEIL
- CHOOSE
- CHR
- CHRCODE
- CONCAT
- CONTAINS
- CONVERT_BASE
- COS
- COSH
- CRC32
- CUME
- CURDATE
- CURTIME
- DATE_COMPARE
- DATE_DIFF
- DATECONVERT
- EXP
- FLOOR
- FV
- GET_DATE_PART
- GREATEST
- IN
- INDEXOF

- INITCAP
- INSTR
- IS_DATE
- IS_NUMBER
- ISNULL
- LAST_DAY
- LEAST
- LEFTSTR
- LENGTH
- LN
- LOG
- LOWER
- LPAD
- LTRIM
- MAKE_DATE_TIME
- MAX
- MD5
- METAPHONE
- MIN
- MOD[MOD]
- MONTHCOMPARE
- MOVINGAVG
- MOVINGSUM
- NPER
- PMT
- POWER
- PV
- RAND
- RATE
- REG_EXTRACT
- REG_MATCH
- REG_REPLACE
- REPLACECHR
- REPLACESTR
- REVERSE
- RIGHTSTR
- ROUND
- RPAD

- RTRIM
- SET_DATE_PART
- SIGN
- SIN
- SINH
- SOUNDEX
- SQRT
- SUBSTR
- TAN
- TANH
- TIMECOMPARE
- TO_CHAR
- TO_DATE
- TO_FLOAT
- TO_INTEGER
- TRUNC
- UPPER
- XOR

注: ディシジョントランスフォーメーションの CURDATE、DATE_COMPARE、DATECONVERT、MONTHCOMPARE の各関数では、日付形式を定義する定数値を使用します。

ディシジョントランスフォーメーションの条件文

ディシジョントランスフォーメーションでは、IF-THEN-ELSE 条件文を使用して入力データを評価します。

それらの条件文の中で、ELSEIF 条件を使用したり、別の IF-THEN-ELSE 文をネストしたりできます。ディシジョントランスフォーメーションの条件文の形式を次に示します。

```
// Primary condition
IF <Boolean expression>
THEN <Rule Block>
// Optional - Multiple ELSEIF conditions
ELSEIF <Boolean expression>
THEN <Rule Block>
// Optional ELSE condition
ELSE <Rule Block>
ENDIF
```

追加の条件文は、ルールブロック内にネストできます。

ディシジョントランスフォーメーションの演算子

ディシジョントランスフォーメーションの演算子は、ディシジョンストラテジを定義するために使用します。
以下の表に、ディシジョントランスフォーメーションの演算子を示します。

演算子のタイプ	演算子	説明
Assignment	:=	ポートに値を割り当てます。
論理	AND	必須の論理条件を追加します。親の論理式が true になるには、この演算子で連結されたすべての論理条件が true になる必要があります。
論理	OR	論理条件を追加します。親の論理式が true になるには、この演算子で連結された論理条件が少なくとも 1 つ true になる必要があります。
論理	NOT	否定の論理条件を指定します。親の論理式が true になるには、この演算子で指定された否定の条件が true になる必要があります。
ディシジョン	=	比較する項目が等しいかどうかを調べます。文字列データ型または数値データ型で使⽤します。
ディシジョン	<>	比較する項目が等しくないかどうかを調べます。文字列データ型または数値データ型で使⽤します。
ディシジョン	<	値が別の値よりも小さいかどうかを調べます。数値データ型で使⽤します。
ディシジョン	<=	値が別の値以下であるかどうかを調べます。数値データ型で使⽤します。
ディシジョン	>	値が別の値よりも大きいかどうかを調べます。数値データ型で使⽤します。
ディシジョン	>=	値が別の値以上であるかどうかを調べます。数値データ型で使⽤します。
数値	-	減算
数値	NEG	否定
数値	+	加算
数値	*	乗算
数値	/	除算
数値	%	モジュロ。数値を別の数値で割った余りを返します。
文字列		文字列を連結します。

ディシジョントランスフォーメーションの NULL 処理

NULL 処理は、Data Integration Service によるディシジョントランスフォーメーションの NULL 値の処理方法を決定します。

NULL 処理を有効にすると、ディシジョントランスフォーメーションは NULL 入力データの元の形を保持します。トランスフォーメーションは、NULL 入力値を使用して関数を評価します。

NULL 処理を無効にすると、ディシジョントランスフォーメーションは NULL 入力値にデフォルト値を割り当てます。トランスフォーメーションはデフォルト値を使用して関数を評価します。例えば、整数型の入力フィールドに NULL 値がある場合、ディシジョントランスフォーメーションは値 0 を入力に割り当て、入力値 0 を使用して関数を評価します。

デフォルトでは、NULL 処理がディシジョントランスフォーメーションで有効になっていません。NULL 処理は **【ストラテジ】** タブで有効にします。NULL 処理は、トランスフォーメーションのストラテジを設定した後で有効にできます。

ディシジョンストラテジの設定

ディシジョンストラテジを設定するには、ソースデータをディシジョントランスフォーメーションに接続し、トランスフォーメーションのビューでプロパティを編集します。

1. ディシジョントランスフォーメーションを開きます。
2. トランスフォーメーションに入力ポートと出力ポートがあることを確認します。
3. **【ディシジョン】** ビューを選択します。
4. **【追加】** をクリックします。
5. ストラテジの名前を入力します。
6. **【Expression】** 領域で、IF-THEN-ELSE 条件文を入力します。
7. 式に関数を追加するには、**【関数】** タブで関数を参照し、関数名をダブルクリックします。

ヒント: 関数を簡単に入力するには、関数名の最初の数文字を入力し、Ctrl-Space キーを押します。

8. 式にポートを追加するには、**【ポート】** タブでポートを参照します。ポート名をダブルクリックして式に追加します。必要に応じて、**【出力ポートの編集】** をクリックして、出力ポートの設定の編集や出力ポートの追加を行います。
9. 必要に応じて、`"/"`の後にコメントを入力してコメント行を追加します。
10. **【検証】** をクリックして、ディシジョン式が有効であるかどうかを確認します。
11. **【OK】** をクリックしてストラテジを保存します。
12. 必要に応じて、その他のストラテジを追加します。ストラテジごとに一意の出力ポートを使用する必要があります。ストラテジ間で出力ポートを共有することはできません。

ディシジョントランスフォーメーションの詳細プロパティ

データ統合サービスでディシジョントランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ディシジョントランスフォーメーションの次の詳細プロパティを設定します。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

パーティション化可能

トランスフォーメーションを複数のスレッドで処理できます。データ統合サービスが1つのスレッドを使用してトランスフォーメーションを処理するようにする場合は、このオプションをクリアします。データ統合サービスは複数のスレッドを使用して残りのマッピングパイプラインステージを処理します。

トランスフォーメーションで CUME、MOVINGSUM、または MOVINGAVG の数値関数のうち 1 つを使用するときは、ディシジョントランスフォーメーションのパーティション化を無効にすることを検討します。これらの関数は行ごとに現在の合計と平均を計算します。CUME、MOVINGSUM、または MOVINGAVG 関数を使用するパーティション化されたトランスフォーメーションは、マッピングを実行するたびに、同じ計算結果を返さない場合があります。

トランスフォーメーションで CUME、MOVINGSUM、または MOVINGAVG 関数を使用しない場合は、トランスフォーメーションのパーティション化を有効にしてパフォーマンスを最適化します。

非ネイティブ環境でのディシジョントランスフォーメーション

非ネイティブ環境でのディシジョントランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限付きでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

第 15 章

重複レコードの例外トランスフォーメーション

この章では、以下の項目について説明します。

- [重複レコードの例外トランスフォーメーションの概要, 257 ページ](#)
- [重複レコードの例外プロセスフロー, 258 ページ](#)
- [重複レコードの例外, 258 ページ](#)
- [重複レコードの例外の「設定」ビュー, 259 ページ](#)
- [ポート, 260 ページ](#)
- [重複レコードの例外トランスフォーメーションの詳細プロパティ, 263 ページ](#)
- [重複レコードの例外マッピングの例, 264 ページ](#)
- [重複レコードの例外トランスフォーメーションの作成, 269 ページ](#)

重複レコードの例外トランスフォーメーションの概要

重複レコード例外トランスフォーメーションは、データ品質プロセスの出力を読み取り、手動確認が必要な重複レコードを特定するアクティブなトランスフォーメーションです。重複レコード例外トランスフォーメーションは、複数グループトランスフォーメーションです。

重複レコード例外トランスフォーメーションでは、別のトランスフォーメーションや別のマッピング内のデータオブジェクトから入力を受け取ります。例外トランスフォーメーションに対する入力には、レコードが重複しているかどうかを判別するために使用される数値一致スコアを含める必要があります。重複レコードの例外トランスフォーメーションで上限および下限の一致スコアのしきい値を設定します。

重複レコードの例外トランスフォーメーションは、以下のアクションのいずれかを実行します。

- 一致スコアが上限しきい値と等しいか、上回る場合、レコードは重複として扱われ、データベースターゲットに書き込まれます。
- 一致スコアが上限しきい値を下回り、下限しきい値を上回る場合、レコードは重複している可能性があるとして扱われ、手動確認のために別のターゲットに書き込まれます。レコードがクラスタに属している場合、クラスタ内のすべてのレコードがターゲットに書き込まれます。
- クラスタの一致率がしきい値を下回る場合、クラスタ内のすべてのレコードが一意的レコードの出力グループに書き込まれます。サイズが 1 のクラスタは、一致スコアに関係なく、一意のグループにルーティングさ

れます。デフォルトでは、例外トランスフォーメーションで一意的レコードがターゲットに書き込まれます。一意的レコードを返すようにトランスフォーメーションを設定できます。

- クラスタ内の一致スコアが 0~100 の範囲外である場合、例外トランスフォーメーションではクラスタ内のすべての行が無視されます。データ統合サービスでは、クラスタ ID が含まれているメッセージが記録されます。

重複レコードの例外プロセスフロー

重複レコードの例外トランスフォーメーションは他のデータ品質トランスフォーメーションの出力を分析し、さまざまなレベルのデータ品質のレコードを含むテーブルを作成します。

複数のデータ品質トランスフォーメーションを単一のマッピングに設定することも、プロセス内のステージ別にマッピングを作成することもできます。

Analyst ツールを使用し、手動確認が必要な重複レコードの確認と更新を行うことができます。

Developer ツールを使用して、以下のタスクを実行します。

1. データ品質の問題に関するスコア値を生成するマッピングを作成します。
2. クラスタモードで一致トランスフォーメーションを使用して、重複レコードの例外のスコア値を生成します。
3. 一致トランスフォーメーションの出力を読み取るには、重複レコードの例外トランスフォーメーションを設定します。一致スコア値に基づいてレコードをデータベーステーブルに書き込むには、このトランスフォーメーションを設定します。
4. 自動統合レコードのターゲットデータオブジェクトを設定します。
5. **【重複レコードテーブルの生成】** オプションをクリックして、重複レコードテーブルを作成し、マッピングキャンパスに追加します。
6. マッピングをワークフローに追加します。
7. 重複している可能性があるレコードの手動確認をユーザーに割り当てるようにヒューマンタスクを設定します。ユーザーは、Analyst ツールでレコードの確認と更新を行うことができます。

重複レコードの例外

重複レコードの例外トランスフォーメーションを使用して、手動確認が必要な重複データのクラスタを識別できます。クラスタ内のレコードの一致スコアで、潜在的な重複が判別されます。トランスフォーメーションの一致スコアの上限および下限のしきい値を設定できます。上限および下限のしきい値によって、類似度が定義されます。

クラスタには、マッチング操作グループと一緒に関連レコードが含まれています。一致トランスフォーメーションでは、重複分析操作および ID 解決操作を使用して、クラスタを作成します。クラスタ内の各レコードに同じクラスタ ID があります。クラスタ内で最も低い一致スコアが上限しきい値と下限しきい値の間である場合、重複レコードの例外トランスフォーメーションでクラスタが重複レコードの例外クラスタとして識別されます。一致トランスフォーメーションでは、クラスタ ID 値カラムをすべてのレコードに割り当てます。重複レコードは同じクラスタ ID を受け取ります。

クラスタ内の最低レコードスコアにより、クラスタタイプが決まります。クラスタに一致スコアが 0.95 の 11 のレコードと、一致スコアが 0.79 の 1 つのレコードがあるとします。上限しきい値が 0.9 で下限しきい値が 0.8 の場合、例外トランスフォーメーションではレコードが一意的レコードテーブルに書き込まれます。

重複レコードの例外の [設定] ビュー

一致スコアしきい値を定義し、重複レコードの例外トランスフォーメーションで各種出力データを書き込む場所を設定します。

以下の図に、設定可能なプロパティを示します。

手動レビューのしきい値
下限しきい値: 0.80
上限しきい値: 0.95

データレーティングのオプション

タイプ	標準出力	重複レコードテーブル
自動統合（上限しきい値を超...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
手動統合（しきい値間）	<input type="checkbox"/>	<input checked="" type="checkbox"/>
一意のレコード（下限しきい...	<input type="checkbox"/>	<input type="checkbox"/>

☐ 一意のレコードのための独立した出力グループを作成
[重複レコードテーブルの生成](#)

以下のプロパティを設定することができます。

下限しきい値

重複レコードのスコア範囲の下限です。トランスフォーメーションは、一致スコアがこの値を下回るレコードを一意のレコードとして処理します。下限しきい値は 0～1 の数値です。

上限しきい値

重複レコードのスコア範囲の上限です。トランスフォーメーションは、一致スコアが上限しきい値以上、または同等のレコードを重複レコードとして処理します。上限しきい値は下限しきい値を超える数値です。

自動統合

すべてのレコードに上限しきい値を超える一致スコアがあるクラスタ。自動統合クラスタは確認が不要です。レコードは重複しています。レコードの結合には、統合トランスフォーメーションを使用できます。デフォルトでは、重複レコードの例外トランスフォーメーションは自動統合クラスタを標準出力ポートに書き込みます。

手動統合

すべてのレコードに下限しきい値以上または同等の一致スコアがあり、少なくとも 1 つのレコードに上限しきい値を下回る一致スコアがあるクラスタ。クラスタに重複レコードが含まれているかどうかを判断するには、クラスタの手動確認を実行する必要があります。デフォルトでは、重複レコードの例外トランスフォーメーションは手動統合レコードを重複レコードテーブルに書き込みます。

一意統合

サイズが 1 であるクラスタ、またはレコードのどれかに下限しきい値を下回る一致スコアがあるクラスタ。一意のレコードクラスタは重複していません。デフォルトでは、重複レコードの例外トランスフォーメーションは一意のレコードを出力テーブルに書き込みません。

標準出力

標準出力ポートに書き込まれるレコードのタイプです。

デフォルトは自動統合レコードです。

重複レコードテーブル

重複レコード出力ポートに書き込まれるレコードのタイプです。デフォルトは手動統合レコードです。

一意のレコードのための独立した出力グループを作成

一意のレコードのために別の出力グループを作成します。一意のレコードのために別のテーブルを作成しない場合は、一意のレコードを他のグループのどれかに書き込むためのトランスフォーメーションを設定できます。あるいは、一意のレコードを出力テーブルに書き込む処理をスキップすることも可能です。デフォルトでは無効になっています。

重複レコードテーブルの生成

重複レコードのクラスタデータを含める、データベースオブジェクトを作成します。このオプションを選択すると、Developer ツールによってデータベースオブジェクトが作成されます。Developer ツールは、オブジェクトをモデルリポジトリに追加し、オブジェクトのインスタンスをマッピングキャンバスに追加し、ポートをオブジェクトにリンクします。

重複レコードテーブルの生成

マッピングで例外トランスフォーメーションのインスタンスから重複レコードテーブルを生成できます。

1. **【設定】** ビューで **【重複レコードテーブルの生成】** をクリックして、テーブルを生成します。
【リレーショナルデータオブジェクトの作成】 ダイアログボックスが表示されます。
2. テーブルが含まれているデータベースへの接続を参照して選択します。
3. データベース内の重複レコードテーブル名を入力します。
4. モデルリポジトリ内の重複レコードテーブルオブジェクト名を入力します。
5. **【完了】** をクリックします。

Developer ツールでは、マッピングキャンバスに新しいテーブルを追加します。

ポート

重複レコードの例外トランスフォーメーションには複数の入力グループと出力グループがあります。

下の図に、入力ポートと出力ポートの例を示します。

ポート

	名前	タイプ	精度	スケール	入力	出力	デフォルト	説明
	☐ 入力							
	☐ データ (3)							
1	Employee	deci...	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Name	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Addr1	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
	☐ 制御 (3)							
1	スコア	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Row_Identifier	string	25	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
	☐ 出力							
	☐ 標準出力 (6)							
1	スコア	double	15	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Row_Identifier	string	25	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	Employee	deci...	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
5	Name	string	50	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
6	Addr1	string	50	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
	☐ クラスタデータ (9)							
1	Row_Identifier	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Sequential_Cluster_ID	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

重複レコードの例外トランスフォーメーションの入力ポート

重複レコードの例外トランスフォーメーションには、入力ポートのデータグループとコントロールグループがあります。

データグループには、ソースデータを受け取るユーザー定義のポートが含まれています。

コントロールポートは、一致トランスフォーメーションがソースデータに追加するメタデータを受け取ります。以下の表に、**コントロールポート**を示します。

ポート	説明
スコア	0～1 の 10 進値。レコードをクラスタにリンクしたレコードとの類似度を特定します。
Row_Identifier	レコードの一意な ID。
Cluster_ID	レコードが属している一致クラスタの ID。

重複レコードの例外トランスフォーメーションの出力ポート

重複レコードの例外トランスフォーメーションには複数の出力グループがあります。デフォルトでは、このトランスフォーメーションによって、重複レコードが**標準出力グループ**に書き込まれます。このトランスフォーメーションによって、一致候補が**クラスタデータグループ**に書き込まれます。一意のレコードの出力グループを追加することができます。

【設定】 ビューのデフォルト設定を変更することによって、このトランスフォーメーションが出力ポートに書き込むレコードタイプを変更できます。

以下の表に、**標準出力グループ**の出力ポートを示します。

ポート	説明
スコア	0～1 の 10 進値。クラスタ内のあるレコードと別のレコードが共通している度合いを識別します。
Row_Identifier	レコードの一意な ID。
Cluster_ID	一致トランスフォーメーションがレコードを割り当てるクラスタの ID です。
ユーザー定義のポート	ソースデータフィールドです。

以下の表に、**クラスタデータグループ**の出力ポートを示します。

ポート	説明
Row_Identifier	レコードの一否 ID。
Sequential_Cluster_ID	ヒューマンタスクでクラスタを識別します。ワークフローでは、クラスタをヒューマンタスクのインスタンスに割り当てるために、連続したクラスタ ID 値が使用されます。
Cluster_ID	そのレコードが属しているクラスタを特定します。一致トランスフォーメーションでは、クラスタ ID をすべてのレコードに割り当てます。
スコア	0～1 の 10 進値。レコードをクラスタにリンクしたレコードとの類似度を特定します。
Is_Master	レコードがクラスタ内の優先されるレコードであるかどうかを示す文字列値。デフォルトでは、クラスタ内の最初の行が優先されるレコードです。値は Y または N です。
Workflow_ID	タスク内のレコードのワークフローを識別する ID です。ワークフロー外でマッピングを実行する場合、ワークフロー ID は DummyWorkflowID です。
ユーザー定義のポート	ソースデータポートです。

ポートの作成

各入力ポートをデータグループに追加します。入力ポートを追加する場合、Developer ツールによって、同じ名前の出力ポートが標準出力グループ、クラスタデータグループ、および一意のレコードグループに追加されます。

1. データ入力グループを選択します。
グループが強調表示されます。

2. **【新規作成（挿入）】** をクリックします。

Developer ツールによって、フィールドが標準出力グループ、クラスタデータグループ、および一意のレコードグループに追加されます。

3. 必要に応じて、フィールド名を変更します。

Developer ツールによって、他のグループのフィールド名が変更されます。

4. データソースに追加する必要がある残りのポートを入力します。

重複レコードの例外トランスフォーメーションの詳細プロパティ

重複レコードの例外トランスフォーメーションには、ソートの動作、キャッシュメモリの動作、およびトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

ソート

トランスフォーメーションが **【クラスタ ID】** ポートデータの投入行をソートするかどうかを決定します。このプロパティは、デフォルトで有効になっています。

投入行が事前にソートされていない場合は、このプロパティを選択します。

キャッシュファイルディレクトリ

データ統合サービスが現在のトランスフォーメーションの一時データを書き込むディレクトリを指定します。入力データの量が利用可能なシステムメモリより大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

キャッシュファイルサイズ

トランスフォーメーションの入力データをソートするためにデータ統合サービスが使用するシステムメモリの量を決定します。デフォルト値は 400,000 バイトです。

データをソートする前に、データ統合サービスは指定されているメモリ量を割り当てます。ソート操作によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュファイルディレクトリに書き込みます。ソート操作にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

注: 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

重複レコードの例外マッピングの例

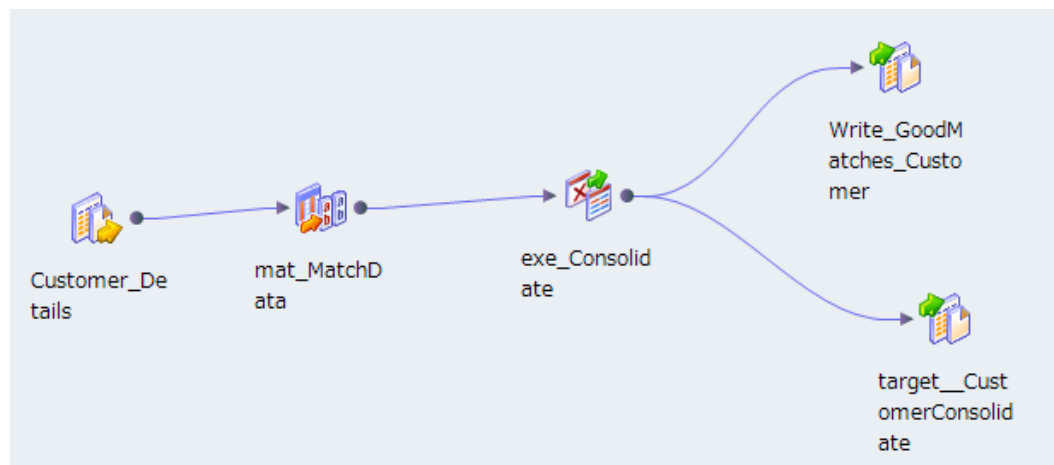
組織が顧客データを確認するデータプロジェクトを実施します。組織は顧客データに複数の重複レコードがあると判断しています。組織は重複している可能性があるレコードの一部を手動で確認する必要があります。

データ品質マッピングを作成して、重複顧客レコードを識別します。マッピングには一致トランスフォーメーションが含まれています。重複レコードの例外トランスフォーメーションは、一致トランスフォーメーションから結果を受け取ります。重複レコードの例外トランスフォーメーションによって、ステータスが不明な各レコードクラスがデータベーステーブルに書き込まれます。データアナリストは、Analyst ツールでデータを確認し、重複レコードを判別します。

重複レコードの例外マッピング

顧客レコードを調べて、重複レコードを検出するための重複レコードの例外マッピングを設定します。

下の図に、重複レコードの例外マッピングを示します。



マッピングには次のオブジェクトが含まれます。

Customer_Details

重複レコードが含まれている可能性があるデータソースです。

mat_MatchData

一致トランスフォーメーションでは、顧客データを調べて、レコードが一致しているかどうかを判別します。一致トランスフォーメーションでは、2つのカラムの値の類似度を示す数値スコアが生成されます。アルゴリズムでは、0 から 1 までの小数値としてマッチ率を計算します。2つのカラムの値が同じ場合、アルゴリズムはスコアに 1 を割り当てます。

exc_Consolidate

重複している可能性がある顧客、重複が判明している顧客、または一意の顧客レコードを判別する重複レコードの例外トランスフォーメーションです。

Write_GoodMatches_Customer テーブル

手動確認が不要なすべてのレコードを受け取るテーブルです。重複レコードの例外トランスフォーメーションでは、重複レコードおよび一意のレコードがこのテーブルに書き込まれます。

Target_CustomerConsolidate テーブル

例外トランスフォーメーションによって、重複している可能性があるレコードが Target_CustomerConsolidate テーブルに書き込まれます。このテーブル内のレコードは、Analyst ツールでの手動確認が必要です。

一致トランスフォーメーション

一致トランスフォーメーションでは、顧客データを受け取り、ID 照合を実行します。

「クラスタ-すべて一致」出力タイプの一致トランスフォーメーションを設定します。一致トランスフォーメーションはクラスタ内のマッチングレコードを返します。クラスタ内の各レコードは、スコアが一致のしきい値以上の 1 つ以上のクラスタ内の他のレコードと一致している必要があります。一致のしきい値は.75 です。

一致トランスフォーメーション **【ストラテジ】** タブで **【Division 一致ストラテジ】** を選択します。Division ストラテジは組み込み一致ストラテジで、住所フィールドに基づいて組織を識別します。一致トランスフォーメーションの **【ストラテジ】** タブで、一致を調べる入力ポートを選択します。ストラテジのウェイトを.5 に設定します。

以下の図に、一致トランスフォーメーションの Division ストラテジ設定を示します。

全 ポート 一致タイプ ストラテジ 照合出力 詳細	◎ 照合ストラテジの定義				
	照合ストラテジ	カスタム名	ウェイト	照合フィールド	プロパティ
	Division	Division1	0.5	ADDR1_1,ADDR1_2,COMPANY_1,COMPANY_2,ADDR2_1,ADDR2_2,AD...	ポロケーション: usa, 照合レベル: TYPICAL

一致トランスフォーメーションは、クラスタ情報を各出力レコードに追加します。また、このトランスフォーメーションは、一意の RowID を各レコードに追加します。

重複レコードの例外の入力グループ

重複レコードの例外トランスフォーメーションには 2 つの入力グループがあります。このトランスフォーメーションには、顧客データを受け取るデータグループがあります。このトランスフォーメーションには、行、行 ID、クラスタ ID の一致スコアが含まれるコントロールグループがあります。

以下の図に、例外トランスフォーメーションの入力グループを示します。

	名前	タイプ	精度	スケ...	入力	出力	デフォルト	説明
	コ入力							
	データ (11)							
1	CUST_ID	deci...	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	COMPANY	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	CONTACT	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	TITLE	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	ADDR1	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
6	ADDR2	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
7	ADDR3	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
8	ADDR4	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
9	COUNTRY	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
10	PHONE	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
11	EMAIL	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
	制御 (3)							
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Row_Iden...	string	25	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

データグループには、顧客データが含まれています。顧客データには顧客 ID、連絡先、役職、住所のフィールドがあります。コントロールグループは、各顧客レコードに一致トランスフォーメーションが追加された、追加のメタデータです。コントロールグループには、一致スコア、行、行 ID、クラス ID が含まれています。

重複レコードの例外の [設定] ビューの例

[設定] ビューで上限および下限のしきい値を定義します。トランスフォーメーションで重複顧客レコード、重複している可能性のあるレコード、および一意の顧客レコードが書き込まれる場所を特定します。

以下の図に、重複レコードの例外トランスフォーメーションの **[設定]** ビューを示します。

手動レビューのしきい値

下限しきい値:

上限しきい値:

データレーティングのオプション

タイプ	標準出力	重複レコードテーブル
自動統合 (上限しきい値を超...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
手動統合 (しきい値間)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
一意のレコード (下限しきい...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

☐ 一意のレコードのための独立した出力グループを作成

以下の表に、構成設定を示します。

オプション	設定項目
下限しきい値	.80
上限しきい値	.95
自動統合	標準出力テーブル
手動統合	重複レコードテーブル
一意のレコード	標準出力テーブル

【重複レコードテーブルの生成】をクリックして、重複レコードテーブルを作成します。一意のレコード専用のテーブルは作成しないでください。このトランスフォーメーションによって、一意のレコードが標準出力テーブルに書き込まれます。

標準出力テーブルレコード

Write_GoodMatches_Customer ターゲットテーブルは、標準出力グループから行を受け取ります。このテーブルは、一意のレコードを受け取り、重複しているレコードを受け取ります。これらのレコードには手動確認が不要です。

以下の図に、例外トランスフォーメーションで返される標準出力レコードを示します。

出力										
名前: exc_Consoli.Good_Records										
	Score	Row_Identif...	Cluster_ID	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2	ADDR3
1	1	1 - 4	3	1001622	INVESTEX	NEW YORK	United Parc...	"50 BROAD...	20TH FL"	310 BRI
2	1	1 - 5	4	7121564	"ARTHUR V...	INC."	WHITEHOU...	Federal Exp...	310 BRIER...	767 FIF
3	0.90476190...	1 - 6	4	7121565	OSTERREIC...	NEW YORK	United Parc...	767 FIFTH...	570 LEXING...	6803 S.
4	0.90476190...	1 - 7	4	7121566	K2 ADVISORS	NEW YORK	Courier	570 LEXING...	51 JFK PAR...	6803 S.
5	0.90476190...	1 - 8	4	7121567	MFP INVES...	SHORT HILLS	Federal Exp...	51 JFK PAR...	ONE PARKE...	6803 S.
6	0.90476190...	1 - 9	4	7121568	EASTON	FORT LEE	US Postal In...	ONE PARKE...	3840 HIGH...	6803 S.
7	0.90476190...	1 - 10	4	7121569	RICE VOEL...	MANDEVILLE	Federal Exp...	3840 HIGH...	"2ND & D S...	SW HO
8	1	1 - 1	16	1001590	E-AGENCY	OAKLAND	Federal Exp...	291 3RD ST...	2500 WEST...	6803 S.
9	0.90476190...	1 - 2	16	1001599	BANK ONE	ELGIN	US Postal In...	2500 WEST...	530 CHESN...	6803 S.
10	1	1 - 3	16	1001604	KPMG PEAT...	WOODCLIF...	US Postal 2...	530 CHESN...	"50 BROAD...	20TH FL
11	1	1 - 11	16	1001658	HOUSE INF...	WASHINGT...	Federal Exp...	"2ND & D S...	SW HOUS...	<NULL>
12	0.90476190...	1 - 12	16	1001660	OPPENHEIM...	ENGLEWOOD	US Postal In...	6803 S. TU...	6803 S. TU...	6803 S.
13	0.90476190...	1 - 13	16	1001659	THOMSON...	FT. WORTH	Courier	301 COMME...	301 COMME...	6803 S.
14	1	1 - 14	16	1001658	LYNCH JON...	NEW YORK	US Postal O...	875 3RD AVE.	875 3RD AVE.	6803 S.
15	0.90476190...	1 - 15	16	1001691	EPICENTRIC	SAN FRANC...	US Postal 2...	333 BRYAN...	333 BRYAN...	6803 S.
16	0.90476190...	1 - 16	16	1001664	CHARLES S...	SAN ANTON...	United Parc...	1100 N.E. L...	1100 N.E. L...	6803 S.
17	0.90476190...	1 - 17	16	1001694	BANK OF M...	BOSTON	US Postal 2...	"125 BROA...	38TH FLOO...	"125 BR
18	1	1 - 18	16	1001729	JOSEPH H...	HIGHLAND...	United Parc...	1212 LINCO...	1212 LINCO...	6803 S.
19	0.90476190...	1 - 19	16	1001724	D E SHAW...	NEW YORK	US Postal O...	120 WEST...	120 WEST...	6803 S.
20	0.90476190...	1 - 20	16	1001732	BANG NET...	SAN FRANC...	Courier	808 BRANN...	808 BRANN...	6803 S.

レコードには、以下のフィールドが含まれます。

スコア

クラスタ内のあるレコードと別のレコードの類似度を示す一致スコアです。一致スコアが1のレコードは重複レコードであり、確認は不要です。レコードの一致スコアが下限しきい値を下回るクラスタは、重複クラスタではありません。

Row_Identifier

テーブル内の各行を識別する一意の行番号です。この例では、行 ID に顧客 ID が含まれています。

クラスタ ID

クラスタの一意的 ID です。クラスタ内の各レコードが同じクラスタ ID を受け取ります。サンプル出力データの最初の 4 つのレコードは一意的です。各レコードに一意的クラスタ ID があります。5 行目から 9 行目はクラスタ 5 に属しています。住所フィールドが同一であるため、このクラスタ内のレコードは重複レコードです。

ソースデータフィールド

標準出力テーブルグループは、すべてのソースデータフィールドも受け取ります。

クラスタ出力

Target_CustomerConsolidate テーブルは、クラスタ出力グループからレコードを受け取ります。クラスタ出力グループは、重複レコードである可能性のあるレコードを返します。Target_CustomerConsolidate テーブル内のレコードは、Analyst ツールでの手動確認が必要です。

以下の図に、Target_CustomerConsolidate テーブルのレコードとフィールドの例を示します。

出力										
名前: exc_ConsoLi.Cluster_Data										
	Row_Identifier	Sequential_ID	Cluster_ID	Score	Is_Master	Workflow_ID	CUST_ID	COMPANY	CONTACT	TITLE
1	0	0	3	1	Y	DummyWor...	1001622	INVESTEX	NEW YORK	United P...
2	1	0	3	1	N	DummyWor...	1001622	INVESTEX	NEW YORK	United P...
3	2	1	4	1	Y	DummyWor...	7121564	"ARTHUR V...	INC."	WHITEH...
4	3	1	4	1	N	DummyWor...	7121564	"ARTHUR V...	INC."	WHITEH...
5	4	1	4	0.90476190...	N	DummyWor...	7121565	OSTERREIC...	NEW YORK	United P...
6	5	1	4	0.90476190...	N	DummyWor...	7121566	K2 ADVISORS	NEW YORK	Courier
7	6	1	4	0.90476190...	N	DummyWor...	7121567	MFP INVES...	SHORT HILLS	Federal
8	7	1	4	0.90476190...	N	DummyWor...	7121568	EASTON	FORT LEE	US Postz...
9	8	1	4	0.90476190...	N	DummyWor...	7121569	RICE VOEL...	MANDEVILLE	Federal
10	9	2	16	1	Y	DummyWor...	1001590	E-AGENCY	OAKLAND	Federal
11	10	2	16	1	N	DummyWor...	1001590	E-AGENCY	OAKLAND	Federal
12	11	2	16	0.90476190...	N	DummyWor...	1001599	BANK ONE	ELGIN	US Postz...
13	12	2	16	1	N	DummyWor...	1001604	KPMG PEAT...	WOODCLIF...	US Postz...
14	13	2	16	1	N	DummyWor...	1001658	HOUSE INF...	WASHINGTON	Federal
15	14	2	16	0.90476190...	N	DummyWor...	1001660	OPPENHEIM...	ENGLEWOOD	US Postz...
16	15	2	16	0.90476190...	N	DummyWor...	1001659	THOMSON	FT. WORTH	Courier
17	16	2	16	1	N	DummyWor...	1001658	LYNCH JON...	NEW YORK	US Postz...
18	17	2	16	0.90476190...	N	DummyWor...	1001691	EPICENTRIC	SAN FRANCO...	US Postz...
19	18	2	16	0.90476190...	N	DummyWor...	1001664	CHARLES S...	SAN ANTON...	United P...
20	19	2	16	0.90476190...	N	DummyWor...	1001694	BANK OF M...	BOSTON	US Postz...

レコードには、以下のフィールドが含まれます。

Row_Identifier

テーブル内の各行を識別する一意の番号です。

シーケンシャルクラスタ ID

ヒューマンタスクで確認する各クラスタのシーケンシャル ID です。重複レコードの例外トランスフォーメーションでは、シーケンシャルクラスタ ID をクラスタデータ出力グループのレコードに追加します。

クラスタ ID

クラスタの一意的 ID です。一致トランスフォーメーションでは、クラスタ ID をすべての出力レコードに割り当てます。重複レコードと重複レコードである可能性のあるレコードがクラスタ ID を共有します。一意のレコードがクラスタ ID を受け取りますが、レコードは他のレコードと ID 番号を共有しません。

スコア

クラスタ内のあるレコードと別のレコードの類似度を示す一致スコアです。一致スコアが.95 未満で.80 を上回る、手動確認が必要なレコードです。

Is Master

そのレコードがクラスタの優先レコードかどうかを特定します。

WorkflowID

トランスフォーメーションがワークフローに存在していないため、Workflow_ID は DummyWorkflowID です。

レコードフィールド

レコード内の他のフィールドに、顧客のソースデータが含まれています。

重複レコードの例外トランスフォーメーションの作成

重複レコードの例外トランスフォーメーションを作成する場合、入力ポートを設定します。一致を特定するための上限および下限のしきい値を定義します。重複レコードおよび一意のレコードを書き込むかどうかを設定します。

1. 再利用可能な、または再利用不可能な重複レコードの例外トランスフォーメーションを作成します。
 - 再利用可能なトランスフォーメーションを作成するには、**【ファイル】 > 【新規】 > 【トランスフォーメーション】**を選択して、重複レコードの例外トランスフォーメーションを選択します。
 - 再利用不可能なトランスフォーメーションを作成するには、マッピングを開き、トランスフォーメーションをマッピングキャンバスに追加します。ウィザードで、重複レコードの例外トランスフォーメーションを選択します。
2. **【次へ】**をクリックするか、**【完了】**をクリックします。

【完了】をクリックすると、トランスフォーメーションを作成する前にデフォルトのしきい値とデータルーティングのオプションを更新できます。
3. **【設定】**ビューで、一致スコアの上限および下限のしきい値を設定します。
4. **【データルーティングのオプション】**セクションで、標準出力と例外テーブルのプロパティを設定して、トランスフォーメーションが各レコードタイプを書き込む場所を設定します。

必要に応じて、重複レコード、確認する重複レコードおよび一意のレコードを書き込むかどうかを設定します。
5. 必要に応じて、一意のレコードテーブルを生成します。新しいテーブルのデータベース接続とテーブル名の情報を入力します。一意のレコードテーブルを生成すると、トランスフォーメーションによりモデルリポジトリにデータベースオブジェクトが作成されます。
6. 入力ポートを設定します。入力ポートを追加する場合、Developer tool によって同じポート名が出力グループに追加されます。
 - 再利用可能なトランスフォーメーションを作成する場合は、**【ポート】**タブを選択し、トランスフォーメーションに接続するデータのポートを追加します。
 - 再利用不可能なトランスフォーメーションを作成する場合は、他のオブジェクトをマッピングキャンバスに追加し、入力ポートをトランスフォーメーションにドラッグします。
7. トランスフォーメーション出力ポートを1つ以上のデータターゲットに接続します。出力ポートを**【設定】**ビューで設定した出力オプションに対応するデータオブジェクトに接続します。
 - 再利用可能なトランスフォーメーションを作成する場合は、トランスフォーメーションをマッピングに追加し、出力ポートを接続します。

- 再利用不可能なトランスフォーメーションを作成する場合は、トランスフォーメーションにより、ポートがクラスタデータテーブルに接続されます。出力ポートを他のデータターゲットに接続する必要があります。

第 16 章

式トランスフォーメーション

この章では、以下の項目について説明します。

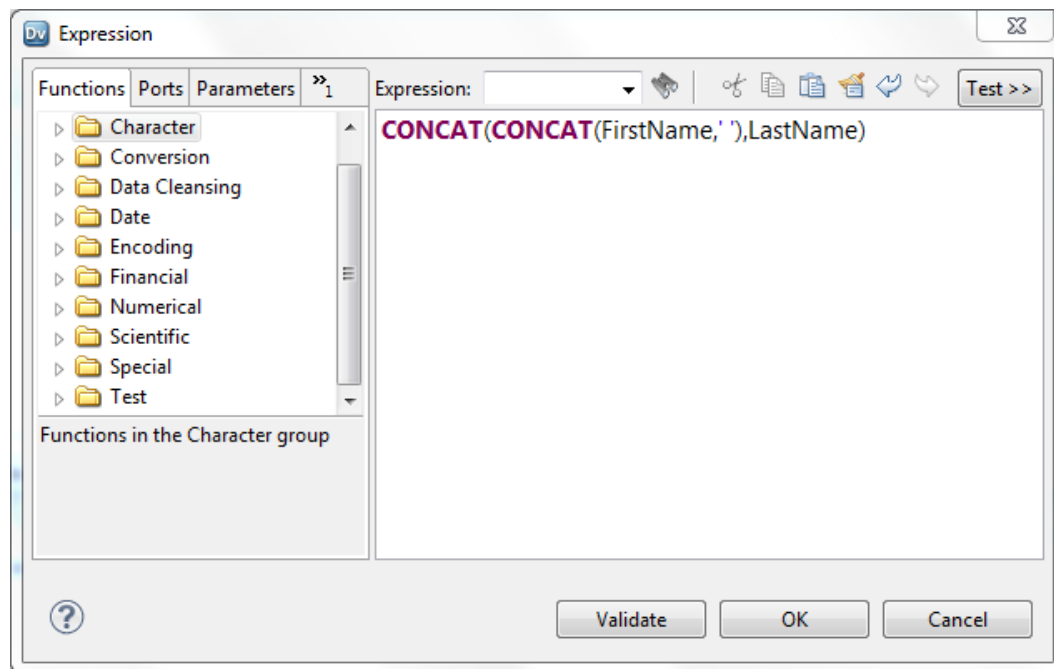
- [式トランスフォーメーションの概要, 271 ページ](#)
- [式トランスフォーメーションのポート, 272 ページ](#)
- [テスト式, 273 ページ](#)
- [ポートセクタ, 274 ページ](#)
- [ウィンドウ化, 277 ページ](#)
- [動的式, 281 ページ](#)
- [式トランスフォーメーションの詳細プロパティ, 285 ページ](#)
- [非ネイティブ環境での式トランスフォーメーション, 286 ページ](#)

式トランスフォーメーションの概要

式トランスフォーメーションは、単一の行での計算の実行または条件文のテストに使用できる、パッシブトランスフォーメーションです。再利用不可能な式トランスフォーメーションでは、マッピング出力を定義するときに、集計のためのマッピング出力式を定義できます。

単一の行で、式を作成して、従業員の給与の調整、名前と姓の連結、または文字列から数値への変換を行う必要がある場合があります。

以下の図に、名前、スペース、姓を連結する式トランスフォーメーションの式を示します。



出力ポートごとに式を作成しておくことで、1つの式トランスフォーメーションで複数の式の入力が可能になります。例えば、国および地方の所得税など、各従業員の給与からの数種の税額を計算したい場合があります。どちらの税額計算でも従業員の給与と税率が必要となります。それぞれの計算に個別の出力ポートを定義します。それぞれの出力ポートに異なる式を定義します。ポート値が変化しないため、給与と税率にパススルーポートを定義できます。

式トランスフォーメーションのポート

式トランスフォーメーションには、式を定義する際に参照できるさまざまなポートタイプが用意されています。

式トランスフォーメーションは、以下のポートタイプを備えています。

入力

アップストリームトランスフォーメーションからデータを受信します。式トランスフォーメーションによってポート値が変更されない場合は、入力ポートの代わりにパススルーポートを定義できます。

出力

式の戻り値を格納します。出力ポートの設定オプションとして式を入力します。また、ポートごとにデフォルト値を設定することもできます。

注: 式の結果が、ゼロによる除算や負の数値の SQRT などの数値エラーになる場合、無限または NaN 値を返します。

パススルー

パススルーポートを定義して、トランスフォーメーション経由で値を変更せずにデータを渡します。パススルーポートを計算で参照することはできませんが、パススルーポートのデータ値を変更することはできません。

変数

式で使用するデータを一時的に格納します。複数の行にまたがるデータを格納できます。変数ポートに値を返す式を定義できます。

動的ポート

動的マッピングでポートを受け取るか返します。動的ポートはアップストリームトランスフォーメーションから1つ以上のカラムを受け取り、カラムごとに生成されたポートを作成できます。動的出力ポートは、1つ以上の生成されたポートを返すことができます。入力ルールを定義して、動的ポートが受け取るカラムを決定できます。動的出力ポートには複数の出力ポートを生成する式を格納できます。

生成されたポート

動的ポート内の1つのカラムを表すポート。式トランスフォーメーション内の生成されたポートは、その式トランスフォーメーションがアップストリームトランスフォーメーションから受け取るカラムに基づいて変わることがあります。

テスト式

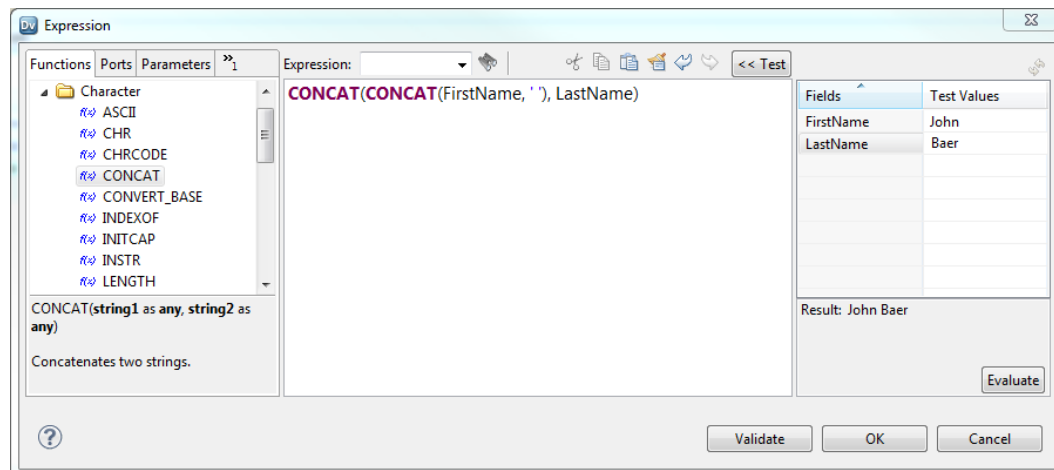
式エディタで設定した式はテストできます。式をテストする場合、サンプルデータを入力してから式を評価します。

式を設定すると、次の方法で式をテストできます。

- 式トランスフォーメーションの出力または変数ポートでテスト
- トランスフォーメーションをマッピングに追加後、式トランスフォーメーションの「マッピング出力」ビューでテスト

例えば、名前、スペース、姓を連結する式を設定後、ポートのサンプルデータを入力し、式を評価して結果を確認できます。

次の図は、名前と姓の例を連結する式の結果を示しています。



サンプルデータの日付形式の文字列

Date/Time または Timestamp with Time Zone データ型のポートを使用する式をテストする場合、必要な日付形式の文字列を使用してポートのサンプルデータを入力する必要があります。

Date/Time データ型のポートのサンプルデータを入力するには、「MM/DD/YYYY HH24:MI:SS」形式を使用します。式の評価を行うと、式エディタには式で指定した形式を使用した結果が表示されます。式の形式文字列を省略すると、式エディタには同じ「MM/DD/YYYY HH24:MI:SS」形式を使用した結果が表示されます。

Timestamp with Time Zone データ型のポートのサンプルデータを入力するには、「MM/DD/YYYY HH24:MI:SS TZR」形式を使用します。式の評価を行うと、式エディタには「YYYY-MM-DD HH24:MI:SS.NS TZR」形式を使用した結果が表示されます。

式のテスト

式エディタで式をテストし、式を評価して結果を確認します。

- 以下のいずれかの方法で式エディタを開きます。
 - 式トランスフォーメーションで、**【開く】** ボタン (🔍) をクリックします。このボタンは出力ポートまたは変数ポートの **【式】** カラムにあります。
 - マッピングに含まれる式トランスフォーメーションを選択します。**【マッピング出力】** ビューで、**【開く】** ボタン (🔍) をクリックします。このボタンは出力の **【式】** カラムにあります。
- 式を設定します。
- 【テスト >>】** をクリックして、テストパネルを開きます。
- 【テスト値】** カラムの各フィールドに、サンプルデータを入力します。
式に含まれるポートまたはパラメータごとにテスト値を入力できます。
- 【評価】** をクリックします。
式の結果がテストパネルの下部に表示されます。

ポートセレクト

トランスフォーメーションでポートを生成した場合、生成したポートが変更されてもトランスフォーメーションが正しく実行されるように設定する必要があります。ポートセレクトを使用して、動的式、ルックアップ条件、またはジョイナ条件でどのポートを使用するかを決定できます。

ポートセレクトとは、式内で参照できるポートの順序付きリストです。生成したポートが動的マッピング内で変更された場合、ポートセレクトに異なるポートが含まれている可能性があります。

例えば、次の式は、動的マッピング内の生成されたポートを参照します。

Salary * 12

動的ソースを使用するようにマッピングを設定できますが、各ソースファイルの給与情報が含まれるカラムは名前が異なります。カラム名は Salary、Monthly_Salary、または Base_Salary です。

次のタスクを実行して、異なるカラム名を受け入れます。

- 「Salary_PortSelector」というポートセレクトを作成します。
- サフィックス「Salary」を持つすべてのポート名を受け入れる選択ルールを作成します。

3. Salary 列名ではなくポートセレクト名を含めるように、式を変更します。式の構文は次のとおりです。
- ```
Salary_PortSelector * 12
```

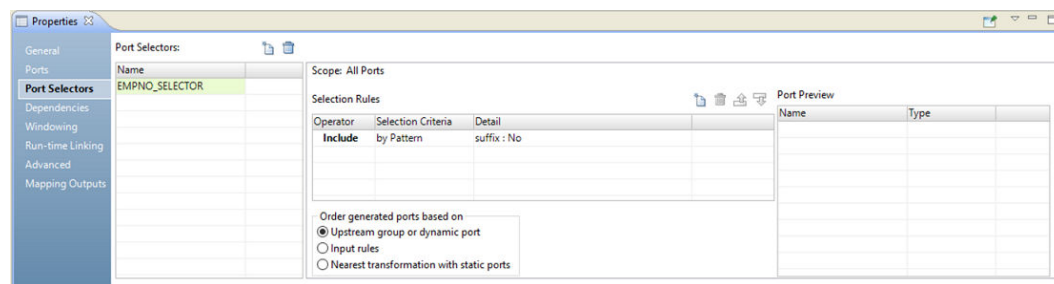
式は、給与を示すあらゆるポート名で正しく実行されます。

## ポートセレクトの設定

ポートセレクトを設定するには、選択ルールを定義して、含める生成済みポートを決定します。選択ルールは、動的ポートに設定できる入力ルールと同じです。

ポートセレクトには、静的または生成済みポートを含めることができます。ポートセレクトの設定は、**ポートセレクト**タブで行います。

次の図は、**ポートセレクト**タブを示しています。



ポートセレクトに次のプロパティを設定します。

### 名前

ポートセレクトを識別します。1つのトランスフォーメーションに複数のポートセレクトを作成し、式で参照できます。

### スコープ

ポートセレクトが適用されるポートグループを識別します。ジョイナまたはルックアップトランスフォーメーションのポートセレクトを作成する場合、スコープを選択する必要があります。これらのトランスフォーメーションには複数の入力グループがあります。ジョイナトランスフォーメーションにはマスタまたは詳細スコープがあります。ルックアップトランスフォーメーションにはインポートまたはルックアップスコープがあります。式トランスフォーメーションには入力グループが1つあります。スコープは常に[すべてのポート]です。

### 選択ルール

ポートセレクトに含めるポートを決定します。選択ルールを作成すると、**ポートのプレビュー**パネルに現在の入力ポートのうち適格なポートが表示されます。これらのポートは変わる可能性があります。さまざまなソースからのポートに対応できるように選択ルールを設定します。

## 選択ルール

ポートセレクトに関連付けられた選択ルールでポートセレクトに含めるポートを決定します。

選択ルールを作成すると、**ポートのプレビュー**パネルに現在の入力ポートのうち適格なポートが表示されます。これらのポートは変わる可能性があります。さまざまなソースからのポートに対応できるように選択ルールを設定します。

次の条件に基づいて選択ルールを作成します。

## 演算子

選択ルールが返すポートを含めるか、除外します。デフォルトは「含める」です。ポートを除外する前にポートを含める必要があります。

## 選択条件

作成する選択ルールのタイプ。カラム名、ポートタイプ、パターン、または複合データ型定義に基づいてルールを作成できます。カラム名に基づいてポートを含めるには、特定の名前を検索するか、名前に含まれる文字列パターンを検索します。

## 詳細

選択条件に適用する値。選択条件がカラム名基準になっている場合は、検索する文字列または名前を設定します。選択条件がポートタイプ基準になっている場合は、含めるポートタイプを選択します。

次の表に、選択条件と条件の詳細を指定する方法を示します。

| 選択条件     | 説明                                    | 詳細                                                                                   |
|----------|---------------------------------------|--------------------------------------------------------------------------------------|
| すべて      | すべてのポートを含めます。                         | 詳細は不要です。                                                                             |
| 名前       | ポート名に基づいてポートをフィルタリングします。              | 値のリストからポート名を選択するか、[ポート] または [ポートリスト] のパラメータを使用します。                                   |
| タイプ      | 各ポートのデータ型に基づいてポートをフィルタリングします。         | リストからデータ型を選択します。                                                                     |
| パターン     | 名前に含まれる文字列または正規表現を使用してポートをフィルタリングします。 | ポート名のパターンタイプとして、プレフィックス、サフィックス、または正規表現を選択します。次に、パターンの値を入力するか、文字列タイプのパラメータを使用します。     |
| 複合データ型定義 | 複合データ型定義を使用してポートをフィルタリングします。          | 複合データ型定義のパターンタイプとして、プレフィックス、サフィックス、または正規表現を選択します。次に、パターンの値を入力するか、文字列タイプのパラメータを使用します。 |

# ポートセレクトタの作成

動的式、ルックアップ条件、または結合条件で使用するポートを決定するために、ポートセレクトタを作成します。

1. **【ポートセレクトタ】** タブをクリックします。
2. **【ポートセレクトタ】** 領域で、**【新規】** をクリックします。  
Developer tool が、すべてのポートが含まれるデフォルトの選択ルールを使用してポートセレクトタを作成します。
3. **【ポートセレクトタ】** 領域で、ポートセレクトタ名を一意的な名前に変更します。
4. ジョイナトランスフォーメーションまたはルックアップトランスフォーメーションに対して作業している場合は、スコープを選択します。  
使用可能なポートは、選択したポートのグループに基づいて変更されます。
5. **【選択ルール】** 領域で、**【演算子】** を選択します。

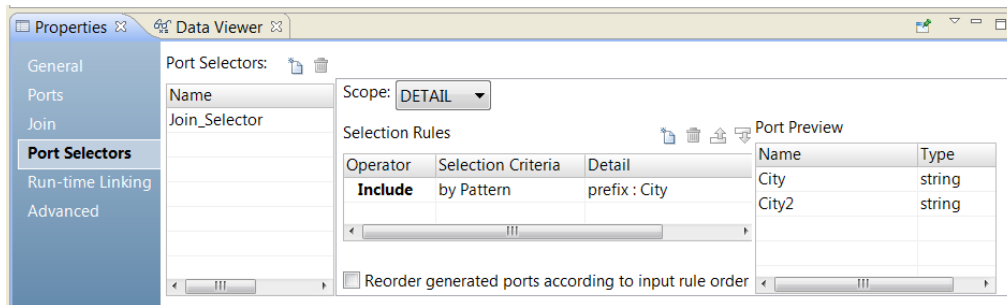


- 含む。ポートセクタのポートを含めるルールを作成します。ポートを除外する前にポートを含める必要があります。
- 除外。ポートセクタから特定のポートを除外するルールを作成します。

6. **【選択条件】** を選択します。

- 名前。特定のポートを名前を選択します。スコープ内のポートのリストからポート名を選択できます。
- タイプ。ポートをタイプで選択します。1つまたは複数のデータ型を選択できます。
- パターン。ポート名の文字のパターンでポートを選択します。特定の文字で検索するか、正規表現を作成できます。

次の図は、[ポートセクタ] タブを示しています。



7. **【詳細】** カラムをクリックします。

**【入力ルールの詳細】** ダイアログボックスが表示されます。

8. ポートをフィルタ処理する基準となる値を選択します。

- 名前。値またはパラメータを基準としてポートリストを作成する場合に選択します。**【選択】** をクリックして、リスト内のポートを選択します。
- タイプ。リストから1つ以上のデータ型を選択します。**【ポートのプレビュー】** 領域に、選択したタイプのポートが表示されます。
- パターン。ポート名のプレフィックスまたはサフィックスから、特定の文字パターンを検索する場合に選択します。または、検索に使用する正規表現を作成することを選択します。パラメータを設定するか、検索に使用するパターンを設定します。

**【ポートのプレビュー】** 領域に、設定したルールどおりにポートセクタのポートが表示されます。

9. ポートセクタのポートの順序を変更するには、**【生成されたポートの順序を入力ルールの順序に従って変更】** を選択します。

## ウィンドウ化

トランスフォーメーションにウィンドウ関数が含まれている場合は、ウィンドウ化プロパティを構成する必要があります。ウィンドウ化は、Spark エンジン上のトランスフォーメーションでのみ利用できます。

ウィンドウ関数は、行のグループに対して動作し、各入力行の戻り値を計算します。

式トランスフォーメーションにウィンドウ関数を定義する前に、ウィンドウ化プロパティを構成してウィンドウを記述する必要があります。ウィンドウ化プロパティには、フレーム仕様、パーティションキー、およびオーダーキーが含まれます。フレーム仕様では、現在の行の全体的な計算に含まれる行を指定します。パーティションキーは、同じパーティション内にある行を決定します。オーダーキーは、パーティション内の行の順序を決定します。

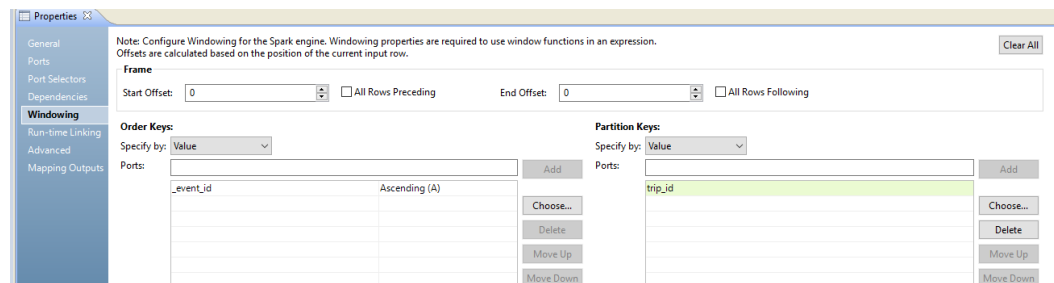
ウィンドウ化プロパティを構成した後、式トランスフォーメーションでウィンドウ関数を定義します。Informatica は、ウィンドウ関数の LEAD と LAG をサポートしています。式トランスフォーメーションでは、集計関数をウィンドウ関数として使用することもできます。

## ウィンドウ化構成

式トランスフォーメーションにウィンドウ関数を含める場合は、その関数に関連付けられているウィンドウ化プロパティを構成します。ウィンドウ化プロパティは、特定の入力行に関連付けられたパーティション化、順序付け、およびフレーム境界を定義します。

[ウィンドウ化] タブでウィンドウ化のトランスフォーメーションを構成します。

次の図は、[ウィンドウ化] タブを示しています。



[ウィンドウ化] タブで、次のプロパティグループを構成します。

### フレーム

現在の行の位置からの物理オフセットに基づいて、現在の入力行のフレームに含まれる行を定義します。

集計関数をウィンドウ関数として使用する場合は、フレームを構成します。ウィンドウ関数の LEAD と LAG は、個々の行を参照し、フレーム仕様を無視します。

### パーティションキー

入力行を別々のパーティションに分割します。パーティションキーを定義しない場合は、すべての行が単一パーティションに属します。

### オーダーキー

パーティション内の行の順序を決定します。選択するポートによって、パーティション内の行の位置が決まります。オーダーキーは昇順または降順にできます。オーダーキーを定義しない場合、行には特定の順序がありません。

## フレーム

フレームは、現在の行に対する相対的な位置に基づいて、現在の入力行の計算に含まれる行を決定します。

LEAD または LAG の代わりに集計関数を使用する場合は、ウィンドウフレームを指定する必要があります。LEAD と LAG は、個々の行を参照し、フレーム仕様を無視します。

開始オフセットと終了オフセットは、現在の入力行の前後に表示される行の数を表します。オフセット「0」は、現在の入力行を表します。例えば開始オフセットが-3 で終了オフセットが0 の場合は、現在の入力行と現在の行より前の3つの行を含むフレームを表します。

次の図は、開始オフセットが-1 で終了オフセットが 1 のフレームを示しています。

| Type      | Category   | Revenue |               |
|-----------|------------|---------|---------------|
| Action    | Video game | 1000    |               |
| Arcade    | Video game | 1000    | ← 1 PRECEDING |
| Sports    | Video game | 2000    |               |
| Adventure | Video game | 3000    |               |
| Strategy  | Video game | 4000    | ← 1 FOLLOWING |

すべての入力行に対して、関数はフレーム内の行に対して集計操作を実行します。前述のフレームに対して SUM のような集計式を構成する場合、式はフレーム内の値の合計を計算し、入力行に対して値 6000 を返します。

現在の入力行を含まないフレームを指定することもできます。例えば開始オフセットが 10 で終了オフセットが 15 の場合は、現在の入力行より後の 10 番目の行から 15 番目の行までの合計 6 行が含まれるフレームを表します。

**注:** 開始オフセットは、終了オフセット以下でなければなりません。

**【先行のすべての行】** および **【後続のすべての行】** というオフセットは、パーティションの最初の行とパーティションの最後の行を表します。例えば、開始オフセットが **【先行のすべての行】** で終了オフセットが -1 の場合、フレームには現在の行より前の 1 行と、それより前のすべての行が含まれます。

次の図は、開始オフセットが -0 で終了オフセットが **【後続のすべての行】** のフレームを示しています。

| Genre     | Recordings | Revenue |                    |
|-----------|------------|---------|--------------------|
| Jazz      | 233        | 5000    |                    |
| Gospel    | 214        | 1000    |                    |
| Country   | 145        | 2000    | All Rows Following |
| Ethnic    | 154        | 9000    |                    |
| Pop       | 317        | 4000    |                    |
| Rock      | 237        | 2100    |                    |
| Classical | 221        | 3200    |                    |
| EDM       | 153        | 950     |                    |
| Hip Hop   | 839        | 2300    |                    |
| Punk      | 415        | 7650    |                    |

## パーティションキーおよびオーダーキー

パーティションキーとオーダーキーを構成して、行のグループを形成し、各パーティション内で行の順序またはシーケンスを定義します。

次のキーを使用して、ウィンドウ内の行のグループ化と順序の指定を行います。

### パーティションキー

すべての入力にわたって計算を実行するのではなく、パーティションの境界を定義するようにパーティションキーを構成します。ウィンドウ関数は、現在の行と同じパーティション内に入る行にわたって動作します。

パーティションキーは、値またはパラメータによって指定できます。ポート名を使用するには、**【値】** を選択します。ソートキーリストパラメータを使用するには、**【パラメータ】** を選択します。ソート基準となるポートのリストを含んだソートキーリストパラメータを作成します。パーティションキーを指定しない場合は、すべてのデータが同じパーティションに含まれます。

## オーダーキー

オーダーキーを使用して、パーティション内の行の順序を決定します。オーダーキーは、パーティション内の特定の行の位置を定義します。

オーダーキーは、値またはパラメータによって指定できます。ポート名を使用するには、**【値】** を選択します。ソートキーリストパラメータを使用するには、**【パラメータ】** を選択します。ソート基準となるポートのリストを含んだソートキーリストパラメータを作成します。また、データを昇順または降順に並べ替えることを選択する必要があります。オーダーキーを指定しない場合、パーティション内の行は特定の順序で配置されません。

## 例

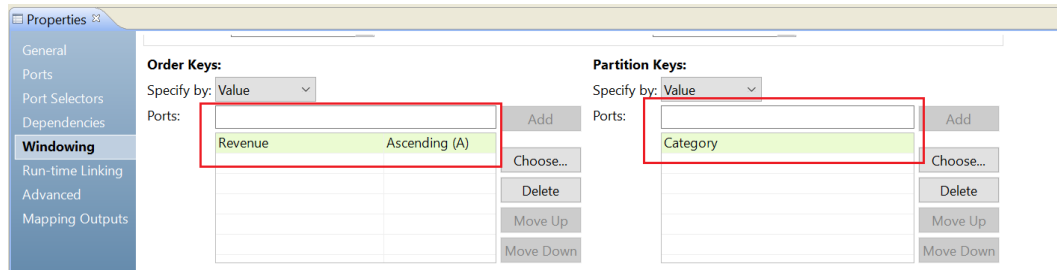
あなたはコーヒーと紅茶の店のオーナーです。売り上げが 1 番目と 2 番目のコーヒーと紅茶を計算したいと考えています。

以下の表に、製品、対応する製品カテゴリ、および各製品の収益を示しています。

| Product    | Category | Revenue |
|------------|----------|---------|
| Espresso   | Coffee   | 600     |
| Black      | Tea      | 550     |
| Cappuccino | Coffee   | 500     |
| Americano  | Coffee   | 600     |
| Oolong     | Tea      | 250     |
| Macchiato  | Coffee   | 300     |
| Green      | Tea      | 450     |
| White      | Tea      | 650     |

カテゴリ別にデータを分割し、収益の降順でデータを並べます。

以下の図に、**【ウィンドウ化】** タブで設定するプロパティを示します。



以下の表は、カテゴリに従って 2 つのパーティションにグループ化されたデータを示しています。各パーティション内では、収益は降順に編成されます。

| Product    | Category | Revenue |
|------------|----------|---------|
| Espresso   | Coffee   | 600     |
| Americano  | Coffee   | 600     |
| Cappuccino | Coffee   | 500     |
| Macchiato  | Coffee   | 300     |
| White      | Tea      | 650     |
| Black      | Tea      | 550     |
| Green      | Tea      | 450     |
| Oolong     | Tea      | 250     |

パーティション化仕様と順序付け仕様に基づいて、売り上げのよいコーヒー 2 つはエスプレッソとアメリカーノ、売り上げのよいお茶の 2 つは白茶と紅茶であると判断しました。

## ウィンドウ化構成のルールとガイドライン

ウィンドウ化のトランスフォーメーションを構成するときには特定のガイドラインが適用されます。

ウィンドウ関数のウィンドウ化プロパティを定義する場合、次のルールとガイドラインを検討します。

- フレームを構成する場合、開始オフセットは終了オフセット以下でなければなりません。そうしないと、フレームは無効になります。
- 集計関数をウィンドウ関数として使用する場合は、フレーム仕様を構成します。LEAD と LAG は、オフセット値に基づいて動作し、フレーム仕様を無視します。
- 複合ポートはパーティションキーまたはオーダーキーとして使用できません。
- ランタイムエラーを避けるため、一意のポート名をパーティションキーおよびオーダーキーに割り当てます。
- パーティションキーとオーダーキーは、動的ポートと、同じ動的ポートの 1 つ以上の生成されたポートの両方は使用できません。動的ポートまたは生成されたポートのいずれかを選択する必要があります。

## 動的式

動的出力ポートの式を設定すると、式が動的式になります。動的式では複数の出力ポートを生成できます。

動的式では、ポートセクタまたは動的ポートを参照できます。ポートセクタまたは動的ポートに複数のポートが含まれる場合、動的式は各ポートに対して実行されます。

動的式を設定すると、Developer tool は、生成されたポートが式に対して有効なタイプかどうかを検証しません。例えば、string 型を必要とする式で decimal 型ポートを含むポートセクタを参照する場合、設計時に式は有効と表示されます。

## 例

式トランスフォーメーションには、生成された以下の入力ポートがあります。

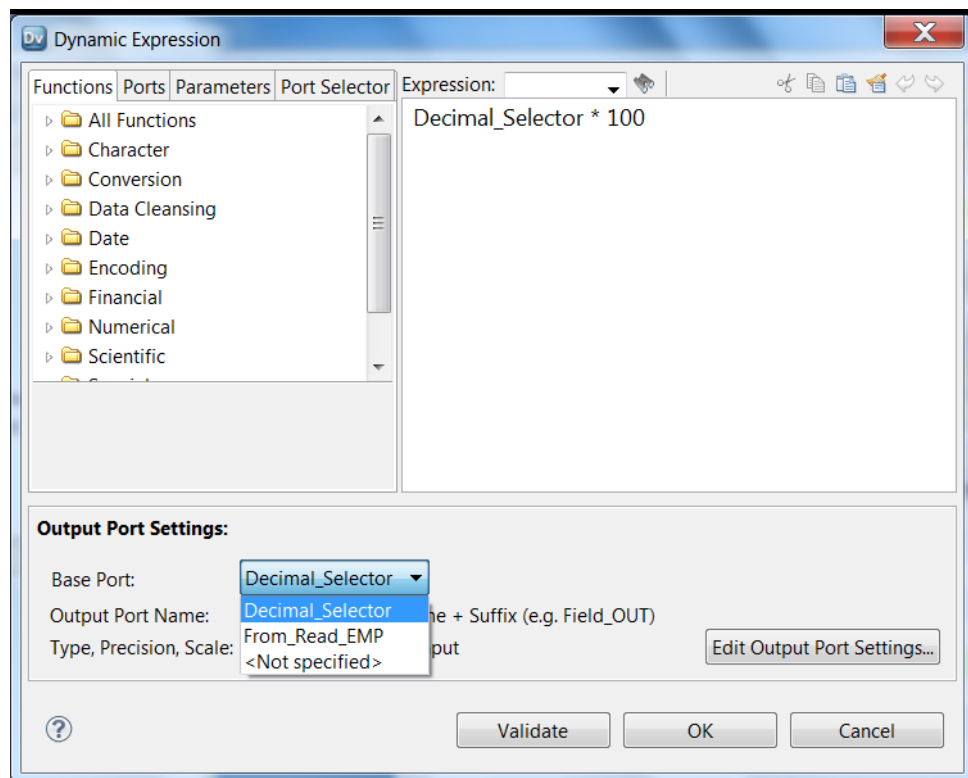
EMPNO   Decimal  
NAME     String  
SALARY   Decimal  
DEPTNO   Decimal

トランスフォーメーションには MyDynamicPort という名前の動的出力ポートが含まれます。出力ポートは動的式の結果を返します。動的式は、ポートセクタの各ポートの値に 100 をかけます。式はポートセクタの各ポートに対して 1 回実行されます。各インスタンスは異なる結果を返すことがあります。式トランスフォーメーションは、各結果に対して別の出力ポートを生成します。

Decimal\_Selector ポートセクタには、decimal データ型のポートを含む選択ルールがあります。

EMPNO   Decimal  
SALARY   Decimal  
DEPTNO   Decimal

次の図は、Decimal\_Selector ポートセクタを参照する動的式を示しています。



出力ポート設定を編集して、出力ポート名と出力ポートプロパティを変更します。また、ベースポートも選択できます。

## 出力ポート設定

動的式への入力として使用するポートを示すことができます。【ベースポート】領域でポートを選択します。

Decimal\_Selector ポートセクタをベースポートに選択すると、動的式は decimal 型ポートを返します。動的式では、NAME ポートのポートは文字列であるために生成されません。

次の図は、トランスフォーメーションで生成されたポートを示しています。

| Properties Data Viewer Tags Notifications |         |           |       |                                     |                                     |                          |                        |  |
|-------------------------------------------|---------|-----------|-------|-------------------------------------|-------------------------------------|--------------------------|------------------------|--|
| General                                   |         |           |       |                                     |                                     |                          |                        |  |
| Ports                                     |         |           |       |                                     |                                     |                          |                        |  |
| Port Selectors                            |         |           |       |                                     |                                     |                          |                        |  |
| Dependencies                              |         |           |       |                                     |                                     |                          |                        |  |
| Parameters                                |         |           |       |                                     |                                     |                          |                        |  |
| Run-time Linking                          |         |           |       |                                     |                                     |                          |                        |  |
| Advanced                                  |         |           |       |                                     |                                     |                          |                        |  |
| Mapping Outputs                           |         |           |       |                                     |                                     |                          |                        |  |
| Name                                      | Type    | Precis... | Scale | Input                               | Output                              | Varia...                 | Expression             |  |
| 1 From_Read_EMP                           | dynamic |           | 0     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | From_Read_EMP          |  |
| 1 EMPNO                                   | decimal | 3         | 0     |                                     |                                     |                          |                        |  |
| 2 NAME                                    | string  | 10        | 0     |                                     |                                     |                          |                        |  |
| 3 SALARY                                  | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 4 DEPTNO                                  | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 2 MyDynamicPort                           | dynamic |           | 0     | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Decimal_Selector * 100 |  |
| 1 EMPNO_OUT                               | decimal | 3         | 0     |                                     |                                     |                          |                        |  |
| 2 SALARY_OUT                              | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 3 DEPTNO_OUT                              | decimal | 4         | 0     |                                     |                                     |                          |                        |  |

From\_Read\_Emp 動的ポートは入力-出力ポートですが、トランスフォーメーションは MyDynamicPort 動的出力ポートのポートのみを返します。

出力ポートの名前のつけ方を設定できます。デフォルトの出力ポート名は、入力ポート名の後に\_OUT をつけます。

ベースポートをポートセレクトに変更できます。

次の図は、式エディタの出力ポート設定を示しています。

**Output Port Settings:**

Base Port: From\_Read\_EMP ▼

Output Port Name: Primary input port name + Suffix (e.g. Field\_O

Type, Precision, Scale: Inherit from primary input

Edit Output Port Settings...

ベースポートを From\_Read\_EMP に設定すると、生成された入力ポートをすべて含む動的ポートを選択することになります。データ統合サービスでは、From\_Read\_EMP のすべてのポートに対して動的式が実行されます。

次の図は、From\_Read\_Emp 入力に基づいて生成された出力ポートを示しています。

| General          |         |           |       |                                     |                                     |                          |                        |  |
|------------------|---------|-----------|-------|-------------------------------------|-------------------------------------|--------------------------|------------------------|--|
| Ports            |         |           |       |                                     |                                     |                          |                        |  |
| Port Selectors   |         |           |       |                                     |                                     |                          |                        |  |
| Dependencies     |         |           |       |                                     |                                     |                          |                        |  |
| Parameters       |         |           |       |                                     |                                     |                          |                        |  |
| Run-time Linking |         |           |       |                                     |                                     |                          |                        |  |
| Advanced         |         |           |       |                                     |                                     |                          |                        |  |
| Mapping Outputs  |         |           |       |                                     |                                     |                          |                        |  |
| Name             | Type    | Precis... | Scale | Input                               | Output                              | Varia...                 | Expression             |  |
| 1 From_Read_EMP  | dynamic |           | 0     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | From_Read_EMP          |  |
| 1 EMPNO          | decimal | 3         | 0     |                                     |                                     |                          |                        |  |
| 2 NAME           | string  | 10        | 0     |                                     |                                     |                          |                        |  |
| 3 SALARY         | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 4 DEPTNO         | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 2 MyDynamicPort  | dynamic |           | 0     | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Decimal_Selector * 100 |  |
| 1 EMPNO_OUT      | decimal | 3         | 0     |                                     |                                     |                          |                        |  |
| 2 NAME_OUT       | string  | 10        | 0     |                                     |                                     |                          |                        |  |
| 3 SALARY_OUT     | decimal | 4         | 0     |                                     |                                     |                          |                        |  |
| 4 DEPTNO_OUT     | decimal | 4         | 0     |                                     |                                     |                          |                        |  |

生成された出力ポートには、NAME\_OUT という名前の出力ポート（string 型）が含まれます。

データ統合サービスは、動的式ごとに出力ポートを生成します。15 個のポートを生成する動的式を作成し、5 個のポートを生成する別の動的式を定義すると、データ統合サービスは 20 個の出力ポートを生成します。各動的出力ポートは、それぞれ異なるポートのグループを生成します。

## 動的式の作成


式トランスフォーメーションで動的式を作成して、動的ポートまたはポートセクタの各ポートで式を 1 回実行します。動的式は、インスタンスごとに個別に生成されたポートに結果を返します。

1. 式トランスフォーメーションで **【プロパティ】** ビューに移動して、**【ポート】** タブをクリックします。
2. **【新しい動的ポート】** をクリックします。

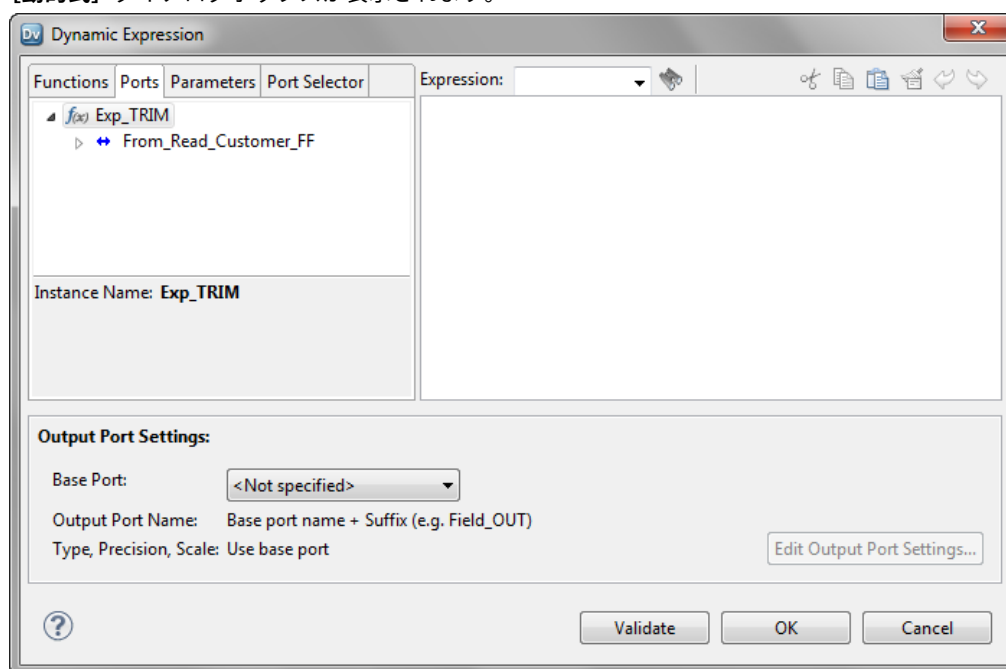
Developer tool で、デフォルトのプロパティを使用して動的ポートが作成されます。

3. 動的ポートの名前を変更し、入力オプションを無効にします。

動的ポートは出力ポートである必要があります。

4. 動的出力ポートの **【式】** カラムで、**【開く】** ボタン (  ) をクリックします。

**【動的式】** ダイアログボックスが表示されます。

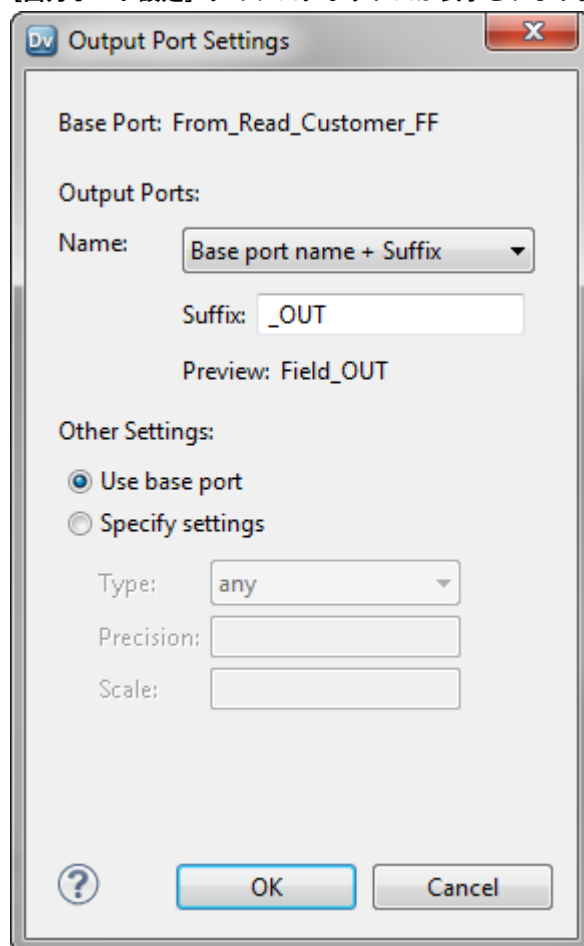


5. 式エディタに式を入力します。式にはポートセクタまたは動的ポートを含めることができます。  
例えば、LTRIM(RTRIM(Dynamic\_Customer))です。ここで、Dynamic\_Customer は動的ポートです。
6. **【検証】** をクリックして式を検証します。
7. **【OK】** をクリックして、**【式の検証】** ダイアログボックスを終了します。
8. **【出力ポート設定】** 領域で、**【ベースポート】** リストから動的出力ポートを選択するか、式で参照したポートセクタを選択します。

Developer tool で、選択に基づいて出力ポートが生成されます。



9. 次の手順を実行して出力ポートの名前を変更します。
  - a. **【出力ポート設定の編集】** をクリックします。  
**【出力ポート設定】** ダイアログボックスが表示されます。



- b. **【名前】** リストで、オプションのいずれかを選択してプレフィックスまたはサフィックスの値を入力します。**【固定文字列 + 自動番号割り当て】** を選択した場合、出力ポート名のテキストを入力します。例えば、出力ポート名に TRIM と入力すると、出力ポート名は TRIM1、TRIM2、TRIM3 と表示されます。
  - c. 必要に応じて、**【その他の設定】** 領域の **【設定の指定】** を選択し、出力ポートのタイプ、精度、およびスケールを変更します。デフォルトでは、出力ポートでベースポートの設定が使用されます。
  - d. **【OK】** をクリックします。
10. **【OK】** をクリックして、**【動的式】** エディタを終了します。

## 式トランスフォーメーションの詳細プロパティ

データ統合サービスが式トランスフォーメーションのデータを処理する方法を決定するプロパティを設定します。

式トランスフォーメーションの以下の詳細プロパティを設定します。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## 行順序を保持

トランスフォーメーションへの入力データの行順序を保持します。データ統合サービスが行順序を変更する可能性がある最適化を実行しないようにする場合に、このオプションを選択します。

データ統合サービスが最適化を実行すると、以前のマッピングで確立された順序が失われる場合があります。ソート済みフラットファイルソース、ソート済みリレーショナルソース、またはソータトランスフォーメーションを使用したマッピングにおける順序を確立できます。行順序を保持するようにトランスフォーメーションを設定すると、データ統合サービスではマッピングの最適化が実行される際に、この設定が考慮されます。データ統合サービスは、行順序を保持できる場合には、トランスフォーメーションの最適化を実行します。最適化により行順序が変更される可能性がある場合には、データ統合サービスはトランスフォーメーションの最適化を実行しません。

# 非ネイティブ環境での式トランスフォーメーション

非ネイティブ環境での式トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

## Blaze エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

ユーザー定義関数を持つ式トランスフォーメーションは、関数に例外エラーがある行に対して NULL 値を返します。

## Spark エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

**注:** 式の結果が、ゼロによる除算や負の数値の SQRT などの、数値エラーになった場合は、NULL 値を返し、出力に行は表示されません。ネイティブ環境では、式は infinite または NaN 値を返します。

## ストリーミングマッピングでの式トランスフォーメーション

ストリーミングマッピングには、Spark エンジン上のバッチマッピングと同じ処理ルールがあります。

## Databricks Spark エンジンでの式トランスフォーメーション

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションに、ステートフル変数ポートが含まれる。
- 式のトランスフォーメーションに、サポートされていない関数が含まれる。

**注:** 式の結果が、ゼロによる除算や負の数値の SQRT などの、数値エラーになった場合は、NULL 値を返し、出力に行は表示されません。ネイティブ環境では、式は infinite または NaN 値を返します。

## 第 17 章

# フィルタトランスフォーメーション

この章では、以下の項目について説明します。

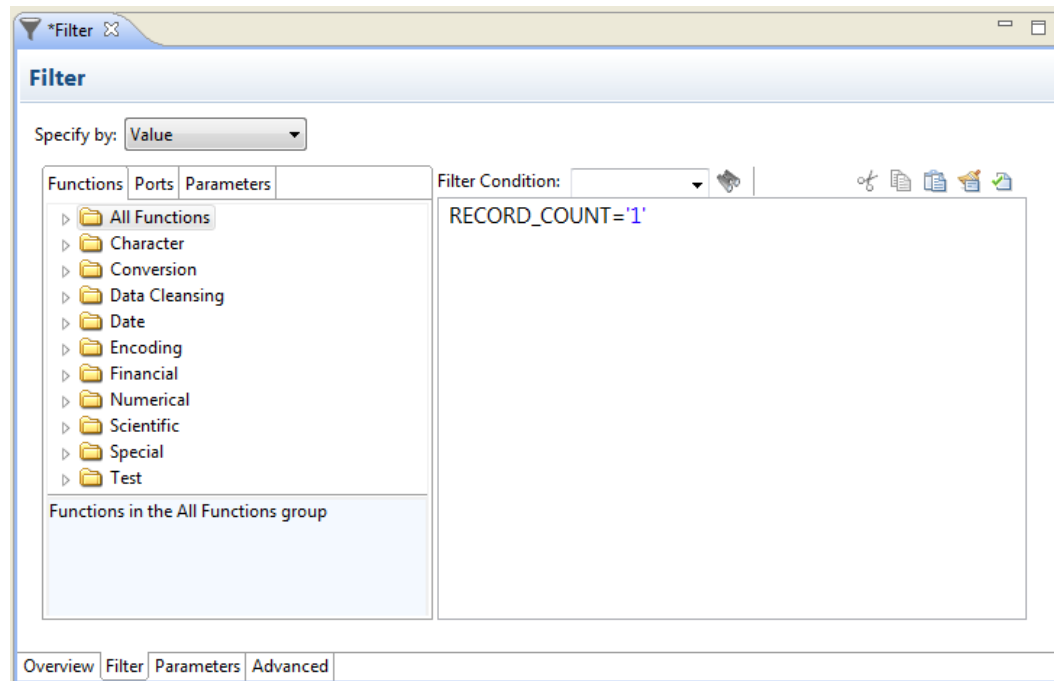
- [フィルタトランスフォーメーションの概要, 288 ページ](#)
- [動的マッピングでのフィルタトランスフォーメーション, 289 ページ](#)
- [フィルタ条件, 290 ページ](#)
- [フィルタトランスフォーメーションの詳細プロパティ, 292 ページ](#)
- [フィルタトランスフォーメーションのパフォーマンスのヒント, 292 ページ](#)
- [非ネイティブ環境でのフィルタトランスフォーメーション, 292 ページ](#)

## フィルタトランスフォーメーションの概要

フィルタトランスフォーメーションは、マッピング内の行をフィルタで除外するために使用します。フィルタトランスフォーメーションはアクティブトランスフォーメーションで、トランスフォーメーション通過する行の数を変更することができます。

フィルタトランスフォーメーションで行が通過を許可されるのは、指定されたフィルタ条件を満たした場合であり、条件を満たさない行は削除されます。データは 1 つ以上の条件に基づいてフィルタできます。

以下の図に、フィルタトランスフォーメーションのフィルタ条件を示します。



行が指定の条件を満たすかどうかを基準にして、データ統合サービスが評価する各行に対し、フィルタ条件は TRUE または FALSE を返します。TRUE を返した各行に対して、データ統合サービスはトランスフォーメーションを通過し、FALSE を返した各行に対しては、データ統合サービスが削除して、メッセージをログに書き込みます。

複数のトランスフォーメーションからのポートを Filter トランスフォーメーションに連結することはできません。フィルタの入力ポートは、1 つのトランスフォーメーションからのものでなければなりません。

## 動的マッピングでのフィルタトランスフォーメーション

動的マッピングでフィルタトランスフォーメーションを使用できます。トランスフォーメーションで動的ポートを設定して、生成されたポートをフィルタ条件で参照できます。

完全なフィルタ条件をパラメータ化できます。式全体が含まれているデフォルト値を使用して、式パラメータを設定します。Developer tool は、パラメータのデフォルト値のフィルタ条件を検証しません。

フィルタ条件では動的ポートを参照できます。動的ポートには、生成された複数のポートを含めることができます。データ統合サービスは、フィルタ条件を展開して、生成された各ポートが含まれるようにします。式に含めるには、生成された各ポートが有効なタイプである必要があります。

フィルタ条件では生成されたポートを参照できます。ただし、生成されたポートが実行時に存在しない場合は、マッピングが失敗します。

# フィルタ条件

フィルタ条件は、TRUE または FALSE を返す式です。

式エディタで条件を入力します。フィルタ条件では大文字と小文字が区別されます。

フィルタとして 1 つの値を返すすべての式を使用できます。例えば給料が\$30,000 以下の従業員の行を除外する場合は、以下のように条件を入力します。

`SALARY > 30000`

AND および OR 論理演算子を使用すると、複数の条件を指定することができます。例えば給料が\$30,000 未満であるか、または\$100,000 を超える従業員を除外する場合は、以下のように条件を入力します。

`SALARY > 30000 AND SALARY < 100000`

フィルタ条件には、ポート、パラメータ、動的ポート、生成されたポートを使用できます。式エディタでポートとパラメータを選択します。

フィルタ条件に動的ポートを使用した場合は、フィルタ条件が展開され、動的ポート内のすべての生成されたポートが取り込まれます。例えば、動的ポート MyDynamicPort に次の 3 つの 10 進ポートが含まれているとします。

Salary  
Bonus  
Stock

このとき、次のフィルタ条件を設定します。

`MyDynamicPort > 100`

このフィルタ条件は次の式に展開されます。

`Salary > 100 AND Bonus > 100 AND Stock > 100`

フィルタ条件には定数を入力できます。FALSE に該当する値はゼロ (0) です。ゼロ以外の値は TRUE とみなされます。例えば、トランスフォーメーションに数値データ型を使用する NUMBER\_OF\_UNITS というポートがあるとします。NUMBER\_OF\_UNITS の値が 0 に等しければ FALSE を返すようにフィルタ条件を設定します。値がゼロでなければ、TRUE が返されます。

**注:** 単一のポートセレクタや動的ポートはブール値として使用できません。

TRUE または FALSE を式の値として指定する必要はありません。TRUE および FALSE は、設定したすべての条件に対する暗黙の戻り値です。フィルタ条件が NULL として評価されると、その行は FALSE になります。

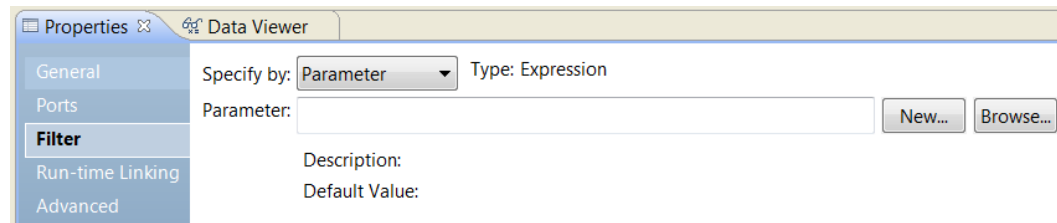
## フィルタ条件のパラメータ化

式パラメータを設定してフィルタ条件を定義できます。式パラメータには式全体が含まれます。

フィルタトランスフォーメーションが動的マッピング内にある場合は、フィルタ条件をパラメータ化する必要があります場合があります。フィルタ条件は、実行時にトランスフォーメーション内で生成されたポートに基づいて変わる可能性があります。

フィルタ条件の式パラメータを使用するには、フィルタトランスフォーメーションプロパティの **【フィルタ】** タブで、[指定元:] の **【パラメータ】** を選択します。

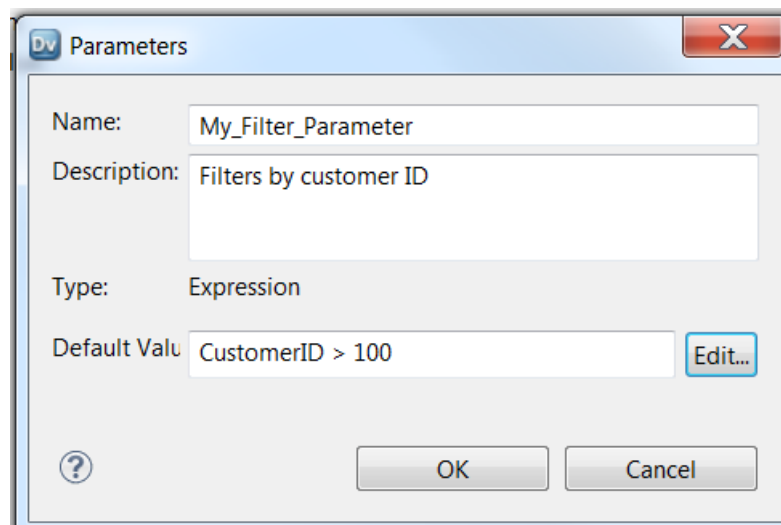
次の図は、パラメータでフィルタ条件を指定する場合の【フィルタ】タブを示しています。



作成済みの式パラメータを参照して選択できます。または、式パラメータを作成できます。

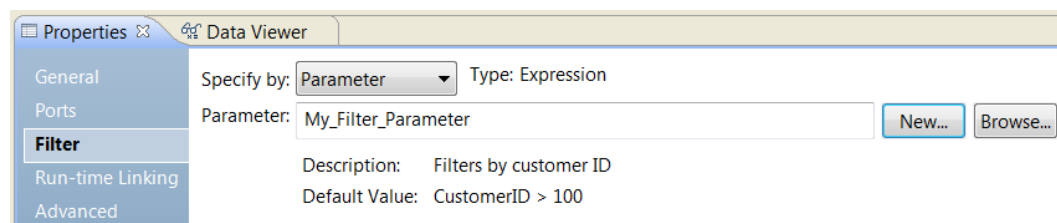
式パラメータを作成するには、【新規】をクリックします。パラメータの名前、説明、デフォルト式の値を入力します。

次の図は、パラメータ入力用のタブを示しています。



【パラメータ】ダイアログボックスにはデフォルト式を入力できます。式エディタを使用する場合は、【編集】をクリックします。式エディタでは、式で使用する関数とポートを選択できます。式を検証することもできます。

次の図は、【フィルタ】タブにフィルタ条件のパラメータを指定したところです。



式パラメータはマッピングパラメータです。パラメータセットまたはパラメータファイルのパラメータは実行時に上書きできます。

## NULL 値を含む行のフィルタ

NULL 値または空白を含む行をフィルタするには、ISNULL 関数と IS\_SPACES 関数を使用してポートの値をテストします。

たとえば FIRST\_NAME ポートで NULL 値を含む行をフィルタで除外したい場合は、次の条件を使用します。

```
IIF(ISNULL(FIRST_NAME),FALSE,TRUE)
```

この条件は、FIRST\_NAME ポートが NULL であれば戻り値として FALSE を返し、その行を無視することを指定しています。NULL が含まれていなければ、行は次のトランスフォーメーションへ渡されます。

## フィルタトランスフォーメーションの詳細プロパティ

Data Integration Service がフィルタトランスフォーメーションのデータを処理する方法を決定するプロパティを設定します。

ログに表示するトレースレベルを設定できます。

**【詳細】** タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## フィルタトランスフォーメーションのパフォーマンスのヒント

フィルタトランスフォーメーションのパフォーマンスを向上させるためのヒントを紹介します。

**マッピングの初期段階でフィルタトランスフォーメーションを使用します。**

マッピング内のソースのできる限り近くにフィルタトランスフォーメーションを配置します。マッピングを介して削除する予定の行を通過させる代わりに、ソースからターゲットへのデータフローの初期段階で不必要なデータを除外することができます。

## 非ネイティブ環境でのフィルタトランスフォーメーション

非ネイティブ環境でのフィルタトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。



- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限なしでサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

## Blaze エンジンでのフィルタトランスフォーメーション

Hive ソースのパーティション化されたカラムでのフィルタトランスフォーメーションがマッピングに含まれている場合、Blaze エンジンはフィルタ条件を満たすデータを持つパーティションのみを読み取れます。フィルタを Hive ソースにプッシュするには、フィルタトランスフォーメーションがソースの後のマッピング内の次のトランスフォーメーションになるように設定します。

## 第 18 章

# 階層型からリレーショナルへのトランスフォーメーション

この章では、以下の項目について説明します。

- [階層型からリレーショナルへのトランスフォーメーションの概要, 294 ページ](#)
- [例 - 階層型からリレーショナルへのトランスフォーメーション, 295 ページ](#)
- [出力リレーショナルポートと【概要】ビュー, 296 ページ](#)
- [階層型からリレーショナルへのトランスフォーメーションのポート, 297 ページ](#)
- [スキーマ参照, 298 ページ](#)
- [ポート設定, 298 ページ](#)
- [階層型からリレーショナルへのトランスフォーメーションの開発, 299 ページ](#)

## 階層型からリレーショナルへのトランスフォーメーションの概要

階層型からリレーショナルへのトランスフォーメーションでは、XML または JSON 階層入力を処理し、リレーショナル出力に変換します。階層型からリレーショナルへのトランスフォーメーションでは、入力ポートから階層入力を読み取り、そのデータをトランスフォーメーション出力ポートでリレーショナル出力に変換します。階層入力をリレーショナル出力に変換するには、スキーマファイルを使用して階層データを定義します。

階層型からリレーショナルへのトランスフォーメーションウィザードを使用すると、データを自動的にマッピングできます。リレーショナル出力ポートへのマッピングは、トランスフォーメーションの【概要】ビューで設定できます。

ウィザードによってトランスフォーメーションが生成されたら、リレーショナル出力ポートからのデータをマッピング内の別のトランスフォーメーションに渡すことができます。

**注:** 階層型からリレーショナルへのトランスフォーメーションは、1 つの.xsd ファイルで最大 10,000 個のスキーマ要素を処理できます。10,000 個を超える要素を処理するには、データを複数のファイルに分割します。

## 例 - 階層型からリレーショナルへのトランスフォーメーション

Harrinder Shipping 社には出荷データを処理するロジスティクス部門があります。在庫データや顧客データを階層形式からデータベーステーブルに保存可能なリレーショナルデータに変換する必要があります。

階層データをリレーショナルデータに変換するマッピングの作成が必要です。会社の在庫システムでは、出荷在庫データを階層形式で生成します。マッピングで階層型からリレーショナルへのトランスフォーメーションを使用し、出荷データを入力し、使用可能なリレーショナル形式で明細を出力する必要があります。

出荷入力には階層形式です。出荷要素には、出荷ごとに顧客および在庫データが含まれるサブ要素があります。

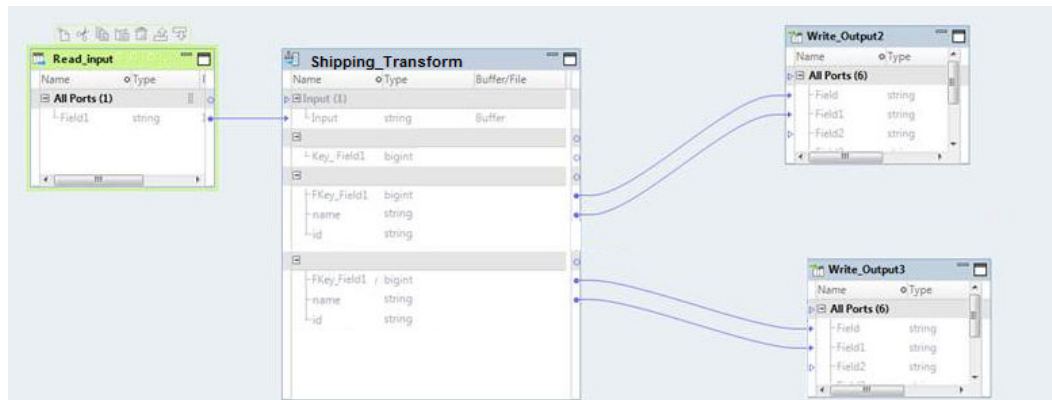
```
Shipments
Shipment
 Items
 Item_Name
 Inventory_ID
 Customer
 Customer_Name
 Customer_ID
 Customer_Address
```

リレーショナル出力では、Customer\_ID 要素は顧客テーブルのプライマリキーであり、出荷テーブルの外部キーです。

| Customer_ID | Customer_Name  | Customer_Address      |
|-------------|----------------|-----------------------|
| 3543766     | Tony Birch     | 6 Moby Drive          |
| 6342562     | Sujita Man     | 22 Dan Street         |
| 6471862     | Dwayne Horace  | 7 Jafendar Boulevard  |
| 7265204     | Carmela Perez  | 23 Dan Street         |
| 4559672     | Delilah Soraya | 28 Jafendar Boulevard |

| Shipment_ID | Inventory_Item | Customer_ID |
|-------------|----------------|-------------|
| 9173327437  | 908274         | 7265204     |
| 9174562342  | 553439         | 7265204     |
| 8484526471  | 546584         | 3543766     |
| 7023847265  | 908274         | 3543766     |
| 9174596725  | 553439         | 3543766     |

次の図は、この例のマッピングを示します。



マッピングには次のオブジェクトが含まれます。

Read\_input

階層データがあるファイルのパスを含むソース。XML ファイルから請求データを読み込みます。

Shipping\_Transform

XML 入力をリレーショナル出力に変換する階層型からリレーショナルへのトランスフォーメーション。

Write\_Output2

変換済みデータの一部を保存するターゲット、リレーショナル形式の顧客テーブル。

Write\_Output3

変換済みデータの他の一部を保存する 2 番目のターゲット、リレーショナル形式の出荷テーブル。

マッピングでは、**Read\_input** フラットファイルを使用して階層入力のターゲットパスを入力します。

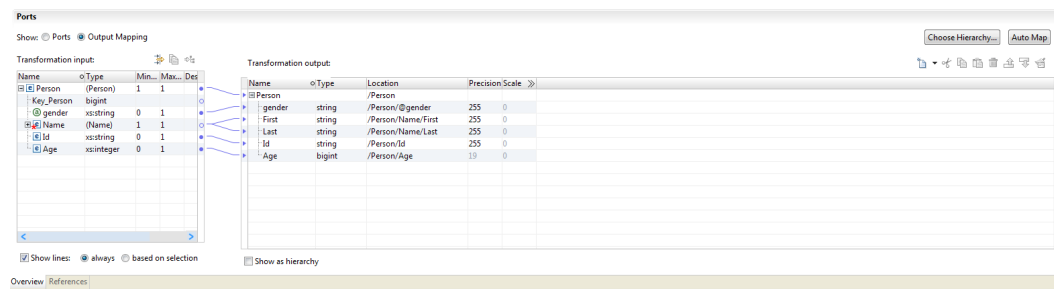
**Shipping\_Transform** トランスフォーメーションでデータが処理され、変換されます。その出力は 2 つの出力ターゲットに保存されます。

## 出力リレーショナルポートと [概要] ビュー

階層型からリレーショナルへのトランスフォーメーションで階層データをリレーショナル出力に変換するには、ウィザードで階層ノードとリレーショナルポート間のリンクを生成します。**[概要]** ビューを使用して、リレーショナルポートと階層ポート間のリンクを表示します。階層出力からポートのグループにノードをリンクして、出力ポートのグループを作成することもできます。

リレーショナルグループのマッピングを表示するには、**[概要]** ビューを使用します。**[出力マッピング]** を選択します。**[概要]** ビューに **[ポート]** パネルが表示されます。

以下の図は、【ポート】 パネルを示しています。



階層スキーマが表示される【トランスフォーメーション入力】領域は左側にあります。リレーショナル出力ポートが表示される【トランスフォーメーション出力】領域は右側にあります。

【トランスフォーメーション出力】領域でリレーショナル出力ポートを定義し、スキーマからポートにノードをリンクできます。スキーマのノードから【トランスフォーメーション出力】領域の空のフィールドにポインタをドラッグしてポートを作成することもできます。ノードを出力スキーマからポートにドラッグすると、Developer tool はそれらの間にリンクを表示します。

## 階層型からリレーショナルへのトランスフォーメーションのポート

階層型からリレーショナルへのトランスフォーメーションのポートは、トランスフォーメーションの【概要】ビューで定義します。

階層型からリレーショナルへのトランスフォーメーションでは、ファイルまたはバッファから入力を読み取ることができます。出力ポートは、トランスフォーメーションからのリレーショナルデータを返します。

階層型からリレーショナルへのトランスフォーメーションを作成すると、Developer tool によってデフォルトの入力ポートが作成されます。入力タイプによって、データ統合サービスが階層型からリレーショナルへのトランスフォーメーションに渡すデータのタイプが決まります。入力タイプによって、入力がデータであるかソースファイルパスであるかが決まります。

以下の入力タイプのいずれかを設定します。

### バッファ

階層型からリレーショナルへのトランスフォーメーションは、入力ポートでソースデータ行を受け取ります。Informatica トランスフォーメーションからデータを受け取るようにトランスフォーメーションを設定する場合は、バッファ入力タイプを使用します。

### ファイル

階層型からリレーショナルへのトランスフォーメーションは、入力ポートでソースファイルパスを受け取ります。階層型からリレーショナルへのトランスフォーメーションはソースファイルを開きます。また、バッファ入力ポートで処理すると大量のシステムメモリが必要な場合がある大きなファイルにも、ファイル入力タイプを使用することができます。

新しいトランスフォーメーションウィザードでトランスフォーメーションを作成する場合は、サンプル入力ファイルを定義できます。サンプル入力ファイルは、入力ファイルの小さなサンプルです。階層型からリレーショナルへの作成時には、サンプル入力ファイルを参照します。また、【データビューア】ビューでトランスフォーメーションをテストするときにもサンプル入力ファイルを使用します。

トランスフォーメーションには、リレーショナルデータを返す 1 つ以上のポートグループが含まれています。

## スキーマ参照

階層型からリレーショナルへのトランスフォーメーションでは、階層スキーマによってトランスフォーメーションの入力階層を定義する必要があります。トランスフォーメーションでスキーマを使用するには、スキーマ参照を定義します。

トランスフォーメーションのスキーマ参照は、トランスフォーメーションの【参照】ビューで定義します。

階層型からリレーショナルへのトランスフォーメーションでは、モデルリポジトリのスキーマオブジェクトが参照されます。このスキーマオブジェクトは、トランスフォーメーションを作成する前のリポジトリに存在することができます。また、スキーマをトランスフォーメーションの【リファレンス】ビューからインポートすることもできます。

スキーマは別のスキーマを参照できます。【参照】ビューには、階層型からリレーショナルへのトランスフォーメーションが参照する各スキーマの名前空間とプレフィックスが表示されます。名前空間が空になっている複数のスキーマを参照した場合、トランスフォーメーションは無効になります。

## ポート設定

【ポート】パネルに、トランスフォーメーションは階層スキーマノードとリレーショナルポート間のマッピングを表示します。トランスフォーメーションではスキーマを使用して階層入力を定義します。スキーマにルート要素にできる複数の要素がある場合、1つのノードをルート要素に選択します。

ウィザードで階層スキーマノードとリレーショナルポート間のリンクを生成します。生成されたリンクを変更する場合、【ポート】パネルを使用してリンクを追加、削除、編集できます。ノードをポートにリンクし、ポートを作成できます。

ノードを【トランスフォーメーション出力】領域にリンクすると、Developer tool によって、場所フィールドが階層内のノードの場所で更新されます。ポートを手動で作成する場合、ノードをポートにマッピングする必要があります。【場所】カラムを更新し、リストからノードを選択します。

複数出現ノードを親要素が含まれているグループにリンクする場合、含める子要素の出現回数を設定できます。または、親グループを、トランスフォーメーション出力内の複数出現子グループで置き替えることもできます。

グループを作成するには、ノードを【トランスフォーメーション出力】領域の空のカラムにリンクします。複数出現子ノードを空の入力カラムまたは出力カラムにリンクすると、Developer tool によって、そのグループを別の出力グループに関連付けるよう求めるメッセージが表示されます。関連付けるグループを選択すると、Developer tool により、グループ間に関連付けるキーが作成されます。

【トランスフォーメーション出力】領域で、関連する出力ポートのグループを設定します。Developer ツールによって、出力グループに関連付けるよう求めるメッセージが表示された場合、グループにキーが追加されます。キーを表すポートを手動で追加することもできます。

# 階層型からリレーショナルへのトランスフォーメーションの開発

新しいトランスフォーメーションウィザードを使用して、階層型からリレーショナルへのトランスフォーメーションを自動生成します。スキーマまたは階層サンプルファイルを選択して、入力階層を定義してください。

1. Developer tool でトランスフォーメーションを作成します。
2. 入力ポートおよびマッピングを設定します。
3. トランスフォーメーションをテストします。

## 階層型からリレーショナルへのトランスフォーメーションの作成

1. Developer tool で、**【ファイル】** > **【新規】** > **【トランスフォーメーション】** をクリックします。
2. 階層型からリレーショナルへのトランスフォーメーションを選択し、**【次へ】** をクリックします。
3. トランスフォーメーションの名前を入力し、モデルリポジトリでトランスフォーメーションを配置する場所を参照して、**【次へ】** をクリックします。
4. スキーマを選択するには、次の方法のいずれかを選択します。
  - モデルリポジトリのスキーマを使用して入力階層を定義するには、**【スキーマオブジェクト】** フィールド付近で、リポジトリのスキーマファイルを参照して選択します。
  - スキーマファイルをインポートするには、**【新規スキーマオブジェクトの作成】** をクリックします。**【新規スキーマオブジェクト】** ウィンドウで、スキーマファイルを参照し選択するか、サンプル階層ファイルからのスキーマ作成を選択できます。
5. 出力階層のルートを選択します。**【階層のルート】** ダイアログボックスで、出力階層ファイルのルート要素となっている、スキーマ内の要素を選択します。ルートオブジェクトの選択をしやすくするため、サンプル階層ファイルを追加できます。サンプルファイルを追加するには、**【サンプルファイル】** フィールド付近でファイルシステムからファイルを参照し選択します。
6. **【完了】** をクリックします。  
ウィザードにより、リポジトリにトランスフォーメーションが作成されます。

## ポートとマッピングの設定

**【概要】** ビューで入力ポートおよび出力ポートを設定します。

1. 入力ポートのデータ型、ポートタイプ、精度、スケールを選択します。
2. マッピングを表示するには、**【概要】** ビューの **【ポート】** 領域で、**【出力マッピング】** を選択します。
3. **【ポート】** グリッドでツリーを展開します。左側の **【トランスフォーメーション入力】** パネルには期待される階層入力が表示され、右側の **【トランスフォーメーション出力】** パネルにはリレーショナル出力が表示されます。
4. ノードをルートとして定義するには、**【階層の選択】** をクリックします。  
Developer tool では、**【トランスフォーメーション入力】** 領域にルートレベルのノードとルートレベルより下のノードだけが表示されます。
5. ポートを階層ノードと関連付ける行を表示するには、**【行の表示】** をクリックします。すべての関連付けられた行を表示するか、選択したポートの行だけを表示するかを選択します。
6. **【トランスフォーメーション出力】** 領域に入力グループまたは入力ポートを追加するには、次のいずれかの方法を使用します。

- **【トランスフォーメーション入力】** 領域の単純要素または複合要素を、**【トランスフォーメーション出力】** 領域の空のカラムにドラッグします。グループノードの場合は、Developer tool によってポートのないリレーショナルグループが追加されます。
  - リレーショナルグループを追加するには、行を選択して右クリックし、**【新規作成】** > **【グループ】** を選択します。
  - リレーショナルポートを追加するには、右クリックして **【新規作成】** > **【フィールド】** を選択します。
7. ポートの場所の階層ノード設定をクリアするには、次のいずれかの方法を使用します。
- **【トランスフォーメーション入力】** 領域で 1 つ以上のノードを選択して右クリックし、**【クリア】** を選択します。
  - リレーショナルポートと階層ノードを結ぶ線を 1 つ以上選択してから、右クリックして **【削除】** を選択します。
8. 出力ポートを階層形式で表示するには、**【階層で表示】** をクリックします。各子グループは親グループの下に表示されます。

## トランスフォーメーションのテスト

**【データビューア】** ビューで、階層型からリレーショナルへのトランスフォーメーションをテストします。

トランスフォーメーションをテストする前に、ファイル入力場所が定義されていることを確認します。データ統合サービスマシン上の入力場所は、**【概要】** ビューの **【ポート】** パネルの **【入力場所】** カラムに定義します。

1. **【データビューア】** ビューを開きます。
2. **【実行】** をクリックします。

トランスフォーメーションが検証されます。エラーがない場合は、Developer tool の **【出力】** パネルに階層ファイルの内容が表示されます。



## 第 19 章

# Java トランスフォーメーション

この章では、以下の項目について説明します。

- [Java トランスフォーメーションの概要, 301 ページ](#)
- [Java トランスフォーメーションの設計, 305 ページ](#)
- [Java トランスフォーメーションのポート, 305 ページ](#)
- [Java トランスフォーメーションの詳細プロパティ, 306 ページ](#)
- [Java コードの開発, 309 ページ](#)
- [Java トランスフォーメーションの Java のプロパティ, 313 ページ](#)
- [Java トランスフォーメーションによるフィルタの最適化, 316 ページ](#)
- [Java トランスフォーメーションの作成, 318 ページ](#)
- [Java トランスフォーメーションのコンパイル, 320 ページ](#)
- [Java トランスフォーメーションのトラブルシューティング, 320 ページ](#)
- [Struct データへの変換例, 322 ページ](#)
- [非ネイティブ環境での Java トランスフォーメーション, 325 ページ](#)

## Java トランスフォーメーションの概要

Java トランスフォーメーションを使用すると、Developer ツールの機能を拡張できます。

Java トランスフォーメーションは、Java プログラミング言語を使用してトランスフォーメーション機能を定義するための、単純なネイティブのプログラミングインタフェースを提供します。Java トランスフォーメーションを使用すると、Java プログラミング言語または外部 Java 開発環境に関する高度な知識がなくても、単純またはやや高度なトランスフォーメーション機能を定義することができます。Java トランスフォーメーションには、アクティブとパッシブの両方があります。

Developer ツールは、Java Development Kit (JDK) を使用して Java コードをコンパイルし、トランスフォーメーションのバイトコードを生成します。バイトコードはモデルリポジトリに格納されます。

Data Integration Service は Java Runtime Environment (JRE) を使用して、生成されたバイトコードをランタイムで実行します。Data Integration Service は、Java トランスフォーメーションを使用してマッピングを実行するとき、JRE を使用してバイトコードを実行し、入力行を処理して出力行を生成します。

Java トランスフォーメーションを作成するには、トランスフォーメーションロジックを定義する Java コードスニペットを記述します。以下のイベントに基づいて、Java トランスフォーメーションのトランスフォーメーション動作を定義します。

- トランスフォーメーションが入力行を受け取ったとき。

- トランスフォーメーションがすべての入力行を処理したとき。

Spark エンジンで実行されるマッピングでは、Java トランスフォーメーションで複合データ型を使用して階層データを処理できます。複合データ型を使用すると、Spark エンジンは、Avro、Parquet、および JSON の複合ファイル内の階層データの読み取り、処理、および書き込みを直接行います。

## 再利用可能および再利用不可能な Java トランスフォーメーション

再利用可能または再利用不可能な Java トランスフォーメーションを作成できます。

再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。再利用不可能なトランスフォーメーションは、単一のマッピングで使用されます。

プロパティの定義や Java コードの作成を行うエディタのビューは、再利用可能な Java トランスフォーメーションを作成しているか再利用不可能な Java トランスフォーメーションを作成しているかによって異なります。

## アクティブ Java トランスフォーメーションとパッシブ Java トランスフォーメーション

Java トランスフォーメーションを作成するときは、そのタイプをアクティブまたはパッシブとして定義します。

トランスフォーメーションタイプは、設定した後で変更することはできません。

Java トランスフォーメーションは、入力データのそれぞれの行に対して 1 回、**【入力時】** タブで定義された Java コードを実行します。

Java トランスフォーメーションは、以下のように、トランスフォーメーションタイプに基づいて出力行を処理します。

- パッシブ Java トランスフォーメーションは、トランスフォーメーションのそれぞれの入力行を処理した後、各入力行に対して 1 つの出力行を生成します。
- アクティブ Java トランスフォーメーションは、トランスフォーメーションのそれぞれの入力行に対して複数の出力行を生成します。

各出力行を生成するには、`generateRow` メソッドを使用します。例えば、トランスフォーメーションに開始日と終了日を表す 2 つの入力ポートが含まれている場合は、`generateRow` メソッドを使用して、開始日と終了日の間の各日付に対して出力行を生成することができます。

## データ型の変換

Java トランスフォーメーションは、Java トランスフォーメーションのポートタイプに基づいて、Developer ツールのデータ型を Java データ型に変換します。

Java トランスフォーメーションは、入力行を読み込むと、入力ポートデータ型を Java データ型に変換します。

Java トランスフォーメーションは、出力行を書き込むと、Java データ型を出力ポートデータ型に変換します。

例えば、Java トランスフォーメーションの整数データ型の入力ポートに対しては、以下の処理が行われます。

1. Java トランスフォーメーションは、入力ポートの整数データ型を Java プリミティブデータ型 `int` に変換します。
2. このトランスフォーメーションで、トランスフォーメーションは入力ポートの値を Java プリミティブデータ型 `int` として扱います。
3. トランスフォーメーションは、出力行を生成すると、Java プリミティブデータ型 `int` を整数データ型に変換します。

以下の表に、Java トランスフォーメーションが Developer ツールのデータ型を Java プリミティブデータ型および複合データ型にマッピングする方法を示します。

| Developer ツールのデータ型      | Java データ型                                                                                                |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| array*                  | java.util.List                                                                                           |
| bigint                  | long                                                                                                     |
| binary                  | byte[]                                                                                                   |
| date/time               | ナノ秒の処理を有効にした場合は、ナノ秒の精度の BigDecimal<br>ナノ秒の処理を無効にした場合は、ミリ秒の精度の long (1970/01/01 00:00:00.000 GMT 以降のミリ秒数) |
| decimal                 | 高精度の処理を無効にした場合は、精度が 15 の double<br>高精度の処理を有効にした場合は、BigDecimal                                            |
| double                  | double                                                                                                   |
| integer                 | 整数型                                                                                                      |
| map*                    | java.util.Map                                                                                            |
| string                  | ストリング                                                                                                    |
| struct*                 | struct フィールド要素のゲッターとセッターによりカスタマイズされた JavaBean クラス                                                        |
| text                    | String                                                                                                   |
| *Spark エンジンでのみサポートされます。 |                                                                                                          |

Java では、java.util.List、java.util.Map、String、byte[]、BigDecimal の各データ型は、複合データ型になります。double、int、long の各データ型は、プリミティブデータ型になります。

Developer ツールでは、array、struct、map の各データ型は、複合データ型になります。

Java トランスフォーメーションは、プリミティブデータ型の NULL 値をゼロに設定します。**【入力時】** タブでは、isNull API メソッドおよび setNull API メソッドを使用して、入力ポートの NULL 値を出力ポートの NULL 値に設定できます。例については、[「setNull」 \(ページ 336\)](#)を参照してください。

**注:** decimal データ型は、高精度が有効なときに BigDecimal にマッピングされます。BigDecimal は、+などの演算子と一緒に使用できません。decimal ポートまたは decimal データ型の要素を持つ複合ポートを使用する式が Java コードに含まれていて、そのポートがいずれかの演算子と一緒に使用された場合、Java コードはコンパイルに失敗します。

## Spark エンジンでの複合データ型の変換

Spark エンジンで実行するマッピングで、Java トランスフォーメーションに複合ポートを追加できます。複合ポートは、複合データ型に割り当てられるポートです。Java トランスフォーメーションは、複合データ型を対

応する Java 複合データ型に変換します。さらに、複合データ型の要素を対応する Java データ型に変換します。これは、プリミティブデータ型のボックス化されたバージョンです。

## array データ型

Java トランスフォーメーションは、入力行を読み込むと、array データ型を Java List データ型に変換し、array 要素のデータ型を Java データ型のボックス化されたバージョンに変換します。

例えば、Developer tool の複合データ型 `array<integer>` を Java データ型 `List<Integer>` に変換します。

Java トランスフォーメーションは、出力行を書き込むときに、Java List データ型を array データ型に変換し、List 要素のデータ型を Developer tool データ型のボックス化されていないバージョンに変換します。

## Struct データ型

Java トランスフォーメーションは、struct データ型の入力行を読み込むと、等価な Java Bean クラスを生成します。Java Bean クラスの名前は、struct データ型の名前と同じです。struct 要素のデータ型を Java データ型のボックス化されたバージョンに変換します。

struct データ型で特殊文字や Java 予約語（`final` や `private` など）を使用している場合、Java トランスフォーメーションはクラス名の特殊文字をアンダースコア（`_`）に置換します。トランスフォーメーションのプロパティタブの [Java] ビューにコード全体を開いて、生成されたクラスを確認できます。

Java トランスフォーメーションは、プレフィックスとしてアンダースコア（`_`）がついたクラスメンバーフィールドを生成し、メンバーフィールドのゲッターとセッターも生成します。

生成される Java Bean クラスは、外部クラスとして型定義ライブラリ名の名前空間を持ちます。この外部クラスの名前は、型定義ライブラリの名前と同じです。struct ポートの Java データ型名の形式は次のとおりです。

`type_library_name.struct_type_name`

例えば、型ライブラリ名が `m_Type_Definition_Library` であるとします。struct ポートの複合データ型定義は次のとおりです。

```
Customer {
 name string
 age integer
}
```

Java トランスフォーメーションは、Java Bean クラスと、メンバーフィールドのゲッターとセッターを生成します。外部クラスと内部クラスのコードスニペットは次のようになります。

```
public static final class m_Type_Definition_Library {
 public static final class Customer implements Serializable {
 private String _name;
 private Integer _age;

 public Customer() {}
```

string 型の struct 要素名のゲッターとセッターのコードスニペットは次のようになります。

```
public String get_name() {
 return _name;
}

public void set_name(String _name) {
 this._name = _name;
}
```

Java トランスフォーメーションは、struct データ型の出力行を書き込むときに、Java Bean クラスを struct データ型に変換し、このクラスのメンバーフィールドを struct 要素に変換します。メンバーフィールドのデータ型は、Developer tool データ型のボックス化されていないバージョンに変換されます。

## Map データ型

Java トランスフォーメーションは、入力行を読み込むと、map データ型を Java Map データ型に変換し、map 要素のデータ型を Java データ型のボックス化されたバージョンに変換します。

例えば、Developer tool の複合データ型 `map<string, bigint>` は、Java データ型 `Map<String, long>` に変換されます。

Java トランスフォーメーションは、出力行を書き込むときに、Java Map データ型を `map` データ型に変換し、Map 要素のデータ型を Developer tool データ型のボックス化されていないバージョンに変換します。

## Java トランスフォーメーションの設計

Java トランスフォーメーションを設計するときは、作成するトランスフォーメーションのタイプなどの要素について考慮する必要があります。

Java トランスフォーメーションの設計時には、以下の点について検討します。

- アクティブまたはパッシブ、どちらの Java トランスフォーメーションを作成する必要があるか。  
パッシブ Java トランスフォーメーションは、トランスフォーメーション内のそれぞれの入力行に対して 1 つの出力行を生成します。  
アクティブ Java トランスフォーメーションは、トランスフォーメーションのそれぞれの入力行に対して複数の出力行を生成します。
- Java トランスフォーメーションに関数を定義する必要があるか。定義する必要がある場合、各関数にどの式を含めるか。  
例えば、入力または出力ポートの値をルックアップする式、あるいは Java トランスフォーメーション変数の値をルックアップする式を呼び出す関数を定義できます。
- 再利用可能な Java トランスフォーメーションと再利用不可能な Java トランスフォーメーションのどちらを作成するか。  
再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。  
再利用不可能なトランスフォーメーションは、単一のマッピングで使用できます。

## Java トランスフォーメーションのポート

Java トランスフォーメーションは、入力ポートと出力ポートを持つことができます。

再利用不可能な Java トランスフォーメーションのポートの作成や編集を行うには、エディタの **【ポート】** タブを使用します。再利用可能な Java トランスフォーメーションのポートの作成や編集を行うには、エディタの **【概要】** ビューを使用します。

ポートのデフォルト値を指定できます。トランスフォーメーションにポートを追加したら、ポート名を Java コードスニペット内で変数として使用できます。

### ポートの作成

作成した Java トランスフォーメーションには、入力グループおよび出力グループが 1 つずつ含まれています。ポートを作成すると、Developer ツールはそのポートを現在選択されている行またはグループの下に追加します。

## デフォルトポート値の設定

Java トランスフォーマーセッションでは、ポートのデフォルト値を定義できます。

Java トランスフォーマーセッションは、ポートのデータ型に基づいて、ポートのデフォルト値を適用してポート変数を初期化します。

### 入出力ポート

Java トランスフォーマーセッションは、Java コードスニペットに値が割り当てられていない未接続の n 入力または出力ポートの値を初期化します。

Java トランスフォーマーセッションでは、次の Java データ型に基づいて、ポートを初期化します。

#### プリミティブデータ型

ポートのデフォルト値を NULL 以外の値に定義した場合、トランスフォーマーセッションはポート変数の値をそのデフォルト値に初期化します。それ以外の場合、ポート変数の値は 0 に初期化されます。

#### 複合データ型

ポートのデフォルト値を定義した場合、トランスフォーマーセッションは新規のオブジェクトを作成し、そのオブジェクトをデフォルト値に初期化します。それ以外の場合、トランスフォーマーセッションはポート変数を NULL に初期化します。例えば、String ポートのデフォルト値を定義した場合、トランスフォーマーセッションは新規の String オブジェクトを作成し、その String オブジェクトをデフォルト値に初期化します。

**注:** Java コードで値が NULL の入力ポート変数にアクセスすると、NullPointerException が発生します。

入力ポートをパーティションキーおよびソートキーとして有効化し、ソート方向を割り当てることができます。データ統合サービスは、データをパーティション化し、ソートキーとソート方向によって各パーティション内のデータをソートします。トランスフォーマーセッションの範囲が [すべての入力] に設定されると、パーティションキーとソートキーは有効になります。

データのパーティション化およびソートには以下のプロパティを使用します。

#### パーティションキー

1 つのパーティションにグループ化するデータの行数を決定する入力ポート。

#### ソートキー

各パーティション内のソート基準を決定する入力ポート。

#### 方向

昇順または降順。デフォルトは昇順です。

## Java トランスフォーマーセッションの詳細プロパティ

Java トランスフォーマーセッションには、トランスフォーマーセッションコードとトランスフォーマーセッション両方の詳細プロパティが含まれます。

マッピング内でトランスフォーマーセッションを使用する際には、トランスフォーマーセッションのプロパティをオーバーライドできます。

**【詳細】** タブで、Java トランスフォーマーセッションの以下の詳細プロパティを定義できます。

#### トレースレベル

このトランスフォーマーセッションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## パーティション化可能

トランスフォーメーションを複数のスレッドで処理できます。データ統合サービスが1つのスレッドを使用してトランスフォーメーションを処理するようにする場合は、このオプションをクリアします。データ統合サービスは複数のスレッドを使用して残りのマッピングパイプラインステージを処理します。

Java コードで Java トランスフォーメーションを1つのスレッドで処理する必要がある場合に、Java トランスフォーメーションのパーティション化を無効にします。

## 高精度 10 進演算を有効にする

精度が 38 以下である decimal データ型ポートを、Java BigDecimal データ型ポートとして処理します。

高精度を無効にして、decimal データ型ポートを Java Double データ型ポートとして処理します。

次の表に、高精度オプションを有効にしたか無効にしたかに基づいて Java トランスフォーメーションが decimal データ型入力ポートの値を扱う方法を示します。

| 例                                          | 高精度処理が有効                      | 高精度処理が無効                                                               |
|--------------------------------------------|-------------------------------|------------------------------------------------------------------------|
| 10 進型入力ポートが値 40012030304957666903 を受け取ります。 | Java トランスフォーメーションは値をそのままにします。 | Java トランスフォーメーションは値を以下の値に変換します。<br>4.00120303049577 x 10 <sup>19</sup> |

Java トランスフォーメーションに decimal ポート、または decimal データ型の要素を持つ複合ポートが含まれる場合、トランスフォーメーションではマッピングと同じ精度モードを使用する必要があります。例えば、Java トランスフォーメーションで高精度を有効にする場合は、マッピングで高精度を有効にする必要があります。

## 日付/時刻にナノ秒を使用する

date/time データ型ポートを精度がナノ秒の Java BigDecimal データ型ポートに変換します。

ナノ秒処理を無効にすると、生成される Java コードは date/time データ型ポートを精度がミリ秒の Java Long データ型ポートに変換します。

## クラスパス

**【インポート】** タブでインポートする非標準 Java パッケージに関連付けられた JAR ファイルまたはクラスファイルのディレクトリのクラスパスを設定します。

Java コードのコンパイルには、JAR ファイルまたはクラスファイルのディレクトリが Developer tool クライアントマシン上でアクセス可能でなければなりません。

オペレーティングシステムに応じてクラスパスの各項目を以下のように区切ります。

- UNIX の場合、クラスパスの各項目を区切るにはコロンを使用します。
- Windows の場合、クラスパスの各項目を区切るにはセミコロンを使用します。

例えば、**【インポート】** タブで Java コンバータパッケージをインポートし、そのパッケージを converter.jar で定義する場合は、Java トランスフォーメーションの Java コードをコンパイルする前に converter.jar ファイルの場所をクラスパスに追加する必要があります。

**注:** 組み込みの Java パッケージの場合、クラスパスを設定する必要はありません。例えば、java.io は組み込み Java パッケージであるため、java.io に対してクラスパスを設定する必要はありません。

## アクティブ

トランスフォーメーションは、それぞれの入力行に対して複数の出力行を生成できます。



Java トランスフォーメーションを作成した後は、このプロパティを変更できません。このプロパティを変更する必要がある場合、新しい Java トランスフォーメーションを作成します。

### トランスフォーメーション範囲

データ統合サービスでトランスフォーメーションロジックを着信データに適用するために使用する方法を定義します。以下の値のいずれかを選択できます。

- 行。トランスフォーメーションロジックを、データの 1 つの行ごとに適用します。手続きの結果がデータの単一の行に依存する場合は [行] を選択してください。
- Transaction。トランスフォーメーションロジックをトランザクションのすべての行に適用します。手続きの結果が同一トランザクションのすべての行に依存し、他のトランザクションの行には依存していない場合には、[Transaction] を選択します。[Transaction] を選択した場合、すべての入力グループを同じトランザクション制御ポイントに接続する必要があります。
- すべての入力。トランスフォーメーションロジックをすべての入力データに適用します。[すべての入力] を選択すると、データ統合サービスはトランザクション境界を削除します。[すべての入力] は、手続きの結果がソース内のすべての行に依存する場合に選択します。

### ステートレス

トランスフォーメーションへの入力データの行順序を保持します。データ統合サービスが行順序を変更する可能性がある最適化を実行しないようにする場合に、このオプションを選択します。

データ統合サービスが最適化を実行すると、以前のマッピングで確立された順序が失われる場合があります。ソート済みフラットファイルソース、ソート済みリレーショナルソース、またはソータトランスフォーメーションを使用したマッピングにおける順序を確立できます。行順序を保持するようにトランスフォーメーションを設定すると、データ統合サービスではマッピングの最適化が実行される際に、この設定が考慮されます。データ統合サービスは、行順序を保持できる場合には、トランスフォーメーションの最適化を実行します。最適化により行順序が変更される可能性がある場合には、データ統合サービスはトランスフォーメーションの最適化を実行しません。

## Developer ツールクライアントのクラスパスの設定

Developer ツールクライアントのクラスパスに jar ファイルまたはクラスファイルのディレクトリを追加できます。

Developer ツールクライアントが動作しているマシン用にクラスパスを設定するには、次の作業のいずれかを実行します。

- CLASSPATH 環境変数を設定します。CLASSPATH 環境変数は Developer ツールクライアントマシン上で設定します。これは、マシン上で実行されている java プロセスすべてに適用されます。
- 再利用不可能な Java トランスフォーメーションの場合は、Java トランスフォーメーションの詳細プロパティでクラスパスを設定します。これは、この Java トランスフォーメーションを含むマッピングに適用されます。Developer tool クライアントは、Java コードをコンパイルするときにクラスパス内のファイルをインクルードします。

jar ファイルまたはクラスファイルのディレクトリを Java トランスフォーメーションのクラスパスに追加するには、以下の手順を実行します。

1. **【詳細】** タブで、**【クラスパス】** の横にある **【値】** カラムの下矢印アイコンをクリックします。  
**【クラスパスの編集】** ダイアログボックスが表示されます。
2. クラスパスを追加するには、以下の手順を実行します。
  - a. **【追加】** をクリックします。  
**【名前を付けて保存】** ウィンドウが表示されます。



- b. **【名前を付けて保存】** ウィンドウで、jar ファイルの保存先のディレクトリに移動します。
  - c. **【OK】** をクリックします。
- 【クラスパスの編集】** ダイアログボックスにクラスパスが表示されます。
3. jar ファイルまたはクラスファイルのディレクトリを削除するには、jar ファイルまたはクラスファイルのディレクトリを選択して **【削除】** をクリックします。
- ディレクトリの一覧からディレクトリが削除されます。

## データ統合サービスのクラスパスの設定

実行時に必要な jar ファイルまたはクラスファイルのディレクトリをデータ統合サービスノードのクラスパスに追加することができます。

実行時に必要な jar ファイルは、データ統合サービスノードの次のディレクトリに格納します。

`$INFA_HOME/services/shared/jars`

この場所に格納された jar ファイルは動的にロードされます。個々のマッピングの実行時に必要なクラスファイルは、このディレクトリから検出されてロードされます。

**注:** Java トランスフォーメーションでは、このディレクトリにある jar ファイルをマッピングレベルのクラスパスに追加します。

## Java コードの開発

特定のトランスフォーメーションイベントに対するトランスフォーメーションの動作を定義する Java コードを記述してコンパイルするには、**【Java】** ビューのコードエントリタブを使用します。

コードエントリタブでは、任意の順序でコードスニペットを開発できます。**【コード全体】** タブでは、Java コード全体の表示は可能ですが、編集はできません。

コードスニペットの開発後は、コードスニペットまたは Java コード全体をコンパイルして、**【Java】** ビューの**【結果】** ウィンドウの**【コンパイル】** プロパティでコンパイルの結果を表示することができます。

各コードエントリタブは、Java コードの記述、表示、およびコンパイルに使用するコンポーネントで構成されます。

## コードプロパティ

Java トランスフォーメーション API メソッドなどの Java コードの表示や入力ができるコントロールを提供します。以下の表に、**【コード】** プロパティに用意されているコントロールを示します。

| コントロール                  | 説明                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ナビゲータ                   | <p>入力ポート、出力ポート、および呼び出し可能な Java トランスフォーメーション API メソッドを表示します。</p> <p>ナビゲータ内の項目の説明を表示するには、その項目をクリックします。</p> <p><b>【Java コード】</b> ウィンドウに項目を追加するには、その項目をダブルクリックします。また、ナビゲータから <b>【Java コード】</b> ウィンドウに項目をドラッグすることもできます。</p> <p>ナビゲータは、以下のコードエントリタブに用意されています。</p> <ul style="list-style-type: none"><li>- ヘルパ</li><li>- 入力時</li><li>- 最後</li></ul>                                                              |
| <b>【Java コード】</b> ウィンドウ | <p>トランスフォーメーションの Java コードを表示または入力できます。 <b>【Java コード】</b> ウィンドウには、基本的な Java 構文ハイライトを使用して Java コードが表示されます。</p> <p><b>注:</b> <b>【コード全体】</b> タブでは、Java トランスフォーメーションのクラスコード全体の表示は可能ですが、編集はできません。</p> <p><b>【Java コード】</b> ウィンドウは、以下のコードエントリタブに用意されています。</p> <ul style="list-style-type: none"><li>- インポート</li><li>- ヘルパ</li><li>- 入力時</li><li>- 最後</li><li>- 関数</li><li>- 最適化インタフェース</li><li>- コード全体</li></ul> |
| <b>【新しい関数】</b> コマンド     | <p><b>【関数の定義】</b> ダイアログボックスを開きます。このダイアログボックスを使用して、Java 式を呼び出す関数を定義します。</p> <p><b>【関数】</b> コマンドは、<b>【関数】</b> タブに用意されています。</p>                                                                                                                                                                                                                                                                              |
| 編集ツールバー                 | <p>切り取り、コピー、貼り付けなど、Java コードを編集するためのツールアイコンをクリックできます。</p> <p>編集ツールバーは、以下のコードエントリタブに用意されています。</p> <ul style="list-style-type: none"><li>- インポート</li><li>- ヘルパ</li><li>- 入力時</li><li>- 最後</li><li>- 関数</li><li>- 最適化インタフェース</li></ul>                                                                                                                                                                        |

## コンパイルプロパティ Compilation properties

Java コードをコンパイルおよびデバッグするコントロールを提供します。以下の表に、コンパイルプロパティのコントロールを示します。

| コントロール       | 説明                                                                                                                                                                                                 |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 【コンパイル】 コマンド | トランスフォーメーション用の Java コードをコンパイルします。                                                                                                                                                                  |
| 【結果】 ウィンドウ   | Java トランスフォーメーションクラスのコンパイル結果が表示され、コード内のエラーの発生源を見つけることができます。<br>コード内のエラーを見つけるには、【結果】 ウィンドウでエラーメッセージを右クリックし、スニペットコードとコード全体のどちらでエラーを表示するかを選択します。<br>【結果】 ウィンドウでエラーメッセージをダブルクリックしてエラーの発生源を見つけることもできます。 |

## Java コードスニペットの作成

Java コードスニペットを作成してトランスフォーメーションの動作を定義するには、コードエントリタブの【Java コード】 ウィンドウを使用します。

1. 適切なコードエントリタブをクリックします。

以下の表に、【Java】 ビューのコードエントリタブで完了できるタスクを示します。

| タブ    | 説明                                                                                                                                                                                                |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| インポート | アクティブまたはパッシブな Java トランスフォーメーションに対して、サードパーティ製、組み込み、またはカスタムの Java パッケージをインポートします。パッケージのインポート後、それらのパッケージを他のコードエントリタブで使用できます。                                                                         |
| ヘルパ   | アクティブまたはパッシブな Java トランスフォーメーション内の Java トランスフォーメーションクラスのユーザー定義変数およびメソッドを宣言します。変数およびメソッドを宣言すると、それらを【インポート】 タブを除く他のすべてのコードエントリタブで使用できます。                                                             |
| 入力時   | 入力行を受け取った際のアクティブまたはパッシブな Java トランスフォーメーションの動作を定義します。このタブで定義した Java コードは、入力行ごとに 1 回実行されます。<br>このタブでは、入出力ポートのデータ、変数、および Java トランスフォーメーション API メソッドにアクセスして使用することもできます。                               |
| 最後    | すべての入力データを処理した後のアクティブまたはパッシブな Java トランスフォーメーションの動作を定義します。<br>このタブでは、アクティブなトランスフォーメーションの出力データの設定や、Java トランスフォーメーション API メソッドの呼び出しも行うことができます。                                                       |
| 関数    | Java トランスフォーメーションの式を呼び出す関数を、Java プログラミング言語を使用して定義します。例えば、入出力ポートの値や Java トランスフォーメーションの変数の値をルックアップする式を呼び出す関数を定義できます。<br>【関数】 タブで、手動で関数を定義するか、または【新しい関数】 をクリックして関数を簡単に定義できる【関数の定義】 ダイアログボックスを呼び出します。 |

| タブ         | 説明                                                                                                               |
|------------|------------------------------------------------------------------------------------------------------------------|
| 最適化インタフェース | 初期選択または最適化にプッシュインを定義します。ナビゲータで、最適化方式を選択します。最適化を有効にするには、コードスニペットを更新します。入力ポートおよびフィルタロジックをプッシュする関連付けられた出力ポートを定義します。 |
| コード全体      | 読み取り専用。このタブでは、Java トランスフォーメーションのクラスコード全体を表示し、コンパイルすることができます。                                                     |

- スニペット内の入力または出力カラムの変数にアクセスするには、ナビゲータに表示された【入力】または【出力】リストを展開し、ポートの名前をダブルクリックします。
- スニペット内の Java トランスフォーメーション API を呼び出すには、ナビゲータに表示された【呼び出し可能な API】リストを展開し、メソッドの名前をダブルクリックします。必要に応じて、メソッドに適切な入力値を設定してください。
- コードエントリタブのタイプに基づいて、適切な Java コードを書き込みます。  
【コード全体】タブの【Java コード】ウィンドウで、Java トランスフォーメーションのクラスコード全体を表示します。

## Java パッケージのインポート

【インポート】タブでは、アクティブまたはパッシブな Java トランスフォーメーションに対して Java パッケージをインポートできます。

サードパーティ製、組み込み、またはカスタムの Java パッケージをインポートできます。Java パッケージのインポート後、インポートされたパッケージを他のコードエントリタブで使用できます。

**注:** 【インポート】タブでは、静的変数、インスタンス変数、またはユーザーメソッドの宣言または使用はできません。

Developer ツールで、Java トランスフォーメーションを含むメタデータをエクスポートまたはインポートしても、Java トランスフォーメーションが必要とするサードパーティ製またはカスタムのパッケージが格納されている JAR ファイルやクラスファイルはエクスポートまたはインポートされません。

Java トランスフォーメーションを含むメタデータをインポートする場合は、必要なサードパーティ製またはカスタムのパッケージが格納された JAR ファイルまたはクラスファイルを、Developer ツールクライアントおよび Data Integration Service ノードにコピーする必要があります。

例えば Java I/O パッケージをインポートするには、【インポート】タブに以下のコードを入力します。

```
import java.io.*;
```

標準以外の Java パッケージをインポートするときは、Java トランスフォーメーションのでクラスパスにパッケージまたはクラスを追加します。

## Helper コードの定義

【ヘルパ】タブで、アクティブ Java トランスフォーメーションまたはパッシブ Java トランスフォーメーション内の Java トランスフォーメーションクラスのユーザー定義変数およびメソッドを宣言できます。

【ヘルパ】タブで変数およびメソッドを宣言すると、【インポート】タブを除くすべてのコードエントリタブでその変数およびメソッドを使用できます。

[ヘルパ] タブでは、以下のタイプのコード、変数、およびメソッドを宣言できます。

- 静的コードおよび静的変数。

静的ブロック内では、静的変数および静的コードを宣言できます。マッピング内の再利用可能な Java トランスフォーメーションのすべてのインスタンスが、静的コードおよび静的変数を共有します。静的コードは、Java トランスフォーメーション内の他のどのコードよりも先に実行されます。

例えば、以下のコードは、マッピング内の Java トランスフォーメーションのすべてのインスタンスに対するエラーしきい値を保存する静的変数を宣言します。

```
static int errorThreshold;
```

この変数を使用してトランスフォーメーションのエラーしきい値を保存すると、そのエラーしきい値にマッピング内の Java トランスフォーメーションの全インスタンスからアクセスできます。

**注:** 再利用可能な Java トランスフォーメーションでは、静的変数を同期する必要があります。

- インスタンス変数。

マッピング内の再利用可能な Java トランスフォーメーションの複数のインスタンスは、インスタンス変数を共有しません。重複を避けるためにプレフィックスを追加してインスタンス変数を宣言し、非プリミティブインスタンス変数を初期化します。

たとえば、以下のコードを使用すると、ブール変数を使用して出力行を生成するかどうかを決定できます。

```
// boolean to decide whether to generate an output row
// based on validity of input
private boolean generateRow;
```

- ユーザー定義の静的メソッドおよびインスタンスメソッド

Java トランスフォーメーションの機能を拡張します。[ヘルパ] タブで宣言された Java メソッドでは、出力変数またはローカルで宣言されたインスタンス変数を使用または変更できます。[ヘルパ] タブの Java メソッドからは、入力変数にアクセスできません。

例えば、[ヘルパ] タブの以下のコードを使用して 2 つの整数を追加する関数を宣言します。

```
private int myTXAdd (int num1,int num2)
{
 return num1+num2;
}
```

## Java トランスフォーメーションの Java のプロパティ

特定のトランスフォーメーションイベントに対するトランスフォーメーションの動作を定義する Java コードを記述してコンパイルするには、[Java] ビューのコードエントリタブを使用します。

以下のコードエントリタブがあります。

- インポート
- ヘルパ
- 入力時
- 最後
- 関数
- 最適化インタフェース

Java トランスフォーメーションのクラスコード全体は、[コード全体] タブに表示されます。

## [インポート] タブ

**[インポート]** タブでは、アクティブまたはパッシブな Java トランスフォーメーションの Java パッケージ（サードパーティ製、ビルトイン、またはカスタム）をインポートできます。

Java パッケージをインポートするには、**[インポート]** タブの **[コード]** プロパティの **[Java コード]** ウィンドウで、パッケージをインポートするコードを入力します。

例えば、次のコードを入力すると java.io パッケージをインポートできます。

```
import java.io.*;
```

Java パッケージをインポートするコードをコンパイルするには、**[インポート]** タブの **[コンパイル]** プロパティにある **[コンパイル]** をクリックします。コンパイルの結果は、**[インポート]** タブの **[結果]** ウィンドウに表示されます。

インポートした Java パッケージは、他のコードエントリタブで使うことができます。

## [ヘルパ] タブ

**[ヘルパ]** タブで、アクティブまたはパッシブな Java トランスフォーメーション内の Java トランスフォーメーションクラスのユーザー定義変数およびメソッドを宣言できます。

ユーザー定義変数およびメソッドを宣言するには、**[Java コード]** ウィンドウの **[ヘルパ]** タブの **[コード]** プロパティにコードを入力します。

Java トランスフォーメーションのヘルパコードをコンパイルするには、**[ヘルパ]** タブの **[コンパイル]** プロパティの **[コンパイル]** をクリックします。コンパイルの結果は **[結果]** ウィンドウの **[ヘルパ]** タブに表示されます。

変数およびメソッドを宣言すると、それらを **[インポート]** タブを除く他のすべてのコードエントリタブで使えます。

## [入力時] タブ

**[入力時]** タブでは、入力行を受け取る際のアクティブまたはパッシブ Java トランスフォーメーションの動作を定義します。このタブでは、入出力ポートのデータ、変数、および Java トランスフォーメーション API メソッドにアクセスして使うこともできます。

このタブで定義した Java コードは、入力行ごとに 1 回実行されます。

入力行を受け取る際の Java トランスフォーメーションの動作を定義するには、**[Java コード]** ウィンドウの **[入力時]** タブの **[コード]** プロパティにコードを入力します。

**[入力時]** タブのナビゲータから、以下の変数と API メソッドにアクセスし、それらを定義することができます。

- 入力および出力ポートの変数。入力および出力ポートのデータに変数としてアクセスするには、ポートの名前を変数の名前として使います。たとえば「in\_int」が整数の入力ポートである場合、Java 基本データ型 int で「in\_int」変数として参照することで、このポートのデータにアクセスできます。入力ポートおよび出力ポートを変数として宣言する必要はありません。

入力ポート変数に値は割り当てないでください。**[入力時]** タブの入力変数に値を割り当てると、対応するポートの入力データを現在の行では取得できません。

- インスタンス変数とユーザー定義メソッド。**[ヘルパ]** タブで宣言した任意のインスタンス変数、静的変数、またはユーザー定義メソッドを使います。

たとえば、アクティブ Java トランスフォーメーションに、整数データ型の 2 つの入力ポート（BASE\_SALARY と BONUSSES）、および整数データ型の 1 つの出力ポート（TOTAL\_COMP）があるとします。また、**[ヘルパ]** タブで、2 つの整数を加算して結果を返すユーザー定義メソッド（myTXAdd）を作成

したとします。この場合、**【入力時】** タブで以下の Java コードを使用し、入力ポートの合計値を出力ポートに割り当てて出力行を生成します。

```
TOTAL_COMP = myTXAdd (BASE_SALARY,BONUSES);
generateRow();
```

Java トランスフォーメーションは、入力行を受け取ると 2 つの入力ポート (BASE\_SALARY および BONUSES) の値を加算した値を出力ポート (TOTAL\_COMP) に割り当て、出力行を生成します。

- Java トランスフォーメーション API メソッド。Java トランスフォーメーションによって提供される API メソッドを呼び出すことができます。

Java トランスフォーメーションのコードをコンパイルするには、**【入力時】** タブの **【コンパイル】** プロパティの **【コンパイル】** をクリックします。コンパイルの結果は **【入力時】** タブの **【結果】** ウィンドウに表示されます。

## 【最後】 タブ

**【最後】** タブでは、アクティブまたはパッシブな Java トランスフォーメーションですべての入力データを処理した後のトランスフォーメーションの動作を定義します。このタブでは、アクティブなトランスフォーメーションの出力データの設定や、Java トランスフォーメーション API メソッドの呼び出しも行うことができます。

Java トランスフォーメーションのすべての入力データを処理した後の動作を定義するには、**【最後】** タブの **【コード】** プロパティの **【Java コード】** ウィンドウでコードを入力します。

**【最後】** タブでは、以下の変数および API メソッドにアクセスしてそれらを定義することができます。

- 出力ポート変数。**【ポート】** タブで変数として定義した出力ポートの名前を使用するか、アクティブな Java トランスフォーメーションの出力データを設定できます。
- インスタンス変数とユーザー定義メソッド。**【ヘルパ】** タブで宣言したインスタンス変数またはユーザー定義メソッドを使用します。
- Java トランスフォーメーション API メソッド。Java トランスフォーメーションによって提供される API メソッドを呼び出します。

例えば、以下の Java コードを使用して、データの終わりに達したときにログに情報を書き込みます。

```
logInfo("Number of null rows for partition is: " + partCountNullRows);
```

Java トランスフォーメーションのコードをコンパイルするには、**【最後】** タブの **【コンパイル】** プロパティにある **【コンパイル】** をクリックします。コンパイルの結果は、**【最後】** タブの **【結果】** ウィンドウに表示されます。

## 【関数】 タブ

**【関数】** タブでは、Java トランスフォーメーションの式を呼び出す関数を Java プログラミング言語を使用して定義します。

例えば、入出力ポートの値や Java トランスフォーメーションの変数の値をルックアップする式を呼び出す関数を定義できます。

関数を定義するには、**【関数】** タブの **【コード】** プロパティの **【Java コード】** ウィンドウで手動で定義します。また、**【新しい関数】** をクリックして **【関数の定義】** ダイアログボックスを呼び出せば、簡単に関数を定義することができます。

コードをコンパイルするには、**【関数】** タブの **【コンパイル】** プロパティにある **【コンパイル】** をクリックします。コンパイルの結果は、**【関数】** タブの **【結果】** ウィンドウに表示されます。



## [コード全体] タブ

**[コード全体]** タブでは、Java トランスフォーメーションの Java クラスコード全体を表示してコードをコンパイルできます。ただし、コードを編集することはできません。

**[コード]** プロパティの **[Java コード]** ウィンドウにクラスコード全体が表示されます。

Java トランスフォーメーションのコード全体をコンパイルするには、**[コード全体]** タブの **[コンパイル]** プロパティにある **[コンパイル]** をクリックします。コンパイルの結果は、**[コード全体]** タブの **[結果]** ウィンドウに表示されます。

# Java トランスフォーメーションによるフィルタの最適化

Data Integration Service は、アクティブな Java トランスフォーメーションにフィルタの最適化を適用できます。Java トランスフォーメーションを定義するときは、Java トランスフォーメーションの **[最適化インタフェース]** タブで、フィルタの最適化のコードを追加します。

## Java トランスフォーメーションによる初期選択の最適化

Java トランスフォーメーションに副次作用がない場合は、初期選択の最適化のアクティブまたはパッシブな Java トランスフォーメーションを有効化できます。オプティマイザは、Java トランスフォーメーションを介してフィルタロジックを渡し、必要に応じてフィルタ条件を変更します。

初期選択の最適化のコードスニペットを表示するには、**[最適化インタフェース]** タブのナビゲータで、PredicatePushOptimization を選択します。

### allowPredicatePush

ブール型。初期選択を有効にします。初期選択を有効にするために True の結果とメッセージを返すようにこの関数を変更します。デフォルトは False で、最適化がサポートされていないというメッセージが返されません。

```
public ResultAndMessage allowPredicatePush(boolean ignoreOrderOfOp) {
 // To Enable PredicatePushOptimization, this function should return true
 //return new ResultAndMessage(true, "");
 return new ResultAndMessage(false, "Predicate Push Optimization Is Not Supported");
}
```

### canGenerateOutputFieldEvalError

ブール型。Java トランスフォーメーションがゼロ除算エラーなどの出力フィールドエラーを返すことができるかどうかを示します。Java トランスフォーメーションが出力フィールドエラーを生成しない場合は False を返すようにこの関数を変更します。Java トランスフォーメーションがフィールドエラーを生成できるとき、Data Integration Service では初期選択の最適化は使用できません。

```
public boolean canGenerateOutputFieldEvalError() {
 // If this Java transformation can never generate an output field evaluation error,
 // return false.
 return true;
}
```



## getInputExpr

入力フィールドのどの入力値が出力フィールドを構成しているかを示す Informatica の式を返します。オプティマイザは、Java トランスフォーメーションを介してフィルタロジックをプッシュするために、出力フィールドを構成している入力フィールドを認識する必要があります。

```
public InfaExpression getInputExpr(TransformationField field,
 TransformationDataInterface group) {
 // This should return an Informatica expression for output fields in terms of input fields
 // We will only push predicate that use fields for which input expressions are defined.
 // For example, if you have two input fields in0 and in1 and three output fields out0, out1, out2
 // out0 is the pass-through of in1, out2 is sum of in1 and in2, and out3 is unknown, the code should be:
 //if (field.getName().equals("out0"))
 // return new InfaExpression("in0", instance);
 //else if (field.getName().equals("out1"))
 // return new InfaExpression("in0 + in1", instance);
 //else if (field.getName().equals("out2"))
 // return null;
 return null;
}
```

例えば、マッピングにフィルタ式が含まれている場合は、"out0 > 8"です。out0 は、Java トランスフォーメーションの out0 出力ポートの値です。out0 の値を in0 入力ポート+5 の値として定義できます。オプティマイザは、式"(in0 + 5) > 8"を初期選択の最適化の Java トランスフォーメーション以降にプッシュできます。出力フィールドに入力フィールドの式がない場合は、NULL を返すことができます。オプティマイザでは、入力式がない出力フィールド以降にフィルタ式がプッシュされることはありません。次のコードを含めることができます。

```
if (field.getName().equals("out0"))
 return new InfaExpression("in0 + 5", instance);
else if (field.getName().equals("out2"))
 return null;
```

## inputGroupsPushPredicateTo

フィルタロジックを受け取ることができるグループのリストを返します。Java トランスフォーメーションには入力グループが1つあります。Java トランスフォーメーションのこの関数は変更しないでください。

```
public List<TransformationDataInterface> inputGroupsPushPredicateTo(
 List<TransformationField> fields) {
 // This functions returns a list of input data interfaces to push predicates to.
 // Since JavaTx only has one input data interface, you should not have to modify this function
 AbstractTransformation tx = instance.getTransformation();
 List<DataInterface> dis = tx.getDataInterfaces();
 List<TransformationDataInterface> inputDIs = new ArrayList<TransformationDataInterface>();
 for (DataInterface di : dis){
 TransformationDataInterface tdi = (TransformationDataInterface) di;
 if (tdi.isInput())
 inputDIs.add(tdi);
 }
 if(inputDIs.size() == 1)
 return inputDIs;
 else
 return null;
}
```

## Java トランスフォーメーションによるプッシュイン最適化

副次作用がなく、最適化がマッピング結果に影響しない場合は、最適化にプッシュインでアクティブ Java トランスフォーメーションを有効にできます。

Java トランスフォーメーションで最適化にプッシュインを設定するときは、Java トランスフォーメーションがオプティマイザから受け取るフィルタ条件を格納する方法を定義します。フィルタ条件を調べるコードを追加します。Java トランスフォーメーションがフィルタロジックを吸収できる場合、Java トランスフォーメー

ションは True の条件をオブティマイザに戻します。オブティマイザは、最適化されたマッピングからフィルタトランスフォーメーションを削除します。

Java トランスフォーメーションを設定するときは、最適化中にフィルタ条件をトランスフォーメーションのメタデータとして格納するコードを記述します。実行時にフィルタ条件を取得するコードや、フィルタロジックに従って行を削除するコードも記述します。

Java トランスフォーメーションを定義するときは、Java トランスフォーメーションの【最適化インタフェース】タブで、最適化にプッシュインのコードを追加します。最適化にプッシュインのコードスニペットにアクセスするには、【最適化インタフェース】タブのナビゲータで、FilterPushdownOptimization を選択します。

最適化にプッシュインを有効にするコードスニペットや、オブティマイザからフィルタ条件を取得するコードスニペットが表示されます。最適化を有効にしたり、フィルタロジックをトランスフォーメーションのメタデータとして保存したりするには、コードスニペットを更新します。

### isFilterSupported

最適化にプッシュインを有効にする場合は、True を返します。最適化にプッシュインを無効にする場合は、False を返します。

最適化にプッシュインを有効にするために True を返すようにこの関数を変更します。

```
public ResultAndMessage isFilterSupported() {
 // To enable filter push-into optimization this function should return true
 // return new ResultAndMessage(true, "");
 return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

### pushFilter

オブティマイザからフィルタ条件を受け取ります。

フィルタを調べ、トランスフォーメーションでフィルタロジックが使用可能かどうかを判断するコードを追加します。Java トランスフォーメーションがフィルタを吸収できる場合は、次のメソッドを使用してフィルタ条件をトランスフォーメーションのメタデータとして格納します。

storeMetadata (String key, String data)

このキーは、メタデータの識別子です。任意の文字列をキーとして定義できます。このデータは、実行時に削除する行を指定するために格納するデータです。例えば、このデータは、Java トランスフォーメーションがオブティマイザから受け取るフィルタ条件である可能性があります。

```
public ResultAndMessage pushFilter(InfraExpression condition) {
 // Add code to absorb the filter
 // If filter is successfully absorbed return new ResultAndMessage(true, ""); and the optimizer
 // will remove the filter from the mapping
 // If the filter is not absorbed, return new ResultAndMessage(false, msg);
 return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

## Java トランスフォーメーションの作成

Developer ツールでは、再利用可能または再利用不可能な Java トランスフォーメーションを作成できます。

## 再利用可能な Java トランスフォーメーションの作成

再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。

再利用可能な Java トランスフォーメーションは Developer ツールで作成します。

1. **[Object Explorer]** ビューで、プロジェクトまたはフォルダを選択します。
2. **[ファイル] > [新規] > [トランスフォーメーション]** をクリックします。  
**[新規]** ダイアログボックスが表示されます。
3. Java トランスフォーメーションを選択します。
4. **[次へ]** をクリックします。
5. トランスフォーメーションの名前を入力します。
6. アクティブなトランスフォーメーションを作成する場合は、**[アクティブとして作成]** オプションを選択します。
7. **[完了]** をクリックします。  
トランスフォーメーションがエディタに表示されます。
8. **[ポート]** ビューで、**[新規]** ボタンをクリックして、トランスフォーメーションにポートを追加します。
9. ポートを編集して、名前、データ型、および精度を設定します。  
Java コードスニペットでは、ポート名を変数として使用します。
10. **[Java]** ビューのコードエントリタブで、トランスフォーメーションの Java コードを記述してコンパイルします。
11. **[Java]** ビューの **[関数]** タブで、式を呼び出す関数を定義します。
12. 任意のコードエントリタブで、**[コンパイル]** プロパティの **[結果]** ウィンドウに表示されたエラーメッセージをダブルクリックし、トランスフォーメーションの Java コードのコンパイルエラーを特定して修正します。
13. **[詳細]** ビューで、トランスフォーメーションのプロパティを編集します。

## 再利用不可能な Java トランスフォーメーションの作成

再利用不可能なトランスフォーメーションは、単一のマッピングで使用されます。

Developer ツールで再利用不可能な Java トランスフォーメーションを作成します。

1. マッピングまたはマプレットで、トランスフォーメーションパレットからエディタに Java トランスフォーメーションをドラッグします。
2. **[新しい Java トランスフォーメーション]** ダイアログボックスで、トランスフォーメーションの名前を入力します。
3. アクティブなトランスフォーメーションを作成する場合は、**[アクティブとして作成]** オプションを選択します。
4. **[完了]** をクリックします。  
トランスフォーメーションがエディタに表示されます。
5. **[全般]** タブで、トランスフォーメーションの名前と説明を編集します。
6. **[ポート]** タブで、**[新規]** ボタンをクリックして、トランスフォーメーションにポートを追加します。
7. ポートを編集して、名前、データ型、および精度を設定します。  
Java コードスニペットでは、ポート名を変数として使用します。
8. **[Java]** ビューのコードエントリタブで、トランスフォーメーションの Java コードを記述してコンパイルします。

9. **【Java】** ビューの **【関数】** タブで、式を呼び出す関数を定義します。
10. 任意のコードエントリタブで、**【コンパイル】** プロパティの **【結果】** ウィンドウに表示されたエラーメッセージをダブルクリックし、トランスフォーメーションの Java コードのコンパイルエラーを特定して修正します。
11. **【詳細】** ビューで、トランスフォーメーションのプロパティを編集します。

## Java トランスフォーメーションのコンパイル

の Developer ツールでは、Java コンパイラを使用して Java コードをコンパイルしてトランスフォーメーション用のバイトコードを生成します。

Java コンパイラは、Java コードをコンパイルし、コンパイルの結果を **【結果】** ウィンドウのコードエントリタブの **【コンパイル】** プロパティに表示します。Java コンパイラは、の Developer ツールと一緒に java/bin ディレクトリにインストールされます。

Java トランスフォーメーションのコード全体をコンパイルするには、**【コード全体】** タブの **【コンパイル】** プロパティの **【コンパイル】** をクリックします。

作成した Java トランスフォーメーションには、Java トランスフォーメーションの基本的な機能を定義する Java クラスが含まれています。Java クラスのコード全体には、トランスフォーメーションのテンプレートクラスコードに加えて、コードエントリタブで定義した Java コードが格納されています。

Java トランスフォーメーションをコンパイルすると、の Developer ツールはコードエントリタブのコードをトランスフォーメーションのテンプレートクラスに追加し、トランスフォーメーションのクラスコード全体を生成します。その後、デベロッパツールは Java コンパイラを呼び出してクラスコード全体をコンパイルします。Java コンパイラは、トランスフォーメーションをコンパイルし、トランスフォーメーションのバイトコードを生成します。

コンパイルの結果は **【結果】** ウィンドウに表示されます。コンパイルの結果を使用して、Java コードエラーを特定および検出します。

## Java トランスフォーメーションのトラブルシューティング

すべてのコードエントリタブの **【コンパイル】** プロパティの **【結果】** ウィンドウで、Java コードエラーを確認および修正できます。

Java トランスフォーメーションのエラーは、コードエントリタブのコード内のエラー、または Java トランスフォーメーションクラスのコード全体内のエラーが原因で発生する可能性があります。

Java トランスフォーメーションをトラブルシューティングするには、以下の手順を実行します。

1. Java スニペットコードまたはトランスフォーメーションのクラスコード全体からエラーのソースを検出します。
2. エラーのタイプを特定します。エラーのタイプを特定するには、**【結果】** ウィンドウに表示されるコンパイルの結果、およびエラーの場所を使用します。
3. コードエントリタブで Java コードを修正します。
4. トランスフォーメーションを再度コンパイルします。

## コンパイルエラーのソースの検出

コンパイルエラーのソースを探すには、コードエントリタブまたは **【コード全体】** タブの **【コンパイル】** プロパティの **【結果】** ウィンドウに表示されるコンパイル結果を使用します。

**【結果】** ウィンドウでエラーメッセージをダブルクリックすると、コードエントリタブまたは **【コード全体】** タブの **【Java コード】** ウィンドウで、エラーの原因となったソースコードが強調表示されます。

**【コード全体】** タブでは、エラーを探すことはできますが、Java コードを編集することはできません。**【コード全体】** タブで見つけたエラーを修正するには、適切なコードエントリタブでコードを変更します。トランスフォーメーションのクラスコード全体にユーザーコードを追加したことが原因で発生したエラーを表示する場合などは、**【コード全体】** タブを使用する必要があります。

### コードエントリタブまたは **【コード全体】** タブでのエラーの確認

コードエントリタブまたは **【コード全体】** タブでコンパイルエラーを探すことができます。

1. 任意のコードエントリタブまたは **【コード全体】** タブの **【コンパイル】** プロパティの **【結果】** ウィンドウで、エラーメッセージを右クリックします。

2. **【表示】** > **【スニペット】** または **【表示】** > **[[コード全体] タブ]** をクリックします。

選択したタブでエラーのソースが強調表示されます。

**注:** **【コード全体】** タブでは、エラーを表示することはできますが、修正することはできません。エラーを修正するには、適切なコードエントリタブに移動する必要があります。

## コンパイルエラーの原因の特定

コンパイルエラーは、ユーザーコードのエラーが原因で発生する場合があります。

ユーザーコードのエラーは、クラスの非ユーザーコードでのエラーの原因になる可能性もあります。コンパイルエラーは、Java トランスフォーメーションのユーザーコードおよび非ユーザーコードで発生します。

### ユーザーコードのエラー

エラーは、コードエントリタブのユーザーコードで発生する可能性があります。ユーザーコードのエラーには、標準 Java 構文および言語のエラーが含まれます。

ユーザーコードのエラーは、Developer ツールがコードエントリタブのユーザーコードをクラスコード全体に追加した場合にも発生することがあります。

たとえば、Java トランスフォーメーションには整数データ型の `int1` という名前の入力ポートがあるとします。クラスのコード全体は、以下のコードで入力ポートの変数を宣言します。

```
int int1;
```

しかし、**【入力時】** タブで同じ変数名を使用すると、Java コンパイラは変数の再宣言としてエラーを発行します。エラーを修正するには、**【入力時】** タブで変数の名前を変更します。

### 非ユーザーコードのエラー

コードエントリタブのユーザーコードは、非ユーザーコードでのエラーの原因になる場合もあります。

たとえば、Java トランスフォーメーションには整数データ型の `int1` および `out1` という名前の入力ポートと出力ポートがあるとします。ここで、以下のコードを **【入力時】** コードエントリタブに書き込み、入力ポート `int1` の `interest` を計算して出力ポート `out1` に割り当てます。

```
int interest;
interest = CallInterest(int1); // calculate interest
```

```
out1 = int1 + interest;
}
```

トランスフォーメーションをコンパイルすると、Developer ツールは **【入力時】** コードエントリタブのコードを、トランスフォーメーションのクラスコード全体に追加します。Java コンパイラが Java コードをコンパイルする際に中括弧が一致していないと、クラスコード全体のメソッドは不完全なまま終了し、Java コンパイラによってエラーが発行されます。

## Struct データへの変換例

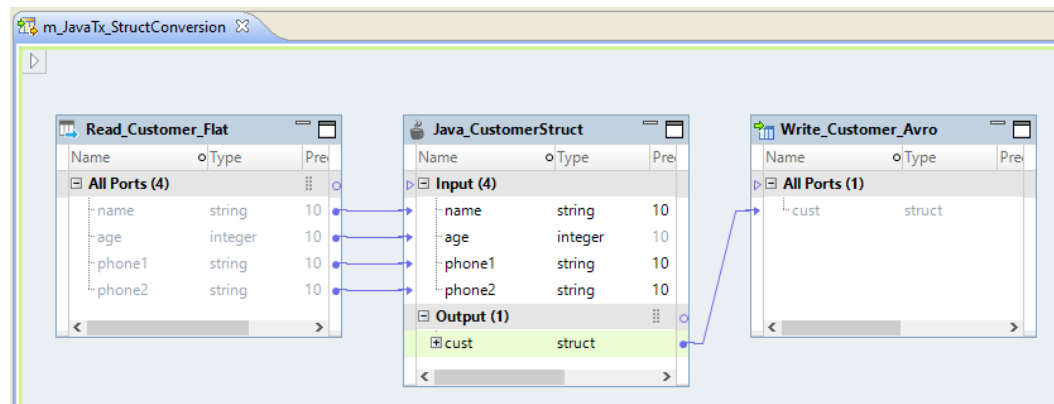
あなたの組織は、フラットファイル内にある大量の顧客データを struct データに変換して Avro ファイルに書き込む必要があります。入力ファイルには、名前、年齢、電話番号といった顧客の詳細情報が含まれています。入力ファイルで顧客名が NULL である場合は、出力ファイルに顧客の詳細を追加しないことにします。

Java トランスフォーメーションでマッピングを開発して、トランスフォーメーション機能を定義できます。Hadoop 環境では、Spark エンジンでマッピングを実行してデータを変換し、struct データを Avro ファイルに書き込みます。

マッピングを作成し、次のトランスフォーメーションを設定します。

- フラットファイルソースから顧客情報を読み取る読み取りトランスフォーメーション
- フラットデータから struct データへの変換と矛盾のあるデータの削除を実行するアクティブなトランスフォーメーションとしての Java トランスフォーメーション
- struct データを Avro ファイルに書き込む書き込みトランスフォーメーション

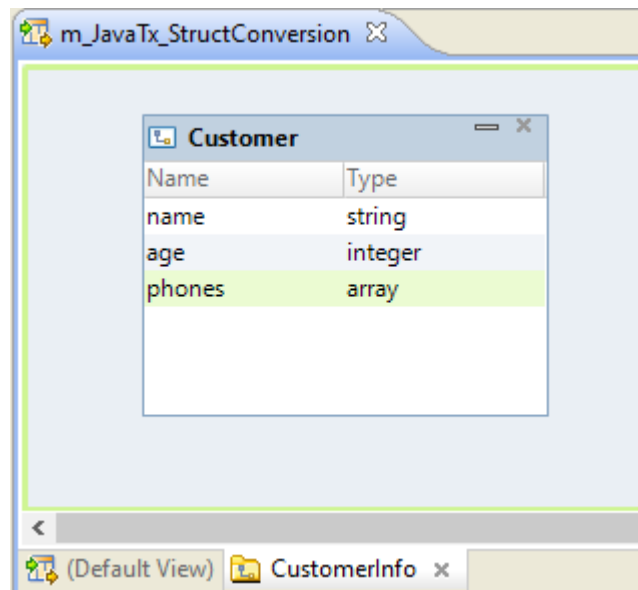
次の図は、読み込みトランスフォーメーション、Java トランスフォーメーション、および書き込みトランスフォーメーションのマッピングを示しています。



マッピングエディタの型定義ライブラリタブで、複合データ型定義 Customer を作成します。複合データ型定義は、struct データのスキーマを表します。型定義ライブラリの名前 CustomerInfo に変更します。次の要素を複合データ型定義に追加します。

- string 型の name
- integer 型の age
- string 要素からなる array 型の phones

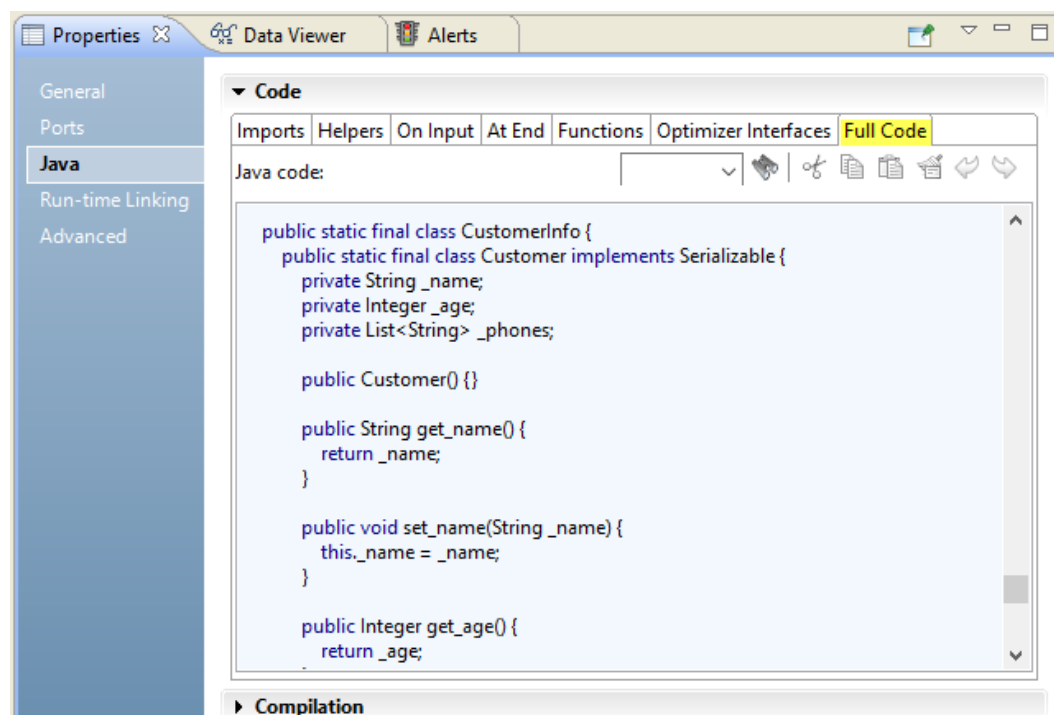
次の図は、型定義ライブラリ内の複合データ型定義を示しています。



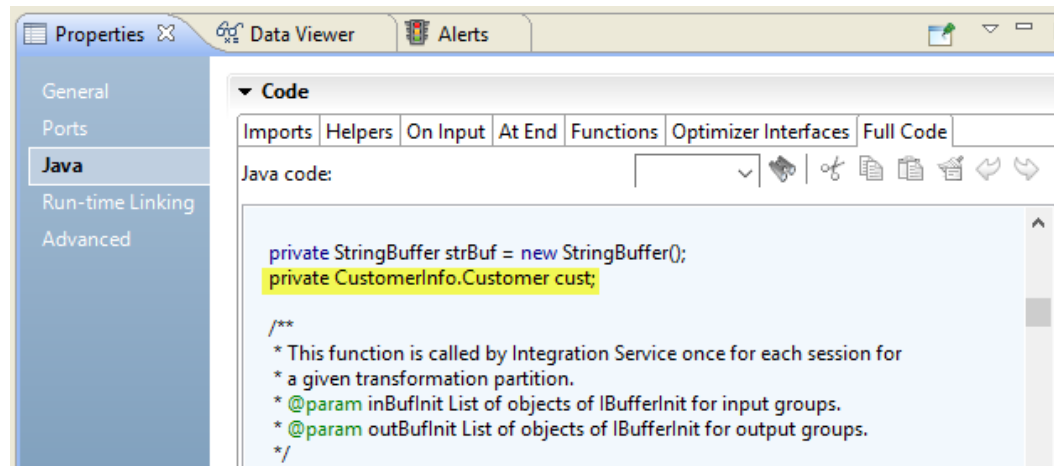
Java トランスフォーメーションで、struct 出力ポートを追加し、ポートの型定義を指定して、作成した複合データ型定義を参照します。Java トランスフォーメーションは、クラス Customer と、メンバーフィールドの読み取りと設定を行うセッタとゲッタを生成します。このクラスには、次のメンバーフィールドがあります。

- \_name
- \_age
- \_phones

次の図は、[Java] ビューの [コード全体] タブに表示されている、struct ポート用に作成されたクラスを示しています。

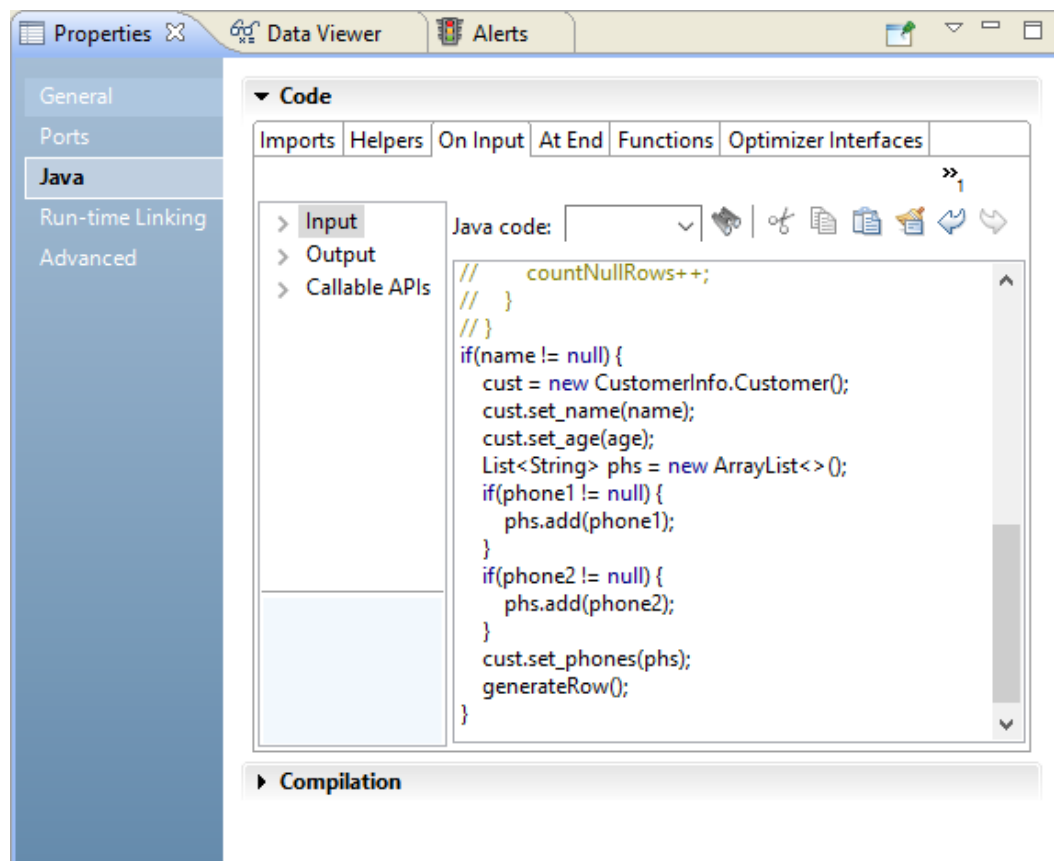


struct ポートの Java データ型は、型定義ライブラリおよび複合データ型定義の名前を使用します。次の図は、生成されたコードの cust フィールドの Java データ型名 CustomerInfo.Customer を示しています。



Java トランスフォーメーションの **[Java]** ビューで、トランスフォーメーションに必要なサードパーティ、ビルトイン、またはカスタムの Java パッケージをインポートします。フラットデータから struct データへの変換と顧客名が NULL のときはその顧客の行の削除を実行する Java コードを記述してコンパイルします。

次の図は、**[入力時]** タブのコードを示しています。



マッピングを検証し、Spark エンジンでマッピングを実行して、変換されたデータを Avro ファイル出力に書き込みます。



# 非ネイティブ環境での Java トランスフォーメーション

非ネイティブ環境での Java トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

## Blaze エンジンでの Java トランスフォーメーション

Java トランスフォーメーション内で外部.jar ファイルを使用するには、次の手順を実行します。

1. 外部.jar ファイルを、データ統合サービスマシン内の<Informatic installation directory>/services/shared/jars にある Informatica のインストールディレクトリにコピーします。次に、データ統合サービスをリサイクルします。
2. Developer tool をホストするマシンが、Java トランスフォーメーションを含むマッピングを開発して実行する場所です。
  - a. 外部.jar ファイルを、ローカルマシンのディレクトリにコピーします。
  - b. Java トランスフォーメーションを、そのローカル.jar ファイルをポイントするインポート文を含めるように編集します。
  - c. Java トランスフォーメーションのクラスパスを更新します。
  - d. トランスフォーメーションをコンパイルします。

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### パーティション化

- 次の制限がトランスフォーメーション範囲プロパティに適用されます。
  - トランスフォーメーション範囲の値トランザクションは有効ではありません。
  - パーティションキーの入力ポートを有効にした場合、トランスフォーメーション範囲は [すべての入力] に設定される必要があります。
  - ステートレスは、トランスフォーメーション範囲が行の場合に有効にされる必要があります。

## Spark エンジンでの Java トランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### 一般的な制限

Java トランスフォーメーションのサポートには、Spark エンジンでは次の制限があります。

- トランスフォーメーションロジックを Hadoop にプッシュするときに、トランスフォーメーション内の Java コードは、出力を標準出力に書き込むことはできません。Java コードは、標準エラーに出力を書き込むことはできます。このエラーはログファイルに表示されます。
- 日時の値の場合、Spark エンジンで最大マイクロ秒の精度がサポートされます。日時の値にナノ秒が含まれている場合、後続の桁は切り詰められます。

## パーティション化

Java トランスフォーメーションには、パーティション化と一緒に使用される場合に次の制限があります。

- パーティション化が可能なプロパティが、Java トランスフォーメーションで有効である必要があります。トランスフォーメーションは 1 つのパーティション内では実行できません。
- 次の制限がトランスフォーメーション範囲プロパティに適用されます。
  - トランスフォーメーション範囲の値トランザクションは有効ではありません。
  - パーティションキーの入力ポートを有効にした場合、トランスフォーメーション範囲は [すべての入力] に設定される必要があります。
  - ステートレスは、トランスフォーメーション範囲が行の場合に有効にされる必要があります。

## マッピング検証

マッピング検証は、次の場合に失敗します。

- Java トランスフォーメーション内の式から未接続のルックアップトランスフォーメーションを参照した。
- 複合データ型のポートをパーティションまたはソートキーとして選択した。
- 日時でナノ秒処理を有効にし、Java トランスフォーメーションに、日時型の要素を持つ複合データ型のポートが含まれる。例えば、array<data/time>型のポートは、日時でナノ秒処理を有効にした場合に有効ではありません。

マッピングは、次の場合に失敗します。

- Java トランスフォーメーションに小数位ポートまたは decimal データ型の要素を持つ複合ポートが含まれているときに、Java トランスフォーメーションとマッピングが異なる精度モードを使用する場合。  
マッピングで高精度が有効にされている場合でも、マッピングに decimal データ型の要素を持つ複合ポートが含まれているときや、マッピングがストリーミングマッピングであるときなど、マッピングがデータを低精度モードで処理することがあります。Java トランスフォーメーションとマッピングの両方で高精度が有効にされていても、マッピングが低精度モードでデータを処理した場合、マッピングは失敗します。

## 外部.jar ファイルの使用

Java トランスフォーメーション内で外部.jar ファイルを使用するには、次の手順を実行します。

1. 外部.jar ファイルを、データ統合サービスマシン内の次の場所にある Informatica のインストールディレクトリにコピーします。  
<Informatic installation directory>/services/shared/jars
2. データ統合サービスをリサイクルします。
3. Developer tool をホストするマシンが、Java トランスフォーメーションを含むマッピングを開発して実行する場所です。
  - a. 外部.jar ファイルを、ローカルマシンのディレクトリにコピーします。
  - b. Java トランスフォーメーションを、そのローカル.jar ファイルをポイントするインポート文を含めるように編集します。
  - c. Java トランスフォーメーションのクラスパスを更新します。
  - d. トランスフォーメーションをコンパイルします。

## ストリーミングマッピングでの Java トランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

データ統合サービスは次のプロパティを無視します。

- パーティション化可能。データ統合サービスはプロパティを無視して、複数のスレッドを使用してトランスフォーメーションを処理できます。
- ステートレス。データ統合サービスはプロパティを無視して、入力データの行順序を維持します。

## 第 20 章

# Java トランスフォーメーション API のリファレンス

この章では、以下の項目について説明します。

- [Java トランスフォーメーション API メソッドの概要, 328 ページ](#)
- [defineJExpression, 329 ページ](#)
- [failSession, 330 ページ](#)
- [generateRow, 330 ページ](#)
- [getInRowType, 331 ページ](#)
- [getMetadata, 332 ページ](#)
- [incrementErrorCount, 332 ページ](#)
- [invokeJExpression, 333 ページ](#)
- [isNull, 334 ページ](#)
- [logError, 334 ページ](#)
- [logInfo, 335 ページ](#)
- [resetNotification, 335 ページ](#)
- [setNull, 336 ページ](#)
- [storeMetadata, 337 ページ](#)

## Java トランスフォーメーション API メソッドの概要

エディタの **【Java】** ビューのコードエントリタブで、API メソッドを Java コードに追加してトランスフォーメーションの動作を定義することができます。

API メソッドをコードに追加するには、コードエントリタブのナビゲータで **【呼び出し可能な API】** リストを展開し、コードに追加するメソッドの名前をダブルクリックします。

また、ナビゲータから Java コードスニペットにメソッドをドラッグするか、Java コードスニペットに API メソッドを手動で入力することもできます。

Java トランスフォーメーションの Java コードに追加できる API メソッドは次のとおりです。

`defineJExpression`

Java 式を定義します。

failSession

エラーメッセージ付きの例外をスローし、マッピングを失敗させます。

generateRow

アクティブな Java トランスフォーメーションの出力行を生成します。

getInRowType

トランスフォーメーションの現在の行の入力タイプを返します。

incrementErrorCount

マッピングのエラーカウントを増やします。

invokeJExpression

defineJExpression メソッドを使用して定義した Java 式を呼び出します。

isNull

入力カラムの NULL 値の有無を確認します。

logError

ログにエラーメッセージを書き込みます。

logInfo

ログに情報メッセージを書き込みます。

resetNotification

Data Integration Service マシンがリスタートモードで実行されている場合に、マッピングの実行後に Java コードで使用する変数をリセットします。

setNull

アクティブまたはパッシブ Java トランスフォーメーションの出力カラムの値を NULL に設定します。

## defineJExpression

式（式の文字列および入力パラメータを含む）を定義します。defineJExpression メソッドの引数には、式の構文を定義する入力パラメータと文字列値を含む JExprParamMetadata オブジェクトの配列が含まれています。

以下の構文を使用します。

```
defineJExpression(
 String expression,
 Object[] paramMetadataArray
);
```

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              | タイプ | データ型         | 説明                                         |
|--------------------|-----|--------------|--------------------------------------------|
| 式                  | 入力  | String       | 式を表す文字列。                                   |
| paramMetadataArray | 入力  | オブジェクト<br>[] | 式の入力パラメータを含む JExprParamMetadata オブジェクトの配列。 |

defineJExpression メソッドは、**【インポート】** タブと **【関数】** タブを除く任意のコードエントリタブで Java コードに追加することができます。

defineJExpression メソッドを使用するには、式の入力パラメータを表す JExprParamMetadata オブジェクトの配列をインスタンス化する必要があります。パラメータのメタデータ値を設定し、その配列をパラメータとして defineJExpression メソッドに渡します。

例えば、以下の Java コードでは、2 つの文字列の値をルックアップする式を作成します。

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
defineJExpression(":lkp.mylookup(x1,x2)",params);
```

**注:** 式に渡す一連のパラメータには、先頭に文字 x を付けて番号を示す必要があります。例えば、3 つのパラメータを式に渡す場合は、各パラメータに x1、x2、および x3 という名前を付けます。

## failSession

エラーメッセージ付きの例外をスローし、マッピングを失敗させます。

以下の構文を使用します。

```
failSession(String errorMessage);
```

次の表に、パラメータを示します。

| パラメータ        | パラメータの<br>タイプ | データ型   | 説明            |
|--------------|---------------|--------|---------------|
| errorMessage | Input         | String | エラーメッセージの文字列。 |

failSession メソッドを使用して、マッピングを終了します。コードエントリタブの try/catch ブロック内では、failSession メソッドを使用しないでください。

**【インポート】** および **【関数】** タブを除くコードエントリタブでは、Java コードに failSession メソッドを追加できます。

以下の Java コードは、input1 入力ポートに NULL 値が存在するかどうかについてテストする方法と、input1 が NULL の場合はマッピングに失敗することを示しています。

```
if(isNull("input1")) {
 failSession("Cannot process a null value for port input1.");
}
```

## generateRow

アクティブな Java トランスフォーメーションの出力行を生成します。

以下の構文を使用します。

```
generateRow();
```

generateRow メソッドを呼び出すと、Java トランスフォーメーションは出力ポート変数の現在の値を使用して出力行を生成します。1 つの入力行に対応する複数の行を生成するには、各入力行に対して generateRow

メソッドを複数呼び出すことができます。アクティブな Java トランスフォーメーションで generateRow メソッドを使用しない場合、トランスフォーメーションは出力行を生成しません。

**【インポート】** および **【関数】** タブ以外のすべてのコードエントリタブで、Java コードに generateRow メソッドを追加できます。

generateRow メソッドを呼び出すことができるのは、アクティブトランスフォーメーションのみです。パッシブトランスフォーメーションで generateRow メソッドを呼び出すと、Data Integration Service でエラーが発生します。

以下の Java コードを使用すると、1 つの出力行が生成され、出力ポートの値が変更され、別の出力行が生成されます。

```
// Generate multiple rows.
if(!isNull("input1") && !isNull("input2"))
{
 output1 = input1 + input2;
 output2 = input1 - input2;
}
generateRow();
// Generate another row with modified values.
output1 = output1 * 2;
output2 = output2 * 2;
generateRow();
```

## getInRowType

トランスフォーメーションの現在の行の入力タイプを返します。このメソッドは、挿入、更新、削除、またはリジェクトの値を返します。

以下の構文を使用します。

```
rowType getInRowType();
```

次の表に、パラメータを示します。

| パラメータ   | パラメータのタイプ | データ型   | 説明                                                                                                                                               |
|---------|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| rowType | アウトプット    | String | アップデートストラテジのタイプを返します。以下のいずれかの値となります。 <ul style="list-style-type: none"><li>- DELETE</li><li>- INSERT</li><li>- REJECT</li><li>- UPDATE</li></ul> |

**【入力時】** コードエントリタブで、Java コードに getInRowType メソッドを追加できます。

getInRowType メソッドは、アップデートストラテジを設定するように設定されたアクティブなトランスフォーメーションで使用できます。アップデートストラテジを設定されていないアクティブなトランスフォーメーションでこのメソッドを呼び出すと、Data Integration Service でエラーが発生します。

# getMetadata

Java トランスフォーメーションのメタデータを実行時に取得します。getMetadata メソッドは、pushFilter 関数でオプティマイザが Java トランスフォーメーションに渡すフィルタ条件など、storeMetadata メソッドを使用して保存するメタデータを取得します。

以下の構文を使用します。

```
getMetadata (String key);
```

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 | パラメータのタイプ | データ型   | 説明                                                      |
|-------|-----------|--------|---------------------------------------------------------|
| key   | Input     | String | メタデータを特定します。getMetadata メソッドでは、取得するメタデータを決定するキーが使用されます。 |

getMetadata メソッドは、次のコードエントリタブで Java コードに追加できます。

- ヘルパ
- 入力時
- 最後
- 最適化インタフェース
- 関数

最適化にプッシュインのフィルタ条件を取得するように、getMetadata メソッドを設定することができます。getMetadata メソッドは、格納する各フィルタ条件をオプティマイザから取得できます。

```
// Retrieve a filter condition
String mydata = getMetadata ("FilterKey");
```

# incrementErrorCount

エラーカウントを増やします。エラー数がエラーしきい値に達すると、マッピングは失敗します。

以下の構文を使用します。

```
incrementErrorCount(int nErrors);
```

次の表に、パラメータを示します。

| パラメータ   | パラメータのタイプ | データ型    | 説明           |
|---------|-----------|---------|--------------|
| nErrors | 入力        | Integer | エラーカウントの増分数。 |

incrementErrorCount メソッドは、**【インポート】** タブと **【関数】** タブを除く任意のコードエントリタブで Java コードに追加することができます。



以下の Java コードでは、トランスフォーメーションの入力ポートが NULL 値の場合にエラーカウントが増加します。

```
// Check if input employee id and name is null.
if (isNull ("EMP_ID_INP") || isNull ("EMP_NAME_INP"))
{
 incrementErrorCount(1);
 // if input employee id and/or name is null, don't generate a output row for this input row
 generateRow = false;
}
```

## invokeJExpression

式を呼び出し、式の値を返します。

以下の構文を使用します。

```
(datatype)invokeJExpression(
 String expression,
 Object[] paramMetadataArray);
```

invokeJExpression メソッドの入力パラメータは、式および式の入力パラメータを含むオブジェクトの配列を表す文字列値です。

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              | パラメータ<br>のタイプ | データ型         | 説明                     |
|--------------------|---------------|--------------|------------------------|
| 式                  | 入力            | String       | 式を表す文字列。               |
| paramMetadataArray | 入力            | オブジェクト<br>[] | 式の入力パラメータを含むオブジェクトの配列。 |

invokeJExpression メソッドは、**【インポート】** タブと **【関数】** タブを除く任意のコードエントリタブで Java コードに追加することができます。

invokeJExpression メソッドを使用する場合は、以下のルールおよびガイドラインに従います。

- 戻りデータ型。invokeJExpression メソッドの戻りデータ型はオブジェクトです。関数の戻り値は、適切なデータ型でキャストする必要があります。  
Integer、Double、String、および byte [] のデータ型で値を返すことができます。
- 行タイプ。invokeJExpression メソッドの戻り値の行タイプは INSERT です。  
戻り値に異なる行タイプを使用するには、高度なインタフェースを使用します。
- NULL 値。パラメータとして NULL 値を渡した場合、または invokeJExpression メソッドの戻り値が NULL の場合、この値は NULL インジケータとして処理されます。  
例えば、式の戻り値が NULL で戻りデータ型が String の場合、NULL 値の文字列が返されます。
- Date データ型。Date データ型の入力パラメータは、String データ型に変換する必要があります。  
式中の文字列を Date データ型として使用するには、to\_date()関数を使用して、文字列を Date データ型に変換します。  
また、Date データ型を String データ型として返す式の戻り型をキャストする必要があります。

以下の例は、文字列「John」と「Smith」を連結し、文字列「John Smith」を返します。

```
(String)invokeJExpression("concat(x1,x2)", new Object [] { "John ", "Smith" });
```

**注:** 式に渡す一連のパラメータには、先頭に文字 x を付けて番号を示す必要があります。例えば、3 つのパラメータを式に渡す場合は、各パラメータに x1、x2、および x3 という名前を付けます。

## isNull

入力カラムの値を調べ、NULL 値の有無を確認します。

以下の構文を使用します。

```
Boolean isNull(String strColName);
```

次の表に、パラメータを示します。

| パラメータ      | パラメータの<br>タイプ | データ型   | 説明        |
|------------|---------------|--------|-----------|
| strColName | Input         | String | 入力カラムの名前。 |

isNull メソッドは、**【入力時】** コードエントリタブで Java コードに追加できます。

以下の Java コードは、SALARY 入力カラムの値が NULL であるかどうかを確認してから totalSalaries インスタンス変数に追加する方法を示しています。

```
// if value of SALARY is not null
if (!isNull("SALARY")) {
 // add to totalSalaries
 TOTAL_SALARIES += SALARY;
}
```

以下の Java コードを使用しても同じ結果が得られます。

```
// if value of SALARY is not null
String strColName = "SALARY";
if (!isNull(strColName)) {
 // add to totalSalaries
 TOTAL_SALARIES += SALARY;
}
```

## logError

ログにエラーメッセージを書き込みます。

以下の構文を使用します。

```
logError(String msg);
```

次の表に、パラメータを示します。

| パラメータ | パラメータの<br>タイプ | データ型   | 説明            |
|-------|---------------|--------|---------------|
| msg   | Input         | String | エラーメッセージの文字列。 |

logError メソッドは、【インポート】 タブと 【関数】 タブを除く任意のコードエントリタブで Java コードに追加することができます。

以下の Java コードでは、入力ポートが NULL の場合にエラーがログに記録されます。

```
// check BASE_SALARY
if (isNull("BASE_SALARY")) {
 logError("Cannot process a null salary field.");
}
```

このコードを実行すると、以下のメッセージがログに出力されます。

[JTX\_1013] [ERROR] Cannot process a null salary field.

## logInfo

ログに情報メッセージを書き込みます。

以下の構文を使用します。

```
logInfo(String msg);
```

次の表に、パラメータを示します。

| パラメータ | パラメータの<br>タイプ | データ型   | 説明           |
|-------|---------------|--------|--------------|
| msg   | Input         | String | 情報メッセージの文字列。 |

【インポート】 および 【関数】 タブを除くすべてのコードエントリタブで、Java コードに logInfo メソッドを追加できます。

以下の Java コードは、Java トランスフォーメーションがメッセージしきい値である 1,000 行を処理した後に、ログにメッセージを書き込む方法を示しています。

```
if (numRowsProcessed == messageThreshold) {
 logInfo("Processed " + messageThreshold + " rows.");
}
```

## resetNotification

Data Integration Service マシンがリスタートモードで実行されている場合に、マッピングの実行後に Java コードで使用する変数をリセットします。

リスタートモードでは、Data Integration Service の初期化は解除されませんが、Data Integration Service が要求後にリセットされて次の要求を処理できるようになります。

Java トランスフォーメーションの場合は、resetNotification メソッドを使用すると、マッピングの実行後に Java コードの変数がリセットされます。

以下の構文を使用します。

```
public int resetNotification(IGroup group) {
 return EStatus.value;
}
```

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 | パラメータのタイプ | データ型          | 説明                                                                                                                                            |
|-------|-----------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| int   | 出力        | EStatus.value | 戻り値。value は次の値のいずれかになります。 <ul style="list-style-type: none"><li>- SUCCESS。成功です。</li><li>- FAILURE。失敗です。</li><li>- NOIMPL。実装されていません。</li></ul> |
| group | 入力        | IGroup        | 入力グループ。                                                                                                                                       |

resetNotification メソッドは、**ヘルパ** タブのコードエントリタブで Java コードに追加できます。

resetNotification メソッドは「呼び出し可能な API」リストには表示されません。

例えば、Java コードで out5\_static という名前の静的変数を宣言し、その変数を 1 に初期化とします。以下の Java コードでは、次のマッピングの実行後に out5\_static 変数が 1 にリセットされます。

```
public int resetNotification(IGroup group) {
 out5_static=1;
 return EStatus.SUCCESS;
}
```

このメソッドは必須ではありません。ただし、Data Integration Service をリスタートモードで実行している場合、resetNotification メソッドが実装されていない Java トランスフォーメーションがマッピングに含まれていると、JSDK\_42075 警告メッセージがログに出力されます。

## setNull

アクティブまたはパッシブな Java トランスフォーメーションの出力カラムの値を NULL に設定します。

以下の構文を使用します。

```
setNull(String strColName);
```

次の表に、パラメータを示します。

| パラメータ      | パラメータのタイプ | データ型   | 説明        |
|------------|-----------|--------|-----------|
| strColName | Input     | String | 出力カラムの名前。 |

setNull メソッドは、アクティブまたはパッシブな Java トランスフォーメーションの出力カラムの値を NULL に設定します。出力カラムを NULL に設定したら、出力行を生成するまで値は変更できません。

**インポート** および **関数** タブを除くすべてのコードエントリタブで、Java コードに setNull メソッドを追加できます。

以下の Java コードは、入力カラムの値をチェックし、出力カラムの対応する値を NULL に設定する方法を示しています。

```
// check value of Q3RESULTS input column
if(isNull("Q3RESULTS")) {
 // set the value of output column to null
 setNull("RESULTS");
}
```

また、以下の Java コードを使用して同じ結果を得ることもできます。

```
// check value of Q3RESULTS input column
String strColName = "Q3RESULTS";
if(isNull(strColName)) {
 // set the value of output column to null
 setNull(strColName);
}
```

## storeMetadata

実行時に getMetadata メソッドを使用して取得できる Java トランスフォーメーションのメタデータを格納します。

以下の構文を使用します。

```
storeMetadata (String key String data);
```

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 | パラメータ<br>のタイプ | データ型   | 説明                                                                      |
|-------|---------------|--------|-------------------------------------------------------------------------|
| key   | Input         | String | メタデータを特定します。 storeMetadata メソッドでは、メタデータを特定するキーが必要です。 キーを任意の文字列として定義します。 |
| data  | Input         | String | Java トランスフォーメーションのメタデータとして格納するデータ。                                      |

storeMetadata メソッドは、次のコードエントリタブで Java コードに追加できます。

- ヘルパ
- 入力時
- 最後
- 最適化インタフェース
- 関数

最適化にプッシュインのフィルタ条件を受け入れるように、アクティブなトランスフォーメーションで storeMetadata メソッドを設定することができます。 storeMetadata メソッドは、オプティマイザがマッピングから Java トランスフォーメーションにプッシュするフィルタ条件を格納します。

```
// Store a filter condition
storeMetadata ("FilterKey", condition);
```

## 第 21 章

# Java 式

この章では、以下の項目について説明します。

- [Java 式の概要, 338 ページ](#)
- [\[関数の定義\] ダイアログボックスを使用した式の定義, 339 ページ](#)
- [単純なインタフェースに関する作業, 341 ページ](#)
- [高度なインタフェースに関する作業, 343 ページ](#)
- [JExpression クラス API リファレンス, 347 ページ](#)

## Java 式の概要

Java トランスフォーメーションの式を、Java プログラミング言語を使用して呼び出すことができます。

式を使用すると、Java トランスフォーメーションの機能が拡張されます。たとえば、Java トランスフォーメーションの式を呼び出して、入出力ポートの値、あるいは Java トランスフォーメーション変数の値をルックアップできます。

Java トランスフォーメーションの式を呼び出すには、Java コードを生成するか、または Java トランスフォーメーション API メソッドを使用します。式を呼び出して、適切なコードエントリタブでの式の結果を使用します。式を呼び出す Java コードを生成することも、API メソッドを使用して式を呼び出す Java コードを記述することもできます。

以下の表に、Java トランスフォーメーションの式の作成および呼び出しに使用できる方法を示します。

| メソッド                     | 説明                                                                                             |
|--------------------------|------------------------------------------------------------------------------------------------|
| <b>[関数の定義]</b> ダイアログボックス | 式を呼び出す関数を作成し、式のコードを生成できます。                                                                     |
| 単純なインタフェース               | 式を呼び出す 1 つのメソッドを呼び出し、その式の結果を取得できます。                                                            |
| 高度なインタフェース               | 式を定義し、式を呼び出して、その式の結果を使用できます。<br>オブジェクト指向プログラミングについての知識があり、式の呼び出しをさらに制御したい場合は、高度なインタフェースを使用します。 |

## 式の関数タイプ

Java トランスフォーメーションの式は、**【関数の定義】** ダイアログボックスを使用するか、または単純/高度なインタフェースを使用して作成できます。

入出力ポート変数、あるいは Java コード内の変数を入力パラメータとして使用する式を入力できます。

**【関数の定義】** ダイアログボックスを使用する場合、Java トランスフォーメーションで使用する前に式を検証できます。

Java トランスフォーメーションでは、以下のタイプの式の関数を呼び出すことができます。

| 式の関数タイプ          | 説明                                            |
|------------------|-----------------------------------------------|
| トランスフォーメーション言語関数 | 一般的な式を扱うように設計された、SQL に似た関数です。                 |
| ユーザー定義関数         | トランスフォーメーション言語関数に基づいて Developer ツールで作成する関数です。 |
| カスタム関数           | カスタム関数 API を使用して作成する関数です。                     |

また、未接続のトランスフォーメーションおよびビルトイン変数も式で使用できます。例えば、未接続のルックアップトランスフォーメーションを式で使用できます。

## 【関数の定義】 ダイアログボックスを使用した式の定義

Java 式を定義する場合、関数を設定し、式を作成し、式を呼び出すコードを生成します。

関数を定義して式を作成するには、**【関数の定義】** ダイアログボックスを使用します。

式関数を作成して Java トランスフォーメーションで式を使用するには、以下の高度なタスクを実行します。

1. 式を呼び出す関数を設定します。関数名、説明、およびパラメータの設定を含みます。関数パラメータは、式を作成する場合に使用します。
2. 式の構文を作成し、式を検証します。
3. 式を呼び出す Java コードを生成します。

生成されたコードが、**【関数】** コードエントリタブに表示されます。

Java コードを生成したら、生成した関数を適切なコードエントリタブで呼び出し、単純なインタフェースと高度なインタフェースのどちらを使用するかに基づいて、式を呼び出すか JExpression オブジェクトを取得します。

**注:** 式の作成時に式を検証する場合、**【関数の定義】** ダイアログボックスを使用する必要があります。

## 手順 1. 関数の設定

式を呼び出す Java 関数の関数名、説明、および入力パラメータを設定します。

関数を設定する場合、以下のルールおよびガイドラインを使用します。

- トランスフォーメーション内に既に存在する Java 関数、または Java の予約語と名前が重複しない、ユニークな関数名を使用します。
- パラメータ名、Java データ型、精度、および位取りを設定する必要があります。入力パラメータは、トランスフォーメーションで Java コードの関数を呼び出す際に渡す値です。
- 式に Date データ型を渡すには、入力パラメータの String データ型を使用します。

Date データ型を返す式の場合、単純なインタフェースでは戻り値を String データ型として、高度なインタフェースでは String データ型または Long データ型として使用できます。

## 手順 2. 式の作成と検証

式を作成する場合、設定したパラメータを関数で使用します。

式では、トランスフォーメーション言語の関数、カスタム関数、または他のユーザー定義関数も使用できます。式の作成および検証は、**【関数の定義】** ダイアログボックスで実行できます。

## 手順 3. 式の Java コードの生成

関数と関数パラメータの定義、および式の定義と検証を完了したら、式を呼び出す Java コードを生成できます。

Developer は、生成された Java コードを **【関数】** コードエントリタブに格納します。生成した Java コードを使用して、コードエントリタブの式を呼び出す関数を呼び出します。単純な Java コード、または高度な Java コードを生成できます。

式を呼び出す Java コードを生成した後では、その式の編集および再検証を実行できません。コードを生成した後で式を変更する場合、式を再作成する必要があります。

## 【関数の定義】 ダイアログボックスを使用した式の作成と Java コードの生成

式を呼び出す関数を作成するには、**【関数の定義】** ダイアログボックスを使用します。

式を呼び出す関数を作成するには、以下の手順を実行します。

1. Developer で、Java トランスフォーメーションを開きます。または、新規の Java トランスフォーメーションを作成します。
2. **【Java コード】** タブで、**【新しい関数】** をクリックします。  
**【関数の定義】** ダイアログボックスが表示されます。
3. 関数名を入力します。
4. 必要に応じて、式の説明を入力します。  
最大で 2,000 文字まで入力できます。
5. 関数の引数を作成します。  
引数を作成する場合、引数の名前、データ型、精度、位取りを設定します。
6. **【式】** タブで、作成した引数を使用して式を作成します。
7. 式を検証するには、**【検証】** をクリックします。



8. 必要に応じて、**【式】** ボックスに式を入力します。その後、**【検証】** をクリックして式を検証します。
  9. 高度なインタフェースを使用して Java コードを生成するには、**【詳細コードの生成】** オプションを選択します。次に、**【生成】** をクリックします。
- Developer で、**【関数】** コードエントリタブの式を呼び出す関数が生成されます。

## Java 式のテンプレート

式の単純または高度な Java コードを使用すると、式の Java コードを生成できます。

式の Java コードは、式のテンプレートに基づいて生成されます。

以下の例は、単純な Java コード用に生成された Java 式のテンプレートを示しています。

```
Object function_name (Java datatype x1[,
 Java datatype x2 ...])
 throws SDK Exception
{
 return (Object)invokeJExpression(String expression,
 new Object [] { x1[, x2, ...]});
}
```

以下の例は、高度なインタフェースを使用して生成された Java 式のテンプレートを示しています。

```
JExpression function_name () throws SDKException
{
 JExprParamMetadata params[] = new JExprParamMetadata[number of parameters];
 params[0] = new JExprParamMetadata (
 EDataType.STRING, // data type
 20, // precision
 0 // scale
);
 ...
 params[number of parameters - 1] = new JExprParamMetadata (
 EDataType.STRING, // data type
 20, // precision
 0 // scale
);
 ...
 return defineJExpression(String expression,params);
}
```

## 単純なインタフェースに関する作業

単純なインタフェースで式を呼び出すには、invokeJExpression Java API メソッドを使用します。

### invokeJExpression

式を呼び出し、式の値を返します。

以下の構文を使用します。

```
(datatype)invokeJExpression(
 String expression,
 Object[] paramMetadataArray);
```

invokeJExpression メソッドの入力パラメータは、式および式の入力パラメータを含むオブジェクトの配列を表す文字列値です。

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              | パラメータ<br>のタイプ | データ型         | 説明                     |
|--------------------|---------------|--------------|------------------------|
| 式                  | 入力            | String       | 式を表す文字列。               |
| paramMetadataArray | 入力            | オブジェクト<br>[] | 式の入力パラメータを含むオブジェクトの配列。 |

invokeJExpression メソッドは、**【インポート】** タブと **【関数】** タブを除く任意のコードエントリタブで Java コードに追加することができます。

invokeJExpression メソッドを使用する場合は、以下のルールおよびガイドラインに従います。

- 戻りデータ型。invokeJExpression メソッドの戻りデータ型はオブジェクトです。関数の戻り値は、適切なデータ型でキャストする必要があります。  
Integer、Double、String、および byte [] のデータ型で値を返すことができます。
- 行タイプ。invokeJExpression メソッドの戻り値の行タイプは INSERT です。  
戻り値に異なる行タイプを使用するには、高度なインタフェースを使用します。
- NULL 値。パラメータとして NULL 値を渡した場合、または invokeJExpression メソッドの戻り値が NULL の場合、この値は NULL インジケータとして処理されます。  
例えば、式の戻り値が NULL で戻りデータ型が String の場合、NULL 値の文字列が返されます。
- Date データ型。Date データ型の入力パラメータは、String データ型に変換する必要があります。  
式中の文字列を Date データ型として使用するには、to\_date()関数を使用して、文字列を Date データ型に変換します。  
また、Date データ型を String データ型として返す式の戻り型をキャストする必要があります。

**注:** 式に渡す一連のパラメータには、先頭に文字 x を付けて番号を示す必要があります。例えば、3 つのパラメータを式に渡す場合は、各パラメータに x1、x2、および x3 という名前を付けます。

## 単純なインタフェースの例

**【ヘルパ】** および **【入力時】** コードエントリタブで、invokeJExpression API メソッドを使用する式を定義し、呼び出すことができます。

以下の例は、Java トランスフォーメーションの NAME および ADDRESS 入力ポートに対してルックアップを完了する方法、および COMPANY\_NAME 出力ポートに戻り値を割り当てる方法を示しています。

**【入力時】** コードエントリタブで、以下のコードを入力します。

```
COMPANY_NAME = (String)invokeJExpression(":lkp.my_lookup(X1,X2)", new Object [] {str1 ,str2});
generateRow();
```

# 高度なインタフェースに関する作業

高度なインタフェースでは、オブジェクト指向の API メソッドを使用して、式の定義、呼び出し、および結果の取得を行うことができます。

以下の表に、高度なインタフェースで使用可能なクラスおよび API メソッドを示します。

| クラスまたは API メソッド            | 説明                                                          |
|----------------------------|-------------------------------------------------------------|
| EDataType クラス              | 式のデータ型を列挙します。                                               |
| JExprParamMetadata クラス     | 式内の各パラメータのメタデータが含まれています。パラメータのメタデータには、データ型、精度、および位取りが含まれます。 |
| defineJExpression API メソッド | 式を定義します。式の文字列およびパラメータが含まれます。                                |
| invokeJExpression API メソッド | 式を呼び出します。                                                   |
| JExpression クラス            | メタデータの作成、呼び出し、式の結果の取得、および戻りデータ型のチェックを実行するメソッドが含まれます。        |

## 高度なインタフェースを使用した式の呼び出し

高度なインタフェースを使用して、式の定義、呼び出し、および結果の取得を行うことができます。

1. **【ヘルパ】** または **【入力時】** コードエントリタブで、式の各引数に対する JExprParamMetadata クラスのインスタンスを作成し、メタデータの値を設定します。必要に応じて、defineJExpression メソッドで JExprParamMetadata オブジェクトをインスタンス化できます。
2. defineJExpression メソッドを使用して、式の JExpression オブジェクトを取得します。
3. 適切なコードエントリタブで、invokeJExpression メソッドを使用して式を呼び出します。
4. isResultNull メソッドを使用して、戻り値の結果をチェックします。
5. getResultDataType メソッドまたは getResultMetadata メソッドを使用して、戻り値のデータ型またはメタデータをそれぞれ取得できます。
6. 適切な API メソッドを使用して、式の結果を取得します。getInt メソッド、getDouble メソッド、getStringBuffer メソッド、および getBytes メソッドを使用できます。

## 高度なインタフェースに関する作業のルールとガイドライン

高度なインタフェースを使用するときは、ルールとガイドラインに注意する必要があります。

以下のルールおよびガイドラインを使用します。

- パラメータとして NULL 値を渡す場合、または式の結果が NULL の場合、その値は NULL インジケータとして処理されます。例えば、式の結果が NULL で戻り値のデータ型が String の場合、NULL 値の文字列が返されます。式の結果は、isResultNull メソッドを使用して確認できます。

- Date データ型の入力パラメータを String データ型に変換すると、式で使えるようになります。式中の文字列を Date データ型として使用するには、to\_date()関数を使用して、文字列を Date データ型に変換します。

Date データ型を String データ型または Long データ型として返す式の結果を取得できます。

Date データ型を String データ型として返す式の結果を取得するには、getStringBuffer メソッドを使用します。Date データ型を Long データ型として返す式の結果を取得するには、getLong メソッドを使用します。

## EDataType クラス

式で使用される Java データ型を列挙します。式の戻りデータ型を取得、または JExprParamMetadata オブジェクトのパラメータのデータ型を割り当てます。EDataType クラスをインスタンス化する必要はありません。

以下の表に、式で値が列挙される Java データ型を示します。

| データ型         | 列挙された値 |
|--------------|--------|
| INT          | 1      |
| DOUBLE       | 2      |
| STRING       | 3      |
| BYTE_ARRAY   | 4      |
| DATE_AS_LONG | 5      |

以下の Java コード例は、EDataType クラスを使用して、String データ型を JExprParamMetadata オブジェクトに割り当てる方法を示しています。

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata (
 EDataType.STRING, // data type
 20, // precision
 0 // scale
);
...
```

## JExprParamMetadata クラス

式のパラメータを表すオブジェクトをインスタンス化し、パラメータのメタデータを設定します。

入力パラメータのメタデータを設定するには、JExprParamMetadata オブジェクトの配列を defineJExpression メソッドへの入力として使用します。JExprParamMetadata オブジェクトのインスタンスは、**【関数】** コードエントリタブまたは defineJExpression で作成できます。

以下の構文を使用します。

```
JExprParamMetadata paramMetadataArray[] = new JExprParamMetadata[numberOfParameters];
paramMetadataArray[0] = new JExprParamMetadata(datatype, precision, scale);
...
paramMetadataArray[numberOfParameters - 1] = new JExprParamMetadata(datatype, precision, scale);;
```

次の表に、引数を示します。

| 引数   | 引数のタイプ | 引数のデータ型   | 説明          |
|------|--------|-----------|-------------|
| データ型 | 入力     | EDatatype | パラメータのデータ型。 |
| 精度   | 入力     | Integer   | パラメータの精度。   |
| 位取り  | 入力     | Integer   | パラメータの位取り。  |

たとえば、以下の Java コードを使用して、2 つの JExprParamMetadata オブジェクトの配列を、String データ型、精度 20、位取り 20 でインスタンス化します。

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
return defineJExpression("LKP.LKP_addresslookup(X1,X2)",params);
```

## defineJExpression

式（式の文字列および入力パラメータを含む）を定義します。defineJExpression メソッドの引数には、式の構文を定義する入力パラメータと文字列値を含む JExprParamMetadata オブジェクトの配列が含まれています。

以下の構文を使用します。

```
defineJExpression(
 String expression,
 Object[] paramMetadataArray
);
```

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              | タイプ | データ型         | 説明                                         |
|--------------------|-----|--------------|--------------------------------------------|
| 式                  | 入力  | String       | 式を表す文字列。                                   |
| paramMetadataArray | 入力  | オブジェクト<br>[] | 式の入力パラメータを含む JExprParamMetadata オブジェクトの配列。 |

defineJExpression メソッドは、**【インポート】** タブと **【関数】** タブを除く任意のコードエントリタブで Java コードに追加することができます。

defineJExpression メソッドを使用するには、式の入力パラメータを表す JExprParamMetadata オブジェクトの配列をインスタンス化する必要があります。パラメータのメタデータ値を設定し、その配列をパラメータとして defineJExpression メソッドに渡します。

例えば、以下の Java コードでは、2 つの文字列の値をルックアップする式を作成します。

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
defineJExpression("lkp.mylookup(x1,x2)",params);
```

**注:** 式に渡す一連のパラメータには、先頭に文字 x を付けて番号を示す必要があります。例えば、3 つのパラメータを式に渡す場合は、各パラメータに x1、x2、および x3 という名前を付けます。

# JExpression クラス

式の作成および呼び出しを実行するメソッド、式の値を返すメソッド、および戻りデータ型をチェックするメソッドが含まれています。

以下の表に、JExpression クラスのメソッドを示します。

| メソッド名             | 説明                         |
|-------------------|----------------------------|
| invoke            | 式を呼び出します。                  |
| getResultDataType | 式の結果のデータ型を返します。            |
| getResultMetadata | 式の結果のメタデータを返します。           |
| isResultNull      | 式の結果の結果値をチェックします。          |
| getInt            | 式の結果の値を Integer データ型で返します。 |
| getDouble         | 式の結果の値を Double データ型で返します。  |
| getStringBuffer   | 式の結果の値を String データ型で返します。  |
| getBytes          | 式の結果の値を byte [] データ型で返します。 |

## 高度なインタフェースの例

高度なインタフェースを使用して、Java トランスフォーメーションのルックアップ式を作成したり呼び出したりすることができます。

次の例は、式を呼び出す関数を作成する方法、および戻り値を取得するための式を呼び出す方法を示しています。この例では、String データ型の 2 つの入力ポート（NAME および COMPANY）の値を myLookup 関数に渡します。myLookup 関数は、ルックアップ式を使用して ADDRESS 出力ポートの値をルックアップします。

**注:** この例では、LKP\_addresslookup という名前のマッピング内に未接続のルックアップトランスフォーメーションが存在すると想定しています。

**【ヘルパ】** タブで、以下の Java コードを使用します。

```
JExpression addressLookup() throws SDKException
{
 JExprParamMetadata params[] = new JExprParamMetadata[2];
 params[0] = new JExprParamMetadata (
 EDataType.STRING, // data type
 50, // precision
 0 // scale
);
 params[1] = new JExprParamMetadata (
 EDataType.STRING, // data type
 50, // precision
 0 // scale
);
 return defineJExpression(":LKP.LKP_addresslookup(X1,X2)",params);
}
JExpression lookup = null;
boolean isJExprObjCreated = false;
```

式を呼び出して ADDRESS ポートの値を返すには、**【入力時】** タブで以下の Java コードを使用します。

```
...
if(!isJExprObjCreated)
{
```

```

 lookup = addressLookup();
 isJExprObjCreated = true;
 }
 lookup = addressLookup();
 lookup.invoke(new Object [] {NAME,COMPANY}, ERowType.INSERT);
 EDataType addressDataType = lookup.getResultDataType();
 if(addressDataType == EDataType.STRING)
 {
 ADDRESS = (lookup.getStringBuffer()).toString();
 } else {
 logError("Expression result datatype is incorrect.");
 }
}
...

```

## JExpression クラス API リファレンス

JExpression クラスには、式の作成および呼び出しを行う API メソッド、式の値を返す API メソッド、および戻りデータ型をチェックする API メソッドが含まれています。

JExpression クラスには、以下の API メソッドが含まれています。

- getBytes
- getDouble
- getInt
- getLong
- getResultDataType
- getResultMetadata
- getStringBuffer
- 呼び出し
- isResultNull

### getBytes

式の結果の値を byte [] データ型で返します。AES\_ENCRYPT 関数でデータを暗号化する式の結果を取得します。

以下の構文を使用します。

```
objectName.getBytes();
```

以下の Java コードの例では、JExprEncryptData が JExpression オブジェクトである場合に、AES\_ENCRYPT 関数を使用してバイナリデータを暗号化する式の結果を取得します。

```
byte[] newBytes = JExprEncryptData.getBytes();
```

### getDouble

式の結果の値を Double データ型で返します。

以下の構文を使用します。

```
objectName.getDouble();
```

以下の Java コードの例では、JExprSalary が JExpression オブジェクトである場合に給与の値を Double データ型として返す式の結果を取得します。

```
double salary = JExprSalary.getDouble();
```

## getInt

式の結果の値を Integer データ型で返します。

以下の構文を使用します。

```
objectName.getInt();
```

たとえば、以下の Java コードを使用して、findEmpID が JExpression オブジェクトである場合に社員の ID 番号を整数として返す式の結果を取得します。

```
int empID = findEmpID.getInt();
```

## getLong

式の結果の値を Long データ型として返します。Date データ型を使用する式の結果を取得します。

以下の構文を使用します。

```
objectName.getLong();
```

以下の Java コード例を使用して、JExprCurrentDate が JExpression オブジェクトである場合に日付の値を Long データ型として返す式の結果を取得します。

```
long currDate = JExprCurrentDate.getLong();
```

## getResultDataType

式の結果のデータ型を返します。EDatatype の値を返します。

以下の構文を使用します。

```
objectName.getResultDataType();
```

以下の Java コードの例では、式を呼び出し、結果のデータ型を dataType 変数に割り当てます。

```
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);
EDatatype dataType = myObject.getResultDataType();
```

## getResultMetadata

式の結果のメタデータを返します。getResultMetadata を使用して、式の結果の精度、位取り、およびデータ型を取得し、式の戻り値のメタデータを JExprParamMetadata オブジェクトに割り当てることができます。結果のメタデータを取得するには、getScale、getPrecision、getDataType オブジェクトメソッドを使用します。

以下の構文を使用します。

```
objectName.getResultMetadata();
```

以下の Java コードの例では、myObject の戻り値の位取り、精度、およびデータ型を変数に割り当てます。

```
JExprParamMetadata myMetadata = myObject.getResultMetadata();
int scale = myMetadata.getScale();
int prec = myMetadata.getPrecision();
int datatype = myMetadata.getDataType();
```

**注:** getDataType オブジェクトメソッドは、データ型の整数値を EDatatype に列挙されたとおりに返します。



## getStringBuffer

式の結果の値を String データ型で返します。

以下の構文を使用します。

```
objectName.getStringBuffer();
```

以下の Java コード例を使用して、JExprConcat が JExpression オブジェクトである場合に 2 つの連結された文字列を返す式の結果を取得します。

```
String result = JExprConcat.getStringBuffer();
```

## 呼び出し

式を呼び出します。invoke の引数には、入力パラメータおよび行タイプを定義するオブジェクトが含まれています。invoke メソッドを使用する前に JExpression オブジェクトをインスタンス化する必要があります。行タイプには、ERowType.INSERT、ERowType.DELETE、および ERowType.UPDATE を使用します。

以下の構文を使用します。

```
objectName.invoke(
 new Object[] { param1[, ... paramN]},
 rowType
);
```

次の表に、引数を示します。

| 引数         | データ型        | 入力/<br>アウトプ<br>ット | 説明                      |
|------------|-------------|-------------------|-------------------------|
| objectName | JExpression | 入力                | JExpression のオブジェクト名です。 |
| parameters | -           | 入力                | 式の入力値が含まれるオブジェクトの配列です。  |

例えば、**【関数】** コードエントリタブで、address\_lookup()という名前の関数を作成したとします。この関数は、式を表す JExpression オブジェクトを返します。NAME および COMPANY の入力ポートを使用する式を呼び出すには、以下のコードを使用します。

```
JExpression myObject = address_lookup();
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);
```

## isResultNull

式の結果の値をチェックします。

以下の構文を使用します。

```
objectName.isResultNull();
```

以下の Java コード例を使用して式を呼び出し、戻り値が NULL でなかった場合に式の戻り値を ADDRESS 変数に割り当てます。

```
JExpression myObject = address_lookup();
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);
if(!myObject.isResultNull()) {
 String address = myObject.getStringBuffer();
}
```

## 第 22 章

# ジョイナトランスフォーメーション

この章では、以下の項目について説明します。

- [ジョイナトランスフォーメーションの概要, 350 ページ](#)
- [ジョイナトランスフォーメーションの詳細プロパティ, 351 ページ](#)
- [ジョイナキャッシュ, 352 ページ](#)
- [ジョイナトランスフォーメーションポート, 353 ページ](#)
- [動的マッピングでのジョイナトランスフォーメーション, 354 ページ](#)
- [ジョイナトランスフォーメーションのポートセクタ, 354 ページ](#)
- [結合条件の定義, 357 ページ](#)
- [結合タイプ, 361 ページ](#)
- [ジョイナトランスフォーメーションでのソート済み入力, 363 ページ](#)
- [同じソースのデータの結合, 367 ページ](#)
- [ソースパイプラインのブロック, 369 ページ](#)
- [ジョイナトランスフォーメーションのパフォーマンスのヒント, 370 ページ](#)
- [ジョイナトランスフォーメーションのルールとガイドライン, 371 ページ](#)
- [非ネイティブ環境でのジョイナトランスフォーメーション, 371 ページ](#)

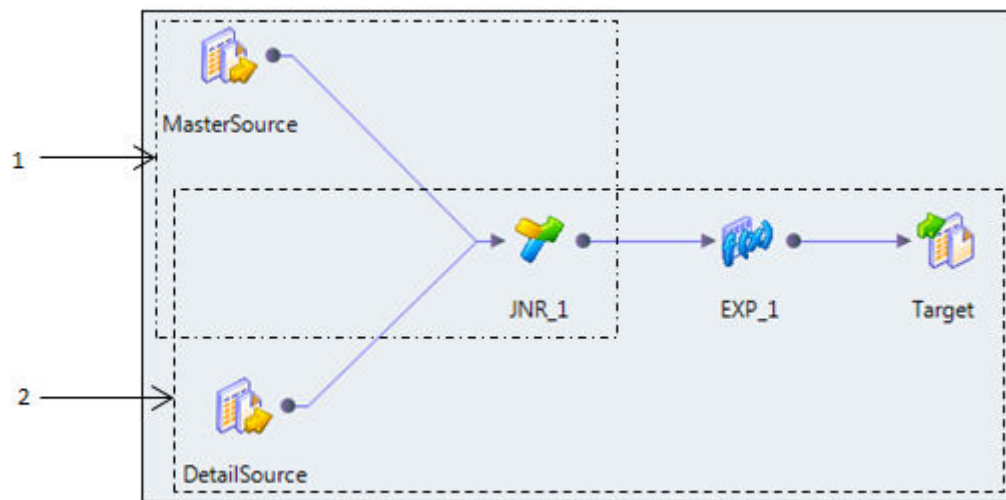
## ジョイナトランスフォーメーションの概要

ジョイナトランスフォーメーションを使用すると、異なる場所またはファイルシステムにある 2 つの関連する異種ソースからのソースデータを結合できます。同じソースからのデータを結合することもできます。ジョイナトランスフォーメーションは、アクティブな複数グループトランスフォーメーションです。

Joiner トランスフォーメーションは、一致するカラムが少なくとも 1 つあるソースを結合します。Joiner トランスフォーメーションでは、2 つのソース間の 1 つ以上のカラムのペアと一致する条件を使用します。

2 つの入力パイプラインには、マスターパイプラインと明細パイプライン、またはマスターブランチと明細ブランチがあります。マスターパイプラインは Joiner トランスフォーメーションで終了しますが、明細パイプラインはターゲットまで継続します。

次の図に、ジョイナトランスフォーメーションを含むマッピング内のマスタパイプラインおよび明細パイプラインを示します。



1. マスタパイプライン
2. 明細パイプライン

マッピング内の3つ以上のソースを結合する場合は、ジョイナトランスフォーメーションからの出力を他のソースパイプラインと結合します。すべてのソースパイプラインを結合するまで、ジョイナトランスフォーメーションをマッピングに追加します。

## ジョイナトランスフォーメーションの詳細プロパティ

Data Integration Service がジョイナトランスフォーメーションのデータを処理する方法を決定するプロパティを設定します。

【詳細】 タブで、以下のプロパティを設定します。

### ジョイナのデータキャッシュサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってデータキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

### ジョイナのインデックスキャッシュサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってインデックスキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

### キャッシュディレクトリ

データ統合サービスがインデックスキャッシュファイルとデータキャッシュファイルを作成するディレクトリ。このディレクトリが存在し、キャッシュファイルを格納するのに十分なディスク容量を備えていることを確認します。

キャッシュのパーティション化時のパフォーマンスを向上させるには、セミコロンで区切って複数のディレクトリを入力します。キャッシュのパーティション化により、トランスフォーメーションを処理する各パーティションに個別のキャッシュが作成されます。

デフォルトは CacheDir システムパラメータです。このプロパティには、別のシステムパラメータまたはユーザー定義のパラメータを設定できます。

### ソート済み入力

入力データがグループで事前にソートされていることを示します。ソートされたデータを結合するには、[ソート済み入力] を選択します。ソート済み入力を使用するとパフォーマンスを向上させることができます。

### マスタのソート順

マスタソースデータのソート順を指定します。マスタソースデータが昇順である場合は、昇順を選択します。昇順を選択した場合には、ソート済み入力も有効にします。デフォルトは [自動] です。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

### 関連項目：

- [「キャッシュサイズ」 \(ページ 72\)](#)

## ジョイナキャッシュ

ジョイナトランスフォーメーションを使用するマッピングを実行すると、データ統合サービスはメモリにインデックスキャッシュおよびデータキャッシュを作成してトランスフォーメーションを実行します。メモリキャッシュ内の使用可能スペースよりも多くのスペースが必要な場合、データ統合サービスはオーバーフローしたデータをキャッシュファイルに格納します。

ジョイナトランスフォーメーションを使用するマッピングを実行するときに、データ統合サービスによってマスターソースと明細ソースから同時に行が読み込まれ、マスター行に基づいてインデックスキャッシュおよびデータキャッシュが構築されます。データ統合サービスでは、明細ソースデータおよびキャッシュされたマスターデータに基づいて結合が実行されます。

ジョイナトランスフォーメーションのタイプにより、データ統合サービスがキャッシュに格納する行数が決まります。

以下の表に、ジョイナトランスフォーメーションのタイプごとにデータ統合サービスがキャッシュに格納する情報を示します。

| ジョイナトランスフォーメーションのタイプ | インデックスキャッシュ                                                                                                                                                                                                                                                      | データキャッシュ                                                                                                                                                     |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 未ソートの入力              | ジョイン条件に、一意のインデックスキーを持つすべてのマスター行を格納します。                                                                                                                                                                                                                           | すべてのマスター行を格納します。                                                                                                                                             |
| 別のソースを使用したソート済み入力    | ジョイン条件に、一意のインデックスキーを持つマスター行を 100 行格納します。                                                                                                                                                                                                                         | インデックスキャッシュに格納された行に対応するマスター行を格納します。マスターデータに同じキーを持つ複数の行が含まれる場合、データ統合サービスによって 100 件以上の行がデータキャッシュに格納されます。                                                       |
| 同じソースを使用したソート済み入力    | ジョイン条件に、一意のインデックスキーを持つすべてのマスター行または明細行を格納します。データ統合サービスで、マスターパイプラインよりも高速な明細パイプラインを処理する場合、明細行を格納します。それ以外の場合は、マスター行を格納します。格納される行数は、マスターパイプラインと明細パイプラインの処理の速さによって異なります。他よりも速く行を処理するパイプラインがある場合、データ統合サービスはすでに処理されたすべての行をキャッシュし、他のパイプラインが行の処理を終えるまでそれらの行をキャッシュしたままにします。 | インデックスキャッシュに格納された行に、データを格納します。インデックスキャッシュにマスターパイプラインのキーが格納されると、データキャッシュにはマスターパイプラインのデータが格納されます。インデックスキャッシュに明細パイプラインのキーが格納されると、データキャッシュには明細パイプラインのデータが格納されます。 |

## ジョイナトランスフォーメーションポート

ジョイナトランスフォーメーションにはさまざまなポートタイプがあり、ポートタイプによって Data Integration Service が結合を実行する方法が決まります。

ジョイナトランスフォーメーションには、以下のポートタイプがあります。

### マスタ

マッピング内のマスタソースにリンクするポート。

### 詳細

マッピング内の詳細ソースにリンクするポート。

### 動的ポート

動的マッピングでポートを受け取るか返します。動的ポートはアップストリームトランスフォーメーションから 1 つ以上のカラムを受け取り、カラムごとに生成されたポートを作成できます。動的出力ポートは、1 つ以上の生成されたポートを返すことができます。入力ルールを定義して、動的ポートが受け取るカラムを決定できます。

ポートはマスタポートから明細ポートに変更できます。明細ポートからマスタポートに変更することもできます。1 つのポートのポートタイプを変更すると、すべてのポートのポートタイプが変更されます。したがって、マスタポートを明細ポートに変更すると、すべてのマスタポートが明細ポートに、すべての明細ポートがマスタポートに変更されます。

# 動的マッピングでのジョイナトランスフォーメーション

動的マッピングでジョイナトランスフォーメーションを使用できます。結合条件で、動的ポートおよび生成されたポートを参照できます。

動的マッピングとは、ソース、ターゲット、トランスフォーメーションのロジックが実行時に変更される可能性のあるマッピングです。パラメータとルールを設定してデータ構造を変更できます。動的マッピングでジョイナトランスフォーメーションを使用する場合、ソースの構造が変わることがあります。入力ポートおよび結合条件のポートも変わります。

以下のタスクを実行することで、動的マッピングにジョイナトランスフォーメーションを設定できます。

## 動的ポートを定義する。

動的ソースのさまざまな入力カラムに対応できるように、動的ポートと生成されたポートを定義します。結合条件に動的ポートまたは生成されたポートを含めることができます。

## ポートセクタを定義する。

結合条件で使用するポートを含むポートセクタを定義します。ポートセクタに含めるポートを決定する選択ルールを設定します。ポートセクタをパラメータ化すると、実行時に特定のポートを含めることができます。

## 結合条件をパラメータ化する。

結合条件全体をパラメータ化できます。必要となる可能性がある各結合条件について、式パラメータを設定します。

動的マッピングの詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

# ジョイナトランスフォーメーションのポートセクタ

ジョイナトランスフォーメーションでポートが生成済みであり、トランスフォーメーションに実行時に生成された他のポートがある場合、有効な結合条件を設定する必要があります。

例えば、動的マッピングに次の結合条件を持つジョイナトランスフォーメーションが含まれます。

```
CustomerID = CustomerNo
```

CustomerID は、ジョイナトランスフォーメーションで生成されたポートです。マッピングに動的ソースがあるため、マッピングを複数の異なるソースファイル形式で実行することが可能です。顧客番号を含むカラムには、各ソースファイルに次のような異なる名前があります。CustomerID、CustomerNum、または CustNO。

ジョイナトランスフォーメーション内でポートセクタを作成すると、動的ソースの異なる顧客カラム名に対応することができます。「Cust」というプレフィックスが付いたポート名を含む選択ルールで、ポートセクタを設定します。

次に、結合条件を設定して、カラム名「CustomerID」ではなく、ポートセクタ名を含めます。

```
Customer_PortSelector = CustomerNo
```

結合条件は、「Cust」で始まる任意のポート名に対して有効です。

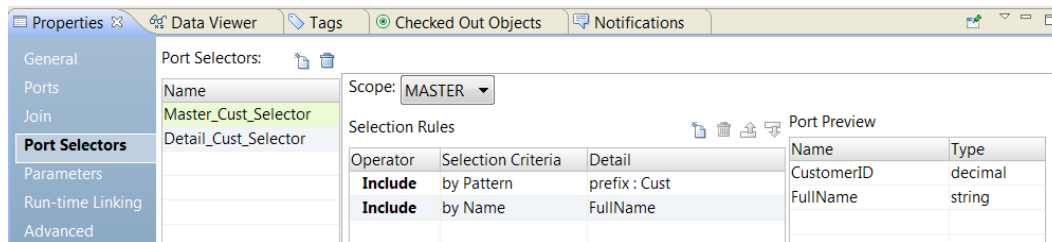
ポートセクタには 1 つ以上のポートを含めることができます。結合条件のマスタグループと詳細グループに同じ数のポートが含まれる場合、結合条件に複数のポートを含めることができます。

## 選択ルール

ポートセレクトを設定するには、選択ルールを定義して、含める生成済みポートを決定します。選択ルールは、動的ポートに設定できる入力ルールと同じです。

ポートセレクトには、ポートまたは生成済みポートを含めることができます。ポートセレクトの設定は、**ポートセレクト**タブで行います。

次の図は、**ポートセレクト**タブを示しています。



ポートセレクトに次のプロパティを設定します。

### 名前

ポートセレクトを識別します。1つのトランスフォーメーションに複数のポートセレクトを作成し、式で参照できます。

### スコープ

ポートセレクトが適用されるポートグループを識別します。スコープとしてマスタまたは詳細を選択できます。

### 選択ルール

ポートセレクトに含めるポートを決定します。選択ルールを作成すると、**ポートのプレビュー**パネルに現在の入力ポートのうち適格なポートが表示されます。これらのポートは変わる可能性があります。さまざまなソースからのポートに対応できるように選択ルールを設定します。

以下の条件に基づいて選択ルールを作成できます。

#### 演算子

選択ルールが返すポートを含めるか、除外します。デフォルトは「含める」です。ポートを除外する前にポートを含める必要があります。

#### 選択条件

作成する選択ルールのタイプ。ポートタイプまたはカラム名に基づいてルールを作成できます。カラム名に基づいてポートを含めるには、特定の名前を検索するか、名前に含まれる文字列パターンを検索します。

#### 詳細

選択条件に適用する値。選択条件がカラム名基準になっている場合は、検索する文字列または名前を設定します。選択条件がポートタイプ基準になっている場合は、含めるポートタイプを選択します。

次の表に、選択条件と条件の詳細を指定する方法を示します。

| 選択条件 | 説明                                    | 詳細                                                                               |
|------|---------------------------------------|----------------------------------------------------------------------------------|
| すべて  | すべてのポートを含めます。                         | 詳細は不要です。                                                                         |
| 名前   | ポート名に基づいてポートをフィルタリングします。              | 値のリストからポート名を選択するか、[ポート] または [ポートリスト] のパラメータを使用します。                               |
| タイプ  | 各ポートのデータ型に基づいてポートをフィルタリングします。         | リストからデータ型を選択します。                                                                 |
| パターン | 名前に含まれる文字列または正規表現を使用してポートをフィルタリングします。 | ポート名のパターンタイプとして、プレフィックス、サフィックス、または正規表現を選択します。次に、パターンの値を入力するか、文字列タイプのパラメータを使用します。 |

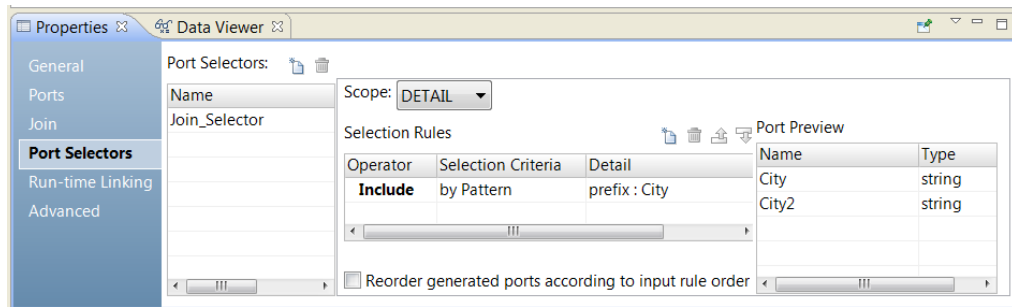
## ポートセレクトの作成

動的式、ルックアップ条件、または結合条件で使用するポートを決定するために、ポートセレクトを作成します。

1. **【ポートセレクト】** タブをクリックします。
2. **【ポートセレクト】** 領域で、**【新規】** をクリックします。  
Developer tool が、すべてのポートが含まれるデフォルトの選択ルールを使用してポートセレクトを作成します。
3. **【ポートセレクト】** 領域で、ポートセレクト名を一意的な名前に変更します。
4. ジョイントトランスフォーメーションまたはルックアップトランスフォーメーションに対して作業している場合は、スコープを選択します。  
使用可能なポートは、選択したポートのグループに基づいて変更されます。
5. **【選択ルール】** 領域で、**【演算子】** を選択します。
  - 含む。ポートセレクトのポートを含めるルールを作成します。ポートを除外する前にポートを含める必要があります。
  - 除外。ポートセレクトから特定のポートを除外するルールを作成します。
6. **【選択条件】** を選択します。
  - 名前。特定のポートを名前で選択します。スコープ内のポートのリストからポート名を選択できます。
  - タイプ。ポートをタイプで選択します。1 つまたは複数のデータ型を選択できます。
  - パターン。ポート名の文字のパターンでポートを選択します。特定の文字で検索するか、正規表現を作成できます。

次の図は、**【ポートセレクト】** タブを示しています。





7. **【詳細】** カラムをクリックします。  
**【入力ルールの詳細】** ダイアログボックスが表示されます。
8. ポートをフィルタ処理する基準となる値を選択します。
  - 名前。値またはパラメータを基準としてポートリストを作成する場合に選択します。 **【選択】** をクリックして、リスト内のポートを選択します。
  - タイプ。リストから1つ以上のデータ型を選択します。 **【ポートのプレビュー】** 領域に、選択したタイプのポートが表示されます。
  - パターン。ポート名のプレフィックスまたはサフィックスから、特定の文字パターンを検索する場合に選択します。または、検索に使用する正規表現を作成することを選択します。パラメータを設定するか、検索に使用するパターンを設定します。**【ポートのプレビュー】** 領域に、設定したルールどおりにポートセクタのポートが表示されます。
9. ポートセクタのポートの順序を変更するには、**【生成されたポートの順序を入力ルールの順序に従って変更】** を選択します。

## 結合条件の定義

結合条件は、データ統合サービスが2つの行を結合するために使用する両方の入力ソースのポートを含みます。

選択した結合タイプに応じて、データ統合サービスは行を結果セットに追加するか、または行を破棄します。Joiner トランスフォーメーションは、ジョイントタイプ、ジョイン条件、および入力データのソースに基づいて結果セットを生成します。

ジョイン条件を定義する前に、マスターソースおよび明細ソースが最適なパフォーマンスに設定されていることを確認してください。データ統合サービスは、マッピングの実行中にマスターソースの各行を明細ソースと比較します。未ソートジョイナトランスフォーメーションのパフォーマンスを高めるには、行の比較の少ないソースをマスターソースとして使用します。ソート済みジョイナトランスフォーメーションのパフォーマンスを高めるには、重複キー値の少ないソースをマスターとして使用します。

ジョイン条件では、ジョイナトランスフォーメーションの入力ソースからの1つ以上のポートを使用します。ポートの数が増えると、2つのソースの結合に要する時間も長くなります。条件内でのポートの順番によって、ジョイナトランスフォーメーションのパフォーマンスに影響を与える場合があります。結合条件で複数のポートを使用する場合、データ統合サービスは指定された順にポートを比較します。

Char データ型と Varchar データ型を結合する場合、データ統合サービスは Char 値に埋め込まれている空白の数を文字列の一部として数えます。

```
Char(40) = "abcd"
Varchar(40) = "abcd"
```

この Char 値には "abcd" と 36 個の空白が埋め込まれています。Char フィールドは後ろに空白が含まれているため、データ統合サービスはこの2つのフィールドを結合しません。

**注:** ジョイナトランスフォーメーションは、NULL 値の一致は検出しません。例えば、EMP\_ID1 と EMP\_ID2 の両方に NULL 値を持つ行が含まれている場合、データ統合サービスはこれらを一致とは見なさないため、2 つの行の結合は実行されません。NULL 値を持つ行を結合するには、NULL 入力をデフォルト値で置き換えてから、デフォルト値で結合します。

単純条件タイプまたは詳細条件タイプを定義できます。式パラメータも定義できます。式パラメータは、結合式を含むパラメータです。マッピングパラメータを使用して、実行時にパラメータ値を変更できます。

## 単純条件タイプ

ソート済みまたは未ソートのジョイナトランスフォーメーションの単純条件タイプを定義します。

単純条件には、指定されたマスタと明細ソースを比較する 1 つ以上の条件が含まれます。単純条件では以下の形式を使用する必要があります。

```
<master_port> operator <detail_port>
```

ソート済みジョイナトランスフォーメーションの条件では、等号演算子を使用する必要があります。

未ソートのジョイナトランスフォーメーションの条件では、=、!=、>、>=、<、<=のどれでも使用できます。

例えば、テーブルを含む 2 つのソース、EMPLOYEE\_AGE と EMPLOYEE\_POSITION があり、どちらにも従業員 ID 番号が含まれる場合、以下の条件は両方のソースに含まれる従業員の行に一致します。

```
EMP_ID1 = EMP_ID2
```

Developer tool は、単純条件のデータタイプを検証します。条件の両方のポートは、同じデータタイプを持つ必要があります。データタイプが一致しない 2 つのポートを条件で使用しなければならない場合、データタイプが一致するように変換します。

単純条件で、結合条件のリストを設定できます。複数の結合条件を設定する場合、結合を実行するにはすべての条件が true である必要があります。

例えば、単純条件で次の文を設定する場合があります。

```
StoreID = StoreNO
Dept = Department
Salary > Commission
```

同じ文を詳細条件として表示すると、結合条件は次の式として表示されます。

```
StoreID = StoreNO AND Dept = Department AND (Salary > Commission)
```

## 詳細条件タイプ

未ソートジョイナトランスフォーメーションの詳細な条件タイプを定義します。

詳細条件には、ブール値または数値を評価するための式を含めることができます。詳細条件には、演算子=、!=、>、>=、<、<=を使用できます。

結合条件には定数を入力できます。FALSE に該当する値はゼロ (0) です。ゼロ以外の値は TRUE とみなされます。例えば、トランスフォーメーションに数値データ型を使用する NUMBER\_OF\_UNITS というポートがあるとします。NUMBER\_OF\_UNITS の値が 0 に等しければ FALSE を返すようにフィルタ条件を設定します。値がゼロでなければ、TRUE が返されます。

**注:** 単一の動的ポートやポートセレクトは結合条件のブール値として使用できません。

結合条件に式を入力するには、**[結合]** タブで詳細条件タイプを選択します。式エディタを使用して、条件にポート、パラメータ、式、ポートセレクト、および演算子を含めます。生成されたポートを使用できます。ポートタイプが数値の場合、式エディタに単一のポートを入力できます。ただし、1 つのポートセレクトを式として入力することはできません。

例えば、従業員のフルネームで一致させてソースを結合する場合、マスターソースに FirstName ポートと LastName ポートを含めます。明細ソースには FullName ポートを含めます。マスターポートを連結し、両方のソースのフルネームを一致させるには、次の条件を定義します。

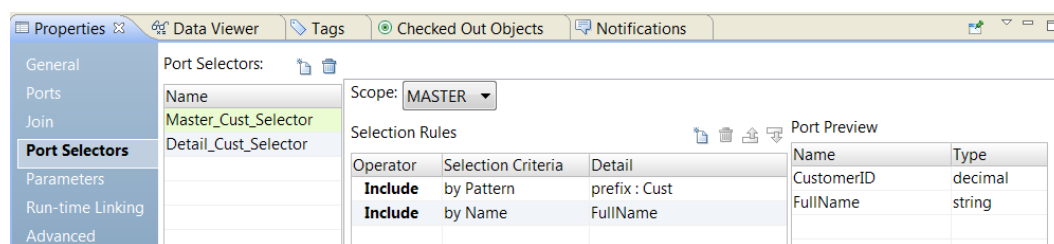
CONCAT(FirstName, LastName) = FullName

## 結合条件のポートセレクト

結合条件にポートセレクトを含めることができます。結合条件では、マスタグループおよび詳細グループからポートセレクトをそれぞれ参照する必要があります。

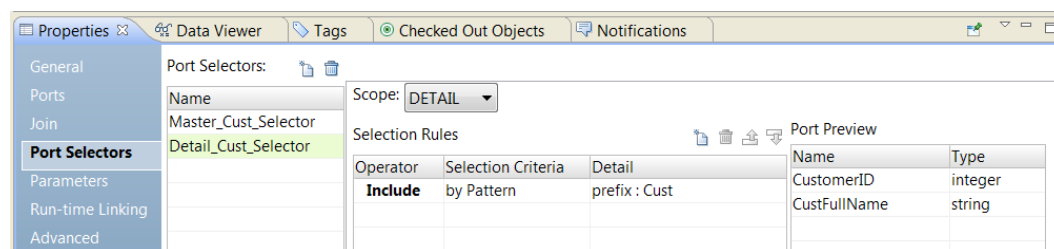
例えば、ジョイントランスフォーメーションに動的ポートが含まれるとします。複数の生成されたポートの結合条件での比較が必要になる場合があります。

次の図は、マスタグループのポートセレクトのフィールドを示しています。



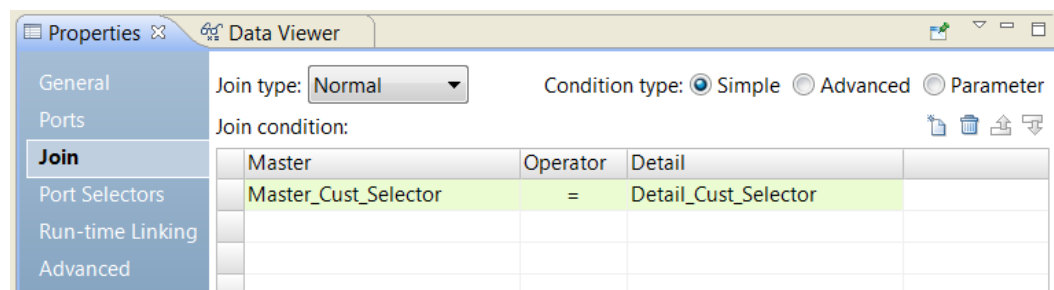
Master\_Cust\_Selector には、CustomerID および FullName ポートが含まれます。

次の図は、詳細グループのポートセレクトのフィールドを示しています。



Detail\_Cust\_Selector には、CustomerNo および CustFullName ポートが含まれます。これらのポートには Cust というプレフィックスがあります。

次の簡単な結合条件を作成します。



結合条件は、Master\_Cust\_Selector の各ポートを Detail\_Cust\_Selector と比較します。結合条件は次のとおりです。CustomerID = CustomerNo AND FullName = CustFullName

各ポートセレクトには、同じ数のポートが含まれている必要があります。ポートは同じタイプである必要があります。

**注:** ポートセクタのスコープを変更して、単純タイプの結合条件が有効でなくなった場合、Developer tool は条件タイプを詳細タイプに切り替えることがあります。【結合】タブで、結合条件タイプを単純タイプに戻すことができます。

## 結合条件の動的ポート

ポートセクタでは動的ポートを参照できます。

動的ポートには、1 つ以上の生成されたポートを含めることができます。結合条件に動的ポートが含まれている場合は、マスタポートの数と明細ポートの数が一致していなければなりません。

例えば、動的ポート A に、2 つの生成されたポートがあるとします。

CustomerID  
OrderID

動的ポート B にも 2 つの生成されたポートがあります。

CustomerNo  
OrderNo

このとき、次の結合条件は有効です。

DynamicPortA = DynamicPortB

この結合条件は次の式に展開されます。

CustomerID = CustomerNo AND OrderID = OrderNo

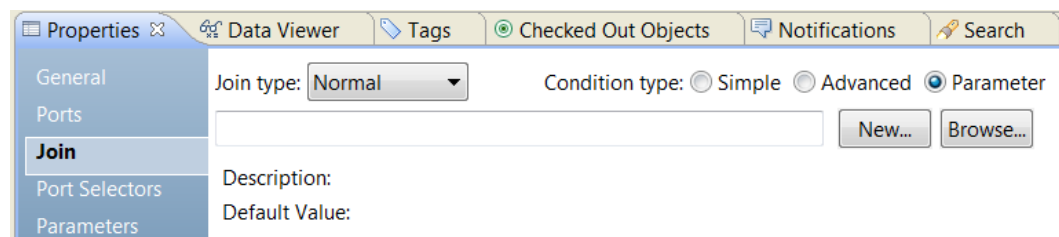
ポートセクタに動的ポートと同数のポートが含まれている場合は、結合条件でポートセクタと動的ポートを参照できます。

## 式パラメータ

結合条件を含む式パラメータを定義できます。ジョイントランスフォーメーションでパラメータを結合条件として選択できます。

結合条件にパラメータを使用するには、【結合】タブでパラメータの条件タイプを選択します。

次の図は、パラメータ条件タイプを選択する場所を示しています。



既存のパラメータを参照するか、パラメータを作成できます。パラメータを作成するには、【新規】をクリックして、パラメータを定義します。式エディタで式を作成します。

**注:** 式パラメータに他のパラメータを含めることはできません。式パラメータにパラメータを埋め込むと、データ統合サービスで実行時検証エラーが発生します。

# 結合タイプ

ジョイナトランスフォーメーションでは、さまざまなタイプのソースから結合を生成することができます。

ジョイナトランスフォーメーションは、以下の結合タイプをサポートしています。

- ノーマル
- マスター外部
- 明細外部
- 完全外部

**注:** Normal または Master Outer ジョインは、Full Outer や Detail Outer ジョインよりも高速に動作します。

どちらのソースのデータも含まないフィールドが結果セットにある場合、ジョイナトランスフォーメーションは空のフィールドを NULL 値で埋めます。あるフィールドが NULL を返すことが分かっていて、NULL をターゲットに挿入したくない場合は、対応するポートにデフォルト値を設定できます。

## ノーマル結合

ノーマル結合の場合、Data Integration Service は条件に基づいて、一致しないマスタソースおよび明細ソースのデータの行をすべて破棄します。

例えば、PARTS\_SIZE および PARTS\_COLOR という、自動車部品に関する 2 つのデータソースがあるとし

ます。

PARTS\_SIZE データソースはマスタソースで、次のデータが格納されています。

| PART_ID1 | DESCRIPTION | SIZE   |
|----------|-------------|--------|
| 1        | Seat Cover  | Large  |
| 2        | Ash Tray    | Small  |
| 3        | Floor Mat   | Medium |

PARTS\_COLOR データソースは明細ソースで、次のデータが格納されています。

| PART_ID2 | DESCRIPTION | COLOR  |
|----------|-------------|--------|
| 1        | Seat Cover  | Blue   |
| 3        | Floor Mat   | Black  |
| 4        | Fuzzy Dice  | Yellow |

両方の PART\_ID を照合して 2 つのテーブルを結合するには、次のように条件を設定します。

PART\_ID1 = PART\_ID2

これらのテーブルに対してノーマル結合を実行すると、結果セットには以下のデータが含まれます。

| PART_ID | DESCRIPTION | SIZE   | COLOR |
|---------|-------------|--------|-------|
| 1       | Seat Cover  | Large  | Blue  |
| 3       | Floor Mat   | Medium | Black |

次の例は、相当する SQL 文を示しています。

```
SELECT * FROM PARTS_SIZE, PARTS_COLOR WHERE PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2
```

## マスタ外部結合

マスタ外部結合は、明細ソースのデータのすべての行およびマスタソースの一致する行をすべて保持します。一致しないマスタソースの行は無視されます。

サンプルのテーブルに対して同じ条件でマスタ外部結合を実行すると、結果セットには以下のデータが含まれます。

| PART_ID | DESCRIPTION | SIZE   | COLOR  |
|---------|-------------|--------|--------|
| 1       | Seat Cover  | Large  | Blue   |
| 3       | Floor Mat   | Medium | Black  |
| 4       | Fuzzy Dice  | NULL   | Yellow |

Fuzzy Dice にはサイズが指定されていないため、Data Integration Service はフィールドに NULL を入力します。

以下の例は、相当する SQL 文を示しています。

```
SELECT * FROM PARTS_SIZE RIGHT OUTER JOIN PARTS_COLOR ON (PARTS_COLOR.PART_ID2 = PARTS_SIZE.PART_ID1)
```

## 明細外部結合

明細外部結合は、マスタソースのデータのすべての行および明細ソースの一致する行をすべて保持します。一致しない明細ソースの行は無視されます。

サンプルのテーブルに対して同じ条件で明細外部結合を実行すると、結果セットには以下のデータが含まれます。

| PART_ID | DESCRIPTION | SIZE   | COLOR |
|---------|-------------|--------|-------|
| 1       | Seat Cover  | Large  | Blue  |
| 2       | Ash Tray    | Small  | NULL  |
| 3       | Floor Mat   | Medium | Black |

Ash Tray で色が指定されていないため、Data Integration Service はフィールドに NULL を入力します。

以下の例は、相当する SQL 文を示しています。

```
SELECT * FROM PARTS_SIZE LEFT OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2)
```

## 完全外部結合

完全外部結合では、マスタソースと明細ソースの両方のデータのすべての行が保持されます。

サンプルのテーブルに対して同じ条件で完全外部結合を実行すると、結果セットには以下のデータが含まれます。

| PARTED | DESCRIPTION | SIZE  | Color |
|--------|-------------|-------|-------|
| 1      | Seat Cover  | Large | Blue  |

| PARTED | DESCRIPTION | SIZE   | Color  |
|--------|-------------|--------|--------|
| 2      | Ash Tray    | Small  | NULL   |
| 3      | Floor Mat   | Medium | Black  |
| 4      | Fuzzy Dice  | NULL   | Yellow |

Ash Tray では色が、Fuzzy Dice ではサイズが指定されていないため、Data Integration Service はフィールドに NULL を入力します。

以下の例は、相当する SQL 文を示しています。

```
SELECT * FROM PARTS_SIZE FULL OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2)
```

## ジョイナトランスフォーメーションでのソート済み入力

[ソート済み入力] オプションを使用して、ジョイナトランスフォーメーションのパフォーマンスを向上させることができます。データがソートされているときは、ソート済み入力を使用します。

ジョイナトランスフォーメーションでソート済みデータを使うように設定すると、Data Integration Service はディスクの入出力を最小化してパフォーマンスを向上させます。大量のデータセットを扱う場合に、パフォーマンスを最大限に向上させることができます。

ソート済みデータを使用するようにマッピングを設定するには、Data Integration Service がジョイナトランスフォーメーションを処理する場合にソート済みデータを使用できるように、マッピング内でソート順を確立して維持します。以下の手順を実行して、マッピングの設定を行います。

1. 結合するデータのソート順を設定します。
2. ソート済みデータのソート順を維持するトランスフォーメーションを追加します。
3. ソート済みデータを使用するようにジョイナトランスフォーメーションを設定し、ソートの基点ポートを使用するように結合条件を設定します。ソートの基点は、ソート済みデータのソースを表します。

## ソート順の設定

ソート順は、Data Integration Service からソート済みデータがジョイナトランスフォーメーションに確実に渡されるように設定します。

ソート順を設定するには、以下のいずれかの方法を使用します。

- ソート済みフラットファイルを使用する。ソート済みデータがフラットファイルに入っている場合は、ソートカラムの順序が各ソースファイル内で一致していることを確認します。
- ソート済みリレーショナルデータを使用する。リレーショナルデータオブジェクト内のソート済みポートを使用して、ソースデータベースのカラムをソートします。各リレーショナルデータオブジェクト内のソート済みポートの順序は同一に設定してください。
- ソータートランスフォーメーションを使用して、リレーショナルデータまたはフラットファイルデータをソートします。ソータートランスフォーメーションをマスタおよび明細パイプラインに配置します。各ソータートランスフォーメーションで、同じ順のソートキーポートおよび同じソート順方向を使用するよう設定します。

未ソートデータまたは正しくソートされていないデータを、ソート済みデータを使用するように設定されているジョイナトランスフォーメーションに渡すと、マッピングの実行が失敗します。Data Integration Serviceは、ログファイルにエラーを記録します。

## マッピングへのトランスフォーメーションの追加

ジョイナトランスフォーメーションでソート済みデータのソート順を維持するトランスフォーメーションをマッピングに追加します。

ジョイナトランスフォーメーションをソートの基点のすぐ後に置くことで、ソート済みデータを維持できます。

ソートの基点とジョイナトランスフォーメーションの間にトランスフォーメーションを追加する場合は、以下のガイドラインに従って、ソート済みデータを維持します。

- ソートの基点とジョイナトランスフォーメーションの間に、以下のトランスフォーメーションは配置しないでください。
  - ランク
  - 共有体
  - 未ソートアグリゲータ
  - 前のトランスフォーメーションが1つ含まれるマブレット
- 以下のガイドラインに従うのであれば、ソートの基点とジョイナトランスフォーメーションの間にソート済みアグリゲータトランスフォーメーションを配置することができます。
  - アグリゲータトランスフォーメーションをソート済み入力に合わせて設定します。
  - アグリゲータトランスフォーメーション内のカラムのグループのポートは、ソートの基点でのポートと同じものを使用してください。
  - ポートのグループは、ソートの基点でのポートと同じ順序である必要があります。
- 別のパイプラインを持つジョイナトランスフォーメーションの結果セットを結合する場合は、最初のジョイナトランスフォーメーションからの出力されるデータがソート済みであることを確認してください。

## 結合条件のルールとガイドライン

ソート済みジョイナトランスフォーメーションの結合条件を作成する場合、特定のルールとガイドラインが適用されます。

結合条件条件を作成するときには、以下のガイドラインに従ってください。

- 等号演算子を使用する単純な条件タイプを定義する必要があります。
- ソートの基点とジョイナトランスフォーメーションの間にソート済みアグリゲータトランスフォーメーションを使用する場合は、そのソート済みアグリゲータトランスフォーメーションを、結合条件を定義するときのソートの基点として扱います。
- 結合条件で使用するポートは、ソートの基点でのポートと一致している必要があります。
- 複数の結合条件を設定する場合、最初の結合条件内のポートは、ソートの基点での最初のポートと一致している必要があります。
- 複数の条件を設定する場合、条件の順序は、ソートの基点でのポートの順序と一致している必要があり、またどのポートもスキップしてはなりません。
- ソートの基点でのソート済みポートの数は、ジョイン条件でのポートの数以上にすることができます。
- Decimal データ型のポートを結合する場合、各ポートの精度は同じ精度範囲に収まっている必要があります。



以下の有効な精度範囲のいずれかを使用できます。

- Decimal 0～18
- Decimal 19～28
- Decimal 29～38
- Decimal 39 以上

例えば、条件 `DecimalA = DecimalB` を定義し、`DecimalA` が精度 15 で `DecimalB` が 25 の場合、この条件は有効ではありません。

## 結合条件とソート順の例

この例では、ソート済みポートを含むマスタパイプラインと明細パイプラインを結合するジョイナトランスフォーメーションを示します。

以下のソート済みポートを使ってマスタパイプラインと明細パイプラインにソータトランスフォーメーションを設定します。

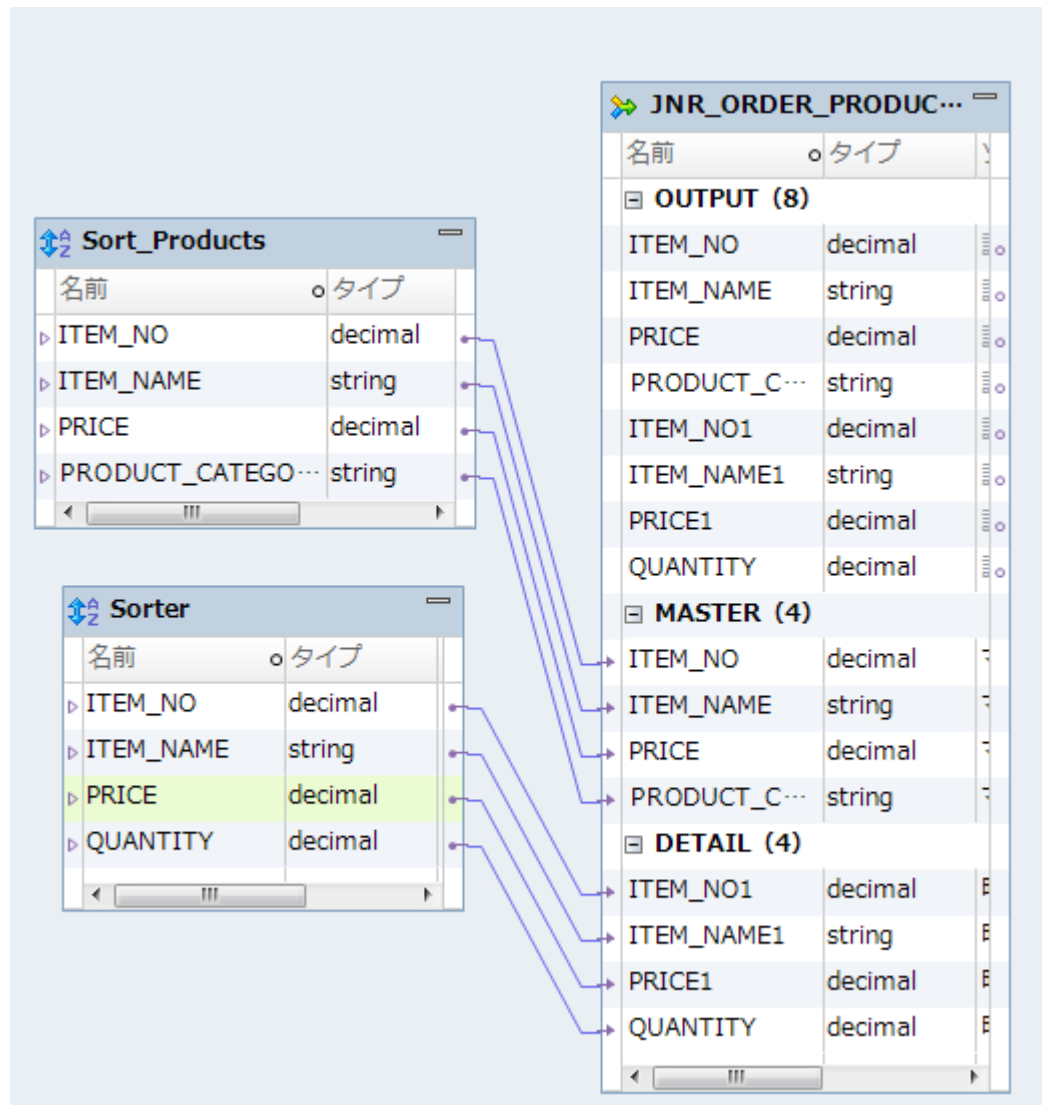
- ITEM\_NO
- ITEM\_NAME
- PRICE

ジョイン条件を設定するときには、以下のガイドラインを使ってソート順を維持します。

- 最初のジョイン条件で ITEM\_NO を使用する必要があります。
- 2 番目のジョイン条件を追加する場合は、ITEM\_NAME を使用する必要があります。
- ジョイン条件で PRICE を使用したい場合、2 番目のジョイン条件でも ITEM\_NAME を使用する必要があります。

ITEM\_NAME をスキップして ITEM\_NO と PRICE で結合すると、ソート順が失われ、データ統合サービスはマッピングの実行に失敗します。

以下の図に、ITEM\_NO、ITEM\_NAME、および PRICE の各ポートでソートおよび結合するように設定されたマッピングを示します。



Joiner トランスフォーメーションを使ってマスタパイプラインと明細パイプラインを結合する場合、次のジョイン条件のうちどれか 1 つを設定できます。

ITEM\_NO = ITEM\_NO

または

ITEM\_NO = ITEM\_NO1

ITEM\_NAME = ITEM\_NAME1

または

ITEM\_NO = ITEM\_NO1

ITEM\_NAME = ITEM\_NAME1

PRICE = PRICE1

# 同じソースのデータの結合

データの一部分に対して計算を実行し、トランスフォーメーションが実行されたデータと元のデータを結合する場合、同じソースのデータを結合できます。

同じソースのデータを結合する場合は、1つのマッピング内で元のデータを維持しながら元データの一部分にトランスフォーメーションを行うことができます。次の方法で同じソースのデータを結合できます。

- 同じパイプラインの2つのブランチを結合します。
- 同じソースの2つのインスタンスを結合します。

## 同じパイプラインの2つのブランチの結合

同じソースのデータを結合するときには、パイプラインの2つのブランチを作成します。

パイプラインをブランチに分岐する場合は、パイプラインの少なくとも片方のブランチの、マッピング入力とジョイナトランスフォーメーションの間にトランスフォーメーションを追加する必要があります。ソート済みデータを結合し、ジョイナトランスフォーメーションをソート済み入力用に設定する必要があります。

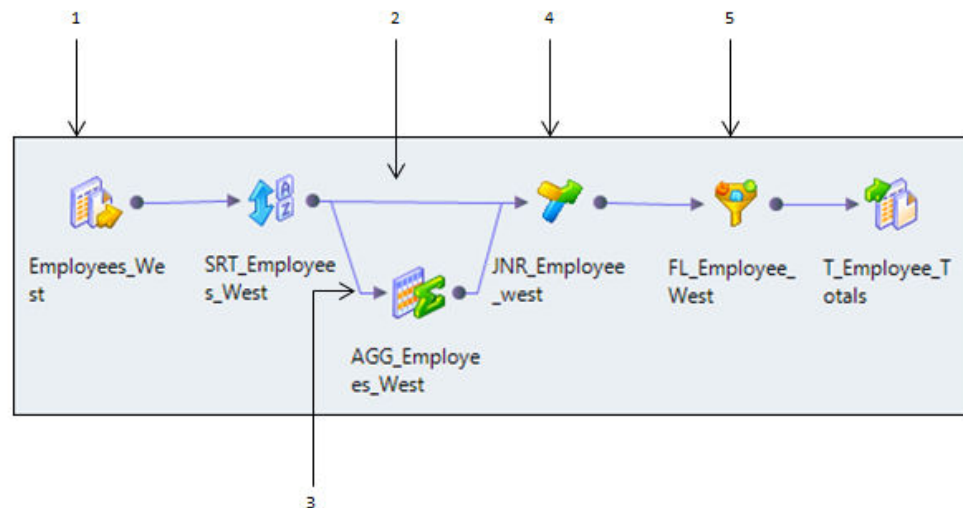
たとえば、以下のポートを含むソースがあるとします。

- Employee
- Department
- Total Sales

ターゲットに、所属する部署の平均売上を超える売上を実現した従業員を表示するとします。そのためには、以下のトランスフォーメーションを含むマッピングを作成します。

- ソートートランスフォーメーション。データをソートします。
- ソート済みアグリゲータトランスフォーメーション。売上データの平均を算出し、部署ごとにグループ化します。この集計を実行すると、個々の従業員のデータは失われます。従業員データを維持するには、パイプラインのブランチをアグリゲータトランスフォーメーションに渡し、同じデータのブランチをジョイナトランスフォーメーションに渡して元のデータを維持する必要があります。パイプラインの両方のブランチを結合すると、集計済みデータと元のデータを結合することになります。
- ソート済みジョイナトランスフォーメーション。集計されたソート済みデータを元のデータと結合します。

- フィルタトランスフォーメーション。平均売上データと各従業員の売上データを比較し、平均売上に達していない従業員をフィルタで除外します。



1. Employees\_West ソース
2. パイプラインブランチ 1
3. パイプラインブランチ 2
4. ソート済みジョイナトランスフォーメーション
5. 平均売上より低い従業員を除外する

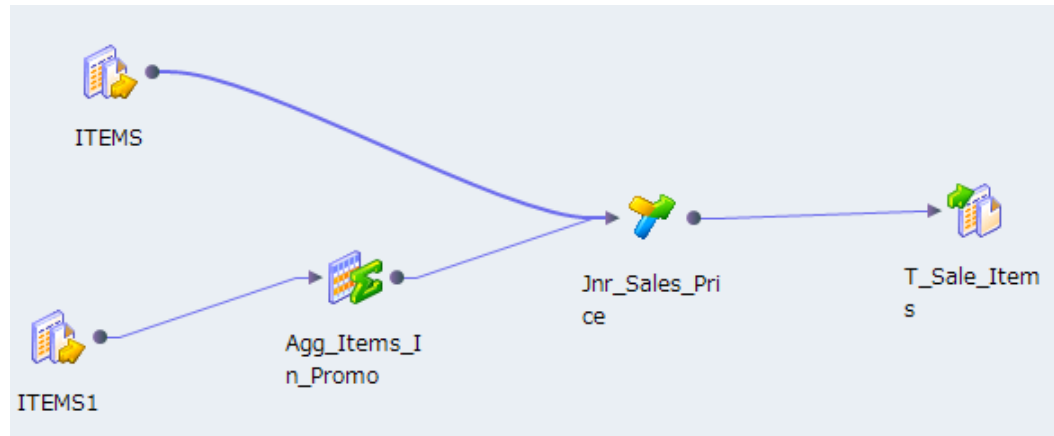
ジョイナトランスフォーメーションが一方のブランチのデータを受け取るのが、もう一方のブランチより大きく遅れる場合、2つのブランチの結合によってパフォーマンスが低下することがあります。ジョイナトランスフォーメーションは、最初のブランチのすべてのデータをキャッシュに格納し、キャッシュがいっぱいになるとキャッシュをディスクに書き込みます。次にジョイナトランスフォーメーションが2番目のブランチからデータを受け取る際に、ディスクからデータを読み込む必要があります。

## 同じソースの2つのインスタンスの結合

ソースの2つ目のインスタンスを作成することによって、同じソースのデータを結合できます。

2つ目のソースインスタンスを作成すると、2つのソースインスタンスからのパイプラインを結合できます。未ソートデータを結合したい場合は、同じソースのインスタンスを2つ作成して、パイプラインを結合する必要があります。

以下の図に、Joiner トランスフォーメーションで結合される同じソースの 2 つのインスタンスを示します。



同じソースの 2 つのインスタンスを結合すると、データ統合サービスは各ソースインスタンスに対してソースデータを読み込みます。パフォーマンスは、パイプラインの 2 つのブランチを結合する場合よりも低くなる可能性があります。

## 同じソースのデータ結合のガイドライン

パイプラインのブランチを結合するか、ソースの 2 つのインスタンスを結合するかの判断には、特定のガイドラインが適用されます。

パイプラインのブランチを結合するか、ソースの 2 つのインスタンスを結合するかを判断するときは、以下のガイドラインに従ってください。

- ソースが大きい場合、あるいはソースデータを 1 回しか読み込まない場合は、パイプラインの 2 つのブランチを結合します。
- ソート済みデータを使う場合は、パイプラインの 2 つのブランチを結合します。ソースデータが未ソートデータで、ソータートランスフォーメーションを使ってデータをソートする場合には、データのソート後にパイプラインをブランチに分岐します。
- ソースとジョイナトランスフォーメーションの間のパイプラインにブロッキングトランスフォーメーションを追加する必要がある場合は、ソースの 2 つのインスタンスを結合します。
- 一方のパイプラインの処理がもう一方のパイプラインよりも遅い場合は、ソースの 2 つのインスタンスを結合します。
- 未ソートのデータを結合する必要がある場合、ソースの 2 つのインスタンスを結合します。

## ソースパイプラインのブロック

ジョイナトランスフォーメーションを含むマッピングを実行すると、Data Integration Service はマッピングの設定とジョイナトランスフォーメーションがソート済み入力用に設定されているかどうかによって、ソースデータのブロックとブロック解除を行います。

## 未ソートジョイナトランスフォーメーション

Data Integration Service は未ソートジョイナトランスフォーメーションを処理するとき、明細行を読み込む前にマスタ行をすべて読み込みます。Data Integration Service は、マスタソースの行をキャッシュに格納する間、明細ソースをブロックします。

Data Integration Service がマスタ行をすべて読み込んでキャッシュに格納した後、明細ソースのブロックを解除して明細行を読み込みます。未ソートジョイナトランスフォーメーションを含むマッピングは、データフロー検証に違反する場合があります。

## ソート済みジョイナトランスフォーメーション

Data Integration Service は、ソート済みのジョイナトランスフォーメーションを処理するとき、マッピングの設定に基づいてデータをブロックします。ジョイナトランスフォーメーションへのマスタおよび明細入力がある異なるソースから生じている場合、ブロックロジックが可能になります。

ターゲットロード順グループのすべてのソースを同時にブロックせずにジョイナトランスフォーメーションを処理できる場合、Data Integration Service はブロックロジックを使用してジョイナトランスフォーメーションを処理します。それ以外の場合、ブロックロジックは使用しません。その代わりに、より多くの行をキャッシュに格納します。

ブロックロジックを使用してジョイナトランスフォーメーションを処理できる場合、Data Integration Service がキャッシュに格納する行は少なくなり、パフォーマンスが向上します。

### マスタ行のキャッシュ

Data Integration Service は、ジョイナトランスフォーメーションを処理する場合、両方のソースから同時に行を読み込み、マスタ行に基づいてインデックスキャッシュおよびデータキャッシュを作成します。

Data Integration Service は次に、明細ソースデータとキャッシュデータに基づいて結合を実行します。Data Integration Service がキャッシュに格納する行数は、ソースデータ、およびジョイナトランスフォーメーションがソート済み入力用に設定されているかどうかによって異なります。

未ソートジョイナトランスフォーメーションのパフォーマンスを向上させるには、行の比較的小さいソースをマスタソースとして使用します。ソート済みジョイナトランスフォーメーションのパフォーマンスを向上させるには、重複キー値の少ないソースをマスタとして使用します。

## ジョイナトランスフォーメーションのパフォーマンスのヒント

ジョイナトランスフォーメーションのパフォーマンスを向上させるためのヒントを紹介します。

ジョイナトランスフォーメーションでは、中間結果を格納するための追加領域を実行時に必要とするため、パフォーマンスが低下することがあります。Joiner トランスフォーメーションを最適化するためには、Joiner パフォーマンスカウンタの情報を表示します。

以下のヒントを使用して、ジョイナトランスフォーメーションのパフォーマンスを向上させることができます。

**重複キー値が少ない方のソースをマスタとして指定します。**

データ統合サービスは、ソート済みジョイナトランスフォーメーションを処理するとき、一度に 100 個の一意的なキーの行をキャッシュに格納します。マスタソースに同じキー値を持つ多数の行が含まれる場合、データ統合サービスはより多くの行をキャッシュに格納する必要があり、それによってパフォーマンスが低下することがあります。

**行数が少ない方のソースをマスタとして指定します。**

ジョイナトランスフォーメーションは詳細ソースの各行をマスタソースと比較します。マスタ内の行が少なければ、結合のための比較が繰り返される回数も少なくなり、その結果、結合プロセスが高速になります。

**可能な場合は、データベース内で結合を実行します。**

データベース内で結合を実行すると、マッピングの実行中に実行する場合よりも処理が高速になります。パフォーマンスは、使用するデータベース結合の種類によっても変わってきます。ノーマル結合は、外部結合よりも高速で、結果的にレコード数が少なくて済みます。場合によっては、例えば2つの異なるデータベースまたはフラットファイルシステムとテーブルを結合する場合は、これが不可能なこともあります。

**可能な場合は、ソート済みデータを結合します。**

ソート済み入力を使用するようにジョイナトランスフォーメーションを設定してください。データ統合サービスは、ディスクの入出力を最小化することによってパフォーマンスを向上させます。パフォーマンスは、大量のデータセットを扱う場合に最大限に向上させることができます。未ソートジョイナトランスフォーメーションの場合、行の比較的少ないソースをマスタソースとして指定します。

**結合条件を最適化する。**

データ統合サービスは、小さい方のグループから行を読み取り、大きい方のグループで一致する行を見つけ結合操作を実行することで、1つの結合オペランドのデータセットのサイズを小さくしようとします。データセットのサイズを小さくすると、データ統合サービスで大きい方のグループソースから不要な行が読み取られなくなるため、マッピングのパフォーマンスが向上します。データ統合サービスによって、結合条件が大きい方のグループソースに移動され、小さい方のグループと一致する行のみが読み取られます。

**準結合最適化方式を使用する。**

一方の入力グループに他方よりも多くの行が含まれており、結合条件に基づいて、小さい方のグループに一致するものがない行が大きい方のグループに多数含まれている場合は、準結合最適化方式を使用するとマッピングのパフォーマンスが向上します。

## ジョイナトランスフォーメーションのルールとガイドライン

ジョイナトランスフォーメーションを使用するときは、特定のルールとガイドラインが適用されます。

ジョイナトランスフォーメーションは、ほとんどのトランスフォーメーションからの入力を受け付けます。ただし、いずれかの入力パイプラインにアップデートストラテジトランスフォーメーションが含まれている場合、ジョイナトランスフォーメーションは使用できません。

## 非ネイティブ環境でのジョイナトランスフォーメーション

非ネイティブ環境でのジョイナトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。

- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

## Blaze エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 不等な結合とマップ側の結合を含むトランスフォーメーションが無効。
- ジョイナトランスフォーメーションの式が、未接続のルックアップトランスフォーメーションを参照する。

ジョイナトランスフォーメーションが、詳細外部結合または完全な外部結合用に設定されると、マップ側の結合は無効になります。

## Spark エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 結合条件がバイナリデータ型、またはバイナリ式が含まれる。

### ストリーミングマッピングでのジョイナトランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

#### マッピングの検査

マッピング検証は、次の場合に失敗します。

- ジョイナトランスフォーメーションがアグリゲータトランスフォーメーションからのダウンストリームである。
- ジョイナトランスフォーメーションがランクトランスフォーメーションからのダウンストリームである。
- ストリーミングパイプラインに複数のジョイナトランスフォーメーションが含まれている。
- ジョイナトランスフォーメーションが、ストリーミングパイプラインからのデータと非ストリーミングパイプラインからのデータを結合する。

#### 一般的なガイドライン

結合条件を指定するには、[結合] タブの詳細条件タイプから TIME\_RANGE 関数を選択して、結合条件式を入力します。TIME\_RANGE 関数は、結合するストリーミングイベントの時間範囲を決定します。

## Databricks Spark エンジンでのジョイナトランスフォーメーション

マッピング検証は、次の場合に失敗します。

- 結合条件がバイナリデータ型、またはバイナリ式が含まれる。



## 第 23 章

# キージェネレータトランスフォーメーション

この章では、以下の項目について説明します。

- [キージェネレータトランスフォーメーションの概要, 373 ページ](#)
- [Soundex ストラテジ, 374 ページ](#)
- [文字列ストラテジ, 374 ページ](#)
- [NYSIIS ストラテジ, 375 ページ](#)
- [キージェネレータの出力ポート, 375 ページ](#)
- [グループ化ストラテジの設定, 376 ページ](#)
- [キー作成のプロパティ, 376 ページ](#)
- [キージェネレータトランスフォーメーションの詳細プロパティ, 377 ページ](#)
- [非ネイティブ環境でのキージェネレータトランスフォーメーション, 377 ページ](#)

## キージェネレータトランスフォーメーションの概要

キージェネレータトランスフォーメーションは、選択されたカラムのデータ値に基づいてレコードをグループに整理するアクティブなトランスフォーメーションです。このトランスフォーメーションを使用して、一致トランスフォーメーションに渡す前にレコードをソートします。

キージェネレータトランスフォーメーションでは、選択されたカラムのグループキーの作成にグループ化ストラテジが使用されます。ストラテジとしては、文字列、Soundex、および NYSIIS があります。選択されたフィールドの値が同じレコードには、同じグループキーが割り当てられます。一致トランスフォーメーションは、グループキー値が同じレコードをまとめて処理します。その結果、一致トランスフォーメーション内の重複分析がスピードアップします。

一致トランスフォーメーションが実行する必要がある比較処理の数は、データセット内のレコード数に応じて急激に増加します。この急激な増加により、大量のコンピューティングリソースが消費される可能性があります。キージェネレータトランスフォーメーションは、グループキーを作成することによって、一致トランスフォーメーションがより小さいグループでレコードを比較できるようにし、その結果、処理時間が短縮されます。

フィールド一致を実行する場合は、グループキー生成のカラムに、一致ニーズに対して有用なグループを提供する可能性の高いカラムを選択します。たとえば、姓のカラムは、名のカラムよりも有用なグループキーデータを提供する可能性があります。ただし、一致トランスフォーメーションの重複分析では、姓カラムを使用しないでください。

キージェネレータトランスフォーメーションでは、各レコードに一意の ID を作成することもできます。一致トランスフォーメーションに渡す各レコードには、一意の ID が含まれている必要があります。データに ID が存在しない場合は、キージェネレータトランスフォーメーションを使用して作成します。

# Soundex ストラテジ

Soundex ストラテジは、単語を分析し、単語の発音を表すコードからグループキーを作成します。

Soundex コードは単語の 1 文字目で始まり、後に続く子音を表す一連の数字が続きます。Soundex ストラテジを使用して、音が似た単語に同じコードを割り当てます。ストラテジが返す英数字の数を定義するには、Soundex 深度を設定します。

このストラテジは、単語のスペルではなく音に焦点を当て、代替のスペルやスペルの小さな差異をグループ化することができます。例えば、Smyth と Smith の Soundex コードは同じです。

Soundex ストラテジは、発音の間違った単語をグループ化することもできます。例えば、名前 Edmonton と Edmonson の Soundex コードは同じです。

## Soundex ストラテジのプロパティ

キージェネレータトランスフォーメーションでグループキーの作成に使用される Soundex 設定を決定するには、Soundex ストラテジのプロパティを設定します。

以下の表に、Soundex ストラテジのプロパティを示します。

| プロパティ      | 説明                                                                                                             |
|------------|----------------------------------------------------------------------------------------------------------------|
| Soundex 深度 | Soundex ストラテジによって返される英数字の数を決定します。デフォルトの深度は 3 です。この深度では、文字列の最初の文字と次の 2 つの子音を表す 2 つの数字で構成される Soundex コードが作成されます。 |

関連項目：

- [「文字列ストラテジのプロパティ」 \(ページ 375\)](#)
- [「キー作成のプロパティ」 \(ページ 376\)](#)
- [「グループ化ストラテジの設定」 \(ページ 376\)](#)

# 文字列ストラテジ

文字列ストラテジは、入力データのサブ文字列からグループキーを作成します。

入力カラム内のサブ文字列の長さや場所を指定できます。例えば、入力文字列の最初の 4 文字からキーを作成するようにこのストラテジを設定することができます。

## 文字列ストラテジのプロパティ

キージェネレータ変換でグループキーの作成に使用されるサブ文字列を決定するには、文字列ストラテジのプロパティを設定します。

以下の表に、文字列ストラテジのプロパティを示します。

| プロパティ | 説明                                                                              |
|-------|---------------------------------------------------------------------------------|
| 左から開始 | 左から右に入力フィールドを読み取るように変換を設定します。                                                   |
| 右から開始 | 右から左に入力フィールドを読み取るように変換を設定します。                                                   |
| 開始位置  | スキップする文字数を指定します。例えば、 <b>【開始位置】</b> に3と入力した場合、サブ文字列は入力フィールドの指定された側から4文字目から始まります。 |
| 長さ    | グループキーとして使用する文字列の長さを指定します。入力フィールドすべてを使用するには、0と入力します。                            |

関連項目：

- [「Soundex ストラテジのプロパティ」 \(ページ 374\)](#)
- [「キー作成のプロパティ」 \(ページ 376\)](#)
- [「グループ化ストラテジの設定」 \(ページ 376\)](#)

## NYSIIS ストラテジ

NYSIIS ストラテジは、単語を分析し、単語の発音を表す文字からグループキーを作成します。

Soundex ストラテジが文字列の最初の母音のみを考慮するのに対し、NYSIIS ストラテジは文字列全体の母音を分析します。NYSIIS ストラテジは、すべての文字を6つの文字のいずれかに変換し、ほとんどの母音を文字Aに変換します。

## キージェネレータの出力ポート

キージェネレータ変換の出力ポートは、一致変換がレコードの処理に使用するIDとグループキーを作成します。

以下の表に、キージェネレータ変換の出力ポートを示します。

| プロパティ      | 説明                              |
|------------|---------------------------------|
| SequenceID | ソースデータセット内の各レコードを特定するIDを作成します。  |
| GroupKey   | 一致変換でレコードの処理に使用されるグループキーを作成します。 |

再利用可能なキージェネレータ変換を作成するときは、**【概要】**ビューを使ってポートを表示します。再利用できない変換をマッピングに追加するときは、**【プロパティ】**ビューの**【ポート】**タブを使ってポートを表示します。

# グループ化ストラテジの設定

グループ化ストラテジを設定するには、**【ストラテジ】** ビューでプロパティを編集します。

キージェネレータストラテジを設定する前に、キージェネレータトランスフォーメーションに入力ポートを追加します。

1. **【ストラテジ】** ビューを選択します。
2. **【新規】** ボタンをクリックします。
3. グループ化ストラテジを選択します。
4. **【OK】** をクリックします。
5. **【入力】** カラムで、入力ポートを選択します。
6. プロパティフィールドで選択矢印をクリックして、ストラテジのプロパティを設定します。
7. キー作成のプロパティを設定します。

関連項目：

- [「Soundex ストラテジのプロパティ」 \(ページ 374\)](#)
- [「文字列ストラテジのプロパティ」 \(ページ 375\)](#)
- [「キー作成のプロパティ」 \(ページ 376\)](#)

## キー作成のプロパティ

分析するデータに適したキー作成プロパティを設定します。

以下の表にキー作成のプロパティを示します。

| プロパティ              | 説明                                                                                                                                                        |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| ソート結果              | [GroupKey] フィールドを使用して、キージェネレータトランスフォーメーションの出力をソートします。フィールド一致操作では、このオプションを選択するか、一致トランスフォーメーションにソート済みデータを提供していることを確認する必要があります。このオプションは ID 一致操作では選択しないでください。 |
| シーケンスキーを自動的に生成     | 入力データの順序を使用してシーケンスキーフィールドを生成します。                                                                                                                          |
| フィールドをシーケンスキーとして使用 | 指定されたカラムに対してシーケンスフィールドを生成します。                                                                                                                             |
| シーケンスキーフィールド       | シーケンスキーフィールドの名前を指定します。                                                                                                                                    |

関連項目：

- [「Soundex ストラテジのプロパティ」 \(ページ 374\)](#)
- [「文字列ストラテジのプロパティ」 \(ページ 375\)](#)
- [「グループ化ストラテジの設定」 \(ページ 376\)](#)

# キージェネレータトランスフォーメーションの詳細プロパティ

キージェネレータトランスフォーメーションには、キャッシュメモリの動作とトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

## キャッシュファイルディレクトリ

データ統合サービスが現在のトランスフォーメーションの一時データを書き込むディレクトリを指定します。入力データの量が利用可能なシステムメモリより大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

## キャッシュファイルサイズ

トランスフォーメーションの入力データをソートするためにデータ統合サービスが使用するシステムメモリの量を決定します。このパラメータを使用して、キャッシュファイルサイズを指定できます。

データをソートする前に、データ統合サービスは指定されているメモリ量を割り当てます。ソート操作によってこれを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュディレクトリに書き込みます。ソート操作にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

キャッシュファイルサイズを指定しない場合、トランスフォーメーションはデータ統合サービスの実行オプションの最大メモリ値を適用します。

**注:** 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

# 非ネイティブ環境でのキージェネレータトランスフォーメーション

非ネイティブ環境でのキージェネレータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## 第 24 章

# ラベラトランスフォーメーション

この章では、以下の項目について説明します。

- [ラベラトランスフォーメーションの概要, 378 ページ](#)
- [ラベラトランスフォーメーションを使用する状況, 379 ページ](#)
- [ラベラトランスフォーメーションでの参照データの使用, 380 ページ](#)
- [ラベラトランスフォーメーションのストラテジ, 382 ページ](#)
- [ラベラトランスフォーメーションのポート, 382 ページ](#)
- [文字ラベル適用のプロパティ, 383 ページ](#)
- [トークンラベル適用のプロパティ, 385 ページ](#)
- [文字ラベル適用ストラテジの設定, 388 ページ](#)
- [トークンラベル適用ストラテジの設定, 388 ページ](#)
- [ラベラトランスフォーメーションの詳細プロパティ, 389 ページ](#)
- [非ネイティブ環境でのラベラトランスフォーメーション, 389 ページ](#)

## ラベラトランスフォーメーションの概要

ラベラトランスフォーメーションは、入力ポートフィールドを分析し、各フィールドのデータを説明するテキストラベルを書き込む、パッシブなトランスフォーメーションです。

ラベラトランスフォーメーションは、ポートに含まれる情報のタイプを把握するときに使用します。ポート上の情報のタイプがわからないときや、ポート上で必要な情報のタイプが含まれていないレコードを特定するときに、ラベラトランスフォーメーションを使用します。

ラベルは、入力文字列を説明する、1 つ以上の文字から成る文字列です。各文字列に含まれるデータに基づいて入力文字列にラベルを割り当てるように、ラベラトランスフォーメーションを設定します。

トランスフォーメーションを設定するときは、検索する文字または文字列のタイプを指定し、関連する文字または文字列を検索するときに出力として書き込まれるラベルを指定します。ラベル適用操作を設定するときは、検索する文字または文字列のタイプ、および使用するラベルを入力します。あるいは、参照データオブジェクトを使用して、文字、文字列、およびラベルを指定します。

文字ラベル適用またはトークンラベル適用を実行するように、ラベラトランスフォーメーションを設定します。

### 文字ラベル適用

句読点やスペースを含めて、入力文字列の文字構造を説明するラベルを書き込みます。カラム内の行ごとにラベルが 1 つ書き込まれます。例えば、ラベラトランスフォーメーションで郵便番号 10028 に "nnnnn" ("n" は数字) というラベルを適用できます。

## トークンラベル適用

入力文字列の情報のタイプを説明するラベルを書き込みます。入力データで特定された各トークンのラベルが書き込まれます。例えば、文字列"John J. Smith"にトークン"Word Init Word"を使用してラベルを適用するようにラベラトランスフォーメーションを設定できます。

トークンは入力文字列内の区切られた値です。

ラベラは、指定されたラベルに一致する文字または文字列を検出すると、ラベル名を新しい出力ポートに書き込みます。

ラベラトランスフォーメーションでは、文字とトークンの特定に参照データが使用されます。参照データオブジェクトは、ラベラストラテジで操作を設定するときに選択します。

# ラベラトランスフォーメーションを使用する状況

ラベラトランスフォーメーションは、ポートの各値を説明するラベルを書き込みます。

以下の例に、ラベラトランスフォーメーションで実行できるいくつかのタイプの分析を示します。

## 連絡先データが含まれるレコードの検索

名のリストが含まれる参照テーブルのあるトランスフォーメーションを設定します。トークンラベル適用ストラテジを作成して、参照テーブル内の値に一致するすべての文字列にラベルを適用します。出力データを確認するとき、ラベルが含まれるレコードは人物を特定できる可能性が高くなります。

## 企業レコードの検索

Inc、Corp、Ltdなどの企業の接尾辞のリストが含まれるトークンセットのあるトランスフォーメーションを設定します。トークンラベル適用ストラテジを作成して、参照テーブル内の値に一致するすべての文字列にラベルを適用します。出力データを確認するとき、ラベルが含まれるレコードは企業を特定できる可能性が高くなります。

**注:** 企業名を特定するには、目的の企業の接尾辞のトークンセットを使用します。特定するすべての企業がテーブルに含まれていることが確かである場合は、企業名の参照テーブルを使用できます。例えば、ニューヨーク証券取引所の企業を一覧表示する参照テーブルを使用できます。

## 電話番号データの検索

電話番号の文字構造を定義する文字セットのあるトランスフォーメーションを設定します。例えば、米国の電話番号のさまざまなパターンの句読記号と数字を認識する文字セットを使用できます。このデータを確認して、電話番号として正しい数字が含まれないレコードを検索することができます。

カラムデータを分析するために、文字ラベルでは次の文字を使用できます。

c=punctuation character n=digit s=space

以下の表に、電話番号の構造の例を示します。

| 文字構造                   | 電話番号           |
|------------------------|----------------|
| cnnncsnnnnnnnnnnnnnnnn | (212) 555-1212 |
| nnnnnnnnnnnnnnnn       | 2125551212     |
| cnnncnnnnnnnnnnnnnnnn  | +212-555-1212  |

# ラベラトランスフォーメーションでの参照データの使用

Informatica の Developer ツールでは、インストール時に、ラベラトランスフォーメーションで利用できるさまざまなタイプの参照データオブジェクトもインストールされます。参照データオブジェクトを作成することもできます。

参照データオブジェクトをラベラトランスフォーメーションのストラテジに追加すると、このトランスフォーメーションがストラテジの入力データから参照データオブジェクトを検索します。このトランスフォーメーションでは、検索されたすべての値を参照データオブジェクトからの有効な値に、または指定した値に置き換えます。

次の表に、使用できる参照データのタイプを示します。

| 参照データのタイプ | 説明                                                                                                                                                    |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 文字セット     | 文字、数字、句読記号など、さまざまなタイプの文字を特定します。<br>文字のラベル適用操作を使用します。                                                                                                  |
| 確率モデル     | トークンラベル適用操作にあいまい一致機能を追加します。パーサートランスフォーメーションは、確率モデルを使用して、文字列内の情報のタイプを推測することができます。あいまい一致機能を有効にするには、Developer ツールで確率モデルをコンパイルします。<br>トークンのラベル適用操作を使用します。 |
| 参照テーブル    | データベーステーブル内のエントリに一致する文字列を検索します。<br>トークンのラベル適用操作と文字のラベル適用操作を使用します。                                                                                     |
| 正規表現      | 定義した条件に一致する文字列を特定します。正規表現を使用して、大きな文字列内の文字列を検索することができます。<br>トークンのラベル適用操作を使用します。                                                                        |
| トークンセット   | 文字列に含まれる情報のタイプに基づいて、文字列を特定します。<br>トークンのラベル適用操作を使用します。<br>Informatica は、単語、電話番号、郵便番号、製品コードの定義など、トークンセットのさまざまなタイプのトークン定義とともにインストールされます。                 |

## 文字セット

文字セットには、特定の文字および文字範囲を識別する式が含まれます。文字セットは、ラベラトランスフォーメーション、およびトークン解析モードを使用するパーサートランスフォーメーションで使用できます。

文字範囲は、連続する文字コードの範囲を指定します。例えば、文字範囲 "[A-C]" は大文字の "A"、"B"、および "C" に一致します。この文字範囲は、小文字の "a"、"b"、または "c" には一致しません。

文字セットを使用すると、トークン解析操作やラベル適用操作の一部として特定の文字または文字範囲を識別できます。例えば、電話番号が格納されたカラムのすべての数字にラベルを適用することができます。数字にラベルを適用した後、パーサートランスフォーメーションでパターンを識別し、問題のあるパターンを別の出力に書き込むことができます。



## 確率モデル

確率モデルは、トークンに含まれている情報のタイプと入力文字列内のトークンの位置によってトークンを特定します。

確率モデルには、参照データ値とラベル値が含まれます。参照データ値は、トランスフォーメーションに接続する入力ポートのデータを表します。ラベル値は、参照データ値に含まれる情報のタイプを説明します。モデルの各参照データ値にラベルを割り当てます。

確率モデルのラベルに参照データ値をリンクするには、モデルをコンパイルします。コンパイル処理では、データ値とラベル間の、一連の論理的な関連付けが生成されます。モデルを読み取るマッピングを実行すると、データ統合サービスはモデルのロジックをトランスフォーメーション入力データに適用します。データ統合サービスは、入力データ値を最も正確に描写しているラベルを返します。

Developer tool で確率モデルを作成します。モデルリポジトリは、確率モデルオブジェクトを格納します。Developer ツールは、データ値、ラベル、およびコンパイルデータを、Informatica ディレクトリ構造内のファイルに書き込みます。

## 参照テーブル

参照テーブルは、少なくとも 2 つのカラムが含まれるデータベーステーブルです。一方のカラムにはデータ値の標準バージョンまたは必要なバージョンが含まれ、もう一方のカラムにはデータ値の代替バージョンが含まれます。参照テーブルをトランスフォーメーションに追加すると、テーブルに存在する値が入力ポートデータで検索されます。作業するデータプロジェクトに役立つデータを含むテーブルを作成できます。

## 正規表現

ラベル適用操作において、正規表現とは、入力データに含まれる特定の文字列の識別に使用できる式を指します。正規表現は、トークンラベル適用モードを使用するラベラトランスフォーメーションで使用できます。

ラベラトランスフォーメーションでは、正規表現を使用して入力パターンを一致させ、1 つのラベルを作成します。正規表現の出力が複数になる場合、複数のラベルは生成されません。

## トークンセット

トークンセットには、特定のトークンを識別する式が含まれます。トークンセットは、トークンラベル適用モードを使用するラベラトランスフォーメーションで使用できます。

トークンセットを使用すると、トークンラベル適用操作の一部として特定のトークンを識別できます。例えば、トークンセットを使用して、"AccountName@DomainName" という形式のすべての電子メールアドレスにラベルを適用できます。トークンにラベルを適用したら、パーサトランスフォーメーションを使用して、指定した出力ポートに電子メールアドレスを書き込むことができます。

Developer ツールには、さまざまなパターンを識別するために使用できるシステム定義のトークンセットが用意されています。システム定義のトークンセットの例をいくつか示します。

- 単語
- 数字
- 電話番号
- 電子メールアドレス
- 郵便番号
- 国の識別番号（社会保障番号など）
- クレジットカード番号

# ラベラトランスフォーメーションのストラテジ

入力データにラベルを割り当てるには、ラベル適用ストラテジを使用します。ラベル適用ストラテジを設定するには、ラベラトランスフォーメーションの【ストラテジ】ビューで設定を編集します。

ラベル適用ストラテジを作成するときは、操作を 1 つ以上追加し、操作ごとに特定のラベル適用タスクを実装します。

ラベラトランスフォーメーションには、ストラテジを作成するためのウィザードが用意されています。ラベル適用ストラテジを作成するときは、文字ラベル適用またはトークンラベル適用のいずれかのモードを選択します。その後、そのラベル適用モードに固有の操作を追加します。

**重要:** 操作やストラテジの順序は変更が可能です。各操作で前の操作の結果を読み取るため、ストラテジ内の操作の順序によってストラテジの出力が変わることがあります。

## 文字ラベル適用操作

文字ラベル適用操作は、データの文字パターンを示すラベルを作成する場合に使用します。

文字ラベル適用ストラテジには、以下のタイプの操作を追加できます。

### 文字セットを使用して文字にラベルを適用する

定義済みの文字セット（数字や英文字など）を使用して文字にラベルを適用します。Unicode 文字セットと Unicode 以外の文字セットを選択できます。

### 参照テーブルを使用して文字にラベルを適用する

参照テーブルにあるカスタムラベルを使用して文字にラベルを適用します。

## トークンラベル適用操作

トークンラベル適用操作は、データの文字列を示すラベルを作成する場合に使用します。

ラベラトランスフォーメーションでは、入力文字列に含まれる複数のトークンを識別してラベルを適用することができます。例えば、米国の電話番号と電子メールアドレスのトークンセットを使用するようにラベラトランスフォーメーションを設定できます。このラベラトランスフォーメーションで入力文字列 "555-555-1212 someone@somewhere.com" を処理すると、出力文字列は "USPHONE EMAIL" になります。

ラベル適用ストラテジに追加できるトークンラベル適用操作のタイプを次に示します。

### 参照テーブルのラベル

参照テーブルのエントリに一致する文字列にラベルを適用します。

### トークンセットのラベルトークン

トークンセットデータまたは確率モデルデータに一致するラベル文字列パターン。

# ラベラトランスフォーメーションのポート

トランスフォーメーション内で設定するラベル適用操作に必要な入力ポートと出力ポートを選択します。

ラベラトランスフォーメーションでは、以下のポートを使用します。

### 入力ポート

アップストリームオブジェクトから文字列の入力を読み取ります。

### ラベル適用後の出力ポート

トランスフォーメーションの操作によって定義されたラベルを書き込みます。

### トークン化された出力ポート

出力の各ラベルに対応する入力文字列を渡します。このポートは、マプレットまたはマッピング内でラベラトランスフォーメーションのパーサートランスフォーメーションのダウンストリームを追加する場合にこのポートを選択し、パターンベースの解析モードで実行されるようにパーサートランスフォーメーションを設定します。パーサートランスフォーメーションは、トークンラベル適用の出力を、トークン化された出力ポートのデータと関連付けます。

### スコア出力ポート

トークンラベル適用操作の確率的な一致手法によって生成されるスコア値を書き込む場合に選択します。

確率モデルが使用されるトークンラベル適用操作を実行すると、ラベルが適用された文字列ごとに数値スコアが生成されます。このスコアは、入力文字列と確率モデルで定義されたパターンの間の類似度を表します。

## 文字ラベル適用のプロパティ

文字ラベル適用操作のプロパティは、ラベラトランスフォーメーションの【ストラテジ】ビューで設定します。

### 全般プロパティ

全般プロパティは、ストラテジで定義するすべての文字ラベル適用操作に適用されます。ストラテジに名前を付けたり、入出力ポートを指定したりするには、全般プロパティを使用します。

次の表は、全般プロパティの説明です。

| プロパティ | 説明                               |
|-------|----------------------------------|
| 名前    | ストラテジの名前を入力します。                  |
| 入力    | ストラテジ操作で読み取ることができる入力ポートを特定します。   |
| 出力    | ストラテジ操作で書き込むことができる出力ポートを特定します。   |
| 説明    | ストラテジの説明を入力します。これはオプションのプロパティです。 |

### 参照テーブルのプロパティ

文字ラベル適用ストラテジを定義するときは、文字セットと参照テーブルを使用してラベルに演算子を追加できます。参照テーブルのプロパティを使用して、参照テーブルの使用方法を指定します。

以下の表に、参照テーブルのプロパティを示します。

| プロパティ  | 説明                           |
|--------|------------------------------|
| 名前     | 操作の名前を入力します。                 |
| 参照テーブル | 文字へのラベルの適用に使用する参照テーブルを指定します。 |

| プロパティ                   | 説明                                       |
|-------------------------|------------------------------------------|
| ラベル                     | 参照テーブルのエントリに一致する入力文字の置換テキストを指定します。       |
| ストラテジ内の他のラベルをオーバーライドします | このラベル適用操作で他のラベル適用操作をオーバーライドするかどうかを指定します。 |

## 文字セットのプロパティ

文字ラベル適用ストラテジを定義するときは、文字セットと参照テーブルを使用してラベルに演算子を追加できます。文字セットのプロパティを使用して、文字セットの使用方法を指定します。

以下の表に、文字セットのプロパティを示します。

| プロパティ    | 説明                                                                                                                 |
|----------|--------------------------------------------------------------------------------------------------------------------|
| 名前       | 操作の名前を入力します。                                                                                                       |
| 文字セットの選択 | 文字列へのラベルの適用に使用する文字セットを指定します。<br>文字セットに一致する入力文字列の置換テキストをオーバーライドできます。 <b>[ラベル]</b> カラムで選択矢印をクリックして、カスタム置換テキストを入力します。 |
| フィルタテキスト | 入力する文字またはワイルドカードを使用して、文字セットのリストをフィルタリングします。                                                                        |
| 文字セットの追加 | カスタム文字セットを定義する場合に選択します。                                                                                            |
| 編集       | カスタム文字セットの内容を編集します。                                                                                                |
| インポート    | コンテンツセットに格納されている文字セットの再利用不可能なコピーを作成できます。元の文字セットを変更しても、ラベラトランスフォーメーションに格納したコピーには反映されません。                            |
| 削除       | カスタム文字セットを削除します。                                                                                                   |
| 実行順序の指定  | トークンセットをデータに適用する順序を設定します。上下の矢印を使用して順序を変更できます。                                                                      |

## フィルタのプロパティ

ラベル適用操作中にスキップする値を指定できます。ラベル適用操作を適用しない値を指定するには、**[テキストを無視する]** プロパティを使用します。

以下の表に、フィルタのプロパティを示します。

| プロパティ     | 説明                                                                 |
|-----------|--------------------------------------------------------------------|
| 検索用語      | ラベル適用を実行する前にフィルタする文字列を指定します。この機能を使用して、定義したラベル適用ストラテジに対する例外を指定できます。 |
| 大文字小文字の区別 | フィルタする文字列を検索語句と照合するときに大文字と小文字を区別するかどうかを指定します。                      |

| プロパティ | 説明                            |
|-------|-------------------------------|
| 大文字   | フィルタする文字列を大文字に変換します。          |
| 開始    | フィルタする文字列の検索を開始する文字の位置を指定します。 |
| 終了    | フィルタする文字列の検索を終了する文字の位置を指定します。 |

## トークンラベル適用のプロパティ

トークンラベル適用操作のプロパティは、ラベラトランスフォーメーションの【ストラテジ】ビューで設定します。

### 全般プロパティ

全般プロパティは、ストラテジで定義するすべてのトークンラベル適用操作に適用されます。ストラテジに名前を付けたり、入出力ポートを指定したり、ストラテジで確率的な一致方法を有効にするかどうかを指定するには、全般プロパティを使用します。

次の表は、全般プロパティの説明です。

| プロパティ           | 説明                                                                                          |
|-----------------|---------------------------------------------------------------------------------------------|
| 名前              | ストラテジの名前を入力します。                                                                             |
| 入力              | ストラテジ操作で読み取ることができる出力ポートを識別します。                                                              |
| 出力              | ストラテジ操作で書き込むことができる出力ポートを識別します。                                                              |
| 説明              | ストラテジを説明します。このプロパティはオプションです。                                                                |
| 確率的な一致方法を使用     | ストラテジが確率モデルを使用してトークンのタイプを識別できるように指定します。                                                     |
| 反転有効            | 右から左に入力データを読み取るようにストラテジを設定します。このプロパティは、確率的な一致に対して無効になっています。                                 |
| 区切り文字           | 入力データ内のサブ文字列を評価するためにトランスフォーメーションが使用する文字を指定します。デフォルトはスペースです。<br>このプロパティは確率的なラベル適用で無効になっています。 |
| トークン化された出力フィールド | 複数のラベルを出力ポートに書き込むようにストラテジを設定します。パーサートランスフォーメーションでパターンベースの解析の入力データを作成するには、このフィールドを選択します。     |
| スコア出力フィールド      | 確率的な一致で生成されるスコア値が含まれるフィールドを特定します。スコア出力フィールドは、確率的な一致方法を使用するオプションを選択するときに設定します。               |
| 出力の区切り文字        | 出力ポートのデータ値を区切る文字を指定します。デフォルトはコロンです。                                                         |

## トークンセットのプロパティ

トークンセットプロパティは、トークンセットを使用するためにラベル適用操作を設定するときに適用されません。

次の表は、全般プロパティの説明です。

| プロパティ      | 説明                                                                                                              |
|------------|-----------------------------------------------------------------------------------------------------------------|
| トークンセットの選択 | トランスフォーメーションが文字列にラベルを付けるために使用するトークンセットを指定します。                                                                   |
| フィルタテキスト   | トークンセットまたは正規表現のリストをフィルタリングします。フィルタとしてテキスト文字とワイルドカード文字を使用します。                                                    |
| トークンセットの追加 | カスタムのトークンセットを定義するために使用します。                                                                                      |
| 正規表現の追加    | 入力パターンに一致する正規表現を定義するために使用します。                                                                                   |
| 編集         | カスタムのトークンセットまたは正規表現の内容を編集します。                                                                                   |
| インポート      | モデルリポジトリのフォルダーからトークンセットまたは正規表現の再利用不可能なコピーをインポートします。トークンセットまたは正規表現のソースオブジェクトが更新されても、データ統合サービスは再利用不可能なコピーを更新しません。 |
| 削除         | カスタムのトークンセットまたは正規表現を削除します。                                                                                      |
| 実行順序の指定    | トークンセットまたは正規表現をデータに適用する順序を設定します。上下の矢印を使用して順序を変更できます。                                                            |

## カスタムラベルプロパティ

トークンラベル適用操作を設定するときは、**【カスタムラベル】** ビューを選択して、特定の検索語句に対するラベルを作成できます。

以下の表に、カスタムラベルプロパティを示します。

| プロパティ     | 説明                                  |
|-----------|-------------------------------------|
| 検索用語      | 検索する文字列を特定します。                      |
| 大文字小文字の区別 | 検索語句と照合するときに大文字と小文字を区別するかどうかを指定します。 |
| カスタムラベル   | 適用するカスタムラベルを特定します。                  |

## 確率的な一致のプロパティ

確率的な一致方法を使用するオプションを選択すると、確率モデルをラベル適用操作に追加できます。トークンセットまたは参照テーブルが使用されている確率モデルをストラテジに追加することはできません。

以下の表に、確率的な一致に関連付けられているプロパティを示します。

| プロパティ    | 説明                                                 |
|----------|----------------------------------------------------|
| 名前       | 操作の名前を入力します。                                       |
| フィルタテキスト | 入力する文字またはワイルドカードを使用して、リポジトリ内の確率モデルのリストをフィルタリングします。 |
| 確率モデル    | この操作で使用する確率モデルを特定します。                              |

## 参照テーブルのプロパティ

参照テーブルプロパティは、参照テーブルを使用するためにラベル適用操作を設定するときに適用されます。

以下の表に、参照テーブルのプロパティを示します。

| プロパティ      | 説明                                                                                                                                            |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 名前         | 操作の名前を入力します。                                                                                                                                  |
| 参照テーブル     | トークンへのラベルの適用に使用する参照テーブルを指定します。                                                                                                                |
| ラベル        | 入力文字列が参照テーブルのエントリに一致するときに新しいポートに書き込まれるテキストを指定します。                                                                                             |
| 大文字小文字の区別  | 入力文字列を参照テーブルのエントリと照合するときに大文字と小文字を区別するかどうかを指定します。                                                                                              |
| 有効な値で一致を置換 | ラベルが適用された文字列を参照テーブルの「有効」カラムのエントリに置き換えます。                                                                                                      |
| 最頻値        | トークンラベル適用の方法を指定します。参照テーブルのエントリに一致する入力文字列にラベルを適用する場合は、「含める」を選択します。参照テーブルのエントリに一致する入力文字列にラベルを適用しない場合は、「占有」を選択します。                               |
| 優先順位の設定    | ストラテジで、参照テーブルへのラベル適用操作がトークンセットへのラベル適用操作よりも優先されるかどうかを指定します。このプロパティを設定すると、トークンセットへのラベル適用の前に参照テーブルへのラベル適用が実行され、トークンセットの分析は参照テーブルのラベル分析を上書きできません。 |

# 文字ラベル適用ストラテジの設定

ラベル適用ストラテジを設定するには、ラベラトランスフォーメーションの【ストラテジ】ビューで設定を編集します。

1. 【ストラテジ】ビューを選択し、【新規】をクリックしてストラテジを作成します。  
ストラテジウィザードが開きます。
2. ストラテジの名前を入力します。
3. 【入力】フィールドと【出力】フィールドをクリックして、ストラテジのポートを定義します。
4. 必要に応じて、ストラテジの説明を入力します。
5. 文字ラベル適用モードを選択します。
6. 【次へ】をクリックします。
7. 設定する文字ラベル適用操作のタイプを選択します。以下の操作を設定できます。
  - 参照テーブルを使用して文字にラベルを適用する。
  - 文字セットを使用したラベル文字。
8. 【次へ】をクリックします。
9. 操作のプロパティを設定し、【次へ】をクリックします。
10. 必要に応じて、【テキストを無視する】プロパティを設定します。
11. 【次へ】をクリックしてその他の操作をストラテジに追加するか、【完了】をクリックします。

ストラテジや操作を処理する順序は変更できます。【ストラテジ】ビューで、ストラテジまたは操作を選択し、【上に移動】または【下に移動】をクリックします。

# トークンラベル適用ストラテジの設定

ラベル適用ストラテジを設定するには、ラベラトランスフォーメーションの【ストラテジ】ビューで設定を編集します。

1. 【ストラテジ】ビューを選択し、【新規】をクリックしてストラテジを作成します。  
ストラテジウィザードが開きます。
2. ストラテジの名前を入力します。
3. 【入力】フィールドと【出力】フィールドをクリックして、ストラテジのポートを定義します。
4. 必要に応じて、ストラテジの説明を入力します。
5. トークンラベル適用モードを選択します。  
選択するモードのプロパティを検証または編集します。
6. 【次へ】をクリックします。
7. 設定するトークンラベル適用操作のタイプを選択します。以下の操作を設定できます。
  - トークンセットを使用してトークンにラベルを適用します。
  - 参照テーブルを使用してトークンにラベルを適用します。
  - 確率的な一致を使用してトークンにラベルを適用します。
8. 【次へ】をクリックします。
9. 操作のプロパティを設定し、【次へ】をクリックします。



確率的な一致を使用するようにストラテジを設定する場合は、その他のデータとして特定されるトークンに使用するラベルを入力します。

10. 必要に応じて、**【カスタムラベル】** プロパティを設定します。
11. **【次へ】** をクリックしてその他の操作をストラテジに追加するか、**【完了】** をクリックします。  
ストラテジや操作を処理する順序は変更できます。 **【ストラテジ】** ビューで、ストラテジまたは操作を選択し、**【上に移動】** または **【下に移動】** をクリックします。

## ラベラトランスフォーメーションの詳細プロパティ

Data Integration Service でラベラトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

**【詳細】** タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

## 非ネイティブ環境でのラベラトランスフォーメーション

非ネイティブ環境でのラベラトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## 第 25 章

# ルックアップトランスフォーメーション

この章では、以下の項目について説明します。

- [ルックアップトランスフォーメーションの概要, 390 ページ](#)
- [接続されたルックアップと接続されていないルックアップ, 391 ページ](#)
- [ルックアップトランスフォーメーションの開発, 394 ページ](#)
- [ルックアップクエリ, 394 ページ](#)
- [ルックアップソースフィルタ, 397 ページ](#)
- [ルックアップ条件, 398 ページ](#)
- [ルックアップキャッシュ, 401 ページ](#)
- [クエリのプロパティ, 401 ページ](#)
- [動的マッピングでのルックアップトランスフォーメーション, 402 ページ](#)
- [動的ポートの定義, 402 ページ](#)
- [ルックアップソースの変更, 403 ページ](#)
- [ポートセクタ, 407 ページ](#)
- [\[ランタイム\] プロパティ, 413 ページ](#)
- [詳細プロパティ, 414 ページ](#)
- [再利用可能なルックアップトランスフォーメーションの作成, 415 ページ](#)
- [再利用不可能なルックアップトランスフォーメーションの作成, 416 ページ](#)
- [接続されていないルックアップトランスレーションの作成, 417 ページ](#)
- [接続されていないルックアップの例, 418 ページ](#)
- [非ネイティブ環境でのルックアップトランスフォーメーション, 420 ページ](#)

## ルックアップトランスフォーメーションの概要

ルックアップトランスフォーメーションはパッシブまたはアクティブなトランスフォーメーションであり、フラットファイル、論理データオブジェクト、参照テーブル、またはリレーショナルテーブルをルックアップし

ます。ルックアップトランスフォーメーションは、ルックアップから 1 行または複数の行を返すことができます。

ルックアップトランスフォーメーションを作成する前に、ルックアップソースを作成します。フラットファイルまたはリレーショナルデータベーステーブルを物理データオブジェクトとしてインポートします。または、ルックアップソースとして使用する論理データオブジェクトまたは参照テーブルを作成します。ルックアップトランスフォーメーションを作成する際、Developer tool がデータオブジェクトまたは参照テーブルからこれらのカラムをトランスフォーメーションのルックアップ用ポートとして追加します。トランスフォーメーションの作成後に、ルックアップの結果を返す 1 つ以上の出力ポートを設定します。ルックアップ条件を設定し、他のルックアッププロパティを設定します。

マッピングの実行またはデータのプレビューを行うと、データ統合サービスはルックアップソースに対しクエリを実行します。データ統合サービスは、トランスフォーメーションのルックアップポート、ルックアッププロパティ、およびルックアップ条件に基づいて、ルックアップソースに対しクエリを実行します。ルックアップトランスフォーメーションは、ルックアップの結果をターゲットまたは別のトランスフォーメーションに返します。

接続されたまたは接続されていないルックアップトランスフォーメーションを設定できます。接続されたトランスフォーメーションは、マッピング内の別のトランスフォーメーションに接続します。接続されていないトランスフォーメーションは、別のトランスフォーメーションの:LKP 式から入力を受け取ります。ルックアップトランスフォーメーションで論理データオブジェクトに対してルックアップを実行する場合は、接続されたルックアップトランスフォーメーションを設定する必要があります。ルックアップトランスフォーメーションの入力ポートをアップストリームトランスフォーメーションまたはアップストリームソースに接続します。出力ポートをダウンストリームトランスフォーメーションまたはダウンストリームターゲットに接続します。

1 つのマッピングで複数のルックアップトランスフォーメーションを使用できます。

ルックアップトランスフォーメーションでは、以下のタスクを実行できます。

- 関連する値を取得する。入力データの値に基づいてルックアップソースから値を取得します。例えば、入力データに従業員 ID が含まれているとします。従業員 ID によってルックアップソースから従業員名を取得します。
- ルックアップソースから複数の行を取得する。
- 計算を行う。ルックアップテーブルから値を取得し、その値を計算に使用します。例えば、消費税率を取得し、税額を計算してターゲットに返します。
- 式を受け付けるトランスフォーメーションの:LKP 式を使用して、接続されていないルックアップを実行する。トランスフォーメーションの別の式を使用して結果をフィルタします。
- 動的マッピングでルックアップトランスフォーメーションを使用するために、ルックアップソースとルックアップ条件をパラメータ化します。

## 接続されたルックアップと接続されていないルックアップ

接続されたまたは接続されていないルックアップトランスフォーメーションを設定できます。接続されたルックアップトランスフォーメーションは、マッピング内の他のトランスフォーメーションに接続する入出力ポートのあるトランスフォーメーションです。接続されていないルックアップトランスフォーメーションは、マッピングに含まれますが、他のトランスフォーメーションに接続されていません。

接続されていないルックアップトランスフォーメーションは、トランスフォーメーション（式トランスフォーメーションやアグリゲータトランスフォーメーションなど）の:LKP 式の結果から入力を受け取ります。:LKP 式は、ルックアップトランスフォーメーションに引数を渡し、ルックアップトランスフォーメーションから結

果を受け取ります。:LKP 式は、ルックアップの結果を式トランスフォーメーションやアグリゲータトランスフォーメーションの別の式に渡して、結果をフィルタすることができます。

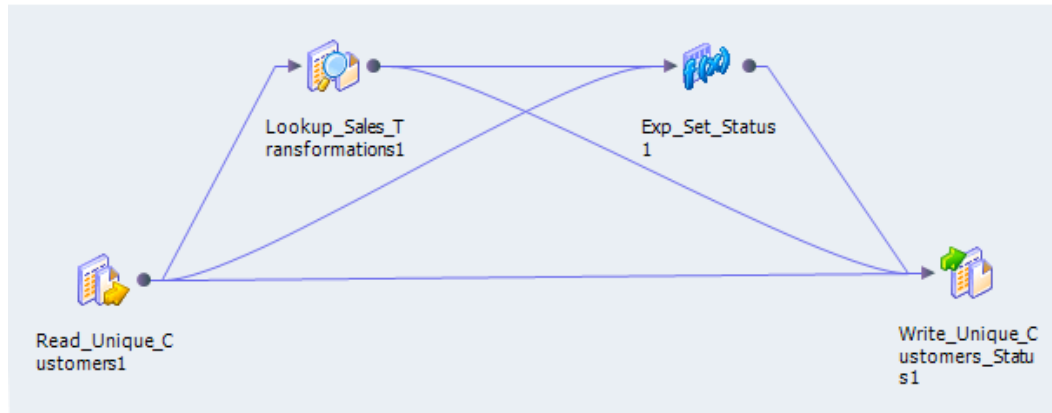
次の表に、接続されたルックアップと接続されていないルックアップの違いを示します。

| 接続されたルックアップ                                                                                                                    | 接続されていないルックアップ                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| 入力値をパイプラインから直接受け取ります。                                                                                                          | 入力値を別のトランスフォーメーションの:LKP 式の結果から受け取ります。                                                       |
| 動的キャッシュまたは静的キャッシュを使用します。                                                                                                       | 静的キャッシュを使用します。                                                                              |
| キャッシュには、ルックアップ条件のルックアップソースカラムおよび出力ポートとなるルックアップソースカラムが含まれています。                                                                  | キャッシュには、ルックアップ条件のすべてのルックアップポート/出力ポートおよびルックアップポート/戻りポートが入っています。                              |
| 同じ行から複数のカラムを返して、動的ルックアップのキャッシュに挿入します。                                                                                          | 各行から戻りポートに 1 つのカラムを返します。                                                                    |
| ルックアップ条件に一致するものがない場合、統合サービスはすべての出力ポートについてデフォルト値を返します。統合サービスで動的キャッシュが設定されている場合、キャッシュに行が挿入されるか、またはキャッシュは未変更のままとなります。             | ルックアップ条件に一致するものがない場合、統合サービスは NULL を返します。                                                    |
| ルックアップ条件に一致するものがある場合、統合サービスはすべてのルックアップ/出力ポートについてルックアップ条件の結果を返します。統合サービスで動的キャッシュが設定されている場合、キャッシュ内の行が更新されるか、または未変更のままのどちらかとなります。 | ルックアップ条件で一致があれば、データ統合サービスは戻りポートにルックアップ条件の結果を返します。                                           |
| 複数の出力値を別のトランスフォーメーションに渡します。ルックアップ/出力ポートを別のトランスフォーメーションにリンクします。                                                                 | 1 つの出力値を別のトランスフォーメーションに返します。ルックアップトランスフォーメーションの戻りポートは、他のトランスフォーメーションの:LKP 式が含まれるポートに値を渡します。 |
| ユーザ定義デフォルト値をサポートします。                                                                                                           | ユーザ定義デフォルト値をサポートしません。                                                                       |

## 接続されたルックアップ

接続されたルックアップトランスフォーメーションとは、マッピング内でソースまたはターゲットに接続されているルックアップトランスフォーメーションのことです。

次の図に、接続されたルックアップトランスフォーメーションとのマッピングを示します。



接続されたルックアップトランスフォーメーションが含まれているマッピングを実行すると、データ統合サービスは以下の手順を実行します。

1. データ統合サービスが値を別のトランスフォーメーションからルックアップトランスフォーメーションの入力ポートに渡します。
2. 各入力行に対して、データ統合サービスがトランスフォーメーションのルックアップポートおよびルックアップ条件に基づいて、ルックアップソースまたはキャッシュにクエリを実行します。
3. トランスフォーメーションがキャッシュを使用していない場合、または静的キャッシュを使用している場合、データ統合サービスはルックアップクエリから値を返します。

トランスフォーメーションが動的キャッシュを使用している場合、Integration Service は行を見つけられなかったキャッシュにその行を挿入します。Integration Service でキャッシュ内に行が検出された場合、キャッシュ内の行が更新されるかまたは未変更のままとなります。行に対して挿入、更新、または変更なしのフラグを設定します。

4. データ統合サービスがクエリからデータを返して、それをマッピング内の次のトランスフォーメーションに渡します。

トランスフォーメーションが動的キャッシュを使用している場合、行をフィルタまたはルータトランスフォーメーションに渡し、ターゲットへの新しい行をフィルタリングすることができます。

**注:** この章では、特に明示しない限り、接続されたルックアップトランスフォーメーションについて説明します。

## 接続されていないルックアップ

接続されていないルックアップトランスフォーメーションは、マッピング内のソースまたはターゲットに接続していないルックアップトランスフォーメーションです。式を使用できるトランスフォーメーションの:LKP式を使用して、ルックアップを呼び出します。

ルックアップ式の構文は、:LKP lookup\_transformation\_name(argument, argument, ...)です。

各引数を記述する順序は、ルックアップトランスフォーメーションのルックアップ条件の順序と一致しなければなりません。ルックアップトランスフォーメーションは、ルックアップトランスフォーメーションの戻りポートを介して、クエリの結果を返します。ルックアップを呼び出すトランスフォーメーションは、:LKP 式が含まれるポートでルックアップの結果値を受け取ります。ルックアップクエリが値を返せなかった場合、ポートは NULL 値を受け取ります。

接続されていないルックアップを実行するときは、マッピング内で同じルックアップを複数回実行できます。ルックアップの結果を別の式でテストし、その結果に基づいて行をフィルタすることができます。

接続されていないルックアップトランスフォーメーションを含むマッピングを実行すると、データ統合サービスは以下の手順を実行します。

1. 接続されていないルックアップトランスフォーメーションは、別のトランスフォーメーション（アグリゲータトランスフォーメーション、式トランスフォーメーション、アップデートストラテジトランスフォーメーションなど）内の:LKP 式の結果から、入力値を受け取ります。
2. データ統合サービスは、ルックアップトランスフォーメーションのルックアップポートおよびルックアップ条件に基づき、ルックアップソースまたはキャッシュに対してクエリを実行します。
3. データ統合サービスはルックアップトランスフォーメーションの戻りポートを介して値を返します。
4. 統合サービスは、:LKP 式の入ったポートに戻り値を渡します。

## ルックアップトランスフォーメーションの開発

ルックアップトランスフォーメーションを開発するときは、ルックアップソースのタイプやルックアップ条件などの要素を考慮する必要があります。

ルックアップトランスフォーメーションを開発するときは、以下の項目について考慮してください。

- フラットファイル、論理データ オブジェクト、参照テーブル、リレーショナルデータオブジェクトのうち、どれからトランスフォーメーションを作成することにするか。ルックアップトランスフォーメーションを作成する前に、ルックアップソースを作成します。フラットファイルまたはリレーショナルデータベーステーブルを物理データオブジェクトとしてインポートします。または、ルックアップソースとして使用する論理データオブジェクトまたは参照テーブルを作成します。
- トランスフォーメーションの出力ポート。
- トランスフォーメーションのルックアップ条件。
- 統合サービスでルックアップデータをキャッシュするかどうか。統合サービスがデータをキャッシュできるのは、データがフラットファイル、参照テーブル、またはリレーショナルデータオブジェクトのときです。

## ルックアップクエリ

統合サービスは、ルックアップトランスフォーメーションで設定したポートおよびプロパティに基づいてルックアップをクエリします。統合サービスは、最初の行がルックアップトランスフォーメーションに入ると、デフォルトのルックアップクエリを実行します。

リレーショナルテーブルにルックアップを使用する場合は、ルックアップクエリをオーバーライドできます。キャッシュされる前に、オーバーライドを使用して、WHERE 句の追加、またはルックアップデータの変換を行えます。

ルックアップクエリに SQL オーバーライドとフィルタを設定すると、統合サービスはフィルタを無視します。

### デフォルトのルックアップクエリ

デフォルトルックアップクエリには、以下の文が含まれます。

## SELECT

SELECT 文には、マッピングのすべてのルックアップポートが含まれます。ルックアップクエリの SELECT 文を表示するには、[カスタムクエリーを使用] プロパティを選択します。

## ORDER BY

ORDER BY 句はカラムを、そのカラムがルックアップトランスフォーメーションに出現する順序と同じ順序に並べかえます。統合サービスが ORDER BY 句を生成します。これは、デフォルトの SQL を生成するときには表示できません。

# ルックアップクエリの SQL オーバーライド

リレーショナルルックアップの場合、ルックアップクエリを上書きできます。キャッシュされる前に、WHERE 句の追加、またはルックアップデータの変換を行えます。

テーブル名とカラム名には予約語とスラッシュ記号を使用できます。

クエリを入力して、デフォルトのルックアップクエリを完全にオーバーライドできます。または、デフォルトのルックアップクエリを表示して編集することもできます。デフォルトのルックアップクエリには、ルックアップポート、出力ポート、および戻りポートが含まれます。

SQL オーバーライドを使用すると、クエリは句 ORDER BY 1 を追加します。この句は、他の句の最初と最後の値を確実に提供するためにデータを順序付けします。

**注:** Hive コマンドラインユーティリティで次のクエリを実行して、SQL を手動で検証できます。

```
CREATE VIEW <table name> (<port list>) AS <SQL>
```

ここで、

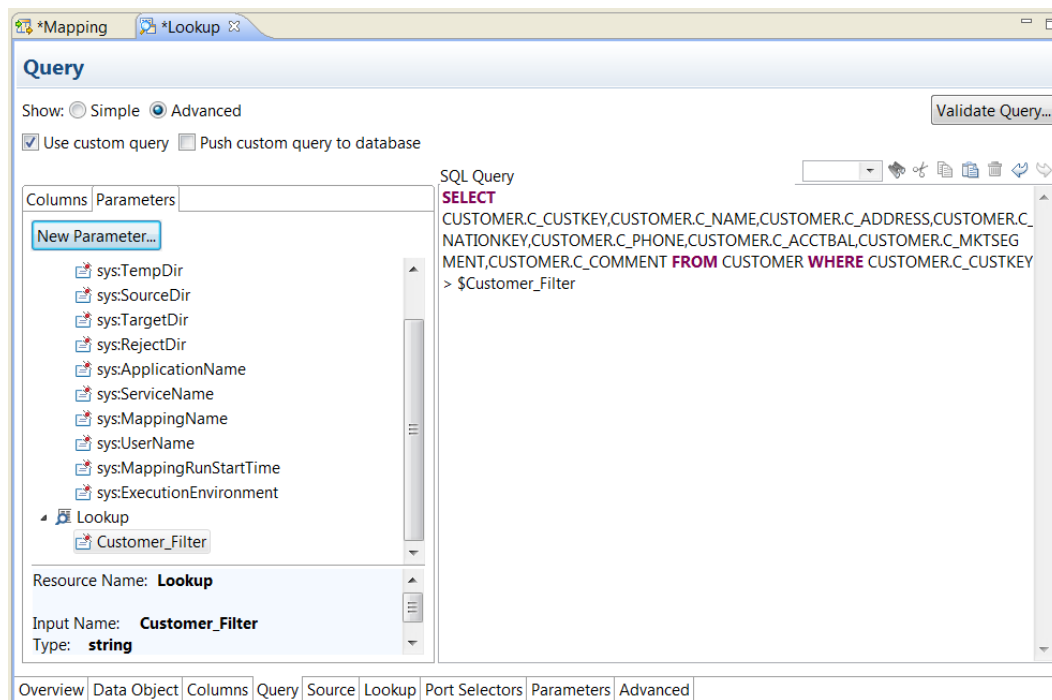
- <table name>は選択した名前
- <port list>は、ソースに含まれるポートのカンマ区切りリスト
- <SQL>は検証するクエリ

# SQL オーバーライドクエリのパラメータ

ルックアップトランスフォーメーションのルックアップクエリでは、システムパラメータまたはユーザー定義のパラメータを使用できます。SQL エディタには、選択可能なシステムパラメータおよびユーザー定義のパラメータのリストが表示されます。

ユーザー定義のパラメータは、ルックアップトランスフォーメーションの [クエリ] タブで参照または作成できます。各パラメータのデフォルト値を定義してください。デフォルト値を上書きするには、ルックアップトランスフォーメーションをマッピングに追加してから、マプレットまたはマッピングパラメータをトランスフォーメーションパラメータにバインドします。

次の図は、ルックアップトランスフォーメーションの [クエリ] タブを示しています。



## 予約語

ルックアップ名またはカラム名に MONTH、YEAR などのデータベース予約語が含まれる場合、統合サービスがデータベースに対して SQL を実行する際にデータベースエラーが発生してマッピングが失敗します。

予約語を格納するファイル (reswords.txt) は、統合サービスがインストールされているディレクトリで作成および管理することができます。統合サービスは、マッピングを初期化するときに、reswords.txt ファイル内を検索し、予約語の前後に引用符を挿入してから、ソース、ターゲット、およびルックアップデータベースに対して SQL を実行します。

引用符で括った識別子に SQL-92 標準を使用するには、場合によって Microsoft SQL Server や Sybase などいくつかのデータベースを有効にすることが必要です。環境 SQL を使用してコマンドを発行します。たとえば Microsoft SQL Server では、次のコマンドを使用できます。

```
SET QUOTED_IDENTIFIER ON
```

## ルックアップクエリのオーバーライドのガイドライン

ルックアップクエリを上書きするときに、一定のルールとガイドラインが適用されます。

ルックアップ SQL クエリを上書きする場合、以下のガイドラインを考慮してください。

- リレーショナルルックアップの場合は、ルックアップ SQL クエリを上書きできます。
- ルックアップキャッシュに追加する行をフィルタリングするには、ルックアップソースフィルタを追加します。これにより、統合サービスは WHERE 句に一致する動的キャッシュとターゲットテーブルにのみ行を挿入します。
- 複数のルックアップトランスフォーメーションでルックアップキャッシュを共有している場合、各ルックアップトランスフォーメーションで同じルックアップ SQL オーバーライドを使用します。
- ルックアップクエリ内のテーブル名やカラム名に予約語が含まれている場合、引用符で予約語を括ってください。



- キャッシュされていないルックアップにルックアップクエリを上書きするには、統合サービスで複数の一致が検出されたときに値を返すように指定します。
- デフォルトの SQL 文にはカラムの追加も削除もできません。
- Developer tool は、SQL クエリの構文を検証しません。接続されていないルックアップクエリの SQL オーバーライドが有効でない場合、マッピングは失敗します。

## ルックアップクエリの上書き

デフォルトのルックアップ SQL クエリを上書きして、カスタマイズしたクエリをルックアップソースに作成することができます。

1. **【プロパティ】** ビューで、**【クエリ】** タブを選択します。
2. **【詳細】** を選択します。
3. **【カスタムクエリを使用】** を選択します。
4. **【SQL クエリ】** 領域でルックアップクエリを編集します。  
テーブル名、カラム名、またはパラメータをダブルクリックしてクエリに追加できます。
5. **【クエリの検証】** をクリックして、ルックアップクエリを検証します。
6. **【カスタムクエリをデータベースにプッシュ】** を選択して、データベース内でルックアップクエリを実行します。

## ルックアップソースフィルタ

キャッシュが有効になっているリレーショナルルックアップトランスフォーメーション用にルックアップソースフィルタを設定できます。ルックアップソースフィルタを追加することで、データ統合サービスがルックアップソーステーブルに対して実行するルックアップの数が制限できます。

ルックアップソースフィルタを設定すると、データ統合サービスはそのフィルタ文の結果に基づいてルックアップを実行します。たとえば、ID が 510 より大きい従業員の名字をすべて取得する必要があるとします。

EmployeeID カラムに対して、次のようにルックアップソースフィルタを設定します。

```
EmployeeID >= 510
```

EmployeeID は、ルックアップトランスフォーメーションの入力ポートです。データ統合サービスはソース行を読み取るときに、EmployeeID の値が 510 より大きいときにキャッシュに対してルックアップを実行します。EmployeeID が 510 以下の場合、ルックアップトランスフォーメーションは姓を取得しません。

プッシュダウン最適化用に設定されたマッピングでルックアップソースフィルタをルックアップクエリに追加すると、データ統合サービスは SQL オーバーライドを示すビューを作成します。次に、データ統合サービスは、このビューに対して SQL クエリを実行し、トランスフォーメーションロジックをデータベースにプッシュします。

## ルックアップでのソース行のフィルタリング

キャッシュが有効になっているリレーショナルルックアップトランスフォーメーション用にルックアップソースフィルタを設定できます。ルックアップのソース行にフィルタを適用することで、統合サービスがルックアップソーステーブルに対して実行するルックアップの数を制限することができます。

1. **【プロパティ】** ビューで、**【クエリ】** タブを選択します。

2. **【フィルタ】** オプションで、**【編集】** をクリックします。
3. SQL エディタで、入力ポートを選択するか、フィルタを適用するルックアップトランスフォーメーションポートを入力します。
4. フィルタ条件を入力します。  
フィルタ条件にはキーワード WHERE を含めないでください。文字列マッピングパラメータおよび変数を文字列識別子で囲みます。
5. **【クエリの検証】** をクリックして、フィルタ条件の構文を検査します。

## ルックアップ条件

データ統合サービスは、ルックアップ条件に基づいてルックアップソースのデータをルックアップします。ルックアップトランスフォーメーションにルックアップ条件を設定する場合、ソースデータ内の 1 つまたは複数のカラムの値がルックアップソースまたはキャッシュ内の値と比較します。

例えば、ソースデータに employee\_number が含まれているとします。ルックアップソーステーブルには、employee\_ID、first\_name、および last\_name が含まれています。以下のルックアップ条件を設定します。

```
employee_ID = employee_number
```

Data Integration Service は、employee\_number ごとにルックアップソースの employee\_ID、last\_name、および first\_name カラムを返します。

Data Integration Service は、ルックアップソースから複数の行を返すことができます。以下のルックアップ条件を設定します。

```
employee_ID > employee_number
```

Data Integration Service は、employee\_ID 番号がソースの従業員番号よりも大きい行をすべて返します。

### データオブジェクトルックアップでの NULL 値

ルックアップ条件への入力が NULL の場合、データオブジェクトのルックアップトランスフォーメーションでは、出力専用ポートに NULL 値が設定され、パススルーポートに入力行の値が設定された 1 行が返されます。

例えば、以下のルックアップ条件は、employee\_ID の値が NULL となっている 1 つ以上の行を含むデータソースに対してルックアップを実行します。

```
employee_ID = employee_number
```

この例では、以下のデータを含むルックアップテーブルを使用します。

| EMPLOYEE_ID | LAST_NAME |
|-------------|-----------|
| 1294765     | Hara      |
| 1356356     | Carver    |
| 1407207     | NULL      |
| 1570348     | Draper    |
| NULL        | Limonov   |

データソースの以下の入力値をルックアップテーブルと比較します。

EMPLOYEE\_NUMBER  
-----  
1294765  
1356356  
1407207  
1648246  
NULL

この例では、ルックアップ条件により以下の結果が生成されます。

1294765,Hara  
1356356,Carver  
1407207,NULL  
NULL,NULL  
NULL,NULL

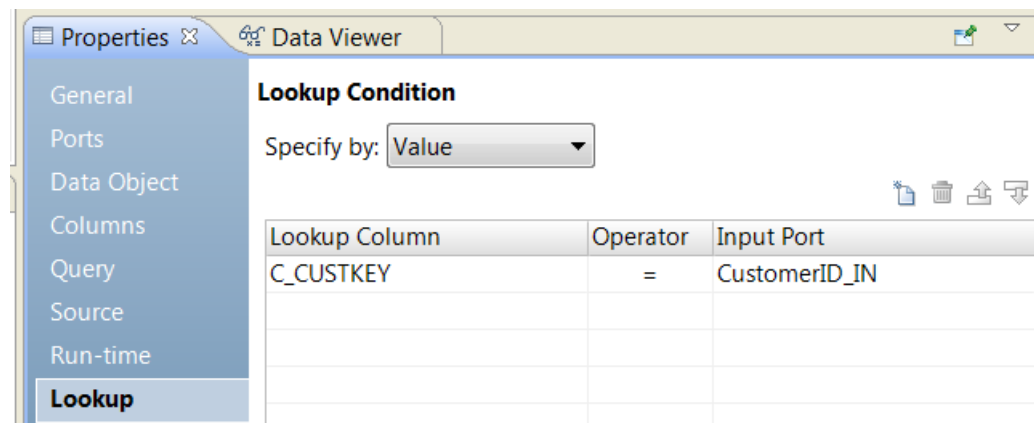
ルックアップ条件は、最初の 2 行で EMPLOYEE\_ID と EMPLOYEE\_NUMBER 間の一致を見つけます。3 番目の行では、ルックアップ条件に参加しない NULL 値を設定した行がルックアップソースに含まれます。これはルックアップ条件に一致するため、非ルックアップカラムに NULL 値を設定した結果が返されます。

4 番目と 5 番目の行では、ルックアップ条件は一致を見つけることができなかったため、両方の値に NULL を返します。5 番目の行でも、ルックアップ条件は一致を見つけることができません。NULL は、NULL を含めどの値にも一致しないためです。

## ルックアップ条件の設定

ルックアップ条件は、ルックアップソースから取得する行を決定する式です。ルックアップ条件の設定は、**【プロパティ】** ビューの **【ルックアップ】** タブで行います。

次の図は、顧客番号でルックアップを実行するルックアップ条件を示しています。



**【ルックアップ】** タブでは、以下のオプションを設定できます。

### 指定元

ルックアップカラムと入力ポート名を選択するには、**【値】** を選択します。式パラメータを設定してルックアップ条件を定義するには、**【パラメータ】** を選択します。

### ルックアップカラム

入力行のカラムと照合するルックアップソースのカラム。ルックアップ条件には複数のカラムを含めることができます。

### 演算子

ルックアップカラムと入力ポート間の検索条件を決定する演算子。=、!=、>、<、>=、<=の各演算子を使用できます。

## 入力ポート

ルックアップソース内を検索する値が含まれる入力ポート。1 つ以上の入力ポートとルックアップソースのポートを比較できます。

# ルックアップトランスフォーメーションの条件のルールとガイドライン

ルックアップトランスフォーメーションの条件の入力時には、特定のルールとガイドラインが適用されます。

ルックアップトランスフォーメーションの条件を入力する際は、以下のルールとガイドラインを考慮してください。

- ルックアップ条件に含めるカラムのデータ型は一致する必要があります。
  - ルックアップ条件でルックアップポートごとに入力ポートを 1 つ使用します。トランスフォーメーションの 1 つまたは複数の条件で、同じ入力ポートを使用できます。
  - ルックアップ条件でポートセレクトまたは動的ポートを使用する場合、ルックアップ条件は、式内のすべてのポートを対象とします。
  - 動的入力ポートまたはポートセレクトを、ルックアップ条件の入力ポートとして使用できます。入力ポート内の生成されたポート数は、ルックアップカラムのポート数と同じである必要があります。
  - 複数のルックアップ条件を使ってルックアップトランスフォーメーションを処理する場合、統合サービスはすべてのルックアップ条件と一致する行を返します。
  - 式パラメータを作成して、再利用できないルックアップトランスフォーメーションのルックアップ条件をパラメータ化できます。
  - ルックアップのパフォーマンスを向上するには、次の順序で条件を入力します。
    - 等しい (=)
    - より小さい (<)、より大きい (>)、より小さいまたは等しい (<=)、より大きいまたは等しい (>=)
    - 等しくない (!=)
  - ルックアップ条件は、=、>、<、>=、<=、または!=を演算子として使用して作成できます。
  - ルックアップ一致に対する統合サービスの処理は、ルックアップトランスフォーメーションの設定に何を使用するか（動的ルックアップキャッシュか、静的ルックアップキャッシュか、キャッシュを使用しないルックアップか）によって異なります。
  - 統合サービスは NULL 値を一致させます。例えば、ルックアップポートと入力ポートの両方に NULL 値があれば、統合サービスはそれらを同じものとみまします。
  - ルックアップ条件のカラムが Decimal データ型の場合、各カラムの精度は同じ精度範囲に収める必要があります。有効な精度範囲は以下のとおりです。
    - Decimal 0～18
    - Decimal 19～28
    - Decimal 29～38
    - Decimal 39 以上
- 例えば、条件を `DecimalA = DecimalB` と定義したとき、`DecimalA` の精度が 15 で `DecimalB` が 25 の場合、このルックアップ条件は有効ではありません。

# ルックアップキャッシュ

大きなルックアップソースまたは小さなルックアップテーブルをキャッシュすると、パフォーマンスを向上できます。ルックアップソースをキャッシュすると、統合サービスは入力行ごとにルックアップソースをクエリする代わりに、ルックアップキャッシュをクエリします。

さまざまなタイプのルックアップキャッシュを、各自の業務要件に応じて作成することができます。動的キャッシュでも静的キャッシュでも作成できます。永続キャッシュでも非永続キャッシュでも作成できます。キャッシュは、複数のルックアップトランスフォーメーション間で共有できます。

ルックアップトランスフォーメーションが動的マッピング内にある場合、永続または非永続キャッシュを使用できます。キャッシュを保持し、パラメータでルックアップソースを変更すると、マッピングは失敗します。フラットファイルルックアップソースの制御ファイルを変更する場合もマッピングは失敗します。

**注:** ルックアップトランスフォーメーションに動的ポートまたはパラメータ化されたルックアップソースが含まれる場合、動的ルックアップキャッシュや永続ルックアップキャッシュは使用できません。

## クエリのプロパティ

リレーショナルルックアップテーブルのルックアップクエリを表示または変更するように、クエリのプロパティを設定します。ルックアップにフィルタを適用したり、ルックアップクエリをカスタマイズすることができます。

次の表で、リレーショナルルックアップを実行するルックアップトランスフォーメーションに関するクエリのプロパティについて説明します。

| プロパティ               | 説明                                                                                                  |
|---------------------|-----------------------------------------------------------------------------------------------------|
| 簡易                  | ルックアップクエリを表示してルックアップにフィルタを適用する場合に選択します。                                                             |
| 詳細                  | リレーショナルルックアップテーブルを含むデータベースで、クエリを表示、クエリのカスタマイズ、クエリの実行をするときに選択します。                                    |
| フィルタ                | フィルタを入力して統合サービスがクエリの対象とする行の数を減らします。このオプションを表示するには、 <b>【簡易】</b> オプションを選択する必要があります。                   |
| カスタムクエリの使用          | ルックアップクエリを上書きするときに選択します。このオプションを表示するには、 <b>【詳細】</b> オプションを選択する必要があります。                              |
| カスタムクエリをデータベースにプッシュ | リレーショナルルックアップテーブルが入ったデータベースでクエリを実行する場合に選択します。このオプションを表示するには、 <b>【詳細】</b> オプションを選択する必要があります。         |
| SQL クエリ             | ルックアップの実行で使用する SQL クエリを表示します。SQL クエリをカスタマイズすることができます。このオプションを表示するには、 <b>【詳細】</b> オプションを選択する必要があります。 |

# 動的マッピングでのルックアップトランスフォーメーション

動的マッピングでルックアップトランスフォーメーションを使用できます。動的ポートを設定し、ソースデータに基づいて各種のポートと送受信を行うことができます。ルックアップソースおよびルックアップ条件をパラメータ化し、各種のポートに基づいてルックアップを実行できます。

動的マッピングとは、ソース、ターゲット、トランスフォーメーションのロジックが実行時に変更される可能性のあるマッピングです。パラメータとルールを設定してデータ構造を変更できます。動的マッピングでルックアップトランスフォーメーションを使用する場合、ルックアップトランスフォーメーションの入力ポートはソースデータに基づいて変更されることがあります。ルックアップ条件のルックアップソースおよびポートの構造が変更されることがあります。

**注:** ルックアップトランスフォーメーションに動的ポートまたはパラメータ化されたルックアップソースが含まれている場合は、ルックアップキャッシュを保持できません。動的キャッシュを設定することもできません。

ルックアップトランスフォーメーションに対して以下のタスクを実行すると、そのトランスフォーメーションを動的マッピングで使用できます。

## 動的ポートの定義

入力カラムの変化に対応できるように、動的ポートと生成されたポートを定義します。

## ルックアップソースのパラメータ化

ルックアップソースを定義するデータオブジェクトにパラメータを割り当てます。再利用不可能なルックアップトランスフォーメーションのルックアップソースをパラメータ化できます。

## ポートセレクトアの定義

ルックアップ条件で使用するポートを指定するポートセレクトアを定義します。再利用不可能なルックアップトランスフォーメーションのポートセレクトアのポートをパラメータ化できます。

## ルックアップ条件のパラメータ化

式パラメータを作成して、完全な式が含まれているデフォルト値を定義します。

動的マッピングの詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

# 動的ポートの定義

ルックアップトランスフォーメーションで動的ポートを定義できます。

ルックアップ条件の入力カラムで動的ポートを参照できます。生成された複数のポートが動的ポートに含まれる場合、ルックアップ条件のルックアップカラム要素でポートセレクトアを使用できます。動的入力ポートには、ルックアップ条件内のポートセレクトアと同じ数のポートが含まれる必要があります。

動的ポートに1つの値が含まれる場合、ルックアップ条件のルックアップカラム要素で1つのポートのみを使用できます。

生成されたポートをルックアップ条件内で参照できます。ただし、動的マッピング内のソースが変わった場合、生成されたポートが存在しない可能性があります。マッピングが失敗します。

# ルックアップソースの変更

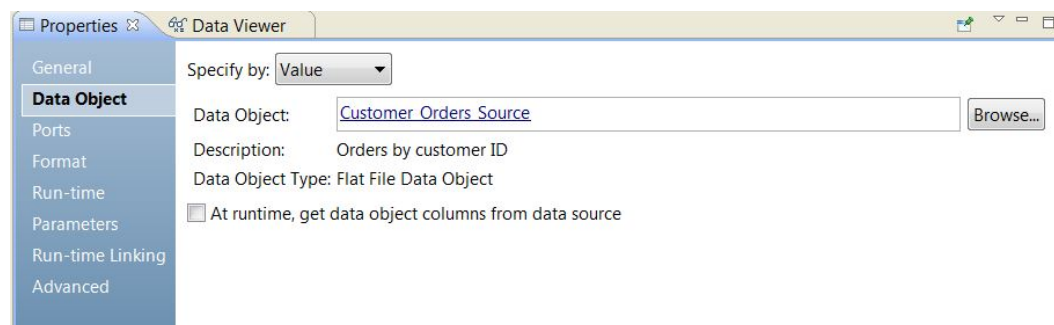
ルックアップソースであるデータオブジェクトを再利用可能なルックアップトランスフォーメーションに変更できます。実行時にルックアップソースとして使用するよう、データオブジェクトを決定するパラメータを設定できます。

物理データオブジェクトからトランスフォーメーションを作成する場合、トランスフォーメーションプロパティの【データオブジェクト】タブに、データオブジェクトに関する情報が表示されます。データオブジェクト名をクリックすると、モデルリポジトリにある物理データオブジェクトの定義を表示することができます。

モデルリポジトリ内の別の物理データオブジェクトを参照することで、トランスフォーメーションのデータオブジェクトを変更できます。データオブジェクトを変更する場合、トランスフォーメーションは、選択されたデータオブジェクトのランタイムプロパティと詳細プロパティを使用します。

データソースの変更内容に基づいて、データオブジェクトの構造を実行時に更新できます。データソースは、データオブジェクトで表される物理ファイルまたはデータベーステーブルです。データ統合サービスでデータカラムをデータソースから取得することを有効にした場合、データ統合サービスはデータソースの構造を調べます。データ統合サービスは、データソースに基づいてトランスフォーメーションインスタンス内のデータオブジェクトポートを更新します。データ統合サービスは、モデルリポジトリ内にある物理データオブジェクトの定義を変更しません。

次の図は、【データオブジェクト】タブを示しています。



【データオブジェクト】タブには以下のフィールドがあります。

## 指定元

特定のデータオブジェクト名を入力するには、【値】を選択します。データオブジェクトをパラメータ化するには、【パラメータ】を選択します。

## データオブジェクト

モデルリポジトリ内のデータオブジェクトの名前。【データオブジェクト】リンクをクリックすると、リポジトリにあるデータオブジェクトの定義を開くことができます。モデルリポジトリ内の別のデータオブジェクトを参照することもできます。

## 説明

リポジトリ内のデータオブジェクトの説明。読み取り専用。

## データオブジェクトタイプ

フラットファイルデータオブジェクト、リレーショナルデータオブジェクト、カスタマイズデータオブジェクトなど、データオブジェクトのタイプを示します。

## 実行時に、データソースからデータオブジェクトのカラムを取得します

データ統合サービスは実行時に、データオブジェクトが参照するテーブルのデータファイルからメタデータとデータ定義の変更を取得し、トランスフォーメーションインスタンスのデータオブジェクトの構造を更新します。



データ統合サービスが実行時にメタデータとデータ定義の変更を取得する方法をプレビューするには、解決済みパラメータとのマッピングを表示します。

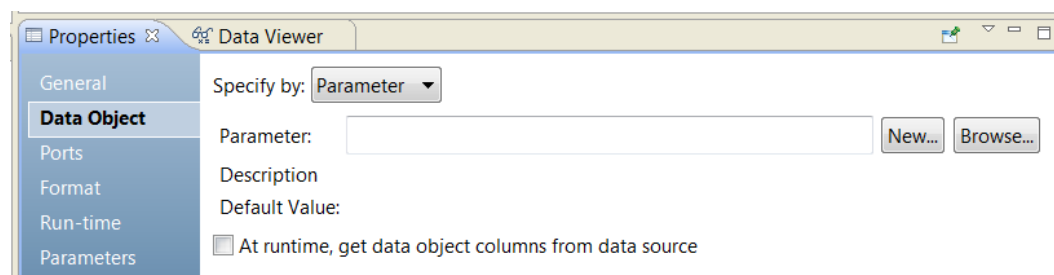
## ルックアップソースのパラメータ化

再利用可能ではないルックアップトランスフォーメーションのルックアップソースのパラメータを設定できません。

データオブジェクトをパラメータ化するには、[データオブジェクト] タブの [指定元:] で [パラメータ] を選択します。[データオブジェクト] タブのプロパティが変化します。

データオブジェクトをパラメータ化するには、リソースタイプパラメータを作成するか、作成済みのリソースパラメータを参照します。パラメータのデフォルト値は、モデルリポジトリ内の物理データオブジェクトの名前です。デフォルトのパラメータ値を作成する場合、リポジトリ内のデータオブジェクトのリストから物理データオブジェクト名を選択します。

次の図は、パラメータでデータオブジェクトを指定した場合の [データオブジェクト] タブを示しています。



[データオブジェクト] タブの [指定元: パラメータ] には、以下のオプションがあります。

### パラメータ

データオブジェクトとして設定したリソースパラメータの名前。読み取り専用。

### 説明

パラメータの説明。読み取り専用。

### 新規

リソースパラメータを作成します。パラメータのデフォルト値としてモデルリポジトリ内のデータオブジェクトを参照し、選択します。

### 参照

リソースパラメータを参照し、パラメータを選択します。

### デフォルト値

データオブジェクトに設定したリソースパラメータのデフォルト値。デフォルト値は、モデルリポジトリ内の物理データオブジェクト名とオブジェクトへのパスです。読み取り専用。

## ポート名とルックアップポートの競合

ルックアップソースをパラメータ化すると、ルックアップトランスフォーメーションの入力ポートは、ルックアップソースのポートと名前が競合する場合があります。

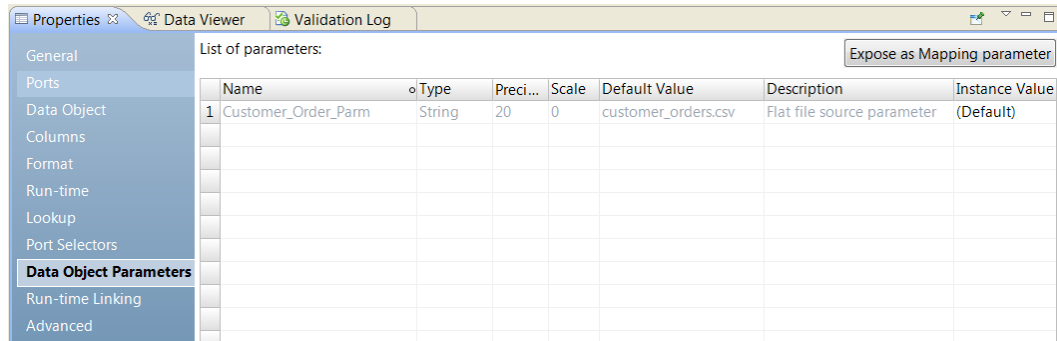
ルックアップトランスフォーメーションの入力ポートがルックアップソースのルックアップポートと名前が競合する場合、Developer tool では、いずれのポートも名前変更されません。Developer tool に検証エラーが表示されます。ルックアップトランスフォーメーションの入力ポート名を変更するか、ルックアップトランスフォーメーションからポートを削除する必要があります。



## パラメータを含むルックアップソース

パラメータを含む物理データオブジェクトからルックアップソースを作成できます。物理データオブジェクトをマッピングに追加すると、**【データオブジェクトパラメータ】** タブにパラメータが表示されます。

次の図は、ルックアップトランスフォーメーションの **【データオブジェクトパラメータ】** タブを示しています。



図には、Customer\_Order\_Parm が示されています。Customer\_Order\_Parm は、フラットファイルデータオブジェクトのソースファイル名のパラメータです。マッピングのソースファイル名を上書きするには、マッピングのパラメータに Customer\_Order\_Parm をバインドします。**【マッピングパラメータとして公開】** をクリックして、マッピングの重複パラメータを作成します。

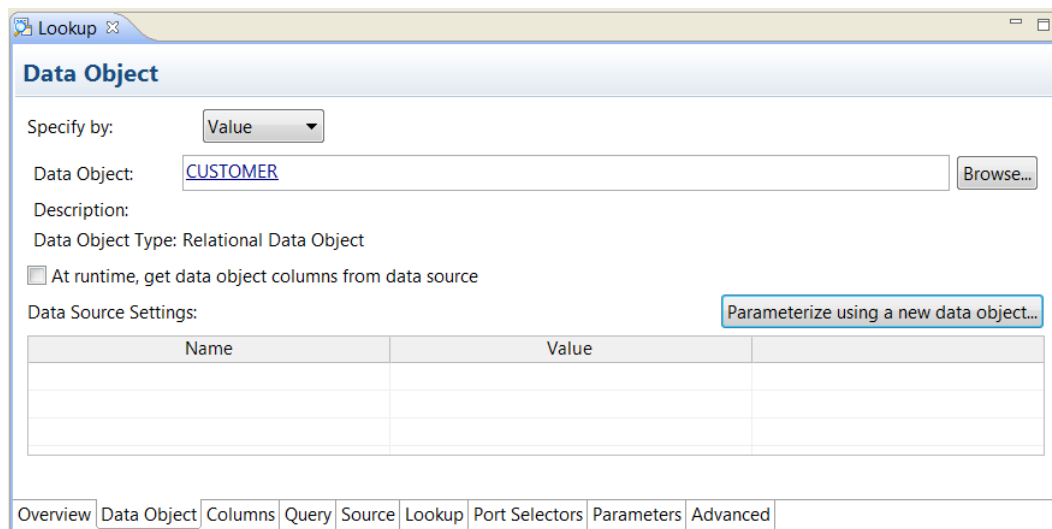
## 重複データオブジェクトでのパラメータの設定

リポジトリ内に重複データオブジェクトを作成し、その物理データオブジェクトのプロパティをパラメータ化できます。接続、リソース名、テーブル所有者、制御ファイル名など、各プロパティのデフォルト値を定義します。

リレーショナルデータオブジェクトとフラットファイルデータオブジェクトの重複データオブジェクトを作成できます。再利用可能なルックアップトランスフォーメーションと再利用不可能なルックアップトランスフォーメーションで重複データオブジェクトを作成できます。

ルックアップトランスフォーメーションの **【データオブジェクト】** タブで、重複データオブジェクトを作成します。データオブジェクトを値として指定する場合に限り、重複オブジェクトを作成できます。重複データオブジェクトを作成するときは、ルックアップトランスフォーメーション内のデータオブジェクト名を重複データオブジェクト名で置き換えます。Developer tool は、データオブジェクトの各プロパティのパラメータを作成します。Developer tool から、パラメータのデフォルト値を入力するように求められます。重複データオブジェクト名の構文は、<original object name>\_Param です。

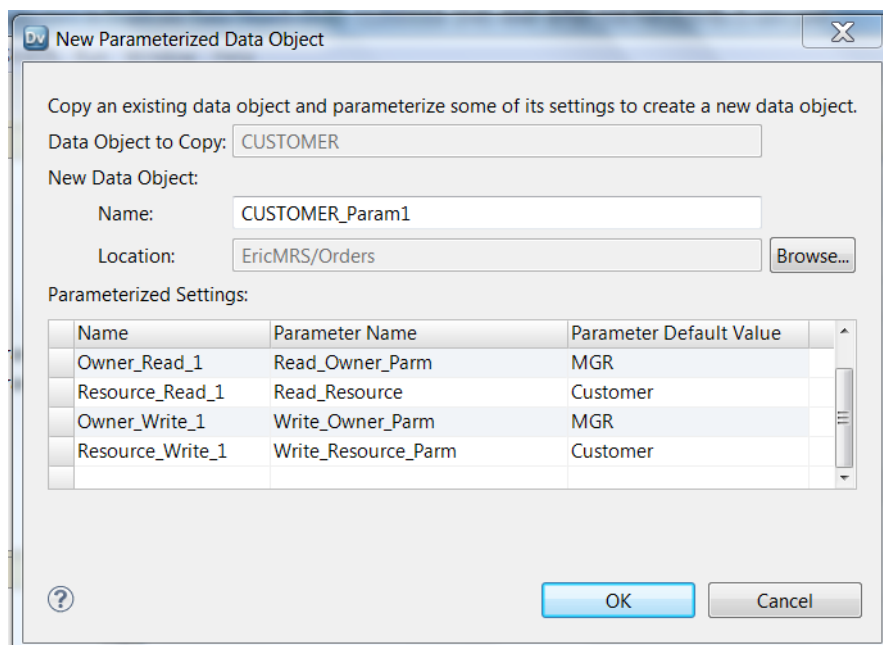
以下の図は、リレーショナルデータオブジェクトとフラットファイルデータオブジェクトの **【データオブジェクト】** タブの **【新しいデータオブジェクトを使用してパラメータ化します】** ボタンを示しています。



重複データオブジェクトを作成し、データオブジェクトをパラメータ化すると、Developer tool はそのデータオブジェクトのパラメータセットを作成します。Developer tool は、データオブジェクトがフラットファイルデータオブジェクトまたはリレーショナルデータオブジェクトのどちらであるかに基づいて、異なるパラメータを作成します。

重複データオブジェクトを作成するときは、**【パラメータ化された新しいデータオブジェクト】** ダイアログボックスで、デフォルトのパラメータ値を設定します。

以下の図は、リレーショナルデータオブジェクトの**【パラメータ化された新しいデータオブジェクト】**を示しています。

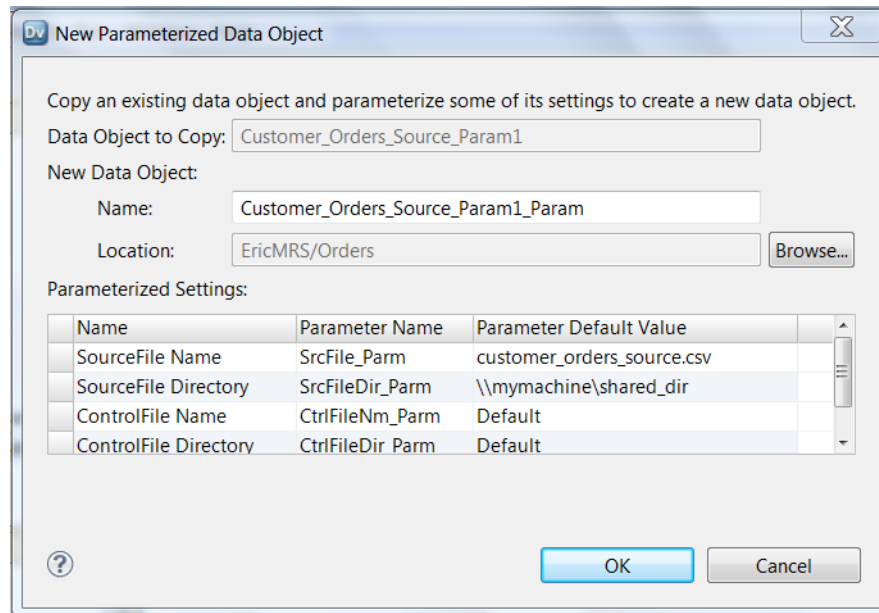


データオブジェクトの名前を変更できます。所有者およびリソースのパラメータのデフォルト値を入力します。

元のデータオブジェクトがパラメータ化されている場合、Developer tool は、元のデータオブジェクトから重複データオブジェクトにパラメータをコピーします。元のプロパティがパラメータ化されていない場合、Developer tool は、重複データオブジェクト内でそのプロパティのパラメータを作成します。Developer tool

は、元のプロパティ値を重複データオブジェクト内のデフォルトのパラメータ値として使用します。Developer tool が元のプロパティ値を特定できない場合、Developer tool はパラメータタイプに基づくデフォルト値を使用してパラメータを作成します。

以下の図は、フラットファイルデータオブジェクトの【パラメータ化された新しいデータオブジェクト】ダイアログボックスを示しています。



ソースファイルとソースファイルディレクトリのデフォルトのパラメータ値を設定します。フラットファイルに制御ファイルが含まれる場合、制御ファイル名とディレクトリを設定します。

デフォルト値を設定したら、Developer tool は重複データオブジェクトを作成します。ルックアップトランスフォーメーションの【データオブジェクト】タブに、重複データオブジェクト名が表示されます。【オブジェクトナビゲータ】に重複データオブジェクトが表示されます。

データオブジェクトの作成後にそのパラメータ値を変更するには、【オブジェクトナビゲータ】で物理データオブジェクトを開きます。【パラメータ】タブをクリックします。

## ポートセレクタ

ルックアップトランスフォーメーションに生成されたポートが含まれている場合は、ルックアップ条件を作成できます。ルックアップ条件では、動的ポートまたはポートセレクタを参照できます。式パラメータを使用して、完全なルックアップ式をパラメータ化することもできます。

生成された複数のポートが動的ポートに含まれる場合は、ルックアップ条件で生成されたポートをフィルタ処理するようにポートセレクタを定義できます。ルックアップソースは動的マッピングで変化することがあります。ルックアップカラムに使用するポートをフィルタ処理するように、ポートセレクタを設定できます。ルックアップソースのポートセレクタには、入力カラムのポートセレクタと同じ数のポートが含まれる必要があります。

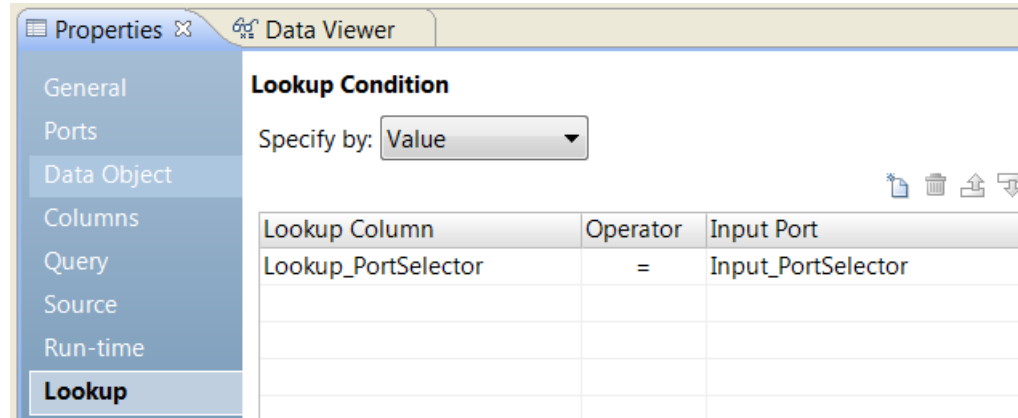
例えば、Lookup\_PortSelector には以下のポートが含まれます。

C\_CustKey  
C\_OrderKey

Input\_PortSelector には以下の入力ポートが含まれます。

CustomerID\_IN  
OrderID\_IN

次の図は、ポートセレクタが含まれているルックアップ条件を示しています。



このルックアップ条件は次の式に展開されます。

$C\_CustKey = CustomerID\_IN \text{ AND } C\_OrderKey = OrderID\_IN$

ルックアップ条件に複数のポートが含まれている場合は、1つの演算子を設定できます。例えば、演算子を大于 (>) に変更できます。このルックアップ条件は次の式に展開されます。

$C\_CustKey > CustomerID\_IN \text{ AND } C\_OrderKey > OrderID\_IN$

動的ポートを含むルックアップ条件を作成することができます。

$Lookup\_PortSelector = Dynamic\_Input\_Port$

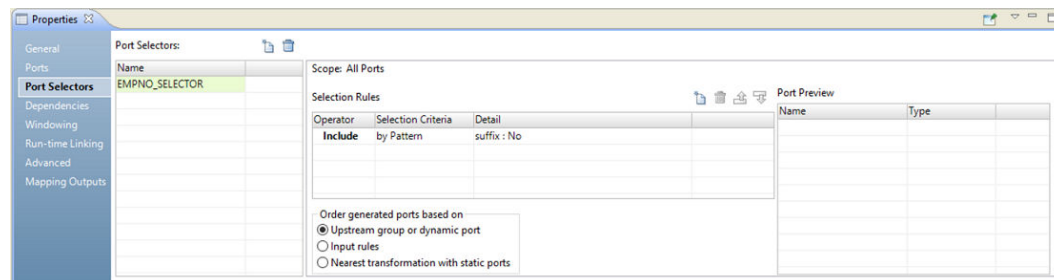
動的ポートには、ポートセレクタと同じ数のポートが含まれる必要があります。

## ポートセレクタの設定

ポートセレクタを設定する際には、選択ルールを定義して、含める生成済みポートを決定します。選択ルールは、動的ポートに設定できる入力ルールと同じです。

ポートセレクタには、静的または生成済みポートを含めることができます。ポートセレクタの設定は、**ポートセレクタ**タブで行います。

次の図は、**ポートセレクタ** タブを示しています。



ポートセレクタに次のプロパティを設定します。

## 名前

ポートセレクトを識別します。1つのトランスフォーメーションに複数のポートセレクトを作成し、式で参照できます。

## スコープ

ポートセレクトが適用されるポートグループを識別します。ジョイナまたはルックアップトランスフォーメーションのポートセレクトを作成する場合、スコープを選択する必要があります。これらのトランスフォーメーションには複数の入力グループがあります。ジョイナトランスフォーメーションにはマスタまたは詳細スコープがあります。ルックアップトランスフォーメーションにはインポートまたはルックアップスコープがあります。式トランスフォーメーションには入力グループが1つあります。スコープは常に「すべてのポート」です。

## 選択ルール

ポートセレクトに含めるポートを決定します。選択ルールを作成すると、**「ポートのプレビュー」** パネルに現在の入力ポートのうち適格なポートが表示されます。これらのポートは変わる可能性があります。さまざまなソースからのポートに対応できるように選択ルールを設定します。

# 選択ルール

ポートセレクトに関連付けられた選択ルールでポートセレクトに含めるポートを決定します。

選択ルールを作成すると、**「ポートのプレビュー」** パネルに現在の入力ポートのうち適格なポートが表示されます。これらのポートは変わる可能性があります。さまざまなソースからのポートに対応できるように選択ルールを設定します。

次の条件に基づいて選択ルールを作成します。

### 演算子

選択ルールが返すポートを含めるか、除外します。デフォルトは「含める」です。ポートを除外する前にポートを含める必要があります。

### 選択条件

作成する選択ルールのタイプ。カラム名、ポートタイプ、パターン、または複合データ型定義に基づいてルールを作成できます。カラム名に基づいてポートを含めるには、特定の名前を検索するか、名前に含まれる文字列パターンを検索します。

### 詳細

選択条件に適用する値。選択条件がカラム名基準になっている場合は、検索する文字列または名前を設定します。選択条件がポートタイプ基準になっている場合は、含めるポートタイプを選択します。

次の表に、選択条件と条件の詳細を指定する方法を示します。

| 選択条件 | 説明                            | 詳細                                                                |
|------|-------------------------------|-------------------------------------------------------------------|
| すべて  | すべてのポートを含めます。                 | 詳細は不要です。                                                          |
| 名前   | ポート名に基づいてポートをフィルタリングします。      | 値のリストからポート名を選択するか、 <b>「ポート」</b> または <b>「ポートリスト」</b> のパラメータを使用します。 |
| タイプ  | 各ポートのデータ型に基づいてポートをフィルタリングします。 | リストからデータ型を選択します。                                                  |

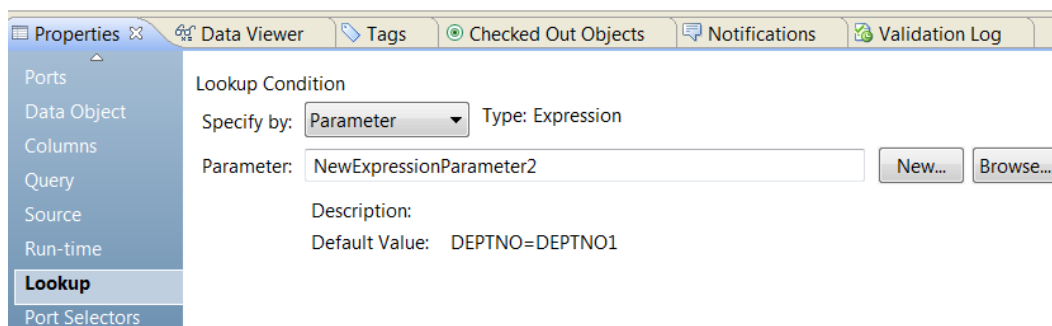
| 選択条件     | 説明                                    | 詳細                                                                                   |
|----------|---------------------------------------|--------------------------------------------------------------------------------------|
| パターン     | 名前に含まれる文字列または正規表現を使用してポートをフィルタリングします。 | ポート名のパターンタイプとして、プレフィックス、サフィックス、または正規表現を選択します。次に、パターンの値を入力するか、文字列タイプのパラメータを使用します。     |
| 複合データ型定義 | 複合データ型定義を使用してポートをフィルタリングします。          | 複合データ型定義のパターンタイプとして、プレフィックス、サフィックス、または正規表現を選択します。次に、パターンの値を入力するか、文字列タイプのパラメータを使用します。 |

## ルックアップ条件のパラメータ化

ルックアップ条件を定義する式パラメータを設定できます。式パラメータには、式エディタで作成した完全な式が含まれています。マッピングパラメータを定義して、実行時に式パラメータを上書きできます。

パラメータを使用してルックアップ条件を指定する場合は、式パラメータを参照するか、パラメータを作成することができます。

次の図は、ルックアップ条件の式パラメータを設定する場所を示しています。



パラメータを作成するには、**新規** をクリックします。パラメータの名前を定義して、デフォルト値を編集します。式パラメータのデフォルト値は、ルックアップ条件を定義するための完全な式です。生成されたポート、動的ポート、およびポートセクタを式に使用できます。

**注:** 式を作成する場合、ルックアップカラムが常に最初の値であり、入力カラムが 2 番目の値です。

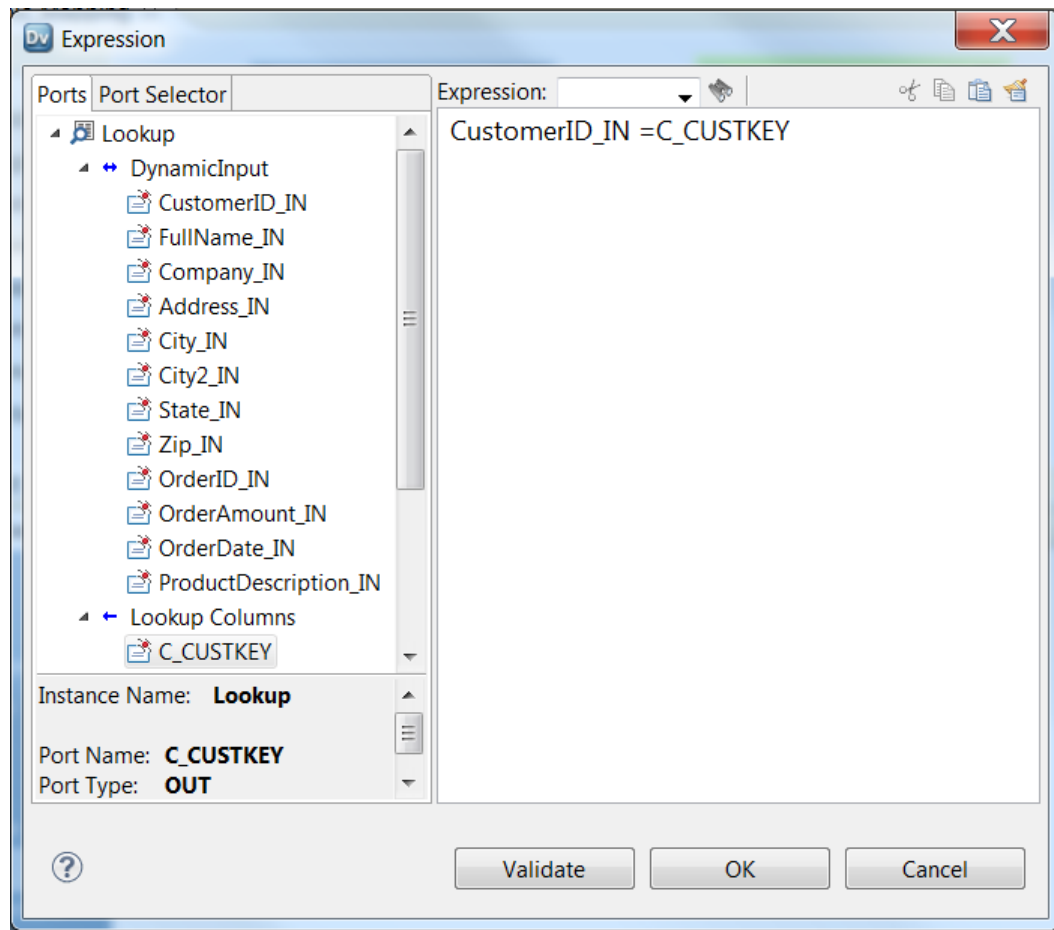
例えば、式パラメータで次のルックアップ条件を作成するとします。

```
CustomerID_IN = C_CUSTKEY
```

CustomerID\_IN はルックアップカラムです。

C\_CUSTKEY は入力カラムです。

次の図は、式エディタのルックアップ式を示しています。



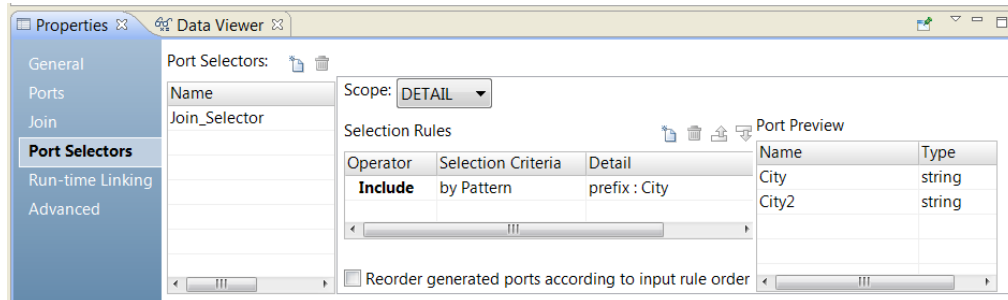
## ポートセレクトタの作成

動的式、ルックアップ条件、または結合条件で使用するポートを決定するために、ポートセレクトタを作成します。

1. **【ポートセレクトタ】** タブをクリックします。
2. **【ポートセレクトタ】** 領域で、**【新規】** をクリックします。  
Developer tool が、すべてのポートが含まれるデフォルトの選択ルールを使用してポートセレクトタを作成します。
3. **【ポートセレクトタ】** 領域で、ポートセレクトタ名を一意的な名前に変更します。
4. ジョイントトランスフォーメーションまたはルックアップトランスフォーメーションに対して作業している場合は、スコープを選択します。  
使用可能なポートは、選択したポートのグループに基づいて変更されます。
5. **【選択ルール】** 領域で、**【演算子】** を選択します。
  - 含む。ポートセレクトタのポートを含めるルールを作成します。ポートを除外する前にポートを含める必要があります。
  - 除外。ポートセレクトタから特定のポートを除外するルールを作成します。
6. **【選択条件】** を選択します。
  - 名前。特定のポートを名前で選択します。スコープ内のポートのリストからポート名を選択できます。

- **タイプ。** ポートをタイプで選択します。1つまたは複数のデータ型を選択できます。
- **パターン。** ポート名の文字のパターンでポートを選択します。特定の文字で検索するか、正規表現を作成できます。

次の図は、[ポートセクタ] タブを示しています。



7. **【詳細】** カラムをクリックします。  
**【入力ルールの詳細】** ダイアログボックスが表示されます。
8. ポートをフィルタ処理する基準となる値を選択します。
  - **名前。** 値またはパラメータを基準としてポートリストを作成する場合に選択します。 **【選択】** をクリックして、リスト内のポートを選択します。
  - **タイプ。** リストから1つ以上のデータ型を選択します。 **【ポートのプレビュー】** 領域に、選択したタイプのポートが表示されます。
  - **パターン。** ポート名のプレフィックスまたはサフィックスから、特定の文字パターンを検索する場合に選択します。または、検索に使用する正規表現を作成することを選択します。パラメータを設定するか、検索に使用するパターンを設定します。**【ポートのプレビュー】** 領域に、設定したルールどおりにポートセクタのポートが表示されます。
9. ポートセクタのポートの順序を変更するには、 **【生成されたポートの順序を入力ルールの順序に従って変更】** を選択します。



# [ランタイム] プロパティ

ランタイムプロパティを設定すると、ルックアップのキャッシュを有効にして設定できます。ランタイムルックアップのプロパティを設定する前に、ルックアップトランスフォーメーションをマッピングに追加する必要があります。

次の表に、フラットファイル、参照テーブル、またはリレーショナルルックアップを実行するルックアップトランスフォーメーションのランタイムプロパティを示します。

| プロパティ                             | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ルックアップ<br>キャッシュが<br>有効            | 統合サービスがルックアップ値をキャッシュするかどうかを指定します。<br>ルックアップキャッシュを有効にすると、統合サービスはルックアップソースに対してクエリを一度実行し、値をキャッシュし、キャッシュ内の値をルックアップします。ルックアップ値をキャッシュに格納すると、大きなルックアップテーブルのパフォーマンスを向上できます。<br>キャッシュを無効にすると、行がトランスフォーメーションに渡されるたびに、統合サービスはルックアップソースに <code>select</code> 文を発行してルックアップ値を求めます。<br>統合サービスはフラットファイルルックアップを常にキャッシュします。                                                                                                                                                     |
| ルックアップ<br>のデータキャ<br>ッシュサイズ        | マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってデータキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。                                                                                                                                                                                                                                                                                                  |
| ルックアップ<br>のインデック<br>スキャッシュ<br>サイズ | マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってインデックスキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。                                                                                                                                                                                                                                                                                               |
| キャッシュフ<br>ァイル名のプ<br>レフィックス        | キャッシュファイルのプレフィックス。永続ルックアップキャッシュに対しキャッシュフ<br>ァイル名のプレフィックスを指定することができます。                                                                                                                                                                                                                                                                                                                                                                                         |
| ルックアップ<br>キャッシュの<br>事前作成          | ルックアップトランスフォーメーションがデータを受け取る前に統合サービスがルックア<br>ップキャッシュを作成できるようになります。パフォーマンス向上のために統合サービス<br>は、複数のルックアップキャッシュファイルと同時に作成することができます。<br>以下のいずれかのオプションを設定します。 <ul style="list-style-type: none"><li>- 自動。統合サービスが値を決定します。</li><li>- 常に許可。ルックアップトランスフォーメーションがデータを受け取る前に統合サービ<br/>スがルックアップキャッシュを作成できるようになります。パフォーマンス向上のため<br/>に統合サービスは、複数のルックアップキャッシュファイルと同時に作成することがで<br/>きます。</li><li>- 常に不許可。統合サービスは、ルックアップトランスフォーメーションが最初の行を受<br/>け取るまではルックアップキャッシュを作成できません。</li></ul> |
| ルックアップ<br>キャッシュデ<br>ィレクトリ名        | ルックアップソースをキャッシュするようにルックアップトランスフォーメーションが設<br>定されている場合に、ルックアップキャッシュファイルの作成に使用されるディレクト<br>リ。<br>デフォルトは <code>CacheDir</code> システムパラメータです。このプロパティには、別のシステムパ<br>ラメータまたはユーザー定義のパラメータを設定できます。                                                                                                                                                                                                                                                                        |
| ルックアップ<br>ソースからの<br>再キャッシュ        | 永続キャッシュをルックアップテーブルと同期させるためにルックアップキャッシュを再<br>構築します。ルックアップトランスフォーメーションに永続ルックアップキャッシュがあ<br>り、ルックアップテーブルが時折変更される場合、データベースからルックアップを再キ<br>ャッシュします。                                                                                                                                                                                                                                                                                                                  |

関連項目：

- [「キャッシュサイズ」 \(ページ 72\)](#)

## 詳細プロパティ

リレーショナルデータベースへの接続と永続ルックアップキャッシュを、詳細プロパティで設定できます。表示されるプロパティは、ルックアップソースのタイプに基づきます。

次の表で、ルックアップソースの各タイプの詳細プロパティについて説明します。

| プロパティ            | ルックアップ<br>ソースのタイ<br>プ                     | 説明                                                                                                                                                                                                                |
|------------------|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ルックアップキャッシュの永続性  | フラットファイル、参照テーブル、リレーショナルルックアップ             | 統合サービスが、2 つ以上のキャッシュファイルを含む永続ルックアップキャッシュを使用するかどうかを示します。ルックアップトランスフォーメーションが永続ルックアップキャッシュを使用するように設定され、永続ルックアップキャッシュファイルが存在しない場合、統合サービスがファイルを作成します。                                                                   |
| 大文字小文字を区別した文字列比較 | フラットファイル                                  | 統合サービスは、文字列カラムに対してルックアップを実行するときに、大文字小文字を区別した比較を使用します。                                                                                                                                                             |
| NULL の順序付け       | フラットファイル                                  | 統合サービスによる NULL 値の順序付けの方法を決定します。NULL 値を昇順にソートするか降順にソートするかを選択できます。デフォルトでは、統合サービスは NULL 値を昇順にソートします。これによって、統合サービスの設定が上書きされ、比較演算子において NULL の扱いを昇順、降順または NULL とします。リレーショナルルックアップの場合、NULL の順序付けはデータベースのデフォルト値によって異なります。 |
| トレースレベル          | フラットファイル、論理データオブジェクト、参照テーブル、リレーショナルルックアップ | このトランスフォーメーションのログに表示される情報の詳細度を設定します。[簡易]、[通常]、[詳細 - 初期化]、[詳細 - データ] から選択できます。デフォルトは [通常] です。                                                                                                                      |
| 更新または挿入          | 参照テーブル、リレーショナルルックアップ                      | 動的ルックアップキャッシュのみに適用されます。統合サービスは、ルックアップトランスフォーメーションに入る行タイプが更新で、インデックスキャッシュに行が存在し、そしてキャッシュデータが既存の行と異なる場合に、キャッシュ内の行を更新します。統合サービスは、新規行の場合、その行をキャッシュに挿入します。                                                             |
| 挿入または更新          | 参照テーブル、リレーショナルルックアップ                      | 動的ルックアップキャッシュのみに適用されます。統合サービスは、ルックアップトランスフォーメーションに入る行タイプが挿入で、それが新規の場合に、行をキャッシュに挿入します。インデックスキャッシュに行が存在する場合であってもデータキャッシュが現在の行と異なる場合は、統合サービスはデータキャッシュで行を更新します。                                                       |
| 更新時に古い値を出力       | 参照テーブル、リレーショナルルックアップ                      | 統合サービスは、行を更新する前にキャッシュ内にある値を出力します。それ以外の場合、統合サービスはキャッシュ内に書き込んだ更新後の値を出力します。                                                                                                                                          |

| プロパティ        | ルックアップソースのタイプ        | 説明                                                                                                                                                                      |
|--------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 動的キャッシュの更新条件 | 参照テーブル、リレーショナルルックアップ | 動的ルックアップキャッシュのみに適用されます。動的キャッシュを更新するかどうかを示す式。統合サービスは、条件が <code>true</code> でデータがキャッシュに存在する場合にキャッシュを更新します。デフォルトは <code>true</code> です。                                    |
| 接続           | 参照テーブル、リレーショナルルックアップ | リレーショナルルックアップソースを含んでいるリレーショナルデータベースへの接続。接続用のパラメータを使用できます。<br>参照テーブルのルックアップの場合、このフィールドは読み取り専用です。                                                                         |
| ソート済み入力      | フラットファイル             | 入力データがグループ単位で事前にソートされていることを示します。                                                                                                                                        |
| 日時形式         | フラットファイル             | 日時形式およびフィールド幅を定義します。ミリ秒、マイクロ秒、ナノ秒の形式は、29 のフィールド幅になります。ポートの日時形式を選択しない場合は、任意の日時形式を入力できます。<br>デフォルトは、「YYYY-MM-DD HH24:MI:SS」です。日時形式を設定しても、ポートのサイズは変わりません。このフィールドは読み取り専用です。 |
| 桁区切り記号       | フラットファイル             | 値は [なし] です。このフィールドは読み取り専用です。                                                                                                                                            |
| 小数点記号        | フラットファイル             | 値はピリオドです。このフィールドは読み取り専用です。                                                                                                                                              |

## 再利用可能なルックアップトランスフォーメーションの作成

フラットファイル、論理データオブジェクト、参照テーブル、またはリレーショナルデータオブジェクトをルックアップするルックアップトランスフォーメーションを作成します。

1. **[Object Explorer]** ビューで、プロジェクトまたはフォルダを選択します。
2. **[ファイル] > [新規] > [トランスフォーメーション]** をクリックします。
3. ルックアップウィザードに移動します。
4. **[フラットファイルデータオブジェクトルックアップ]**、**[論理データオブジェクトルックアップ]**、**[参照テーブルルックアップ]**、または **[リレーショナルデータオブジェクトルックアップ]** を選択します。
5. **[次へ]** をクリックします。  
**[新しいルックアップトランスフォーメーション]** ダイアログボックスが表示されます。
6. Developer ツール内の参照テーブルまたは物理データオブジェクトを選択します。
7. トランスフォーメーションの名前を入力します。
8. **[複数の一致の検出時]** で、ルックアップトランスフォーメーションがルックアップ条件に一致する行を複数見つけたときにどの行を返すのかを設定します。
9. **[完了]** をクリックします。  
ルックアップトランスフォーメーションがエディタに表示されます。

10. 【概要】ビューの【ポート】セクションで、トランスフォーメーションに出力ポートを追加します。  
以下の図に、ルックアップトランスフォーメーションの CUSTOMER\_NO 出力ポートを示します。

**概要**

**全般**

名前: Lookup

説明:

物理データオブジェクト: sales\_transactions1 参照...

複数の一致: 任意の行を返す

**ポート**

| 名前            | タイプ     | 精度 | スケ... | 出力                                  | デフォルト値 | 説明 |
|---------------|---------|----|-------|-------------------------------------|--------|----|
| 1 CUSTOMER_NO | decimal | 5  | 0     | <input checked="" type="checkbox"/> |        |    |

概要 カラム 形式 ルックアップ パラメータ 詳細

11. 【ランタイム】タブ（【プロパティ】ビュー内）で、【ルックアップキャッシュが有効】を選択してルックアップキャッシングを有効にします。  
**注:** ランタイムルックアップのプロパティを設定する前に、ルックアップトランスフォーメーションをマッピングに追加する必要があります。
12. 【ルックアップ】タブ（【プロパティ】ビュー内）で、1つ以上のルックアップ条件を追加します。
13. 【詳細】タブ（【プロパティ】ビュー内）で、トレースレベル、動的ルックアップキャッシュプロパティ、ランタイム接続を設定します。
14. トランスフォーメーションを保存します。

## 再利用不可能なルックアップトランスフォーメーションの作成

マッピングまたはマップレットで再利用不可能なルックアップトランスフォーメーションを作成します。

- マッピングまたはマップレットで、トランスフォーメーションパレットからエディタにルックアップトランスフォーメーションをドラッグします。  
【新規】ダイアログボックスが表示されます。
- 【フラットファイルデータオブジェクトルックアップ】、【論理データオブジェクトルックアップ】、【参照テーブルルックアップ】、または【リレーショナルデータオブジェクトルックアップ】を選択します。
- 【次へ】をクリックします。  
【新しいルックアップトランスフォーメーション】ダイアログボックスが表示されます。
- Developer ツール内の参照テーブルまたは物理データオブジェクトを選択します。
- トランスフォーメーションの名前を入力します。

6. **【複数の一致の検出時】** で、ルックアップトランスフォーメーションがルックアップ条件に一致する行を複数見つけたときにどの行を返すのかを設定します。
7. **【完了】** をクリックします。  
ルックアップトランスフォーメーションがエディタに表示されます。
8. エディタでルックアップトランスフォーメーションを選択します。  
トランスフォーメーションの上にツールバーが表示されます。
9. **【ポート】** タブ（**【プロパティ】** ビュー内）で、出力ポートをトランスフォーメーションに追加します。  
以下の図に、ルックアップトランスフォーメーションの CUSTOMER\_NO 出力ポートを示します。

| プロパティ × データビューア タグ 検証ログ ナビゲータ                            |               |         |    |       |                                     |        |    |
|----------------------------------------------------------|---------------|---------|----|-------|-------------------------------------|--------|----|
| 全般<br>ポート<br>カラム<br>形式<br>ランタイム<br>ルックアップ<br>パラメータ<br>詳細 |               |         |    |       |                                     |        |    |
|                                                          | 名前            | タイプ     | 精度 | スケ... | 出力                                  | デフォルト値 | 説明 |
|                                                          | 1 CUSTOMER_NO | decimal | 5  | 0     | <input checked="" type="checkbox"/> |        |    |
|                                                          |               |         |    |       |                                     |        |    |
|                                                          |               |         |    |       |                                     |        |    |
|                                                          |               |         |    |       |                                     |        |    |
|                                                          |               |         |    |       |                                     |        |    |
|                                                          |               |         |    |       |                                     |        |    |

10. **【ランタイム】** タブ（**【プロパティ】** ビュー内）で、**【ルックアップキャッシュが有効】** を選択してルックアップキャッシングを有効にします。  
**注:** ランタイムルックアップのプロパティを設定する前に、ルックアップトランスフォーメーションをマッピングに追加する必要があります。
11. **【ルックアップ】** タブ（**【プロパティ】** ビュー内）で、1 つ以上のルックアップ条件を追加します。
12. **【詳細】** タブ（**【プロパティ】** ビュー内）で、トレースレベル、動的ルックアップキャッシュプロパティ、ランタイム接続を設定します。
13. トランスフォーメーションを保存します。

## 接続されていないルックアップトランスレーションの作成

式からルックアップを実行するときは、接続されていないルックアップトランスフォーメーションを作成します。フラットファイル、参照テーブル、またはリレーショナルデータオブジェクトに対しては、再利用可能なルックアップトランスフォーメーションまたは再利用不可能な接続されていないルックアップトランスフォーメーションを作成できます。

1. **【Object Explorer】** ビューで、プロジェクトまたはフォルダーを選択します。
2. **【ファイル】** > **【新規】** > **【トランスフォーメーション】** をクリックします。
3. ルックアップウィザードに移動します。
4. **【フラットファイルデータオブジェクトルックアップ】**、**【参照テーブルルックアップ】**、または **【リレーショナルデータオブジェクトルックアップ】** を選択します。
5. **【次へ】** をクリックします。  
**【新しいルックアップ】** ダイアログボックスが表示されます。

6. Developer ツールで、物理データオブジェクトまたは参照テーブルを選択します。
7. トランスフォーメーションの名前を入力します。
8. **【複数の一致の検出時】** で、ルックアップ条件に一致する行が複数見つかった場合にルックアップトランスフォーメーションが返す行を設定します。接続されていないルックアップでは、**【すべてを返す】** は選択しないでください。
9. **【完了】** をクリックします。  
ルックアップトランスフォーメーションがエディタに表示されます。
10. **【概要】** ビューの **【ポート】** セクションで、トランスフォーメーションにポートを追加します。  
:LKP 式の各引数について入力ポートを作成します。作成する各ルックアップ条件について入力ポートを作成します。1つの入力ポートを複数の条件で使用できます。
11. **【概要】** ビューの **【ポート】** セクションで、1つのポートを戻りポートとして設定します。
12. **【ルックアップ】** ビューで、トランスフォーメーションの入力値をルックアップソースまたはキャッシュの値と比較するための1つ以上のルックアップ条件を追加します。  
条件が True の場合、ルックアップは戻りポートの値を返します。ルックアップ条件が false であれば、ルックアップは NULL を返します。
13. アグリゲータトランスフォーメーション、式トランスフォーメーション、アップデートストラテジトランスフォーメーションなどの式を使用できるトランスフォーメーションのポートに対して:LKP 式を作成します。
14. マッピングを作成する際は、接続されていないルックアップトランスフォーメーションをエディタ内のマッピングに追加しますが、マッピング内の他のトランスフォーメーションにポートを接続しないでください。

## 接続されていないルックアップの例

カリフォルニアの小売店が、州内の顧客に販売する商品の各価格に州の消費税を追加します。税額は、顧客が居住する郡によって異なります。消費税を取得するために、郡名を受け取ってその郡の消費税額を返すルックアップトランスフォーメーションを作成します。その郡が消費税を課さない場合、ルックアップトランスフォーメーションは NULL を返します。ルックアップは式トランスフォーメーションから呼び出します。

郡別の消費税の接続されていないルックアップトランスフォーメーションを設定するには、以下の手順に従ってください。

1. 郡別の消費税額を含む、フラットファイルの物理データオブジェクトをインポートします。
2. 接続されていないルックアップトランスフォーメーションを作成します。
3. ルックアップトランスフォーメーションに入力ポートを追加します。
4. 戻りポートを定義します。
5. ルックアップ条件を作成します。
6. ルックアップは式トランスフォーメーションから呼び出します。

### 手順 1。消費税ルックアップソースのモデルリポジトリへのインポート

ルックアップトランスフォーメーションを作成する前に、消費税ファイルがモデルリポジトリに含まれている必要があります。このシナリオでは、消費税ファイルには Sales\_County と County\_SalesTax の2つのフィールドが含まれています。Sales\_County は郡名を含む文字列です。County\_SalesTax は、その郡の税率を含む decimal フィールドです。消費税率ファイルがルックアップソースです。

## 手順 2. 接続されていないルックアップトランスレーションの作成

消費税フラットファイルのデータオブジェクトが含まれる、再利用可能なフラットファイルルックアップトランスフォーメーションを作成します。このシナリオでは、トランスフォーメーション名は Sales\_Tax\_Lookup です。[複数一致] で [最初の行を返す] を選択します。

## 手順 3. ルックアップトランスフォーメーションのポートの定義

[プロパティ] ビューの [ポート] タブで、ルックアップトランスフォーメーションのポートを定義します。

| ポートタイプ | 名前        | タイプ     | 長さ | スケール |
|--------|-----------|---------|----|------|
| 入力     | In_County | String  | 25 |      |
| 出力     | SalesTax  | Decimal | 3  | 3    |

## 手順 4. ルックアップトランスフォーメーションの戻りポートの設定

戻りポートは、ルックアップが取得するフラットファイル内のフィールドです。[カラム] タブの County\_SalesTax カラムが戻りポートです。

ルックアップが True の場合、データ統合サービスはフラットファイルのソースから郡を検索します。データ統合サービスが戻りポートに消費税の値を返します。データ統合サービスが郡を検出できなかった場合、ルックアップの結果は False となり、データ統合サービスが戻りポートに NULL を返します。

## 手順 5. ルックアップ条件の定義

[ルックアップ] ビューで、ルックアップ条件を定義して、入力値をルックアップソースの値と比較します。

ルックアップ条件を追加するには、[ルックアップカラム] をクリックします。

ルックアップ条件の構文は次のとおりです。

```
SALES_COUNTY = IN_COUNTY
```

## 手順 6. 式トランスフォーメーションの作成

フラットファイルから販売レコードを受け取る式トランスフォーメーションを作成します。式トランスフォーメーションは、顧客番号、販売額、および販売した郡を受け取ります。式トランスフォーメーションは、顧客番号、販売額、および消費税を返します。

式トランスフォーメーションは、以下のポートを備えています。

| ポートタイプ | 名前       | タイプ     | 長さ | 精度 | デフォルト値 |
|--------|----------|---------|----|----|--------|
| 入力     | County   | String  | 25 | 10 |        |
| パススルー  | Customer | String  | 10 |    |        |
| パススルー  | SalesAmt | Decimal | 10 | 2  |        |
| 出力     | SalesTax | Decimal | 10 | 2  | 0      |

SalesTax ポートには LKP 式が含まれています。式は、Sales\_Tax\_Lookup トランスフォーメーションを呼び出し、郡名をパラメータとして渡します。Sales\_Tax\_Lookup トランスフォーメーションは、消費税率を式に返します。式トランスフォーメーションは、税率に販売額を掛けます。

SalesTax ポートに以下の式を入力します。

```
(:LKP.Sales_Tax_Lookup(County) * SalesAmt)
```

SalesTax ポートには式の結果が含まれています。ルックアップが失敗すると、ルックアップトランスフォーメーションは NULL を返し、SalesTax ポートには NULL 値が含まれます。

SalesTax ポートに NULL 値が含まれるかどうかを確認する式を追加できます。SalesTax が NULL の場合はゼロを返すように、SalesTax ポートを設定できます。NULL 値が含まれるかどうかを確認してゼロを返すには、以下のルックアップ式を追加します。

```
IIF(ISNULL(:LKP.Sales_Tax_Lookup(County) * SalesAmt),0, SalesTax)
```



# 非ネイティブ環境でのルックアップトランスフォーメーション

非ネイティブ環境でのルックアップトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチおよびストリーミングのマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

## Blaze エンジンでのルックアップトランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

マッピング検証は、次の場合に失敗します。

- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。

マッピング内でルックアップトランスフォーメーションとして Sqoop を使用するデータオブジェクトを追加する場合、データ統合サービスは、Sqoop を介してマッピングを実行しません。JDBC を介してマッピングを実行します。

## Spark エンジンでのルックアップトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ルックアップトランスフォーメーション内のルックアップ条件に、バイナリデータ型が含まれる。
- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。

マッピングは、次の場合に失敗します。

- トランスフォーメーションが未接続であり、ジョイナトランスフォーメーションまたは Java トランスフォーメーションと一緒に使用された。

### 複数一致

複数一致で最初、最後、または任意の値を返すように選択すると、ルックアップトランスフォーメーションは任意の値を返します。

トランスフォーメーションが複数一致でエラーを報告するように設定した場合、Spark エンジンは重複した行を削除し、それらの行をログに含めません。

**注:** HBase ルックアップで一致が検出されない場合は、すべてのカラムが NULL 値である行が生成されます。ルックアップトランスフォーメーションの後にフィルタトランスフォーメーションを追加して、NULL 値を除外することができます。



## ストリーミングマッピングでのルックアップトランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

### マッピングの検査

マッピング検証は、次の場合に失敗します。

- ルックアップがデータオブジェクトである。
- アグリゲータトランスフォーメーションが、不等式ルックアップ条件を使用して設定されたルックアップトランスフォーメーションと同じストリーミングパイプライン内にある。
- ランクトランスフォーメーションが、不等式ルックアップ条件を使用して設定されたルックアップトランスフォーメーションと同じストリーミングパイプライン内にある。
- パイプラインには、不等式条件で設定された複数のパッシブルックアップトランスフォーメーションが含まれています。

次のような場合に、マッピングが失敗します。

- トランスフォーメーションが未接続。

### 一般的なガイドライン

次の一般的なガイドラインを考慮してください。

- データの検索に浮動小数点データ型を使用すると、予期しない結果が返される場合があります。
- ルックアップトランスフォーメーションを使用して、フラットファイル、HDFS データ、Hive データ、リレーショナルデータ、および HBase データ内のデータを検索します。
- DataFrames のクロス結合を回避するために、NULL 値での一致を無視するようにルックアップトランスフォーメーションを設定します。

### HBase ルックアップ

キャッシュされていない HBase テーブルに対してルックアップトランスフォーメーションを使用するには、次の手順を実行します。

1. HBase データオブジェクトを作成します。HBase データオブジェクトのリソースとして HBase テーブルを追加するときに、ROW ID カラムを含めます。
2. HBase データ読み取り操作を作成して、ストリーミングマッピングにインポートします。
3. マッピングにデータ操作をインポートする場合は、**【ルックアップ】** オプションを選択します。
4. **【ルックアップ】** タブで、以下のオプションを設定します。
  - **【ルックアップカラム】**。ROW ID の等式条件を指定します。
  - **演算子**。指定 =
5. HBase テーブル内のすべての日付値の形式が有効な Java 日付形式であることを確認します。この形式は、データオブジェクト読み取り操作の **【詳細プロパティ】** タブの **【日付/時刻形式】** プロパティで指定します。

**注:** HBase ルックアップで一致が検出されない場合は、すべてのカラムが NULL 値である行が生成されます。ルックアップトランスフォーメーションの後にフィルタトランスフォーメーションを追加して、NULL 値を除外することができます。

マッピング検証は、次の場合に失敗します。

- 条件に ROW ID が含まれていない。
- このトランスフォーメーションに IF 条件が含まれている。
- トランスフォーメーションに複数の条件が含まれている。

- 入力カラムが日付型である。

## Databricks Spark エンジンでのルックアップトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。
- ルックアップトランスフォーメーション内のルックアップ条件に、バイナリデータ型が含まれる。
- キャッシュが共有、名前付き、動的、キャッシュを使用しないように設定された。キャッシュは静的キャッシュである必要があります。
- ルックアップソースは Microsoft Azure SQL Data Warehouse ではありません。

マッピングは、次の場合に失敗します。

- トランスフォーメーションが未接続であり、ジョイナトランスフォーメーションと一緒に使用された。

### 複数一致

複数一致で最初、最後、または任意の値を返すように選択すると、ルックアップトランスフォーメーションは任意の値を返します。

トランスフォーメーションが複数一致でエラーを報告するように設定した場合、Spark エンジンは重複した行を削除し、それらの行をログに含めません。

**注:** HBase ルックアップで一致が検出されない場合は、すべてのカラムが NULL 値である行が生成されます。ルックアップトランスフォーメーションの後にフィルタトランスフォーメーションを追加して、NULL 値を除外することができます。

## 第 26 章

# ルックアップキャッシュ

この章では、以下の項目について説明します。

- [ルックアップキャッシュの概要, 423 ページ](#)
- [ルックアップキャッシュのタイプ, 424 ページ](#)
- [キャッシュを使用しないルックアップ, 425 ページ](#)
- [静的ルックアップキャッシュ, 425 ページ](#)
- [永続ルックアップキャッシュ, 426 ページ](#)
- [動的ルックアップキャッシュ, 427 ページ](#)
- [共有ルックアップキャッシュ, 427 ページ](#)
- [キャッシュの比較, 429 ページ](#)
- [ルックアップのキャッシュのパーティション化, 429 ページ](#)

## ルックアップキャッシュの概要

ルックアップトランスフォーメーションを設定して、リレーショナルまたはフラットファイルのルックアップソースをキャッシュできます。大きなルックアップテーブルまたはファイルでルックアップキャッシングを有効にしてルックアップのパフォーマンスを向上させます。

統合サービスは、キャッシュを使用するルックアップトランスフォーメーションのデータの最初の行を処理するときにメモリにキャッシュを作成します。統合サービスでは、ソース行がルックアップトランスフォーメーションに入力されるとキャッシュが作成されます。キャッシュのメモリは、トランスフォーメーションで設定した量に基づいて割り当てられます。統合サービスではインデックスキャッシュに条件値が格納され、データキャッシュに出力値が格納されます。統合サービスは、トランスフォーメーションに入力される各行のキャッシュに対してクエリを実行します。

データがメモリキャッシュに入らない場合、統合サービスはオーバーフローした値をキャッシュファイルに格納します。統合サービスは、キャッシュディレクトリにキャッシュファイルを作成します。デフォルトでは、データ統合サービスは CacheDir システムパラメータで指定されるディレクトリにキャッシュファイルを作成します。マッピングが完了すると、永続キャッシュを使用するようにルックアップトランスフォーメーションを設定していない限り、統合サービスはキャッシュメモリを解放し、キャッシュファイルを削除します。

フラットファイルルックアップまたはルックアップを使用する場合、統合サービスはルックアップソースをキャッシュします。ソート済み入力用にフラットファイルルックアップを設定する場合、条件カラムがグループ化されていない場合は統合サービスはルックアップをキャッシュできません。カラムがグループ化されていてソートされていない場合、統合サービスはソート済み入力の設定されていない場合と同様にルックアップを処理します。

ルックアップトランスフォーメーションにキャッシュを設定していない場合、統合サービスは各入力行のルックアップソースに対してクエリを実行します。ルックアップソースをキャッシュに格納するかどうかに関わらず、ルックアップクエリの結果および処理は同じです。ただし、ルックアップキャッシングを有効にすると、大きなルックアップソースでルックアップパフォーマンスを向上させることができます。

## ルックアップキャッシュのタイプ

異なるタイプのルックアップキャッシュを設定できます。例えば、同じマッピングの複数のルックアップトランスフォーメーション間でキャッシュを共有させる場合には、共有キャッシュが設定できます。

次のタイプのルックアップキャッシュを設定できます。

### 静的キャッシュ

統合サービスがルックアップを処理している間、静的キャッシュは変化しません。統合サービスは、ルックアップを処理するたびに静的キャッシュを再構築します。ルックアップトランスフォーメーションのキャッシュを有効にすると、統合サービスはデフォルトで静的キャッシュを作成します。統合サービスは、最初のルックアップ要求を処理するときにキャッシュを構築します。ルックアップトランスフォーメーションに入力される各行についてキャッシュにある値をルックアップします。ルックアップ条件が True の場合、統合サービスはルックアップキャッシュからの値を返します。

静的キャッシュは以下の理由で使用されます。

- マッピングの実行中、ルックアップソースが変化しない。
- ルックアップは接続されていないルックアップである。接続されていないルックアップには静的キャッシュを使用する必要があります。
- パフォーマンスを向上するため。統合サービスはルックアップトランスフォーメーションの処理中にキャッシュを更新しないため、動的キャッシュでのルックアップトランスフォーメーションよりも速い静的キャッシュでのルックアップトランスフォーメーションを処理します。
- ルックアップ条件が false のとき、接続されたトランスフォーメーションにはデフォルト値を、接続されていないトランスフォーメーションには NULL を返すように統合サービスを設定できます。

### 永続キャッシュ

永続キャッシュは、統合サービスがルックアップを処理しても変更されません。統合サービスはルックアップキャッシュファイルを保存し、そのキャッシュを使用するように設定されているルックアップトランスフォーメーションを次回処理する際に再利用します。永続キャッシュは、ルックアップソースが変化しない場合に使用します。

ルックアップトランスフォーメーションは、必要に応じて永続ルックアップキャッシュを再構築するように設定できます。

### 動的キャッシュ

動的ルックアップキャッシュは、統合サービスがルックアップを処理する間に変化します。統合サービスは、最初のルックアップ要求を処理する際に動的ルックアップキャッシュを構築します。統合サービスは各行の処理時に、データを動的にルックアップキャッシュに挿入または更新し、そのデータをターゲットに渡します。動的キャッシュはターゲットと同期しています。

動的キャッシュは、新規または変更されたレコードに従ってターゲットを更新するときに使用します。マッピングでターゲットデータのルックアップが必要な場合に、動的なキャッシュを使用することもできますが、ターゲットへの接続は遅くなります。

### 共有キャッシュ

共有キャッシュは、同じマッピングの複数のルックアップトランスフォーメーションで使用できます。共有キャッシュを使用してマッピングのパフォーマンスを向上させます。各ルックアップトランスフォーメ

ーションに別々にルックアップキャッシュを生成するのではなく、統合サービスは1つのキャッシュを生成します。

## キャッシュを使用しないルックアップ

キャッシュを使用しないルックアップとは、統合サービスがルックアップソースをキャッシュしない場合です。デフォルトでは、統合サービスはルックアップトランスフォーメーションにルックアップキャッシュを使用しません。

統合サービスは、キャッシュを使用しないルックアップを、キャッシュを使用するルックアップと同じ方法で処理します。ただし、ルックアップキャッシュの作成およびルックアップキャッシュに対するクエリ実行の代わりに、ルックアップソースに対してクエリを実行します。

ルックアップ条件が true の場合、統合サービスはルックアップソースからの値を返します。接続されたルックアップトランスフォーメーションを処理する場合、統合サービスはルックアップ/出力ポートが表す値を返します。未接続のルックアップトランスフォーメーションを処理する場合、統合サービスは戻りポートが表す値を返します。

条件が True でない場合、統合サービスは NULL またはデフォルト値を返します。接続されたルックアップトランスフォーメーションを処理する際に条件が満たされない場合、統合サービスは出力ポートのデフォルト値を返します。未接続のルックアップトランスフォーメーションを処理する際に条件が満たされない場合、統合サービスは NULL を返します。

## 静的ルックアップキャッシュ

静的ルックアップキャッシュは、統合サービスがルックアップトランスフォーメーションを処理する際に更新しないキャッシュです。ルックアップトランスフォーメーションにキャッシュを設定した場合、統合サービスではデフォルトで静的ルックアップキャッシュが作成されます。

統合サービスは、最初のルックアップ要求を処理するときにキャッシュを構築します。トランスフォーメーションに渡す各行に対するルックアップ条件に基づいて、キャッシュに問い合わせます。

ルックアップ条件が true の場合、統合サービスは静的ルックアップキャッシュからの値を返します。接続されたルックアップトランスフォーメーションを処理する場合、統合サービスはルックアップ/出力ポートが表す値を返します。未接続のルックアップトランスフォーメーションを処理する場合、統合サービスは戻りポートが表す値を返します。

条件が True でない場合、統合サービスは NULL またはデフォルト値を返します。接続されたルックアップトランスフォーメーションを処理する際に条件が満たされない場合、統合サービスは出力ポートのデフォルト値を返します。未接続のルックアップトランスフォーメーションを処理する際に条件が満たされない場合、統合サービスは NULL を返します。

# 永続ルックアップキャッシュ

永続ルックアップキャッシュは、同じマッピングの複数の実行に対して統合サービスが再利用するキャッシュです。実行されるマッピング間でルックアップソースが変化しない場合には永続キャッシュを使用します。

ルックアップトランスフォーメーションでルックアップキャッシュを有効にした場合、統合サービスではデフォルトで非永続キャッシュが使用されます。マッピングが完了すると、統合サービスはキャッシュファイルを削除します。次にマッピングを実行したときに、統合サービスはルックアップソースからメモリキャッシュを作成します。

ルックアップトランスフォーメーションを設定して永続ルックアップキャッシュを使用する場合、統合サービスはマッピングの複数の実行に対してキャッシュファイルを保存して再使用します。永続キャッシュによって、ルックアップテーブルの読み取りとルックアップキャッシュの再構築にかかる時間がなくなります。

永続ルックアップキャッシュを使用して最初にマッピングを実行するときは、統合サービスはキャッシュファイルをディスクに保存します。次に統合サービスがマッピングを実行したときは、メモリキャッシュをキャッシュファイルから作成します。

元のルックアップソースが変更された場合は永続ルックアップキャッシュを再構築するように、統合サービスを設定できます。キャッシュを再構築すると、統合サービスは新しいキャッシュファイルを作成し、統合サービスログにメッセージを書き込みます。

## 永続ルックアップキャッシュの再構築

統合サービスは、永続ルックアップキャッシュを再構築するように設定できます。永続ルックアップキャッシュの再構築を設定していない場合でも、統合サービスは永続ルックアップキャッシュを再構築することがあります。

永続ルックアップキャッシュを再構築するときは、次のルールおよびガイドラインに従ってください。

- 最後にキャッシュを構築してからルックアップソースが変更されている場合、統合サービスは永続ルックアップキャッシュを再構築します。
- マッピングにキャッシュを共有する 1 つ以上のルックアップトランスフォーメーションが含まれる場合はキャッシュを再構築できます。
- マッピングの各実行の間にルックアップテーブルが変更されない場合は、永続ルックアップキャッシュを使用するようにルックアップトランスフォーメーションを設定します。統合サービスは、キャッシュファイルを保存して再利用することにより、ルックアップテーブルの読み取りにかかる時間を節約します。
- それ以降のルックアップトランスフォーメーションにルックアップキャッシュの再構築を設定した場合、統合サービスはそれら进行处理するときにキャッシュを再構築せずに共有します。
- マッピングに 2 つの永続ルックアップがあり、2 番目のルックアップトランスフォーメーションにキャッシュの再構築を設定すると、統合サービスは両方のルックアップに対して永続ルックアップキャッシュを再構築します。

統合サービスは、次のような状況で永続ルックアップキャッシュを再構築します。

- 統合サービスはキャッシュファイルを見つけることができません。
- 統合サービスはキャッシュを再利用できません。この場合、ルックアップキャッシュを再構築するか、またはマッピングに失敗します。
- 統合サービスで高精度を有効または無効にします。
- ルックアップトランスフォーメーションまたはマッピングを編集する。  
**注:** トランスフォーメーションの説明を編集すると、統合サービスはマッピングを再構築しません。
- パーティションの数を変更します。
- ルックアップソースへのアクセスに使用するデータベース接続またはファイルの場所を変更します。

- Unicode モードのソート順を変更します。
- 統合サービスのコードページを変更します。

## 動的ルックアップキャッシュ

動的キャッシュは、統合サービスが各行を処理する際に更新されるキャッシュです。動的ルックアップキャッシュを使用してキャッシュがターゲットに同期し続けるようにします。

動的キャッシュをリレーショナルルックアップおよびフラットファイルルックアップと一緒に使用できます。統合サービスは、最初のルックアップ要求を処理するときにキャッシュを構築します。ルックアップトランスフォーメーションに渡す各行に対するルックアップ条件に基づいて、キャッシュにクエリを実行します。統合サービスは、各行を処理するときにルックアップキャッシュを更新します。

ルックアップクエリの結果、行のタイプ、ルックアップトランスフォーメーションのプロパティによって、統合サービスは、キャッシュ内の行を挿入または更新するか、またはキャッシュを変更しません。

## 共有ルックアップキャッシュ

共有ルックアップキャッシュは、マッピング内の複数のルックアップトランスフォーメーションによって共有される静的ルックアップキャッシュです。共有ルックアップキャッシュを使用してキャッシュの構築に必要な時間を短縮できます。

デフォルトでは、統合サービスによりマッピング内の互換性のあるキャッシュ構造体を持つルックアップトランスフォーメーションのキャッシュが共有されます。例えば、1つのマッピングに同じ再利用可能なルックアップトランスフォーメーションのインスタンスが2つあり、両方のインスタンスに同じ出力ポートを使用する場合、2つのルックアップトランスフォーメーションはデフォルトでルックアップキャッシュを共有します。

統合サービスでは、最初のルックアップトランスフォーメーションが処理される際にキャッシュが作成されます。キャッシュを共有するそれ以降のルックアップトランスフォーメーションの処理には、同じキャッシュが使用されます。統合サービスがルックアップキャッシュを共有する場合、統合サービスログにメッセージが書き込まれます。

統合サービスでは、データキャッシュメモリおよびインデックスキャッシュメモリを最初のルックアップトランスフォーメーションに割り当てます。ルックアップキャッシュを共有するそれ以降のルックアップトランスフォーメーションには、メモリを追加で割り当てません。

トランスフォーメーションまたはキャッシュ構造によって共有が許可されていない場合、統合サービスは新しいキャッシュを作成します。

## ルックアップキャッシュの共有のルールおよびガイドライン

ルックアップキャッシュを共有する場合は、次のルールとガイドラインを考慮します。

- 1つ以上の静的キャッシュを動的ルックアップで共有できます。同じマッピングで動的ルックアップが静的ルックアップとキャッシュを共有する場合、静的ルックアップは動的ルックアップが作成したキャッシュを再使用します。
- 複数の動的ルックアップの間でキャッシュを共有することはできません。



- 複数のルックアップトランスフォーメーションが永続ルックアップキャッシュを再構築するように設定すると、統合サービスは最初のルックアップトランスフォーメーションにキャッシュを構築し、それ以降のルックアップトランスフォーメーションの永続ルックアップキャッシュは共有します。
- 最初のルックアップトランスフォーメーションは永続ルックアップキャッシュを再構築しないが、それ以降のルックアップトランスフォーメーションはキャッシュを再構築するように設定すると、トランスフォーメーションはキャッシュを共有できません。統合サービスは、各ルックアップトランスフォーメーションの処理時にキャッシュを構築します。
- それ以降のルックアップトランスフォーメーションのルックアップ/出力ポートは、統合サービスがキャッシュを構築するのに使用するルックアップトランスフォーメーションのポートと一致するか、またはそれらのサブセットである必要があります。ポートの順序が一致する必要はありません。
- キャッシュを共有するルックアップトランスフォーメーションには、次の特性が必要です。
  - ルックアップトランスフォーメーションはルックアップ条件で同じポートを使用する必要があります。
  - ルックアップトランスフォーメーションは、SQL オーバーライドを使用する場合には、同じものを使用する必要があります。
  - ルックアップキャッシングはすべてのルックアップトランスフォーメーションで有効にする必要があります。
  - ルックアップトランスフォーメーションは同じタイプのルックアップソースを使用する必要があります。
  - すべてのリレーショナルルックアップトランスフォーメーションは、同じデータベース接続を使用する必要があります。
  - ルックアップトランスフォーメーションは同じルックアップテーブル名を使用する必要があります。
  - すべてのルックアップトランスフォーメーションのキャッシュの構造には互換性がある必要があります。



## キャッシュの比較

統合サービスの実行は、設定するルックアップキャッシュのタイプに基づいてそれぞれ異なります。

次の表に、ルックアップトランスフォーメーションと、キャッシュを使用しないルックアップ、静的キャッシュおよび動的キャッシュとの比較を示します。

| 使用なし                                                                                                                                         | 静的キャッシュ                                                                                                                                      | 動的キャッシュ                                                                                                                                                                                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 統合サービスはキャッシュを挿入および更新しません。                                                                                                                    | 統合サービスはキャッシュを挿入および更新しません。                                                                                                                    | 統合サービスは、行をターゲットに渡すときに、キャッシュに行を挿入またはキャッシュ内の行を更新します。                                                                                                                                                                                                                  |
| リレーショナルルックアップを使用できます。                                                                                                                        | リレーショナルルックアップまたはフラットファイルルックアップを使用できます。                                                                                                       | リレーショナルルックアップまたはフラットファイルルックアップを使用できます。                                                                                                                                                                                                                              |
| 条件が True の場合、統合サービスはルックアップテーブルまたはキャッシュから値を返します。<br>条件が True でない場合、統合サービスは接続されたトランスフォーメーションについてはデフォルト値を、接続されていないトランスフォーメーションについては NULL を返します。 | 条件が True の場合、統合サービスはルックアップテーブルまたはキャッシュから値を返します。<br>条件が True でない場合、統合サービスは接続されたトランスフォーメーションについてはデフォルト値を、接続されていないトランスフォーメーションについては NULL を返します。 | 条件が True の場合、統合サービスは行タイプに基づいて、キャッシュの行を更新するか、またはキャッシュを未変更のままにします。これは、キャッシュおよびターゲットテーブルに行があることを示しています。更新された行をターゲットテーブルに渡すことができます。<br>条件が True ではない場合、統合サービスは行タイプに基づいて、キャッシュに行を挿入するか、またはキャッシュを未変更のままにします。これは、キャッシュまたはターゲットに行がなかったことを示しています。挿入された行をターゲットテーブルに渡すことができます。 |

## ルックアップのキャッシュのパーティション化

キャッシュのパーティション化では、各パーティションごとに、アグリゲータ、ジョイナ、ランクまたはルックアップの各トランスフォーメーションを処理する個別のキャッシュが作成されます。キャッシュのパーティション化により、各パーティションは並行して個別のキャッシュにクエリを実行するので、マッピングのパフォーマンスが向上します。

統合サービスがマッピングにパーティションを作成する場合、統合サービスはパーティション化されたルックアップトランスフォーメーションにキャッシュのパーティション化を使用することがあります。

次の条件が当てはまる場合、統合サービスは接続されたルックアップトランスフォーメーションにキャッシュのパーティション化を使用します。

- ルックアップ条件に等号演算子だけが含まれる。
- 接続されたルックアップトランスフォーメーションがリレーショナルテーブルのデータを検索する場合、データベースは大文字と小文字が区別された比較に対して設定されます。

統合サービスは接続されていないルックアップトランスフォーメーションにはキャッシュのパーティション化を使用しません。

統合サービスがルックアップトランスフォーメーションにキャッシュのパーティション化を使用しないときには、ルックアップトランスフォーメーションのすべてのパーティションは同じキャッシュを共有します。各パーティションは同じキャッシュに連続してクエリを実行します。

## 第 27 章

# 動的ルックアップキャッシュ

この章では、以下の項目について説明します。

- [動的ルックアップキャッシュの概要, 430 ページ](#)
- [動的ルックアップキャッシュへの使用, 431 ページ](#)
- [動的ルックアップキャッシュプロパティ, 431 ページ](#)
- [動的ルックアップキャッシュおよび出力値, 433 ページ](#)
- [ルックアップトランスフォーメーションの値, 433 ページ](#)
- [SQL オーバーライドおよび動的ルックアップキャッシュ, 436 ページ](#)
- [動的ルックアップキャッシュのマッピング設定, 437 ページ](#)
- [条件付き動的ルックアップキャッシュの更新, 439 ページ](#)
- [式の結果を使用した動的キャッシュの更新, 440 ページ](#)
- [動的ルックアップキャッシュの例, 441 ページ](#)
- [動的ルックアップキャッシュのルールとガイドライン, 442 ページ](#)

## 動的ルックアップキャッシュの概要

動的ルックアップキャッシュを使用してキャッシュがターゲットに同期し続けるようにします。動的キャッシュは、リレーショナルルックアップまたはフラットファイルルックアップとともに使用できます。

統合サービスは、最初のルックアップ要求を処理する際に動的ルックアップキャッシュを構築します。トランスフォーメーションに渡す各行に対するルックアップ条件に基づいて、キャッシュに問い合わせます。統合サービスは、各行を処理するときにルックアップキャッシュを更新します。

ルックアップクエリの結果、行タイプ、ルックアップトランスフォーメーションのプロパティに基づいて、統合サービスは、ソースから行を読み取るときに、動的ルックアップキャッシュに以下のアクションのいずれかを実行します。

### キャッシュに行を挿入

行がキャッシュになく、行をキャッシュに挿入するようにルックアップトランスフォーメーションを設定している場合、統合サービスは行をキャッシュに挿入します。入力ポートまたは生成されたシーケンス ID に基づいて行をキャッシュに挿入するように、トランスフォーメーションを設定することができます。統合サービスはその行に挿入のフラグを設定します。

### キャッシュ内の行を更新

行がキャッシュにあり、キャッシュの行を更新するようにルックアップトランスフォーメーションを設定している場合、統合サービスは行を更新します。統合サービスは入力ポートに基づいてキャッシュ内の行を更新します。統合サービスはその行に更新行のフラグを設定します。

### キャッシュに変更を加えない

行がキャッシュにあり、新しい行のみをキャッシュに挿入するようにルックアップトランスフォーメーションを設定している場合、統合サービスは何も変更しません。または、行がキャッシュになく、ユーザが既存の行のみを更新すると指定した場合です。もしくは、行がキャッシュにあっても、ルックアップ条件に基づき、何も変更されない場合です。統合サービスはその行に未変更のフラグを設定します。

NewLookupRow の値に基づいて、動的ルックアップトランスフォーメーションとともにルータトランスフォーメーションまたはフィルタトランスフォーメーションを設定して、挿入行または更新行をターゲットテーブルにルーティングできます。変更のない行を別のターゲットテーブルやフラットファイルにルーティングしたり、それらを削除したりできます。

## 動的ルックアップキャッシュへの使用

動的ルックアップキャッシュとともに使用しているルックアップトランスフォーメーションがルックアップソースの変更に基づいてキャッシュを更新するように設定できます。

動的ルックアップキャッシュは以下の理由で使用することがあります。

**新規および更新した顧客情報でマスタ顧客テーブルを更新する。**

例えば、ターゲットに顧客が存在するかどうかを判断するために、ルックアップトランスフォーメーションを使用して顧客テーブルにルックアップを実行できます。キャッシュは顧客テーブルを表します。ルックアップトランスフォーメーションは、行をターゲットに渡すときに、キャッシュに行を挿入またはキャッシュ内の行を更新します。

**リレーショナルテーブルの代わりにエクスポートされたフラットファイルをルックアップソースとして使用します。**

データベースへの接続に時間がかかる場合は、リレーショナルテーブルコンテンツをフラットファイルにエクスポートしてそのファイルをルックアップソースとして使用できます。例えば、データベースへの ODBC 接続に時間がかかる場合に、この方法の使用が必要になる場合があります。マッピングでデータベーステーブルをリレーショナルターゲットとして設定し、ルックアップキャッシュの変更をデータベーステーブルに戻すことができます。

## 動的ルックアップキャッシュプロパティ

動的ルックアップのプロパティを設定して動的ルックアップキャッシュを有効にし、キャッシュが更新される方法を設定します。例えば、動的キャッシュで挿入または更新される値を設定できます。

動的ルックアップキャッシュを有効にする場合は以下のプロパティを設定します。

### 複数の一致の検出時

[エラーを報告] に設定されます。

### 動的ルックアップキャッシュ

動的ルックアップキャッシュを有効にします。

このオプションは、ルックアップキャッシュを有効にした後に使用できます。

### 更新でなければ挿入

行タイプが更新で、ルックアップトランスフォーメーションに入力される行に適用されます。有効にすると、統合サービスはキャッシュの既存の行を更新し、新しい行の場合は挿入します。無効にすると、統合サービスは新しい行を挿入しません。

このオプションは、動的キャッシュを有効にした後に使用できます。

### 挿入でなければ更新

行タイプが挿入で、ルックアップトランスフォーメーションに入力される行に適用されます。有効にすると、統合サービスはキャッシュに行を挿入し、既存の行を更新します。無効にすると、統合サービスは既存の行を更新しません。

このオプションは、動的キャッシュを有効にした後に使用できます。

### 更新時に古い値を出力

ルックアップトランスフォーメーションは、キャッシュから既存の値または新しい値を出力できます。有効にすると、統合サービスはキャッシュで値を更新する前にルックアップ/出力ポートから既存の値を出力します。統合サービスがキャッシュの行を更新する場合、入力データに基づいて行を更新する前にルックアップキャッシュの値を出力します。統合サービスがキャッシュ内に行を挿入する場合、NULL 値を出力します。

統合サービスのプロパティを無効にして、ルックアップ/出力ポートと入出力ポートから同じ値を渡します。このプロパティはデフォルトで有効になっています。

このオプションは、動的キャッシュを有効にした後に使用できます。

### 動的キャッシュの更新条件

有効にすると、統合サービスは条件式を使用して、動的キャッシュを更新するかどうかを決定します。統合サービスは、条件が True でデータがキャッシュに存在する場合にキャッシュを更新します。

ルックアップポートまたは入力ポートを使用して式を作成します。式には、入力値またはルックアップキャッシュの値を含めることができます。デフォルトは true です。

このオプションは、動的キャッシュを有効にした後に使用できます。

### NewLookupRow

Developer ツールは、動的キャッシュを設定されたルックアップトランスフォーメーションに、このポートを追加します。

NewLookupRow プロパティには、次のいずれかの値を含めることができます。

- 0 = キャッシュを更新しない。
- 1 = キャッシュに行を挿入する。
- 2 = キャッシュの行を更新する。

ルックアップキャッシュとターゲットテーブルの同期を確保するには、NewLookupRow の値が 1 または 2 である場合に行をターゲットに渡します。

### 関連するポート

統合サービスは、キャッシュのデータを更新するときに関連ポートの値を使用します。統合サービスは、ルックアップ条件で指定される入力ポートとルックアップソースポートを関連付けます。動的ルックアップの残りのルックアップソースポートに対して関連するポートを設定する必要があります。動的ルックアップのすべてのルックアップソースに関連ポートを設定しない場合、マッピングの検証は失敗します。

ルックアップソースポートは次のオブジェクトに関連付けることができます。

| オブジェクト   | 説明                                                                         |
|----------|----------------------------------------------------------------------------|
| 入力ポート    | 入力ポートの値に基づいてキャッシュを更新します。                                                   |
| 関連付けられた式 | 選択して式を入力します。統合サービスは式の結果に基づいてキャッシュを更新します。                                   |
| シーケンス ID | ルックアップキャッシュに挿入された行にプライマリキーを生成します。シーケンス ID は bigint と int 行にのみ関連付けることができます。 |

#### 更新に対する NULL 入力を見捨てる

ユーザーが動的キャッシュを使用するようにルックアップトランスフォーメーションを設定したとき、Developer ツールはルックアップ/出力ポートに対してこのポートプロパティを有効にします。このプロパティは、統合サービスがキャッシュ内のカラムを NULL 入力値で更新しないようにする場合に選択します。

#### 比較で見捨てる

動的キャッシュを使用するようにルックアップトランスフォーメーションを設定したとき、Developer ツールはルックアップ条件に使用されていないルックアップ/出力ポートに対してこのポートプロパティを有効にします。統合サービスは、すべてのルックアップポートの値と、それらに関連付けられているポートの値をデフォルトで比較します。行を更新する前に値を比較して、統合サービスにポートを見捨てる場合はこのプロパティを選択します。このプロパティを使用して、比較のパフォーマンスを向上できます。

## 動的ルックアップキャッシュおよび出力値

動的ルックアップキャッシュを有効にする場合、出力ポートの値は動的ルックアップキャッシュの設定方法によって異なります。ルックアップ/出力ポートの出力値は、統合サービスが行を更新するときに古い値を出力するか、新しい値を出力するかのどちらかをユーザーが選択したかによって異なります。

**更新時に古い値を出力**プロパティを設定してルックアップ/出力ポートに以下の種類の出力値のいずれかを指定できます。

- 更新時に古い値を出力します。統合サービスは行の更新前にキャッシュに存在した値を出力します。
- 更新時に新しい値を出力します。統合サービスはキャッシュに書き込まれた更新値を出力します。ルックアップ/出力ポートの値は出力ポートの値と一致します。

## ルックアップトランスフォーメーションの値

ルックアップトランスフォーメーションには、入力ポート、ルックアップ、出力ポートの値が含まれます。動的ルックアップキャッシュを有効にする場合、出力ポートの値は動的ルックアップキャッシュの設定方法によって異なります。

ルックアップトランスフォーメーションには次のタイプの値が含まれます。

#### 入力値

統合サービスがルックアップトランスフォーメーションに渡す値。

### ルックアップ値

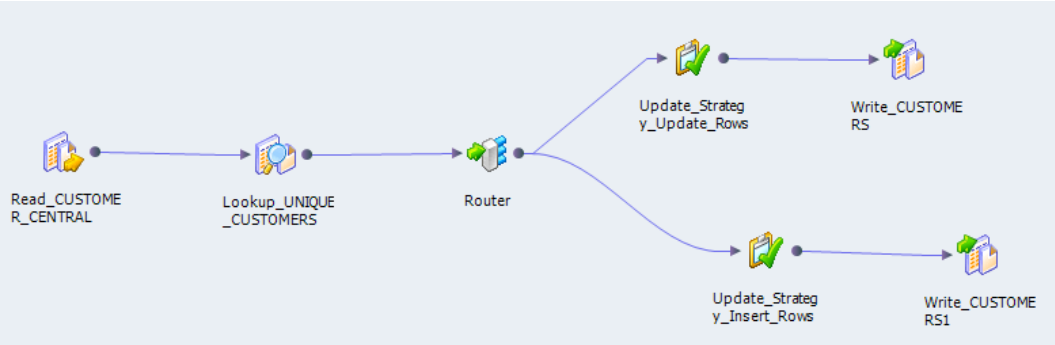
統合サービスがキャッシュに挿入する値。

### 出力値

統合サービスがルックアップトランスフォーメーションの出力ポートから渡す値。ルックアップ/出力ポートの出力値は、統合サービスが行を更新するときに古い値を出力するか、新しい値を出力するかのどちらをユーザーが選択したかによって異なります。

## ルックアップトランスフォーメーションの値の例

例えば、次のオブジェクトを使用してマッピングを作成します。



ルックアップトランスフォーメーションでは、動的ルックアップキャッシュを有効にし、次のルックアップ条件を定義します。

IN\_CUST\_ID = CUST\_ID

デフォルトでは、ルックアップトランスフォーメーションに入力されるすべての行の種類は「挿入」です。キャッシュおよびターゲットテーブルで挿入と更新の両方を実行するには、ルックアップトランスフォーメーションで【挿入でなければ更新】プロパティを選択します。

### 初期キャッシュ値

マッピングを実行すると、統合サービスはターゲットテーブルからルックアップキャッシュを構築します。

以下の表に、ルックアップキャッシュの初期値を示します。

| PK_PRIMARYKEY | CUST_ID | CUST_NAME    | ADDRESS           |
|---------------|---------|--------------|-------------------|
| 100001        | 80001   | Marion James | 100 Main St.      |
| 100002        | 80002   | Laura Jones  | 510 Broadway Ave. |
| 100003        | 80003   | Shelley Lau  | 220 Burnside Ave. |

### 入力値

ソースには、ターゲットテーブルに存在する行と存在しない行が含まれます。統合サービスは、ソース行をルックアップトランスフォーメーションに渡します。

以下の表に、ソース行を示します。

| SQ_CUST_ID | SQ_CUST_NAME  | SQ_ADDRESS        |
|------------|---------------|-------------------|
| 80001      | Marion Atkins | 100 Main St.      |
| 80002      | Laura Gomez   | 510 Broadway Ave. |

| SQ_CUST_ID | SQ_CUST_NAME | SQ_ADDRESS   |
|------------|--------------|--------------|
| 99001      | Jon Freeman  | 555 6th Ave. |

## ルックアップ値

統合サービスは、ルックアップ条件に応じてキャッシュ内の値をルックアップします。既存のカスタマ ID80001 および 80002 についてはキャッシュ内の行を更新します。カスタマ ID99001 についてはキャッシュ内に行を挿入します。統合サービスは新規行に対して新規キー（PK\_PRIMARYKEY）を生成します。

以下の表に、ルックアップから返される行と値を示します。

| PK_PRIMARYKEY | CUST_ID | CUST_NAME     | ADDRESS           |
|---------------|---------|---------------|-------------------|
| 100001        | 80001   | Marion Atkins | 100 Main St.      |
| 100002        | 80002   | Laura Gomez   | 510 Broadway Ave. |
| 100004        | 99001   | Jon Freeman   | 555 6th Ave.      |

## 出力行

統合サービスは、動的キャッシュに対して実行する挿入および更新に基づいて、ルックアップトランスフォーメーションの行にフラグを設定します。統合サービスは、最終的に行をルータトランスフォーメーションに渡し、ここで挿入行のブランチと更新行の別のブランチが作成されます。各ブランチには、アップデートストラテジトランスフォーメーションがあります。アップデートストラテジトランスフォーメーションは、NewLookupRow ポートの値に基づいて行に挿入行または更新のフラグを設定します。

ルックアップ/出力ポートと入出力ポートの出力値は、Integration Service が行を更新するときに古い値を出力するか、新しい値を出力するかのどちらかをユーザーが選択したかによって異なります。ただし、NewLookupRow ポートと、シーケンス ID を使用するルックアップ/出力ポートの出力値は新しい行と更新された行で同じになります。

新しい値を出力するように選択すると、ルックアップ/出力ポートは次の値を出力します。

| NewLookupRow | PK_PRIMARYKEY | CUST_ID | CUST_NAME     | ADDRESS           |
|--------------|---------------|---------|---------------|-------------------|
| 2            | 100001        | 80001   | Marion Atkins | 100 Main St.      |
| 2            | 100002        | 80002   | Laura Gomez   | 510 Broadway Ave. |
| 1            | 100004        | 99001   | Jon Freeman   | 555 6th Ave.      |

古い値を出力するように選択すると、ルックアップ/出力ポートは次の値を出力します。

| NewLookupRow | PK_PRIMARYKEY | CUST_ID | CUST_NAME    | ADDRESS           |
|--------------|---------------|---------|--------------|-------------------|
| 2            | 100001        | 80001   | Marion James | 100 Main St.      |
| 2            | 100002        | 80002   | Laura Jones  | 510 Broadway Ave. |
| 1            | 100004        | 99001   | Jon Freeman  | 555 6th Ave.      |

統合サービスがルックアップキャッシュ内の行を更新するときは、キャッシュ内およびターゲットテーブル内の行のプライマリキー（PK\_PRIMARYKEY）の値を使用します。

統合サービスはシーケンス ID を使用して、キャッシュに存在しない顧客のプライマリキーを生成します。統合サービスは、プライマリキー値をルックアップキャッシュに挿入し、その値をルックアップ/出力ポートに返します。

統合サービスは、入力値に一致する入出力ポートから値を出力します。

**注:** 入力値が NULL で、ユーザーが関連入力ポートに対して [NULL を無視] プロパティを選択した場合、入力値はルックアップ値または入出力ポートからの値と等しくなりません。[NULL を無視] プロパティを選択した場合、ターゲットに NULL 値を渡すと、ルックアップキャッシュとターゲットテーブルが非同期になる可能性があります。NULL 値をターゲットに渡さないことを確認してください。

## SQL オーバーライドおよび動的ルックアップキャッシュ

ルックアップクエリに WHERE 句を追加してキャッシュを作成するのに使用されるレコードをフィルタリングして、キャッシュを使用しないルックアップのデータベーステーブルでルックアップを実行します。ただし、統合サービスは動的キャッシュに行を挿入するときに WHERE 句を使用しません。

動的キャッシュを使ってルックアップトランスフォーメーションに WHERE 句を追加する場合、フィルタトランスフォーメーションをルックアップトランスフォーメーションの前に接続して、キャッシュまたはターゲットテーブルに挿入しない行をフィルタリングします。フィルタトランスフォーメーションを含めない場合、キャッシュとターゲットテーブルの間で結果に矛盾が生じる可能性があります。

たとえば、ルックアップトランスフォーメーションを設定して従業員テーブルの EMP に対して動的ルックアップを行い、EMP\_ID で一致する行を探すとします。下記のルックアップ SQL 上書きを定義します。

```
SELECT EMP_ID, EMP_STATUS FROM EMP ORDER BY EMP_ID, EMP_STATUS WHERE EMP_STATUS = 4
```

最初にマッピングを実行する際、統合サービスはルックアップ SQL オーバーライドに基づいてターゲットテーブルからルックアップキャッシュを作成します。キャッシュ内のすべての行は WHERE 句の条件、EMP\_STATUS = 4 に一致します。

例えば、統合サービスは指定したルックアップ条件に一致するソース行を読み取りますが、EMP\_STATUS の値は 2 です。ターゲットに EMP\_STATUS が 2 の行がある可能性があっても、統合サービスは SQL オーバーライドのため、キャッシュの行を見つけることができません。統合サービスはキャッシュに行を挿入して、行をターゲットテーブルに渡します。行がすでに存在する場合、統合サービスがこの行をターゲットテーブルに挿入すると、結果に矛盾が生じる可能性があります。さらに、キャッシュ内のすべての行が SQL オーバーライドの WHERE 句の条件に一致するとは限りません。

WHERE 句に一致する行のみを確実にキャッシュに挿入するには、フィルタトランスフォーメーションをルックアップトランスフォーメーションの前に追加し、ルックアップ SQL 上書きで WHERE 句内の条件をフィルタ条件として定義します。

上記の例では、次のフィルタトランスフォーメーションのフィルタ条件と SQL オーバーライドの WHERE 句を入力します。

```
EMP_STATUS = 4
```



# 動的ルックアップキャッシュのマッピング設定

動的キャッシュによるルックアップを使用する場合、動的ルックアップキャッシュを更新するようにマッピングを設定し、変更された行をターゲットに書き込む必要があります。

動的ルックアップキャッシュを使用するようにマッピングを設定するには、以下の手順を実行します。

**ルックアップトランスフォーメーションの入力行に「挿入」または「更新」のフラグを設定します。**

デフォルトでは、すべての入力行の行タイプは「挿入」です。アップデートストラテジトランスフォーメーションをルックアップトランスフォーメーションの前に追加して、入力行に別の行タイプを指定します。

**統合サービスが動的キャッシュの入力行を処理する方法を指定します。**

【挿入でなければ更新】オプションまたは【更新でなければ挿入】オプションを選択し、挿入または更新のフラグが設定された行を処理します。

**ターゲットに挿入される行とターゲットの更新される行に、マッピングパイプラインを別々に作成します。**

ルックアップトランスフォーメーションの後にフィルタトランスフォーメーションまたはルータトランスフォーメーションを追加して、挿入行または更新行を個別のマッピングブランチにルーティングします。NewLookupRow の値を使用して、各行に適切なブランチを指定します。

**ルックアップトランスフォーメーションの出力行に行タイプを設定します。**

アップデートストラテジトランスフォーメーションを追加して、行に挿入または更新のフラグを設定します。

## 挿入でなければ更新

**更新でなければ挿入** プロパティを使用して、行タイプが「挿入」のときに動的ルックアップキャッシュの既存の行を更新します。

このプロパティは、行タイプが「挿入」で、ルックアップトランスフォーメーションに入力される行に対して使用します。更新行などの他のタイプの行がルックアップトランスフォーメーションに入力される場合、【挿入でなければ更新】プロパティは統合サービスの行の処理方法に影響しません。

【挿入でなければ更新】プロパティを選択し、ルックアップトランスフォーメーションに入力される行のタイプが「挿入」の場合、統合サービスは、行が新規行のときはキャッシュに挿入します。インデックスキャッシュに行は存在するがデータキャッシュが現在の行と異なる場合に、統合サービスはデータキャッシュ内の行を更新します。

【挿入でなければ更新】を選択せず、ルックアップトランスフォーメーションに入力される行のタイプが「挿入」の場合、統合サービスは、行が新規行のときはキャッシュに挿入し、すでに存在する行のときはキャッシュを変更しません。

以下の表に、ルックアップトランスフォーメーションに入力される行のタイプが「挿入」の場合に、統合サービスがルックアップキャッシュを変更する方法を示します。

| 【挿入でなければ更新】オプション | キャッシュ内で行が見つかったか | データキャッシュが異なるか | ルックアップキャッシュの結果 | NewLookupRow 値 |
|------------------|-----------------|---------------|----------------|----------------|
| クリア - 挿入のみ       | はい              | -             | 変更なし           | 0              |
| クリア - 挿入のみ       | いいえ             | -             | 挿入             | 1              |
| 選択済み             | はい              | はい            | 更新             | 2 <sup>1</sup> |

| 【挿入でなければ更新】オプション | キャッシュ内で<br>行が見つかった<br>か | データキャッ<br>シュが異なる<br>か | ルックアップキ<br>ャッシュの結果 | NewLookupRow 値 |
|------------------|-------------------------|-----------------------|--------------------|----------------|
| 選択済み             | はい                      | いいえ                   | 変更なし               | 0              |
| 選択済み             | いいえ                     | -                     | 挿入                 | 1              |

1. ルックアップ条件にないすべてのルックアップポートに対して [NULL を無視] を選択し、それらのポートすべてに NULL 値が含まれている場合、統合サービスはキャッシュを変更せず、NewLookupRow 値は 0 となります。

## 更新でなければ挿入

挿入でなければ更新プロパティを使用して、行タイプが「更新」のときに動的ルックアップキャッシュの新規の行を挿入します。

ルックアップトランスフォーメーションで **【更新でなければ挿入】** プロパティを選択できます。このプロパティは、行タイプが「更新」で、ルックアップトランスフォーメーションに入力される行に対してのみ使用します。他のタイプの行、例えば挿入行がルックアップトランスフォーメーションに入力される場合、このプロパティは統合サービスの行の処理方法に影響しません。

このプロパティを選択した場合で、ルックアップトランスフォーメーションに入力される行のタイプが「更新」の場合、その行がすでにインデックスキャッシュに存在し、キャッシュデータが既存の行と異なっていれば、統合サービスはその行を更新します。統合サービスは、新規行の場合にその行をキャッシュに挿入します。

このプロパティを選択していない場合、ルックアップトランスフォーメーションに入力される行のタイプが「更新」のときは、統合サービスは行がすでに存在する行ならばキャッシュ内で更新し、行が新規行ならばキャッシュを変更しません。

ルックアップ条件にないすべてのルックアップポートに対して **[NULL を無視]** を選択し、それらのポートすべてに NULL 値が含まれている場合、統合サービスはキャッシュを変更せず、NewLookupRow 値は 0 となります。

以下の表に、ルックアップトランスフォーメーションに入力される行のタイプが「更新」の場合に、統合サービスがルックアップキャッシュを変更する方法を示します。

| 【更新でなければ挿入】オプション | キャッシュ内で<br>行が見つかった<br>か | データキャッ<br>シュが異なる<br>か | ルックアップキ<br>ャッシュの結果 | NewLookupRow 値 |
|------------------|-------------------------|-----------------------|--------------------|----------------|
| クリア（更新のみ）        | はい                      | はい                    | 更新                 | 2              |
| クリア（更新のみ）        | はい                      | いいえ                   | 変更なし               | 0              |
| クリア（更新のみ）        | いいえ                     | -                     | 変更なし               | 0              |
| 選択済み             | はい                      | はい                    | 更新                 | 2              |
| 選択済み             | はい                      | いいえ                   | 変更なし               | 0              |
| 選択済み             | いいえ                     | -                     | 挿入                 | 1              |

## 動的ルックアップキャッシュおよびターゲットの同期

ダウンストリームトランスフォーメーションを設定して、動的ルックアップキャッシュとターゲットが同期するようにします。

動的ルックアップキャッシュを使用すると、統合サービスはルックアップキャッシュに書き込んでからターゲットテーブルに書き込みます。統合サービスがデータをターゲットに書き込まない場合、ルックアップキャッシュとターゲットテーブルは非同期になります。例えば、ターゲットデータベースはデータを拒否することがあります。

ルックアップキャッシュがルックアップテーブルに同期し続けるようにするには、次のガイドラインに従います。

- NewLookupRow 値が 1 または 2 の場合、ルータトランスフォーメーションを使用し、行をキャッシュに格納されたターゲットに渡します。
- NewLookupRow 値が 0 の場合、ルータトランスフォーメーションを使用し、行を削除します。あるいは、行を別のターゲットに出力します。
- アップデートストラテジトランスフォーメーションをルックアップトランスフォーメーションの後に使用して、行にターゲットへの挿入フラグまたは更新フラグを設定します。
- 統合サービスがルックアップキャッシュに書き込む値と同じ値をルックアップトランスフォーメーションがターゲットに出力することを確認します。更新時に新しい値を出力するように選択するときは、ターゲットテーブルに出力ポートの代わりにルックアップ/出力ポートを接続するだけです。更新時に古い値を出力すると選択した場合は、ルックアップトランスフォーメーションの後で、ルータトランスフォーメーションの前に式トランスフォーメーションを追加します。ターゲットテーブルの各ポートに式トランスフォーメーションの出力ポートを追加し、NULL 入力値をターゲットに出力しないように式を作成します。
- アップデートストラテジターゲットテーブルオプションを定義する場合は、[挿入と更新] を [更新] として選択します。これにより、統合サービスは「更新」と記された行を更新し、「挿入」と記された行を挿入します。

## 条件付き動的ルックアップキャッシュの更新

動的ルックアップキャッシュは、Boolean 式の結果に基づいて更新できます。統合サービスは式が true のときにキャッシュを更新します。

例えば、ターゲットテーブルに製品番号、在庫数量、タイムスタンプのカラムがあるとします。在庫数量は最新のソース値で更新する必要があります。ソースデータのタイムスタンプが動的キャッシュ内のタイムスタンプよりも大きい場合に、在庫数量を更新することができます。以下の式のようなルックアップトランスフォーメーションの式を作成します。

```
lookup_timestamp < input_timestamp
```

式にはルックアップポートや入力ポートを含めることができます。ビルトイン変数、マッピング変数、およびパラメータ変数にアクセスできます。ユーザー定義関数を含めたり、未接続のトランスフォーメーションを参照することもできます。

式は true、false、または NULL を返します。式の結果が NULL である場合、その式は false です。統合サービスはキャッシュを更新しません。式の結果が true に変更する必要がある場合は、式に NULL 値のチェックを追加します。式のデフォルト値は true です。

## 条件付きの動的ルックアップキャッシュの処理

統合サービスが動的ルックアップキャッシュを更新するかどうかを決定する条件を作成できます。条件が false または NULL のとき、統合サービスは動的ルックアップキャッシュを更新しません。

条件が false または NULL のとき、ルックアップトランスフォーメーションのプロパティに関係なく NewLookupRow の値は 0 になり、統合サービスは動的ルックアップキャッシュを「挿入」または「更新」に設定された行で更新しません。

キャッシュ内に行が存在し、「挿入でなければ更新」または「更新でなければ挿入」を有効にしている場合、NewLookupRow の値は 1 となり、統合サービスはキャッシュの新しい行を更新します。

キャッシュ内に行が存在せず、「挿入でなければ更新」または「更新でなければ挿入」を有効にしている場合、NewLookupRow の値は 2 となり、統合サービスはキャッシュに新しい行を挿入します。

## 条件付き動的ルックアップキャッシュの設定

統合サービスが動的ルックアップキャッシュを更新するかどうかを決定する式を設定できます。

1. ルックアップトランスフォーメーションを作成します。
2. **【ランタイム】** タブ (**【プロパティ】** ビュー内で、**【ルックアップキャッシュが有効】** を選択します。
3. **【詳細】** タブ (**【プロパティ】** ビュー内で、**【動的ルックアップキャッシュ】** を選択します。
4. 条件を入力するには、**【動的キャッシュの更新条件】** プロパティの下矢印をクリックします。  
式エディタが表示されます。
5. 式の条件を定義します。  
式に対して入力ポート、ルックアップポート、および関数を選択できます。
6. **【検証】** をクリックして式が有効であることを確認します。
7. **【OK】** をクリックします。
8. 必要に応じて、動的ルックアップキャッシュに適用する他の詳細プロパティを選択します。

## 式の結果を使用した動的キャッシュの更新

ルックアップトランスフォーメーションでは、動的ルックアップキャッシュの値を式の結果で更新できます。

例えば、製品テーブルのターゲットには注文数が含まれている数値列があります。ルックアップトランスフォーメーションは、製品の注文を受けるたびに動的キャッシュ order\_count を次の式の結果で更新します。

```
order_count = order_count + 1
```

ルックアップトランスフォーメーションは order\_count を返します。

式が NULL と評価した場合の統合サービスの処理方法を設定できます。

## 式の値が NULL

式のいずれかの値が NULL である場合、式は NULL を返します。ただし、式が NULL でない値を返すように設定できます。

式がルックアップポートを参照していても、ソースデータが新しい場合、ルックアップポートにはデフォルト値が格納されます。デフォルト値は NULL である場合があります。NULL 値をチェックするように IsNull 式を設定できます。

例えば、次の式は lookup\_column が NULL であるかどうかをチェックします。

```
iif (isnull(lookup_column), input_port, user_expression)
```

そのカラムが NULL である場合は、input\_port という値を返します。それ以外の場合は、式の値を返します。

## 式の処理

統合サービスは、式の結果に基づいて動的ルックアップキャッシュの行を挿入および更新できます。式の結果は、ルックアップポートの値が NULL で式に含まれているかどうかに基づいて異なる可能性があります。

統合サービスでは「挿入でなければ更新」を有効にすると、データがキャッシュ内にない場合には、式の結果の行が挿入されます。キャッシュ内にデータが存在しない場合は、ルックアップポートの値が NULL になります。式がルックアップポートの値を参照している場合は、統合サービスは式のデフォルトのポートの値を置き換えます。「挿入でなければ更新」を有効にして、データがキャッシュ内に存在する場合は、統合サービスは式の結果によってキャッシュを更新します。

「更新でなければ挿入」を有効にして、データがキャッシュ内に存在する場合には、統合サービスは式の結果によってキャッシュを更新します。キャッシュ内にデータが存在しない場合は、統合サービスは式の結果を含む行を挿入します。式がルックアップポートの値を参照している場合は、統合サービスは式のデフォルトのポートの値を置き換えます。

## 動的キャッシュの更新のための式の設定

動的ルックアップキャッシュの更新のための式を設定できます。

条件式を作成するには、あらかじめルックアップトランスフォーメーションで動的ルックアップを実行できるようにしておく必要があります。

1. ルックアップトランスフォーメーションを作成します。
2. **【ランタイム】** タブ（**【プロパティ】** ビュー内で、**【ルックアップキャッシュが有効】**）を選択します。
3. **【詳細】** タブ（**【プロパティ】** ビュー内で、**【動的ルックアップキャッシュ】**）を選択します。
4. 必要に応じて、動的ルックアップキャッシュに適用する他の詳細プロパティを選択します。
5. 式を作成するには、**【カラム】** タブ（**【プロパティ】** ビュー）を選択します。
6. 更新するルックアップポートの**【関連ポート】** カラムのドロップダウンの矢印をクリックします。
7. ドロップダウンリストから**【関連する式】**を選択し、**【入力】**をクリックします。

式エディタが表示されます。

8. 式を定義してください。

式に対して入力ポート、ルックアップポート、および関数を選択できます。式の戻り値はルックアップポートのデータタイプと一致する必要があります。

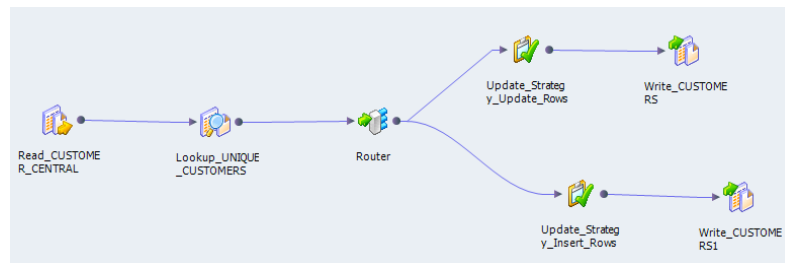
9. **【検証】** をクリックして式が有効であることを確認します。
10. **【OK】** をクリックします。

## 動的ルックアップキャッシュの例

動的ルックアップキャッシュを使用してターゲットの行の挿入および更新ができます。動的ルックアップキャッシュを使用すると、ターゲットに挿入および更新するキャッシュの同じ行を挿入および更新できます。

たとえば、顧客データが含まれているテーブルを更新する必要があるものとします。ソースデータには、ターゲットに挿入または更新する顧客データ行が含まれています。ターゲットとなる動的キャッシュを作成します。ルックアップトランスフォーメーションを設定して動的キャッシュで顧客を検索します。

次の図に、動的ルックアップキャッシュを使用するルックアップトランスフォーメーションを含むマッピングを示します。



ルータトランスフォーメーションは2つのブランチに分割されます。ルータトランスフォーメーションは、一方のブランチに挿入行を渡し、他方のブランチで行を更新します。各ブランチには、ターゲットに行を書き込むアップデートストラテジトランスフォーメーションがあります。両方のブランチには同じターゲットがあります。

マッピングが開始すると、統合サービスは顧客ターゲットテーブルからルックアップキャッシュを作成します。統合サービスは、ルックアップキャッシュにない行を読み込むと、その行をキャッシュに挿入します。

ルックアップトランスフォーメーションは、ルータトランスフォーメーションに各行を返します。ルータトランスフォーメーションは、行に「挿入」または「更新」のどちらがマークされているかに基づき、アップデートストラテジトランスフォーメーションの1つに行をダイレクトします。ルータトランスフォーメーションは、NewLookupRow プロパティに基づいて行に「挿入」または「更新」のどちらがマークされるかを決定します。アップデートストラテジトランスフォーメーションは、各行に「挿入」または「更新」とマークし、ターゲットに渡します。

顧客ターゲットテーブルは、マッピングの実行に伴って変化します。統合サービスは、ルックアップキャッシュに新しい行を挿入し、既存の行を更新します。統合サービスは、ルックアップキャッシュと顧客ターゲットテーブルを同期します。

ターゲットのキーを生成するには、関連するポートのシーケンス ID を使います。統合サービスは、ターゲットテーブルに挿入されるそれぞれの新しい行のプライマリキーとしてシーケンス ID を使用します。

動的ルックアップキャッシュを使用すると、データベースからキャッシュを1回だけ構築するので、セッションのパフォーマンスが向上します。

## 動的ルックアップキャッシュのルールとガイドライン

動的ルックアップキャッシュを使用するときは、次のガイドラインに従います。

- 動的ルックアップキャッシュを使用する場合は、**【複数の一致の検出時】** プロパティを [エラーを報告] に設定する必要があります。プロパティをリセットするには、動的ルックアップを静的ルックアップに変更し、プロパティを変更してから静的ルックアップを動的ルックアップに変更します。
- 同じターゲットロード順グループの動的ルックアップトランスフォーメーションと静的ルックアップトランスフォーメーション間で、キャッシュを共有することはできません。
- 動的ルックアップキャッシュは、リレーショナルルックアップまたはフラットファイルルックアップに対して有効にできます。

- ルックアップトランスフォーメーションは、接続されたトランスフォーメーションでなければなりません。
- 永続または非永続キャッシュのいずれかを使用できます。
- 動的キャッシュが永続でない場合、**[ルックアップソースからのキャッシュの再構築]** が有効になっていない場合でも、統合サービスは常にデータベースからキャッシュを再構築します。
- 作成可能なのは、等価ルックアップ条件だけです。動的キャッシュでデータの範囲をルックアップすることはできません。
- ルックアップ条件にない各ルックアップポートを入力ポート、シーケンス ID、または関連する式に関連付ける必要があります。
- NewLookupRow 値が 1 または 2 の場合、ルータトランスフォーメーションを使用し、行をキャッシュに格納されたターゲットに渡します。
- NewLookupRow 値が 0 の場合、ルータトランスフォーメーションを使用し、行を削除します。あるいは、行を別のターゲットに出力できます。
- 統合サービスがルックアップキャッシュに書き込む値と同じ値がターゲットに出力されることを確認します。更新時に新しい値を出力するように選択するときは、ターゲットテーブルに入力/出力ポートの代わりにルックアップ/出力ポートを接続するだけです。更新時に古い値を出力すると選択した場合は、ルックアップトランスフォーメーションの後で、ルータトランスフォーメーションの前に式トランスフォーメーションを追加します。ターゲットテーブルの各ポートに式トランスフォーメーションの出力ポートを追加し、NULL 入力値をターゲットに出力しないように式を作成します。
- ルックアップ SQL オーバーライドを使用する場合は、カラムを適切なルックアップ用ターゲットに正しくマッピングするようにします。
- ルックアップ SQL オーバーライドに WHERE 句を追加する場合は、ルックアップトランスフォーメーションの前にフィルタトランスフォーメーションを使用します。これにより、統合サービスは WHERE 句に一致する動的キャッシュとターゲットテーブルにのみ行を挿入します。
- 動的キャッシュを使用するように再利用可能なルックアップトランスフォーメーションを設定する場合は、条件を編集したり、マッピングの**動的ルックアップキャッシュ**プロパティを無効にすることはできません。
- アップデートストラテジトランスフォーメーションをルックアップトランスフォーメーションの後に使用して、ターゲットへの挿入または更新フラグを行に設定します。
- ルックアップトランスフォーメーションの**[更新でなければ挿入]** プロパティを使用したい場合は、アップデートストラテジトランスフォーメーションはルックアップトランスフォーメーションの前に使用して、一部またはすべての行を「更新」と定義します。



## 第 28 章

# 一致トランスフォーメーション

この章では、以下の項目について説明します。

- [一致トランスフォーメーションの概要, 444 ページ](#)
- [照合分析, 445 ページ](#)
- [マッチ率の計算, 448 ページ](#)
- [マスターデータの分析, 452 ページ](#)
- [ID 照合分析と永続インデックスデータ, 453 ページ](#)
- [照合マッピングのパフォーマンス, 454 ページ](#)
- [ID 照合分析での照合パフォーマンス, 456 ページ](#)
- [一致トランスフォーメーションのビュー, 459 ページ](#)
- [一致トランスフォーメーションのポート, 460 ページ](#)
- [一致マップレット, 465 ページ](#)
- [照合分析操作の設定, 466 ページ](#)
- [非ネイティブ環境での一致トランスフォーメーション, 467 ページ](#)

## 一致トランスフォーメーションの概要

一致トランスフォーメーションは、レコード間の類似度を分析するアクティブなトランスフォーメーションです。一致トランスフォーメーションを使用して、データセット内または 2 つのデータセット間で重複情報を含むレコードを検索します。

一致トランスフォーメーションは入力ポートの値を分析し、値と値の類似度を表す数値スコアのセットを生成します。複数のポートを選択して、入力レコード間の全体的な類似度を判定することができます。しきい値として最小スコアを指定して、重複情報を含む可能性があるレコードを識別します。

一致トランスフォーメーションは、以下のデータプロジェクトで使用できます。

- 顧客リレーション管理。例えば、ある店舗がメールキャンペーンを企画し、重複する顧客レコードの有無を顧客データベースで確認する必要があるとき。
- 合併と買収。例えば、ある銀行が同じ地域内の別の銀行を買収し、2 つの銀行に共通の顧客がいるとき。
- 規制の準拠。例えば、すべてのデータシステムに重複レコードがないことを要求する、政府や業界の規制下で事業が運営されているとき。
- 財務リスク管理。例えば、銀行が口座名義人間のリレーションを検索するとき。
- マスターデータ管理。例えば、ある小売チェーンに顧客の記録が入ったマスターデータベースがあり、このチェーンの各小売店舗が定期的に記録をマスターデータベースに送信するとき。



- データセット内の重複レコードを特定する必要があるプロジェクト。

## 照合分析

一致トランスフォーメーションにさまざまなタイプの重複分析を定義することができます。定義する重複分析の操作は、マッピング内のデータソースの数およびソースに含まれる情報のタイプに依存します。

一致トランスフォーメーションを設定するときは、以下の項目について検討してください。

- データセットからの選択は、単一のカラムでも複数カラムでもかまいません。
- 単一データソース内のカラムを分析することも、2つのデータソースを分析することもできます。
- 入力ポートフィールド内の未加工データを分析するように一致トランスフォーメーションを設定することも、データ内の ID 情報を分析するようにトランスフォーメーションを設定することもできます。
- 異なるタイプの出力に書き込むように一致トランスフォーメーションを設定することができます。選択する出力のタイプによって、トランスフォーメーションの書き込むレコードの数とレコードの順序が決まります。
- パフォーマンスを向上させるには、照合分析を実行する前に入力レコードをグループごとにソートします。

## カラム分析

一致トランスフォーメーションの設定時に、分析対象のカラムを 1 つ以上選択します。

一致トランスフォーメーションはカラムをペアで分析します。分析対象のカラムを 1 つ選択した場合は、トランスフォーメーションがそのカラムの一時コピーを作成して、ソースカラムを一時カラムと比較します。分析対象のカラムを 2 つ選択した場合は、トランスフォーメーションが選択した 2 つのカラム全体の値を比較します。トランスフォーメーションは一方のカラムの各値を他方のカラムのすべての値と比較します。トランスフォーメーションが分析した各ペアの値のマッチ率が返されます。

一致トランスフォーメーションのストラテジを定義するときに分析するカラムを選択します。ストラテジで分析するカラムとカラムに適用するアルゴリズムを指定します。アルゴリズムは各ペアの値間の類似度を計算します。トランスフォーメーションのアルゴリズムごとに、値間の類似度の測定に使用する条件が異なります。トランスフォーメーションには複数のストラテジを定義でき、さらに各ストラテジに異なるカラムを割り当てることができます。

### カラム分析の例

名字データのカラムの中の値を比較する必要があるとします。そこで、データソースと一致トランスフォーメーションを含んだマッピングを作成します。*Surname* ポートを一致トランスフォーメーションに接続します。マッピングの実行時に、トランスフォーメーションが *Surname* ポートにデータの一時的コピーを作成します。

次の画像は、名字データの一部を示しています。

|   | A       | B         |
|---|---------|-----------|
| 1 | Surname | Surname_1 |
| 2 | Annan   | Annan     |
| 3 | Baker   | Baker     |
| 4 | Barker  | Barker    |
| 5 | Edwards | Edwards   |
| 6 | Parker  | Parker    |
| 7 | Smith   | Smith     |
| 8 | Smith   | Smith     |
| 9 | Zhang   | Zhang     |

マッピングで一連のマッチ率が生成され、次の値に重複の可能性があることが示されます。

- Baker、Barker
- Barker、Parker
- Smith、Smith

ユーザーがデータを確認し、*Baker*、*Barker*、*Parker*の値は重複でないと判断します。他方、*Smith*と *Smith*の値は重複であると判断します。

## 単一ソースでの分析とデュアルソースでの分析

1つまたは2つのデータソースからデータを分析するように一致トランスフォーメーションを設定することができます。トランスフォーメーションのストラテジを定義するときに、各データソースからポートを選択します。

単一ソースでの分析を実行するようにトランスフォーメーションを設定する場合、1つのデータセットから1つまたは複数のポートを選択します。デュアルソースでの分析を実行するようにトランスフォーメーションを設定する場合、各データセットから1つまたは複数のポートを選択します。ポートの選択はペアで選択します。選択するポートの各ペアに関して、トランスフォーメーションはポート内の各値を他のポート内の各値と比較します。単一カラムからのデータに単一ソースでの分析を実行する場合は、トランスフォーメーションが選択したポートの一時コピーを作成します。

**注:** ID 照合分析を実行するときに、データソースを、以前のマッピングで作成した ID データの永続インデックスと比較できます。永続インデックスを使用する ID 分析を指定するには、**【一致タイプ】** オプションを使用してください。

## フィールド一致分析と ID 照合分析

一致トランスフォーメーションを設定することで、フィールド一致分析または ID 照合分析を実行することができます。

フィールド一致分析では、一致トランスフォーメーションがトランスフォーメーションに入力されるソースデータを分析します。フィールド一致分析はどのタイプのデータにも実行できます。ID 照合分析では、一致トランスフォーメーションが入力データから代替データ値のインデックスを生成して、インデックスデータを分析します。入力ポートに ID データが含まれている場合は、ID 照合分析の一致トランスフォーメーションを設定します。ID は、人または組織を特定するデータ値のグループです。

データセットは、1つの ID を様々な形で表すことができます。例えば、次のデータ値はすべて John Smith という名前を表しています。

- John Smith

- Smith, John
- jsmith@email.com
- SMITHJMR

一致トランスフォーメーションでは、レコード内の ID データを読み取って、その ID に関して可能性のある代替バージョンを算出します。このトランスフォーメーションによって、各 ID の現在のバージョンと代替バージョンを含んだインデックスが作成されます。一致トランスフォーメーションはインデックス値を分析しますが、入力レコードの値は分析しません。

## ID ポピュレーションファイル

ID 照合の操作では、ポピュレーションという参照データファイルを読み取ります。ポピュレーションファイルは、ID データの潜在的なバリエーションを定義します。これらのファイルは、Informatica アプリケーションと一緒にインストールされません。ポピュレーションデータファイルは、Informatica から購入してダウンロードします。

コンテンツ管理サービスがアクセスできる場所にファイルをインストールします。Informatica Administrator を使用してコンテンツ管理サービス上の場所を設定します。

## 照合分析でのグループ化

照合分析マッピングでは、トランスフォーメーションが実行する必要があるデータ比較が多数に及ぶため、時間がかかることがあります。比較回数は、選択したポートのデータ値の数に影響されます。

次の表に、1 つのポート上のデータ値の数に応じてマッピングで実行される計算回数を示します。

| データ値の数 | 比較回数   |
|--------|--------|
| 10,000 | 5000 万 |
| 10 万   | 50 億   |
| 100 万  | 5000 億 |

マッピングの実行に要する時間を短縮するために、入力データレコードをグループに割り当てます。グループとは、指定したポート上の、同一の値を含む一連のレコードです。グループ化したデータに照合分析を実行すると、一致トランスフォーメーションが各グループ内のレコードを分析します。トランスフォーメーションが、あるグループのレコードを別のグループのレコードと比較することはありません。グループ化により、マッピング分析の精度を損うことなく、トランスフォーメーションが実行する必要がある比較回数が総じて減少します。

データをグループにまとめるときは、以下のルールとガイドラインを考慮してください。

- データをグループ化するポートは、グループキーポートです。グループキーポートには、さまざまな重複値（住所データセットの市区町村名や都道府県名など）が含まれている必要があります。マッピングデータに使用可能なグループキーポートがない場合は、キージェネレータを使用して現在のマッピングデータからポートを作成します。キージェネレータトランスフォーメーションによるグループキー出力ポートを一致トランスフォーメーションに接続します。  
また、キージェネレータトランスフォーメーションを使用してシーケンス識別子をマッピングデータに追加することもできます。
- フィールド一致操作でグループキーポートを指定する必要があります。ID 分析の一致トランスフォーメーションを設定する場合は、グループキーポートを選択しないでください。ID 分析では、ID インデックスデータのグループキーが生成されます。
- 照合分析で使用する予定のグループキーポートを指定しないでください。

- グループを作成するときは、グループが有効なサイズであることを確認する必要があります。グループサイズが小さすぎると、照合分析でデータセットの中の一部の重複データが検索されないことがあります。グループサイズが大きすぎると、照合分析で偽の重複が返されることがあります。作成されるグループサイズが平均 10,000 レコードとなるグループキーを選択します。
- グループ化により、マッピングデータセット内のレコードの順序が変更されることはありません。

## 一致ペアとクラスタ

一致トランスフォーメーションは異なる数の入力行と出力行の読み取りと書き込みができ、出力行の順序も変更することができます。ユーザーが照合分析の結果の出力形式を決定します。

このトランスフォーメーションは、以下の形式で行を書き込むことができます。

### 一致ペア

トランスフォーメーションは、一致しきい値を満足するマッチ率で一致するレコードの全ペアに対して行を書き込みます。トランスフォーメーションは、レコードの各ペアを単一の行に書き込みます。

レコードは複数の他のレコードと一致することがあるため、1つのレコードが複数の出力行に書き込まれる場合もあります。

### 最良の一致

トランスフォーメーションは、データセット内の各レコードの行を書き込み、別のデータセットの最も類似性の高いレコードを同じ行に追加します。

### クラスタ

トランスフォーメーションは、レコード間の類似度に基づいて出力レコードをクラスタに割り当てます。クラスタとは、各レコードが他の 1 つ以上のレコードと一致（マッチ率が一致しきい値を満足）するレコードの集まりです。トランスフォーメーションは、各レコードを単一の行に書き込みます。

クラスタ内の各レコードは、クラスタ内の他の 1 つ以上のレコードと一致する必要があります。したがって、クラスタ内に相互に一致しないレコードのペアを含めることができます。レコードが他のどのレコードとも一致しない場合、クラスタが単一のレコードから成ることがあります。

**注:** フィールド分析の [クラスタ] オプションは、ID 分析の [クラスタ - すべてに一致] オプションに対応します。ID 分析の [クラスタ - なるべく多くに一致] オプションでは、クラスタの計算と一致ペアの計算を結合します。

[照合出力] ビューで出力オプションを設定します。

### 関連項目：

- [「クラスタ出力オプション」 \(ページ 449\)](#)

## マッチ率の計算

マッチ率は、2つのカラムの値の類似度を示す数値です。アルゴリズムでは、0 から 1 までの小数値としてマッチ率を計算します。2つのカラムの値が同じ場合、アルゴリズムはスコアに 1 を割り当てます。

分析用に複数のカラムペアを選択すると、トランスフォーメーションでは選択したカラムのスコアに基づいて平均スコアを計算します。デフォルトでは、トランスフォーメーションは各カラムペアのスコアに等しい加重を割り当てます。トランスフォーメーションは、データセットのカラムデータの相対的な重要度を推測しません。

トランスフォーメーションがマッチ率の計算に使用する加重値は編集可能です。より高いまたは低い優先順位をデータセットのカラムに割り当てる場合に加重値を編集します。

また、トランスフォーメーションでカラムの NULL 値が検索される場合も適用されるスコアを設定できます。デフォルトでは、トランスフォーメーションは NULL 値をデータエラーとして扱い、NULL を含むすべてのペア値に低いマッチ率を割り当てます。

**注:** 選択するアルゴリズムによって、2 つの値のマッチ率が決まります。アルゴリズムは 2 つの値に単一のスコアを生成します。マッチ率は、照合出力または選択するスコアリング方法のタイプに影響されません。

## 加重スコア

照合分析用に複数のカラムを選択すると、カラムのスコアに基づいて、トランスフォーメーションは各レコードの平均スコアを計算します。平均スコアには、各カラムの比較アルゴリズムに適用するすべての加重値が含まれます。

デフォルトでは、すべてのアルゴリズムは 0.5 の加重値を使用します。選択したカラムが重複情報を含む可能性が高くなる場合は、この値を増やすことができます。選択したカラムの重複した値がレコード間の正確な重複情報を示す可能性が低くなる場合は、加重値を減らすことができます。一致トランスフォーメーションは、ペアのレコードごとに単一のマッチ率として平均スコアを使用します。

## NULL のマッチ率

照合アルゴリズムは、一方または両方の値が NULL であるときに、事前定義されたマッチ率を値のペアに適用します。フィールド一致アルゴリズムが NULL 値に適用するマッチ率は編集可能です。

### NULL のマッチ率とフィールド一致アルゴリズム

フィールド一致アルゴリズムを設定する場合は、アルゴリズムが NULL 値に適用するマッチ率の値を確認してください。フィールド一致アルゴリズムで 2 つの値を比較して一方または両方の値が NULL の場合、デフォルトのスコア 0.5 が適用されます。スコア 0.5 は、データ値間の類似度が低いことを示します。

NULL マッチ率を確認するときは、以下のルールとガイドラインを考慮してください。

- アルゴリズムでプライマリーキーまたはその他の重要なデータを含むカラムを分析する場合、デフォルトのスコアは編集できません。この場合、NULL 値がデータエラーを示し、デフォルトのスコアはデータにとって適切になります。
- アルゴリズムで必要に応じてデータを含めることが可能なカラムを分析する場合、NULL のマッチ率の値を一致しきい値と同じ値に更新します。NULL のマッチ率を一致しきい値に設定した場合、一致分析での NULL 値の影響をキャンセルします。

### NULL のマッチ率と ID 照合アルゴリズム

ID 照合アルゴリズムで 2 つの値を比較して一方または両方の値が NULL の場合、マッチ率に 0 が適用されます。ID 照合分析は、NULL のマッチ率を含むレコードを一意的レコードクラスタに割り当て、クラスタサイズ値 1 を記録します。ID 照合アルゴリズムが NULL 値に適用するスコアは編集できません。

## クラスタ出力オプション

出力データを類似するレコードまたは同一のレコードで構成する場合は、クラスタ出力オプションを選択します。

クラスタ出力オプションを選択した場合、トランスフォーメーションはクラスタ ID 値を各出力レコードに追加します。クラスタ ID 値によってレコードをソートできます。トランスフォーメーションの出力には、すべてのレコードの行が含まれます。レコードが、一致しきい値を満たすマッチ率の他のレコードと一致しない場合、トランスフォーメーションは一意的クラスタ ID をそのレコードに割り当てます。クラスタ出力オプションを選択または更新するには、**[照合出力]** ビューを使用します。

次のクラスタ出力オプションを選択することができます。

## クラスタ

このオプションは、クラスタ ID 値を出力レコードに割り当てる場合に選択します。

### クラスタ - 最良の一致

このオプションは、マッチ率が最高のレコードのペアをクラスタに追加する場合に選択します。レコードが、複数の他のレコードと最高のマッチ率を示す場合があるため、複数のレコードのペアがクラスタ ID 値を共有できます。

### クラスタ - すべてに一致

【クラスタ - すべてに一致】オプションは、【クラスタ】オプションと同様に動作します。

トランスフォーメーションは ID 照合分析で、【クラスタ - すべてに一致】および【クラスタ - なるべく多くに一致】をオプション名として使用します。

**注:** データ統合サービスが複数の一致トランスフォーメーションを同時に実行する場合、各トランスフォーメーションからの出力に対し一意のクラスタ ID 値が生成されます。このため、各トランスフォーメーションが生成するレコードのクラスタ ID 値は不連続になる場合があります。

## 【クラスタ】オプションと【クラスタ - すべてに一致】オプション

フィールド一致分析には【クラスタ】オプションを選択します。ID 照合分析には【クラスタ - すべてに一致】オプションを選択します。

一致トランスフォーメーションでは、次のルールを使用してクラスタを作成します。

- 2つのレコードのマッチ率が一致しきい値を満たす場合は、一致トランスフォーメーションがクラスタにレコードを追加します。
- データセットのレコードがクラスタ内のレコードに1つでも一致すれば、トランスフォーメーションがそのレコードをクラスタに追加します。
- 1つのクラスタのあるレコードが別のクラスタのあるレコードに一致する場合、2つのクラスタはマージされます。
- トランスフォーメーションは、すべてのレコードがクラスタに配属されるまで照合結果を次々と処理していきます。
- レコードが、データセット内のどのレコードとも一致しない場合、トランスフォーメーションは一意のクラスタ ID 値をそのレコードに割り当てます。

## 【クラスタ - なるべく多くに一致】オプション

ID 照合分析には【クラスタ - なるべく多くに一致】オプションを選択します。

トランスフォーメーションでは、次のルールを使用してクラスタを作成します。

- トランスフォーメーションは、現在のレコードでマッチ率が最高のレコードを特定します。マッチ率がしきい値を満足する場合、トランスフォーメーションはそのレコードのペアをクラスタに追加します。
- 一致するレコードのうち一方がクラスタ内にある場合、もう一方のレコードがトランスフォーメーションによって現在のクラスタに追加されます。
- すべてのレコードがクラスタに配属されるまでトランスフォーメーションが照合結果を次々と処理していきます。
- レコードがデータ内の他のどのレコードとも一致しない場合、クラスタが単一のレコードから成ることがあります。

**注: 【照合出力】** ビューの【一致】プロパティを使用すると、トランスフォーメーションが単一のデータソースと永続データストアを比較する方法を指定できます。【一致】プロパティによって、トランスフォーメーションがソースデータまたは永続データストア内の重複を確認するかどうかが決まります。



## 関連項目：

- [「一致ペアとクラスタ」 \(ページ 448\)](#)

## クラスタ分析でのドライバスコアとリンクスコア

一致トランスフォーメーションでクラスタ出力のオプションを選択すると、リンクスコアとドライバスコアのデータを出力に追加することができます。

リンクスコアとは、レコードが同一クラスタのメンバであることを識別するための、2つのレコード間のスコアです。レコード間のリンクによってクラスタの構成が決まります。どのレコードでも、同一クラスタ内の任意のレコードにリンクすることができます。

ドライバスコアとは、クラスタ内でシーケンス ID の値が最高のレコードと、同じクラスタ内の他のレコードとの間のスコアです。ドライバスコアは、クラスタ内のすべてのレコードを単一のレコードに照らして評価する手段の1つです。ドライバスコアを照合出力に追加すると、すべてのクラスタが完了するまで一致トランスフォーメーションがドライバスコアを計算できないため、マッピングの実行速度が遅くなります。

**注:** 照合分析は、定義したストラテジごとにスコアを1セット生成します。ドライバスコアとリンクスコアは、各クラスタ内のさまざまなレコードペアのマッチ率を示します。リンクスコアとドライバスコアは、レコードがトランスフォーメーションに入る順序に依存する可能性があります。ドライバスコアが一致しきい値より低くなる場合もあります。

### クラスタ分析の例

名字データの列を分析するためにフィールド一致ストラテジを設定することにします。そこで、一致しきい値として 0.825 をストラテジに設定します。クラスタ化した出力形式を選択し、トランスフォーメーションでデータビューアを実行します。

データビューアに表示されるデータを下表に示します。

| 名字     | シーケンス ID | クラスタ ID | クラスタ サイズ | ドライバ ID | ドライバスコア | リンク ID | リンクスコア  |
|--------|----------|---------|----------|---------|---------|--------|---------|
| SMITH  | 1        | 1       | 2        | 1-6     | 1       | 1-1    | 1       |
| SMYTH  | 2        | 2       | 2        | 1-3     | 0.83333 | 1-2    | 1       |
| SMYTHE | 3        | 2       | 2        | 1-3     | 1       | 1-2    | 0.83333 |
| SMITT  | 4        | 3       | 1        | 1-4     | 1       | 1-4    | 1       |
| SMITS  | 5        | 4       | 1        | 1-5     | 1       | 1-5    | 1       |
| SMITH  | 6        | 1       | 2        | 1-6     | 1       | 1-1    | 1       |

データビューアには、名字データに関して次の情報が含まれています。

- SMITT と SMITS は、どのレコードにも一致しません（スコアが一致しきい値を満足しない）。一致トランスフォーメーションが、レコードがデータセット内で一意であると判定します。  
SMITT と SMITS は、クラスタサイズが1です。クラスタ出力内で一意のレコードを見つけるには、含まれるレコードが1つだけのクラスタを検索します。
- SMITH と SMITH は、リンクスコアが1です。一致トランスフォーメーションがレコード同士が同一であると判定します。トランスフォーメーションにより、レコードが単一のクラスタに追加されます。
- SMYTH と SMYTHE は、リンクスコアが 0.83333 です。スコアが一致しきい値を超えています。したがって、このトランスフォーメーションにより、レコードが単一のクラスタに追加されます。

# マスターデータの分析

一致トランスフォーメーションで2つのデータソースを分析するときは、ソースをマスターデータセットとして識別する必要があります。このトランスフォーメーションでは、指定したデータセット内の各レコードのデータ値を、2番目のデータセット内のすべてのレコードの対応する値と比較します。

多くの会社において、マスターデータセットは、固定で使用する質の高いデータストアを構成しています。レコードをマスターデータセットに追加する前に、一致トランスフォーメーションを使用してレコードが重複情報をマスターデータに追加しないことを検証します。

## マスターデータの例

ある銀行が、顧客口座記録のマスターデータセットを管理しています。この銀行では、新規の顧客口座である識別されたレコードで毎日マスターデータセットを更新しています。この銀行は、マスターデータセットの中の新しいレコードで顧客情報が重複していないか確認するために、重複分析マッピングを使用します。マスターデータセットと新規口座のテーブルが共通の構造を持ち、テーブルが同じタイプのデータベースを使用しています。したがって、銀行はマスターデータセットを更新する必要があるたびに重複分析のマッピングを再利用することができます。

## マスターデータセット分析の方向性

一致トランスフォーメーションでは、レコードを1つのデータセットから別のデータセットに単方向で比較します。このトランスフォーメーションでは、マスターデータセット内の各レコードを2番目のデータセット内のすべてのレコードと比較します。2番目のデータセット内の各レコードをマスターデータセット内のすべてのレコードと比較することはありません。したがって、マスターデータセットの選択が照合分析の結果に影響する可能性があります。

次の表は、ID 照合分析で比較できる2つのデータセットを示しています。

| データセット 1              | データセット 2           |
|-----------------------|--------------------|
| Alex Bell             | Alexander Bell     |
| Alexander Graham Bell | Thomas Edison      |
| Alva Edison           | Nicola Tesla       |
| Marie Curie           | Irene Joliot Curie |
| Dorothy Crowfoot      | Dorothy Hodgkin    |

データセット 1 をマスターデータセットとして選択し、**【最良の一致】** 出力オプションを選択した場合、出力に以下のレコードが含まれます。

- Alex Bell, Alexander Bell
- Alexander Graham Bell, Alexander Bell

データセット 2 をマスターデータセットとして選択し、**【最良の一致】** 出力オプションを選択した場合、出力に以下のレコードが含まれます。

- Alexander Bell, Alex Bell

データセット 2 がマスターデータセットの場合、トランスフォーメーションによって、Alexander Bell と Alexander Graham Bell が一致する可能性はありません。Alexander Bell は出力データ内で Alex Bell とすでに一致しているためです。



## マッピングの再利用

データをマスターデータセットに定期的に追加する場合、再利用できる重複分析のマッピングを設定します。マスターデータセットと比較するデータソースのポート設定に変更がなければ、マッピングを再利用できます。

マッピングを実行するときは、トランスフォーメーションがマスターデータと新しいデータを指定することを確認します。マッピングは他に設定の更新を行わずに実行することができます。

## ID 照合分析と永続インデックスデータ

マッピングを実行して ID 情報を分析する際に、一致トランスフォーメーションはデータセット内の ID の代替バージョンを格納するインデックスを生成します。デフォルトでは、一致トランスフォーメーションがインデックスデータを一時ファイルに書き込みます。インデックスデータをデータベーステーブルに保存するようにトランスフォーメーションを設定することもできます。

生成するインデックステーブルはデータストアを表し、後続のマッピングで再利用できます。このインデックステーブルをデータソースと比較できます。また必要に応じて、データソースからのインデックスデータに基づいてインデックステーブルを更新できます。トランスフォーメーションはインデックステーブルを再生成しないため、後続のマッピングはより高速に実行されます。さらに、このインデックステーブルは、ID データの信頼できるデータストアを表すものになります。

### 関連項目：

- [「ID 照合分析での照合パフォーマンス」 \(ページ 456\)](#)

## 永続インデックスデータに関するルールとガイドライン

ID 情報のマスターデータセットを分析するように一致トランスフォーメーションを設定するときは、以下のルールとガイドラインを考慮してください。

- マスターデータセットの再利用可能インデックスを生成するには、インデックスデータをデータベーステーブルに書き込むように一致トランスフォーメーションを設定します。インデックスデータの永続ストアはデータベーステーブルで構成されます。
- 別のデータセットの ID をインデックスデータストアと比較するには、データセットをマッピングデータソースとして設定します。一致トランスフォーメーションを設定し、データソースとインデックスデータストアを読み込みます。指定したデータベース接続のデフォルトスキーマからインデックステーブルを選択します。
- 一致トランスフォーメーションが、入力レコードからのシーケンス識別子の値を、レコードに対応するインデックスデータの行に追加します。*SequenceID* 入力ポートはシーケンス識別子を含んでいます。トランスフォーメーションはこのシーケンス識別子を使用して、照合分析の各過程を通してインデックスデータを追跡します。シーケンス ID ポートを切断しないでください。
- 一致トランスフォーメーションをインデックスストアに接続すると、トランスフォーメーションは、ストアを作成したトランスフォーメーションのプロパティ、キーレベル、キータイプ、キーフィールドのプロパティ値を再利用します。また、トランスフォーメーションが、ストアを作成したトランスフォーメーションのポート設定を再利用します。

トランスフォーメーションのプロパティが一致していない場合、ID 分析はマッピングソースデータとインデックスデータを正しく比較できません。

- 一致トランスフォーメーションは、キーフィールドとして選択された入力ポートのデータを使用して ID インデックスを生成します。また、別のポートのデータを ID インデックスに書き込むこともできます。非キ

フィールドデータポートをトランスフォーメーションから切断すると、マッピングの実行時に、対応するインデックスカラムのすべてのデータが消去されます。インデックステーブルの入力ポートデータを保持する場合は、入力データポートを切断しないでください。

- データセットのインデックステーブルデータを生成するときに、一致トランスフォーメーションの照合分析を無効にできます。例えば、データセットのインデックスストアを作成するときに、照合分析を無効にすることが考えられます。照合分析を無効にすると、マッピングがより速く実行されます。

一致分析を無効にすると、一致トランスフォーメーションで持続ステータスコードと持続ステータスの説明を生成し、表示できます。トランスフォーメーションでは、一致スコアや一致分析の結果と関連付いたその他のデータの生成または表示を行いません。例えば、トランスフォーメーションでレコードをクラスタに割り当てるように設定し、一致分析を無効にすると、トランスフォーメーションはクラスタ ID 値の生成や表示を行いません。

- 一致トランスフォーメーションがマッピングソースからのデータに基づいてインデックスストアを更新するかどうかを決定します。一致トランスフォーメーションはシーケンス識別子を使用して、インデックスストア内の行とマッピングデータ内の行が同じレコードを表しているかどうかを判断します。

## 照合マッピングのパフォーマンス

一致トランスフォーメーションのパフォーマンスを決定するデータ要因を、そのトランスフォーメーションを含んだマッピングの実行前に事前に確認することができます。システムにマッピングを実行するためのリソースがあるか確認することができます。また、入力データの類似度を測定するトランスフォーメーションを正しく設定したかどうかを確認できます。

システムに必須リソースがあるか確認する場合に、**[パフォーマンス照合分析]** オプションを使用します。マッピングで入力データの類似度を正確に測定できるかどうかを確認する場合は、**[クラスタ照合分析]** オプションを使用します。

単一のデータソースを読み取る一致トランスフォーメーションに対してパフォーマンス照合分析とクラスタ照合分析を実行します。デュアルソースフィールド照合解析を実行するすべての一致トランスフォーメーションに対してパフォーマンス照合分析を実行します。インデックステーブルに接続する ID 照合ストラテジに対しては、パフォーマンス照合分析やクラスタ照合分析を実行しません。

### パフォーマンス照合分析でのドリルダウン

照合分析のデータをドリルダウンして、一致しきい値以上のレコードペアを表示できます。**[詳細]** ビューのレコードをダブルクリックし、**[データビューア]** を使用して選択したレコードと一致するレコードを表示します。**[データビューア]** には各ペアのレコードのデータが 1 行で表示されます。行にはペアの各レコードの行識別子が示されます。

### クラスタ照合分析でのドリルダウン

クラスタ分析のデータをドリルダウンして、各クラスタのレコードを表示できます。**[詳細]** ビューのクラスタをダブルクリックして、**[データビューア]** にデータを表示します。**[データビューア]** には一度に 1 つのクラスタが表示されます。クラスタのデータには、ドライバスコア、リンクスコア、ドライバ識別子、リンク識別子など、ユーザーが選択したスコアオプションが含まれます。

### 一致トランスフォーメーションロギング

一致トランスフォーメーションを使用するマッピングを実行する場合、Developer ツールのログではマッピングで実行する比較計算の回数を追跡します。ログデータを表示するには、**[データビューア]** の **[ログの表示]** オプションを選択します。

マッピングでは、計算が 100,000 回ごとにログを更新します。

## 一致クラスタ分析のデータの表示

トランスフォーメーションが作成できるクラスタに関して統計データを表示することができます。クラスタの統計は、現在のマッピング設定に基づいたデータセット内のレコード重複レベルに関する概要を示します。

データを表示するには、マッピングキャンバスの中の一致トランスフォーメーションを右クリックして、**[一致クラスタ分析]** を選択します。

分析を実行する前に、トランスフォーメーションが入っているマッピングを検証してください。

一致クラスタ分析は以下のプロパティに関してデータを表示します。

| プロパティ        | 説明                                                                                    |
|--------------|---------------------------------------------------------------------------------------|
| ソース          | 入力データ行の数。                                                                             |
| 最後の実行        | 分析の日付と時間。                                                                             |
| 検出されたクラスタの総数 | マッピング実行時に照合分析が生成するクラスタの数。                                                             |
| 最小クラスタサイズ    | 入っているレコードが最も少ないクラスタのレコード数。最小クラスタのサイズが 1 の場合、そのデータセットには少なくとも 1 つの一意のレコードがあります。         |
| 最大クラスタサイズ    | 最も多くのレコードが入っているクラスタのレコード数。<br>この値が平均クラスタサイズを大きく超える場合は、最も大きいクラスタの中に偽の重複が入っている可能性があります。 |
| 一意のレコードの数    | データセット内で他のレコードと一致しない（マッチ率が一致しきい値未満の）レコードの数。                                           |
| 重複したレコードの数   | データセット内で他のレコードと一致する（マッチ率が一致しきい値を満たす）レコードの数。                                           |
| 比較総数         | マッピングが実行する比較操作の数。                                                                     |
| 平均クラスタサイズ    | クラスタ内の平均レコード数。                                                                        |

## 照合パフォーマンス分析のデータの表示

マッピングが入力データとして読み込むレコードグループに関する統計データを表示することができます。

データを表示するには、マッピングキャンバスの中の一致トランスフォーメーションを右クリックして、**[照合パフォーマンス分析]** を選択します。

分析を実行する前に、トランスフォーメーションが入っているマッピングを検証してください。

照合パフォーマンス分析は、以下のプロパティに関してデータを表示します。

| プロパティ | 説明        |
|-------|-----------|
| ソース   | 入力データ行の数。 |
| 最後の実行 | 分析の日付と時間。 |

| プロパティ            | 説明                                                                                                                                                  |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 検出されたグループの総数     | 選択されたグループキーの値に基づいて、データセットに対し定義されたグループの数。                                                                                                            |
| スループット（レコード/分）   | 照合分析のスピードを推定する変数の値。この値を設定します。この値を使用して照合分析の実行に必要な時間を見積もります。                                                                                          |
| レコード照合の推定時間      | 一致トランスフォーメーションの設定に基づいて見積もった、データセット内のすべてのレコードを分析するためにかかる時間。                                                                                          |
| 生成されるペアの総数       | 入力データの行数とグループ数に基づいた、トランスフォーメーションが実行する必要がある比較の数。                                                                                                     |
| 最小グループサイズ        | 1つのグループの中に存在できる最小レコード数を示す変数の値。この値を設定します。この値を使用して、使用できるサイズのグループをマッピングが作成することを確認します。<br><b>注:</b> 最小グループサイズの値は、マッピング実行時に作成されるグループのサイズを決めるものではありません。   |
| 最小しきい値未満のグループの数  | 最小グループサイズの値よりレコード数が少ないグループの数。<br>多くのグループが最小グループサイズ未満である場合、トランスフォーメーションを編集して別のグループキーを選択する必要があることがあります。                                               |
| 最大グループサイズ        | 1つのグループの中に存在できる最大レコード数を示す変数の値。この値をパフォーマンス分析に設定します。この値を使用して、使用できるサイズのグループをマッピングが作成することを確認します。<br><b>注:</b> この値は、マッピング実行時に作成されるグループのサイズを決めるものではありません。 |
| 最大しきい値を超えるグループの数 | 最大グループサイズの値よりレコード数が多いグループの数。<br>多くのグループが最大グループサイズ以上である場合、トランスフォーメーションを編集して別のグループキーを選択する必要があることがあります。                                                |

## ID 照合分析での照合パフォーマンス

2つのデータセットに対して ID 分析を実行する際にマッピングのパフォーマンスを向上させるには、データベーステーブルから ID インデックスデータを読み取る一致トランスフォーメーションを設定します。マッピングを実行して、マスターデータセットのインデックステーブルを作成します。マッピングを再度実行して、インデックスデータを別のデータソースと比較します。

**[照合タイプ]** ビューのオプションを使用して、インデックスデータを保存するデータベーステーブルを特定します。インデックスデータを別のソースからのデータと比較するトランスフォーメーションを設定するときも、同じオプションを使用してインデックステーブルを選択します。

インデックスデータをデータベーステーブルに書き込むには、以下のタスクを実行します。

1. ID 情報のデータソースを読み取るマッピングを作成します。
2. インデックスデータをデータベースに書き込むようにマッピング内で一致トランスフォーメーションを設定します。
3. そのマッピングを実行して、インデックスデータを生成します。このインデックスデータは、再利用できるデータストアを示しています。

インデックスデータをデータベーステーブルから読み込むには、以下のタスクを実行します。

1. 他の ID データソースを読み取るマッピングを作成します。
2. マッピングに、あらかじめ指定したデータベースからインデックスデータを読み取る一致トランスフォーメーションを設定します。  
マッピングデータソースとインデックスデータの構造が同じ場合は、インデックスデータを生成したマッピングを再利用できます。
3. マッピングを実行してデータソースをインデックスデータと比較します。  
マッピングによりデータソースのインデックスデータが作成されます。このマッピングでは、大きいほうのデータセットに対してインデックスデータを生成する必要ありません。したがって、このマッピングは、両方のデータセットに対してインデックスデータを生成するデュアルソースのマッピングよりも速く実行されます。

## 関連項目：

- [「ID 照合分析と永続インデックスデータ」 \(ページ 453\)](#)

## ID インデックスデータ用のデータストアの作成

ID 情報の入ったデータソースを読み取るマッピングを設定します。インデックスデータをデータベースに書き込むために一致トランスフォーメーションを使用します。

1. マッピングを作成し、データソースをマッピングキャンバスに追加します。
2. そのマッピングキャンバスに一致トランスフォーメーションを追加します。
3. データソース上で、ID 情報の入った出力ポートを選択します。
  - ID 情報のポートを一致トランスフォーメーションに接続します。
  - シーケンス識別子値を含むポートを一致トランスフォーメーションに接続します。
4. 一致トランスフォーメーションで、**[照合タイプ]** ピューを選択します。
5. 照合タイプを **[永続的レコード ID での ID 照合]** に設定します。
6. インデックスデータストアを作成するための次のオプションを設定します。
  - 永続方法を **[データベースを新しい ID で更新する]** に設定します。
  - 照合プロセスの値を確認します。デフォルト値は **[有効]** です。照合プロセスを無効にした場合、マッピングで ID インデックステーブルは作成されますが、データに対する照合分析は実行されません。
  - **[DB 接続]** メニューで、インデックステーブルのデータベースを選択します。
  - **[永続ストア]** メニューで、**[新規作成]** を選択します。**[ストレージテーブルの作成]** ダイアログボックスで、インデックスの名前を入力します。
7. 一致トランスフォーメーションが必要とするその他の設定手順を実行します。例えば、トランスフォーメーションストラテジを設定します。
8. ターゲットデータオブジェクトをマッピングに追加します。
9. 一致トランスフォーメーションの出力ポートをターゲットのデータオブジェクトに接続します。
10. マッピングを実行します。

このマッピングにより、データソースのインデックスデータが、ユーザー指定のデータベーステーブルに書き込まれます。

## 単一ソース分析でのインデックスデータストアの使用

ID 情報のデータソースを読み込むマッピングを設定します。一致トランスフォーメーションを使用して、データソースをマスターデータセットのインデックスデータストアと比較します。

マッピングを設定する前に、データソースにシーケンス識別子の値のカラムがあることを確認します。

時間を節約するために、インデックスデータストアを作成したマッピングをコピーまたは再利用できます。

1. インデックスデータストアを生成したマッピングを開きます。  
あるいは、その代わりにマッピングのコピーを開きます。
2. マッピングのデータソースを確認します。  
必要に応じて、データソースを現在のデータを含むソースに置換します。  
**注:** データソースを削除する場合は、一致トランスフォーメーションのポート接続も削除します。
3. ID 情報を含むデータソースのポートを特定します。
  - ID 情報のポートを一致トランスフォーメーションに接続します。
  - シーケンス識別子値を含むポートを一致トランスフォーメーションに接続します。  
入力ポートと入力ポートの順序は、インデックステーブルを作成したトランスフォーメーションの入力ポートに対応している必要があります。
4. 一致トランスフォーメーションで、**[照合タイプ]** ビューを選択します。
5. 照合タイプを **[永続的レコード ID での ID 照合 (デュアルソース)]** に設定します。
6. ポピュレーション、キーレベル、キータイプ、キーフィールド値を確認します。
7. インデックスデータストアを識別するようにオプションを設定します。
  - 永続方法を設定します。例えば、インデックステーブルの現在のデータを維持する場合は、**[データベースを更新しない]** を選択します。
  - 照合プロセスを **[有効]** に設定します。
  - **[DB 接続]** メニューで、インデックステーブルを含むデータベースを選択します。
  - **[永続ストア]** メニューで、インデックスデータを含むテーブルを参照します。
8. **[照合出力]** ビューで以下のプロパティを設定します。
  - 一致。トランスフォーメーションがインデックスデータをデータベーステーブルから読み込むときに、分析対象のレコードを特定します。
  - 出力。トランスフォーメーションが出力として書き込むレコードをフィルタします。
9. 一致トランスフォーメーションが必要とするその他の設定手順を実行します。  
例えば、トランスフォーメーションストラテジを設定します。
10. マッピングのデータターゲットを確認します。  
必要に応じて、データターゲットを別のターゲットオブジェクトに置換します。
11. 一致トランスフォーメーションの出力ポートをターゲットのデータオブジェクトに接続します。
12. マッピングを実行します。

マッピングは、データソースのレコードをインデックスデータストアと比較します。トランスフォーメーションは、データソースのインデックスデータをデータストアに書き込みます。



# 一致トランスフォーメーションのビュー

一致トランスフォーメーションのユーザーが設定できるオプションは、一連のビューに分けて整理されています。再利用可能なトランスフォーメーションでは、ビューが Developer ツールのキャンバスにタブとして表示されます。ビューを開くには、タブをクリックします。再利用不可能なトランスフォーメーションでは、ビューはキャンバス上にリストとして表示されます。ビューを開くには、リスト上の項目をクリックします。

照合分析操作を設定する場合、次のビューを設定することができます。

## 全般

再利用不可能な一致トランスフォーメーションの名前と説明を更新するときに、[全般] ビューを使用します。説明の入力は任意です。

## ポート

再利用不可能な一致トランスフォーメーションの入力ポートと出力ポートを確認するときに、[ポート] ビューを使用します。

## 概要

再利用可能なトランスフォーメーションの名前と説明を更新するときに、[概要] を使用します。説明の入力は任意です。再利用可能なトランスフォーメーションの入力ポートと出力ポートを作成するときにも、[概要] を使用します。

## 照合タイプ

トランスフォーメーションが実行する重複分析のタイプを選択するときに、[照合タイプ] ビューを使用します。フィールド一致分析または ID 照合分析が選択できます。単一のデータソースまたは 2 つのデータソースを指定することができます。

## ストラテジ

入力データを分析するためのストラテジを 1 つまたは複数定義するときに、[ストラテジ] ビューを使用します。各ストラテジ内で 2 つのデータカラムを選択し、そのカラムに照合分析アルゴリズムを割り当てます。

## 照合出力

出力データの構造と形式を指定するときに、[照合出力] ビューを使用します。

## パラメータ

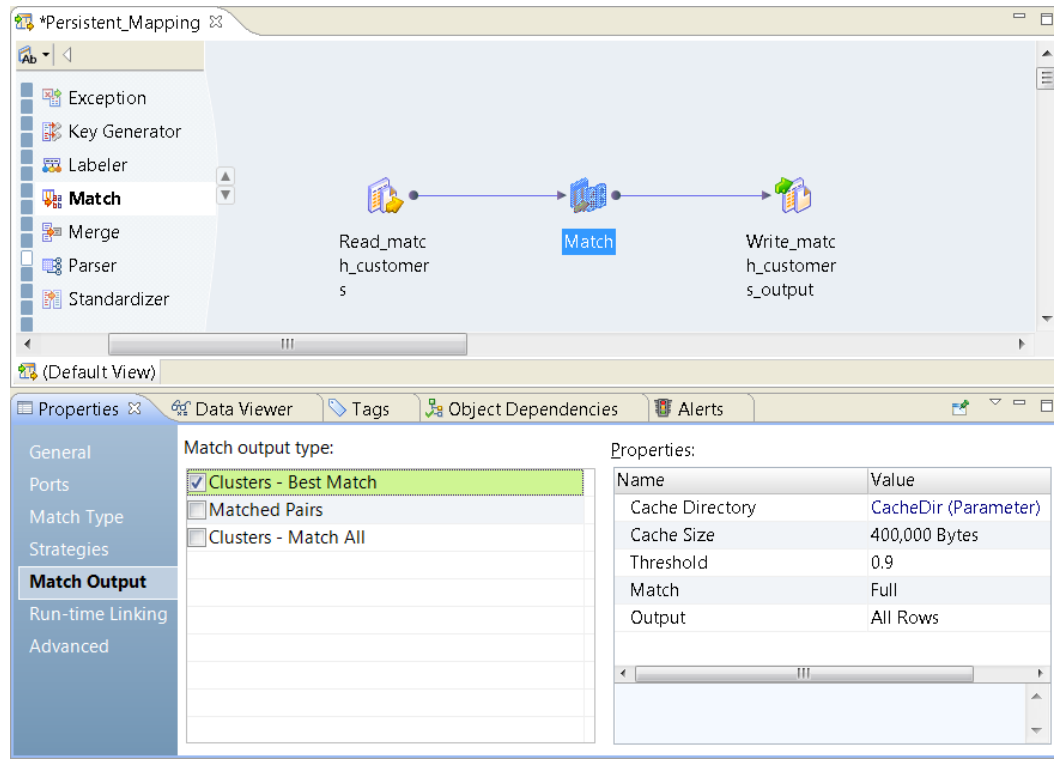
このトランスフォーメーションを含んだマッピングを実行するときに、そのトランスフォーメーションにデータ統合サービスが適用できるパラメータを定義するために、[パラメータ] ビューを使用します。

## 詳細

次のプロパティを指定するときに、[詳細] ビューを使用します。

- トランスフォーメーションに関してマッピングが書き込むログメッセージの詳細レベル。
- トランスフォーメーションを含むマッピングを実行するときに、ID 照合操作が使用するプロセスの数。
- トランスフォーメーションが同一のレコードから出力ポートにデータを直接渡すかどうか。クラスタ化された出力を書き込むようにトランスフォーメーションを設定すると、同一レコードをフィルタリングすることができます。

以下の図は、永続インデックスストアを使った ID 照合分析のために設定する一致トランスレーションでのビューを示しています。



## 一致トランスフォーメーションのポート

一致トランスフォーメーションは、定義した照合分析操作に関するメタデータが入った定義済み入力ポートと出力ポートの組み合わせを含みます。ユーザーが照合タイプおよび照合出力のオプションを設定すると、トランスフォーメーションはポートを選択またはクリアします。

トランスフォーメーションを設定するときは、メタデータポートを確認します。トランスフォーメーションをマッピングに追加するときは、メタデータポートをアップストリームとダウンストリームのマッピングオブジェクトの正しいポートに接続していることを確認します。

## 一致トランスフォーメーションの入力ポート

定義済み入力ポートには、トランスフォーメーションが照合分析に必要とするメタデータが含まれています。

一致トランスフォーメーションの作成後に、次の入力ポートを設定できます。

### Sequenced

マッピングソースのデータセット内の各レコードに対する一意の識別子。入力データセット内の各レコードは一意のシーケンス識別子を含む必要があります。データセットの中にシーケンス識別子が重複したものがあると、一致トランスフォーメーションで重複レコードを正しく識別することはできません。データに一意の識別子が存在しない場合は、キージェネレータートランスフォーメーションを使用して一意の識別子を作成します。



ID データに対してインデックスデータストアを作成すると、一致トランスフォーメーションは各レコードのシーケンス識別子をデータストアに追加します。データソースをインデックスデータストアと比較するようにトランスフォーメーションを設定する場合、トランスフォーメーションが両方のデータセットから共通のシーケンス識別子を検出する可能性があります。それぞれのデータセットでシーケンス識別子が一意である場合に、トランスフォーメーションはシーケンス識別子を分析することができます。

#### GroupKey

レコードがどのグループに属しているかを識別するキー値。

**注:** マッピングパフォーマンスを向上させるには、GroupKey という入力ポートとそれに接続されている出力ポートを同じ精度の値で設定します。

## 一致トランスフォーメーションの出力ポート

定義済み出力ポートには、トランスフォーメーションが実行する分析についてのメタデータが含まれています。

一致トランスフォーメーションの作成後、以下の出力ポートを設定できます。

#### GroupKey

レコードがどのグループに属しているかを識別するキー値。

ダウンストリームのトランスフォーメーション（関連付けトランスフォーメーションなど）が、グループキー値を読み取ることができます。

#### ClusterId

レコードが属しているクラスタの識別子。クラスタ出力に使用されます。

#### ClusterSize

レコードが属しているクラスタ内のレコード数。クラスタに一意のレコードが含まれている場合、クラスタサイズは 1 です。クラスタ出力に使用されます。

#### RowId および RowId1

そのレコードの一意の行識別子。一致トランスフォーメーションは、照合分析の操作中行識別子を使用して行を識別します。識別子は、入力データ内の行番号と一致しない場合があります。

#### DriverId

クラスタ内のドライバレコードの行識別子。クラスタ出力に使用されます。ドライバレコードは、SequenceID 入力ポートに対してクラスタ内で値が最高のレコードです。

#### DriverScore

トランスフォーメーションが、一致ペアの出力とクラスタ化された出力の中にドライバスコアを割り当てます。一致ペアの場合、ドライバスコアはレコードペア間のマッチ率です。クラスタの場合のドライバスコアは、現在のレコードとクラスタ内のドライバの間のマッチ率です。

#### LinkId

現在のレコードと一致し、それをクラスタにリンクしたレコードの行識別子。クラスタ出力に使用されます。

#### LinkScore

クラスタ作成またはクラスタへのレコード追加が起きる 2 つのレコード間のマッチ率。LinkID ポートは、リンクスコアが現在のレコードと同じレコードを特定します。クラスタ出力に使用されます。

#### PersistenceStatus

入力レコードに対する照合分析の結果を表す 8 文字からなるコード。単一ソースの ID 分析で、トランスフォーメーションがデータソースをインデックスデータストアと比較するときに使用されます。

トランスフォーメーションがコードの最初の 3 文字に値を入れます。トランスフォーメーションが各位置で異なる文字を返す可能性があります。トランスフォーメーションは 4 番目から 8 番目の位置に 0 を返します。

一致ペア出力を生成するようにトランスフォーメーションを設定すると、トランスフォーメーションは PersistenceStatus ポートと PersistenceStatus1 ポートを作成します。

#### PersistenceStatusDesc

持続ステータスコードの値に関するテキストによる説明。単一ソースの ID 分析で、トランスフォーメーションがデータソースをインデックスデータスコアと比較するときに使用されます。

一致ペア出力を生成するようにトランスフォーメーションを設定すると、トランスフォーメーションは PersistenceStatusDesc ポートと PersistenceStatusDesc1 ポートを作成します。

## 持続ステータスコードと持続ステータス記述

持続ステータスコードと持続ステータス記述は、一致トランスフォーメーションが分析する異なるタイプのインデックスデータ間のリレーションを説明します。永続 ID データストアを読み取るトランスフォーメーションを設定するときに、トランスフォーメーションがステータスコードとステータス記述を生成します。

トランスフォーメーションが持続ステータスコードを PersistenceStatus ポートに書き込みます。このコードは 8 文字で構成されます。トランスフォーメーションが文字列の 1〜3 番目にコードの値を入力します。トランスフォーメーションは 4 番目から 8 番目の位置に 0 を返します。

トランスフォーメーションが持続ステータス記述を PersistenceStatusDesc ポートに書き込みます。この記述はカンマ区切りの 3 つのテキスト文字列で、それぞれが持続ステータスコードの 1〜3 番目の値を説明します。

トランスフォーメーションはソースデータレコードからのシーケンス識別子の値を使用して、2 つのデータセットのインデックスデータを比較します。

以下の表に、ステータス記述とステータスコードの各位置にトランスフォーメーションが書き込む情報の種類を示します。

| 位置  | 説明                                                                                 |
|-----|------------------------------------------------------------------------------------|
| 1   | レコードを含むデータセットを特定します。                                                               |
| 2   | レコードの重複ステータスを示します。トランスフォーメーションは、トランスフォーメーション入力データとインデックスデータストア間で共通のシーケンス識別子を検索します。 |
| 3   | トランスフォーメーションがデータに実行するアクションを説明します。                                                  |
| 4-8 | ステータスコードの各位置に 0 が含まれます。<br>ステータス記述に位置のについてのテキストは含まれません。                            |

## ステータスコード値とステータス記述値

持続ステータスコードと持続ステータス記述は、トランスフォーメーション入力レコードとデータストアが表すレコード間のリレーションを説明します。トランスフォーメーションはシーケンス識別子の値を使用して、レコードを識別し、データセット内のレコード間のリレーションを判断します。

持続ステータスコードと持続ステータス記述は共通の構造です。ステータスコードとステータス記述は、出力データ文字列の各位置に同じ情報を示します。

### データセットのステータス

ステータスコードとステータス記述の 1 番目の値は、レコードを含むデータセットを特定します。

以下の表に、トランスフォーメーションが1番目の位置に返すことができるステータスコードとステータス記述を示します。

| ステータスコード | ステータスの説明                                    |
|----------|---------------------------------------------|
| S        | ストア。<br>現在のレコードはインデックスデータストアから生成されています。     |
| I        | 入力。<br>現在のレコードはトランスフォーメーション入力データから生成されています。 |

### 重複レコードのステータス

ステータスコードとステータス記述の2番目の値は、トランスフォーメーションインデックスデータと永続データストア間のリレーションを説明します。

以下の表に、トランスフォーメーションが2番目の位置に返すことができるステータスコードとステータス記述を示します。

| ステータスコード | ステータスの説明                                                                                                         |
|----------|------------------------------------------------------------------------------------------------------------------|
| A        | 不完全。<br>インデックスデータストアに現在のレコードのデータが含まれていません。                                                                       |
| E        | 存在する。<br>現在のレコードがインデックスデータストアとトランスフォーメーション入力データに存在します。                                                           |
| I        | 無効。<br>トランスフォーメーションで現在のレコードを分析できません。例えば、[照合タイプ] タブのキーフィールドがレコードデータと互換性がないため、トランスフォーメーションがレコードのインデックスデータを生成できません。 |
| N        | 新規。<br>そのレコードはデータソースにあります。                                                                                       |
| 0        | [ダッシュ]<br>そのレコードはインデックスデータストアにあります。                                                                              |

### データストアのステータス

ステータスコードとステータス記述の3番目の値は、トランスフォーメーションがインデックスデータテーブルに実行するアクションを説明します。

以下の表に、トランスフォーメーションが3番目の位置に返すことができるステータスコードとステータス記述を示します。

| ステータスコード | ステータスの説明                                                                                                                                                                                                                            |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A        | 追加済み。<br>トランスフォーメーションが現在の入力レコードのインデックスデータを永続データストアに追加します。<br>トランスフォーメーション入力データと永続インデックスデータのシーケンス識別子が異なります。                                                                                                                          |
| I        | 無視。<br>トランスフォーメーションが現在の入力レコードのインデックスデータを永続データストアに追加しません。                                                                                                                                                                            |
| N        | トランスフォーメーションは、次のいずれかの説明を返します。 <ul style="list-style-type: none"><li>- 変更なし。<br/>現在のレコードは永続データストアから生成され、トランスフォーメーションは何のアクションも実行しません。</li><li>- 未追加。<br/>ユーザーが定義した一致ポリシーにより、トランスフォーメーションが現在の入力レコードのデータに基づいて永続データストアを更新しません。</li></ul> |
| R        | 削除済み。<br>トランスフォーメーションが、レコードのインデックスデータをインデックスデータストアから削除します。                                                                                                                                                                          |
| U        | 更新済み。<br>トランスフォーメーションがトランスフォーメーション入力レコードのインデックスデータに基づいて永続データストアの行を更新します。<br>トランスフォーメーションの入力データと永続インデックスデータは共通のシーケンス識別子を持ちます。                                                                                                        |

### 持続ステータス記述の例

持続ステータスコード INA00000 には、次の持続ステータス記述があります。

#### 入力、新規、追加

ステータスコードとステータス記述には、レコードに関する次の情報が含まれます。

- レコードはトランスフォーメーション入力データから生成されています。
- 持続データストアにそのレコードのコピーは含まれません。
- トランスフォーメーションがレコードのインデックスデータを永続データストアに追加します。

## 出力ポートと照合出力の選択

選択する照合出力オプションによって、トランスフォーメーションの出力ポートが決まります。例えば、クラスタ化した出力タイプを選択すると、トランスフォーメーションは ClusterId ポートと ClusterSize ポートを作成します。

必要なトランスフォーメーション出力のタイプを選択し、トランスフォーメーションのポートを確認します。

照合出力のタイプを更新する場合、更新後にトランスフォーメーションの出力ポート設定を確認します。マッピングでトランスフォーメーションを使用する場合、マッピングでダウンストリームのオブジェクトに出力ポートを再接続する必要が生じることがあります。

# 一致マッピングレット

一致マッピングレットは、一致トランスフォーメーション内で作成して一致トランスフォーメーションに埋め込むことができるマッピングレットのタイプです。

一致マッピングレットを作成するには、一致トランスフォーメーションの設定を一致マッピングレットとして保存します。一致マッピングレットを作成する場合は、一致トランスフォーメーションの設定を比較トランスフォーメーションおよび加重平均トランスフォーメーションに変換します。

一致マッピングレットの作成後に、トランスフォーメーションを追加して照合分析をカスタマイズすることができます。例えば、2つのストラテジのリンクスコアを評価する式トランスフォーメーションを追加し、一番高いスコアを選択することができます。

一致トランスフォーメーションと違って一致マッピングレットはパッシブで、Analyst ツール内のルールとして使用できます。Analyst ツールで一致マッピングレットを使用し、データプロファイリングプロセスの一部としてレコードを一致させます。

一致トランスフォーメーションは、一致トランスフォーメーション内から作成された一致マッピングレットのみを読み取ることができます。

## 一致マッピングレットの作成

複数のトランスフォーメーションを使用する照合分析操作を定義するため、一致マッピングレットを作成します。

1. エディタで一致トランスフォーメーションを開き、**【ストラテジ】** ビューを選択します。
2. **【照合ルールを使用】** を選択します。
3. **【名前】** フィールドで、**【新規作成】** を選択します。  
**【新しいマッピングレット】** ウィンドウが開きます。
4. **【新しいマッピングレット】** ウィンドウで、マッピングレットの名前を入力し、マッピングレットを保存する場所を選択します。
5. 必要に応じて、**【一致トランスフォーメーションのストラテジを再利用】** を選択して、入力、ストラテジ、および加重を、現在の一致トランスフォーメーションから一致マッピングレットにコピーします。  
**注:** この設定を使用して、一致トランスフォーメーションに現在定義されている一致機能を複製する一致マッピングレットを短時間で作成することをお勧めします。
6. **【完了】** をクリックします。  
一致マッピングレットがエディタに表示されます。
7. 必要に応じて、比較トランスフォーメーションと加重平均トランスフォーメーションを一致マッピングレット内に追加し設定することによって、一致操作を作成します。
8. **【ファイル】** > **【保存】** をクリックして、マッピングレットを保存します。
9. マッピングレットを閉じ、一致トランスフォーメーションを含むエディタを選択します。作成したマッピングレットが**【名前】** フィールドに表示されていることを確認します。
10. 必要に応じて、**【照合フィールド】** ボタンをクリックしてマッピングレット内の照合フィールドを設定します。  
**【照合ルールの設定】** ウィンドウが開きます。
11. **【入力フィールド】** カラムおよび**【使用可能な入力】** カラム内のフィールドをダブルクリックして、入力ポートを一致入力に割り当てます。
12. **【ファイル】** > **【保存】** をクリックして、トランスフォーメーションを保存します。

## 一致マプレットの使用

以前に定義された一致マプレットを一致トランスフォーメーションで選択および設定できます。

1. エディタで一致トランスフォーメーションを開き、**【ストラテジ】** ビューを選択します。
2. **【照合ルールを使用】** を選択します。
3. **【名前】** フィールドで、**【既存を使用】** を選択します。  
**【照合ルールの設定】** ウィンドウが開きます。
4. **【参照】** をクリックして、リポジトリ内の一致マプレットを探します。  
**重要:** 選択できるのは、一致トランスフォーメーションによって作成されたマプレットのみです。  
**【照合マプレットの選択】** ウィンドウが開きます。
5. 一致マプレットを選択し、**【OK】** をクリックします。
6. **【入力フィールド】** および **【使用可能な入力】** カラム内のフィールドをダブルクリックして、入力ポートを一致入力に割り当てます。
7. **【OK】** をクリックします。  
**【照合ルールの設定】** ウィンドウが閉じます。
8. **【ファイル】** > **【保存】** をクリックして、一致トランスフォーメーションを保存します。

## 照合分析操作の設定

一致操作を設定するには、ソースデータを一致トランスフォーメーションに接続し、トランスフォーメーションのビューでプロパティを編集します。

1. 一致トランスフォーメーションを作成し、ソースデータをトランスフォーメーションに接続します。
2. **【照合タイプ】** ビューを選択し、照合タイプを選択します。
3. 選択した照合操作のタイプのプロパティを設定します。  
デュアルソース照合タイプを選択した場合は、**【マスターデータセット】** プロパティを設定します。
4. **【ストラテジ】** ビューを選択し、**【照合ストラテジの定義】** を選択します。
5. **【新規】** をクリックします。  
**新しい照合ストラテジウィザード**が開きます。
6. 一致ストラテジを選択し、**【次へ】** をクリックします。
7. 必要に応じて、加重と NULL 一致の設定を編集します。**【次へ】** をクリックします。
8. **【使用可能】** カラム内のセルをダブルクリックして、分析する入力ポートを選択します。  
**【次へ】** をクリックして別のストラテジを設定するか、**【完了】** をクリックしてウィザードを終了します。  
**注:** ストラテジの設定を編集するには、**【ストラテジ】** ビューでそのストラテジのセル内の矢印をクリックします。
9. **【照合出力】** ビューを選択します。  
照合出力タイプを選択し、そのプロパティを設定します。

**注:** **【ストラテジ】** ビューで一致マプレットを選択または編集することによって、一致ストラテジを設定することもできます。一致マプレットは、一致トランスフォーメーションに埋め込むことができるマプレットのタイプです。

# 非ネイティブ環境での一致トランスフォーメーション

非ネイティブ環境での一致トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングで制限付きでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## Blaze エンジンでの一致トランスフォーメーション

マッピング検証は、一致トランスフォーメーションが ID インデックスデータをデータベーステーブルに書き込むように設定されていると失敗します。

一致トランスフォーメーションでは、クラスタ ID 値はネイティブ環境と非ネイティブ環境で異なって生成されます。非ネイティブ環境では、トランスフォーメーションはグループ ID 値をクラスタ ID に追加します。

## Spark エンジンでの一致トランスフォーメーション

マッピング検証は、一致トランスフォーメーションが ID インデックスデータをデータベーステーブルに書き込むように設定されていると失敗します。

一致トランスフォーメーションでは、クラスタ ID 値はネイティブ環境と非ネイティブ環境で異なって生成されます。非ネイティブ環境では、トランスフォーメーションはグループ ID 値をクラスタ ID に追加します。

## 第 29 章

# フィールド分析での一致トランスフォーメーション

この章では、以下の項目について説明します。

- [フィールド一致分析, 468 ページ](#)
- [フィールド一致分析の処理フロー, 468 ページ](#)
- [フィールド照合タイプのオプション, 469 ページ](#)
- [フィールド一致ストラテジ, 469 ページ](#)
- [フィールド一致の出力オプション, 472 ページ](#)
- [フィールド一致の詳細プロパティ, 474 ページ](#)
- [フィールド一致分析の例, 475 ページ](#)

## フィールド一致分析

単一のデータセット内または 2 つデータセット間で重複しているレコードや類似レコードを見つけるには、フィールド一致分析を実行します。

フィールド一致分析のための一致トランスフォーメーションを設定するときは、以下のビューに関するオプションを設置します。

- 照合タイプ
- ストラテジ
- 照合出力

任意で、パラメータビューと詳細ビューのオプションを設定します。

## フィールド一致分析の処理フロー

次の処理フローは、フィールド一致分析用に一致トランスフォーメーションを設定するために行う手順をまとめたものです。一致トランスフォーメーションを単独で使用するプロセス、または一致トランスフォーメーションや他のトランスフォーメーションを使用するプロセスを定義することができます。

**注:** フィールド一致分析でマッピングに一致トランスフォーメーションを追加する場合、アップストリームのキージェネレータートランスフォーメーションをマッピングに追加します。



一致トランスフォーメーション用にデータを準備するには、以下の手順を実行します。

1. ソースデータレコードをグループに分けて整理します。  
キージェネレータトランスフォーメーションを使用して、グループキー値を各レコードに割り当てます。グループ割り当てによって、一致トランスフォーメーションが実行する必要がある計算の回数を減らすことができます。
2. データソースレコードに一意的シーケンス識別子の値が入っていることを確認します。キージェネレータトランスフォーメーションを使用して値を作成することができます。

一致トランスフォーメーションで次の手順を実行します。

1. 照合タイプとしてフィールド分析を指定し、データソースの数を指定します。  
2つのデータセットを分析するようにトランスフォーメーションを設定する場合、マスターデータセットを選択します。  
**【照合タイプ】** ビューを使用してデータソースのタイプと数を設定します。
2. 照合分析ストラテジを定義します。アルゴリズムを選択し、カラムのペアをアルゴリズムに割り当てます。  
**【ストラテジ】** ビューを使用してストラテジを定義します。
3. トランスフォーメーションが照合分析の結果の生成に使用するメソッドを指定します。
4. 一致しきい値を設定します。一致しきい値とは、2つのレコードが互いの重複であると識別するための最小スコアです。

**【照合出力】** ビューを使用して出力メソッドと一致しきい値を選択します。

**注:** 一致トランスフォーメーションまたは加重平均トランスフォーメーションの一致しきい値を設定することができます。一致マップレットを作成する場合は加重平均トランスフォーメーションを使用します。

## フィールド照合タイプのオプション

**【照合タイプ】** ビューには、デュアルソースでの照合分析に適用されるオプションが1つあります。このオプションはマスターデータセットを特定します。**【照合タイプ】** ビューには、単一ソースでの照合分析のためのオプションがありません。

2つのデータセットを分析するときは、そのうち1つをマスターデータセットとして選択する必要があります。どちらのデータセットもプロジェクトまたは組織内のマスターデータセットに相当しない場合は、大きいほうのデータセットをマスターデータセットとして選択します。

マスターデータセットを指定するには、**【マスターデータセット】** オプションを使用します。

## フィールド一致ストラテジ

**【ストラテジ】** ビューには、入力データに対して定義したストラテジがリストで表示されます。

ストラテジは、トランスフォーメーションがデータソースレコード間の類似性と差異を測定する方法を決定します。

## フィールド一致のアルゴリズム

一致トランスフォーメーションには、2つのカラム間のデータ値を比較するアルゴリズムが含まれています。このアルゴリズムによって、データ値間の差異の程度を計算する方法が異なります。

選択したカラム内に見つかりと想定しているデータ差異のタイプを測定できるアルゴリズムを選択します。

### バイグラム

バイグラムアルゴリズムは、郵便アドレスが1つのフィールドに入力されている場合など、長いテキスト文字列を比較する場合に使用します。

バイグラムアルゴリズムでは、2つのデータ文字列の一致スコアを、両方の文字列に含まれる連続した文字に基づいて計算します。このアルゴリズムでは、連続する文字から成る文字ペアが両方の文字列に共通して存在するかを調べます。そして相手の文字列に一致するものが存在する文字ペアの数を、両方の文字列の文字ペアの総数で割ります。

#### バイグラムの例

次の文字列について考えてみます。

- larder
- lerder

これらの文字列をバイグラムのグループに分けると次のようになります。

```
l a, a r, r d, d e, e r
l e, e r, r d, d e, e r
```

文字列"lerder"の2つ目の"e r"は一致と見なされません。文字列"larder"には文字列"e r"が1つしかなく、2つ目に対応するものはないからです。

バイグラムの一致スコアを計算するには、一致するペアの数（6）を両方の文字列のペアの総数（10）で割ります。この例では、文字列の類似度は60%で、一致スコアは0.60になります。

### ハミング距離

電話番号、郵便番号、製品コードなどの数値フィールドやコードフィールドのように、データ文字の位置が重要な要素である場合には、ハミング距離アルゴリズムを使用します。

ハミング距離アルゴリズムでは、2つのデータ文字列の一致スコアを、データ文字列間で文字が異なる位置の数に基づいて計算します。長さが異なる文字列の場合、長い方の文字列にしかない各文字は文字列間の相違としてカウントされます。

#### ハミング距離の例

次の文字列について考えてみます。

- Morlow
- Marlowes

強調表示された文字は、ハミングアルゴリズムで相違と見なされる位置を示しています。

ハミングの一致スコアを計算するには、一致する文字の数（5）を長い方の文字列の文字数（8）で割ります。この例では、文字列の類似度は62.5%で、一致スコアは0.625になります。

## エディット距離

エディット距離アルゴリズムは、単語や短いテキスト文字列（名前など）を比較する場合に使用します。

エディット距離アルゴリズムでは、文字列を別の文字列に変換するために文字の挿入、削除、または置き換えが必要な最小限の「コスト」を計算します。

### エディット距離の例

次の文字列について考えてみます。

- Levenston
- Levenshtein

強調表示された文字は、文字列をもう一方の文字列に変換するために処理が必要な部分を示しています。

エディット距離アルゴリズムでは、変更されない文字の数（8）を長い方の文字列の文字数（11）で割ります。この例では、文字列の類似度は 72.7% で、一致スコアは 0.727 になります。

## Jaro 距離

2つの文字列を比較するときに文字列内の最初の文字の類似度を優先する場合は、Jaro 距離アルゴリズムを使用します。

Jaro 距離のマッチ率は、両方の文字列から最初の 4 文字を取り出して比較したときの類似度、および識別された文字の転置の数を反映しています。最初の 4 文字の一致の重要度に、[ペナルティ] プロパティに入力した値を使用して重みが設定されます。

### Jaro 距離のプロパティ

Jaro 距離アルゴリズムを設定する場合は、次のプロパティが設定できます。

#### ペナルティ

比較する 2 つの文字列内の最初の 4 文字が同一でない場合の一致スコアのペナルティを指定します。最初の文字が一致しない場合は、ペナルティの値がそのまま減算されます。それ以外の文字が異なる場合は、その位置に基づいてペナルティの端数が減算されます。デフォルトのペナルティ値は 0.20 です。

#### 大文字小文字の区別

Jaro 距離アルゴリズムで文字の比較を行うときに大文字と小文字を区別するかどうかを指定します。

### Jaro 距離の例

次の文字列について考えてみます。

- 391859
- 813995

[ペナルティ] を 0.20（デフォルト値）にしてこれらの文字列を分析した場合、Jaro 距離アルゴリズムで返される一致スコアは 0.513 になります。文字列の類似度は 51.3% となります。

## ハミング距離の反転

ハミング距離反転アルゴリズムは、2 つの文字列間で文字が異なる位置の割合を、文字列を右から左に読み取りながら計算する場合に使用します。

ハミング距離アルゴリズムでは、2 つのデータ文字列の一致スコアを、データ文字列間で文字が異なる位置の数に基づいて計算します。長さが異なる文字列の場合、長い方の文字列にしかない各文字は文字列間の相違としてカウントされます。

## ハミング距離の反転の例

次の文字列について考えてみましょう。この文字列は、ハミング反転アルゴリズムについて説明するために右から左に文字を配置しています。

- 1-999-9999
- **011-01-999-9991**

強調表示された文字は、ハミング距離反転アルゴリズムで相違と見なされる位置を示しています。

ハミングの反転の一致スコアを計算するには、一致する文字の数（9）を長い方の文字列の文字数（15）で割ります。この例では、一致スコアは 0.6 になり、文字列の類似度は 60% となります。

## フィールド一致ストラテジのプロパティ

**【ストラテジ】** ビューで **【ストラテジ】** ウィザードを開き、フィールド一致ストラテジごとにプロパティを設定します。

フィールド一致ストラテジの設定時には、次のプロパティを設定できます。

### 名前

名前でストラテジを特定します。

### 加重

レコードの全体スコアが計算される場合、マッチ率に割り当てられた相対優先順位を指定します。デフォルトは 0.5 です。

### 1 つのフィールドが NULL

ペアのデータ値の一方が NULL の場合、アルゴリズムでこのペアに適用されるマッチ率を定義します。デフォルトは 0.5 です。

### 両方のフィールドが NULL

ペアのデータ値の両方の値が NULL の場合、アルゴリズムでこのペアに適用されるマッチ率を定義します。デフォルトは 0.5 です。

**注:** 一致したカラムの値の一方または両方が NULL の場合、照合アルゴリズムはマッチ率を計算しません。アルゴリズムは、NULL の一致プロパティに定義されたスコアを適用します。NULL の一致プロパティは取り消すことができません。

## フィールド一致の出力オプション

フィールド照合分析の出力形式を定義するには、**【照合出力】** オプションを設定します。

**【照合出力のタイプ】** 領域と **【プロパティ】** 領域のオプションを設定します。

## 照合出力のタイプ

**【照合出力】** ビューには、出力データの形式を指定するオプションがあります。レコードをクラスタ内または一致ペア内に書き込むようにトランスフォーメーションを設定することができます。

以下のいずれかの照合出力のタイプを選択します。

### 最良の一致

マスタデータセット内の各レコードを、第2データセット内でマッチ率が最高のレコードを使って書き込みます。照合操作では、第2データセットでマスタレコードとのマッチ率が最も高いレコードが選択されます。2つ以上のレコードが最高のマッチ率を返す場合、照合操作で第2データセット内の最初のレコードが選択されます。最良の一致では、レコードの各ペアを単一の行に書き込みます。

トランスフォーメーションをデュアルソースでの分析に設定する場合に【**最良の一致**】を選択することができます。

### クラスタ

一致しきい値を満たすマッチ率によって相互にリンクするレコードセットを含むクラスタを書き込みます。クラスタ内の各レコードは、マッチ率のしきい値を満たす1つ以上の他のレコードと一致する必要があります。

単一ソース分析およびデュアルソース分析のためにトランスフォーメーションを設定する場合、【**クラスタ**】を選択できます。

### 一致ペア

一致しきい値を満足するマッチ率で相互に一致するレコードの全ペアを書き込みます。トランスフォーメーションは各ペアを1行に書き込み、各ペアのマッチ率を各行に追加します。レコードが他の複数のレコードと一致する場合、トランスフォーメーションはレコードペアごとに1行を書き込みます。

単一ソース分析およびデュアルソース分析のためにトランスフォーメーションを設定する場合、【**一致ペア**】を選択できます。

## 照合出力のプロパティ

【照合出力】ビューは、キャッシュメモリの動作、マッチ率のしきい値、およびトランスフォーメーション出力に表示されるマッチ率を指定するプロパティが含まれます。

照合出力のプロパティを使用して、トランスフォーメーションがマッチ率の値を出力レコードに追加する方法を指定することもできます。

照合出力タイプの選択後に、以下のプロパティを設定します。

### キャッシュディレクトリ

データ統合サービスがフィールド照合分析中に一時データを書き込むディレクトリを指定します。照合分析が生成するデータ量が利用可能なシステムメモリ量より大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

### キャッシュサイズ

データ統合サービスがフィールド照合分析に割り当てるシステムメモリの量を決定します。デフォルト値は400,000 バイトです。

データをソートする前に、データ統合サービスは指定されているメモリ量を割り当てます。照合分析によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュディレクトリに書き込みます。照合分析にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

**注:** 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

## しきい値

2つのレコードを互いの重複候補とみなす最小マッチ率を設定します。

パラメータをしきい値に割り当てることができます。0 から 1 の範囲で 10 進値を設定します。

## スコアリング方法

トランスフォーメーションの出力に表示されるマッチ率の値を決定します。クラスタ出力のスコアリング方法を選択します。

下表でスコアリング方法について説明します。

| スコアリング方法のオプション | 説明                                      |
|----------------|-----------------------------------------|
| 前後方向           | リンクスコアとドライバスコアをクラスタ内の各レコードに追加します。       |
| リンクスコア         | リンクスコアをクラスタ内の各レコードに追加します。デフォルトのオプションです。 |
| ドライバスコア        | ドライバスコアをクラスタ内の各レコードに追加します。              |
| なし             | マッチ率をクラスタ内のどのレコードにも追加しません。              |

**注:** ドライバスコアをレコードに追加すると、マッピング実行時間が長くなります。マッピングは、クラスタの完了後にドライバスコアの値をレコードに追加するため、すべてのクラスタが完了するまで待機しています。

# フィールド一致の詳細プロパティ

トランスフォーメーションには、実行インスタンスの数、トランスフォーメーションが同じ行を分析する方法、およびログデータのトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

## 実行インスタンス数

トランスフォーメーションが実行時に使用するスレッド数を決定します。

一致トランスフォーメーションでは、フィールド一致分析で単一の実行インスタンスを使用します。ID 照合分析用にトランスフォーメーションを設定するときに、実行インスタンスの数を編集することができます。

## 完全一致のフィルタ

トランスフォーメーションが、照合ストラテジで入力データの同じレコードのペアに比較アルゴリズムを適用するかどうかを決定します。

トランスフォーメーションが同じレコードのペアを検出すると、アルゴリズムはレコード間の類似性のレベルを分析する必要がありません。トランスフォーメーションは、追加の分析を行わずにレコードを出力ステージに直接渡すことができます。同じレコードを出力ステージに直接渡すようにトランスフォーメーションを設定するには、[完全一致のフィルタ] を選択します。入力データに多くの同じ行が含まれる場合、比較アルゴリズムが実行する計算が少なくなり、マッピングが高速になります。

入力データに多くの同じ行が含まれる場合は、このオプションを選択します。入力データに多くの同じ行が含まれない場合は、トランスフォーメーションの実行速度が低下することがあるので、このオプションを選択しないでください。

**注:** オプションを選択するか、または選択を取り消しても、トランスフォーメーションの出力には同じレコードデータが含まれます。オプションを選択するか、または選択を取り消すと、トランスフォーメーションは異なるリンクスコアとドライバスコアを出力レコードに割り当てることがあります。

トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

フィールド一致分析の例

あなたは小売銀行でのデータスチュアートです。過去 7 日間に銀行口座を開設した顧客に関する口座レコードを、まとめて受け取ります。このデータセットに重複レコードがないか検証する必要があります。そこで、全レコードから重複データを探すマッピングを設計します。

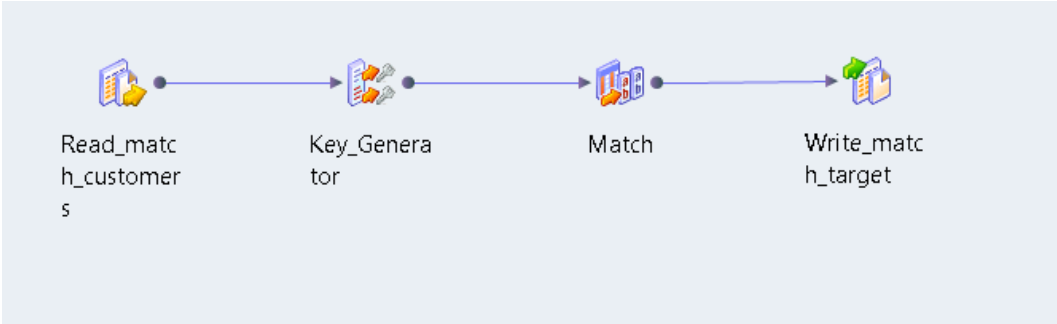
マッピングの作成

複数フィールドから重複データを探すマッピングを作成します。

このマッピングでは、以下のタスクを実行します。

- データソースを読み込む。
- グループキー値とシーケンス ID 値をソースレコードに追加する。
- レコードのフィールドデータを分析する。
- 結果をデータターゲットに書き込む。

次の図に、Developer ツールのマッピングを示します。



作成するマッピングには、以下のオブジェクトが含まれます。

| オブジェクト名              | 説明                                                          |
|----------------------|-------------------------------------------------------------|
| Read_Match_Customers | データソース。<br>口座所有者の詳細（名前、アドレス、口座番号など）が入ります。                   |
| Key_Generator        | キージェネレータートランスフォーメーション。<br>グループキーの値およびシーケンス識別子をソースデータに追加します。 |

| オブジェクト名            | 説明                                    |
|--------------------|---------------------------------------|
| 一致                 | 一致トランスフォーメーション。<br>口座データの重複レベルを分析します。 |
| Write_match_target | データターゲット。<br>フィールド一致分析の結果が入ります。       |

## 入力データサンプル

データセットには、各顧客の口座番号、名前、住所、会社名が入っています。あなたはこのデータセットからモデルリポジトリでデータソースを作成します。データソースをマッピングに追加します。

次に示すデータの抜粋は、顧客の口座データのサンプルです。

| CustomerID | Lastname | 市区町村         | 州  | ZIP   |
|------------|----------|--------------|----|-------|
| 15954467   | JONES    | SCARSDALE    | NY | 10583 |
| 10110907   | JONES    | MINNEAPOLIS  | MN | 55437 |
| 19131127   | JONES    | INDIANAPOLIS | IN | 46240 |
| 10112097   | JONES    | HOUSTON      | TX | 77036 |
| 19133807   | JONES    | PLANTATION   | FL | 33324 |
| 10112447   | JONES    | SCARSDALE    | NY | 10583 |
| 15952487   | JONES    | HOUSTON      | TX | 77002 |
| 10112027   | JONES    | OAKLAND      | CA | 94623 |

## キージェネレータトランスフォーメーションの設定

キージェネレータトランスフォーメーションを設定するときは、分析対象にするデータソースポートを接続します。さらにグループキーのデータが入ったポートを指定します。レコードに一意の識別子が入っていない場合、シーケンス ID ポートを使用して一意の識別子をレコードに追加します。

グループキーのポートを指定するときは、以下のガイドラインを考慮してください。

- ポートデータで規則的に繰り返される値の入ったポートを選択する。可能であれば、サイズの似通ったグループを形成するポートを選択するのが望ましい。
- 重複分析と関連性のないポートを選択する。

現在の例では、City ポートをグループキーとして選択します。口座名が同じ都市で 2 回以上現れる場合、その口座に重複データが含まれている可能性があります。口座名が異なる都市で 2 回以上現れる場合、その口座が重複している可能性は低いといえます。

**ヒント：**グループキーのポートを選択する前にデータソースに対しカラムプロファイルを実行してください。プロファイルの結果が、各値がポートに現れる回数を示している可能性があります。



## 一致トランスフォーメーションの設定

再利用不可能な一致トランスフォーメーションをマッピングに追加して、フィールド分析を実行します。

以下のタスクを実行して、一致トランスフォーメーションの設定を行ってください。

1. 実行する照合分析のタイプを選択する。
2. 入力ポートをトランスフォーメーションに接続する。
3. レコードデータを比較するようにストラテジを設定する。
4. トランスフォーメーションで作成する照合出力データのタイプを選択する。
5. 出力ポートをデータターゲットに接続する。

### 照合操作のタイプの選択

**【照合タイプ】** ビューで、照合操作を選択するオプションを使用することができます。口座データを比較するために、単一ソースでフィールド照合分析を実行するようにトランスフォーメーションを設定します。

### 入力ポートの接続

顧客口座データのポートを一致トランスフォーメーションに接続します。

一致トランスフォーメーションは、設定済み入力ポートを使用してレコードを処理する順序を決めます。このトランスフォーメーションでは、シーケンス識別子を使用して入力ポートからのレコードを、出力として書き込む一致ペアまたはクラスタまで追跡します。このトランスフォーメーションではグループキーを使用して、処理するレコードを並び替えます。

以下に示す一致トランスフォーメーションの設定済みポートを、キージェネレータトランスフォーメーションの設定済み出力ポートに接続してください。

- SequenceID
- GroupKey

### フィールド分析用のストラテジ設定

**【ストラテジ】** ビューのオプションを使用してストラテジを設定します。トランスフォーメーションがレコードデータに対して実行する分析のタイプは、ストラテジによって決まります。

以下のストラテジを作成します。

- 顧客 ID 番号の分析に、エディット距離アルゴリズムを使用するストラテジを作成します。CustomerID\_1 というポートと、CustomerID\_2 というポートを選択します。
- 名字データの分析には、Jaro 距離アルゴリズムを使用するストラテジを作成します。Lastname\_1 というポートと、Lastname\_2 というポートを選択します。  
**注:** Jaro 距離アルゴリズムでは、異なる文字で始まる類似文字列に追加ペナルティを適用します。したがって、Jaro 距離アルゴリズムが PATTON と PATTEN に高いマッチ率を適用する一方、BAYLOR と TAYLOR に低いマッチ率を適用するという可能性があります。
- ZIP コードデータの分析には、逆ハミング距離アルゴリズムを使用するストラテジを作成します。Zip\_1 というポートと、Zip\_2 というポートを選択します。

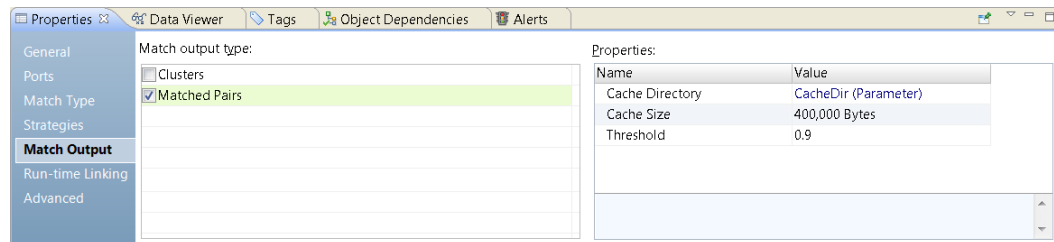
関連項目：

- [「フィールド一致のアルゴリズム」 \(ページ 470\)](#)

## 照合出力のタイプの選択

【照合出力】ビューに対するオプションを使用して照合分析の結果を出力するときの形式を定義します。

次の図に、単一ソースでのフィールド照合分析の場合の【照合出力】ビューを示します。



このトランスフォーメーションを、一致ペア内の出力レコードを整理するように設定します。このトランスフォーメーションはレコードをペアで返すため、分析の結果には一意のレコードが含まれません。

## 出力ポートの接続

一致トランスフォーメーションの出力ポートをマッピングのデータターゲットに接続します。データターゲットに書き込むレコードデータが入っているポートを選択します。

このトランスフォーメーションでは、各一致ペアのレコードを特定する設定済みポートが含まれています。設定済みポート名は **RowId** と **RowId1** です。各行の ID 値で、出力データ内のレコードを特定します。

行 ID 値は一致ストラテジで選択するポートに対応します。ストラテジを設定するときは、サフィックスが **\_1** または **\_2** のポート名を選択します。RowId の値で、サフィックスが「\_1」のポートを含んだレコードを特定します。RowId1 の値で、サフィックスが「\_2」のポートを含んだレコードを特定します。

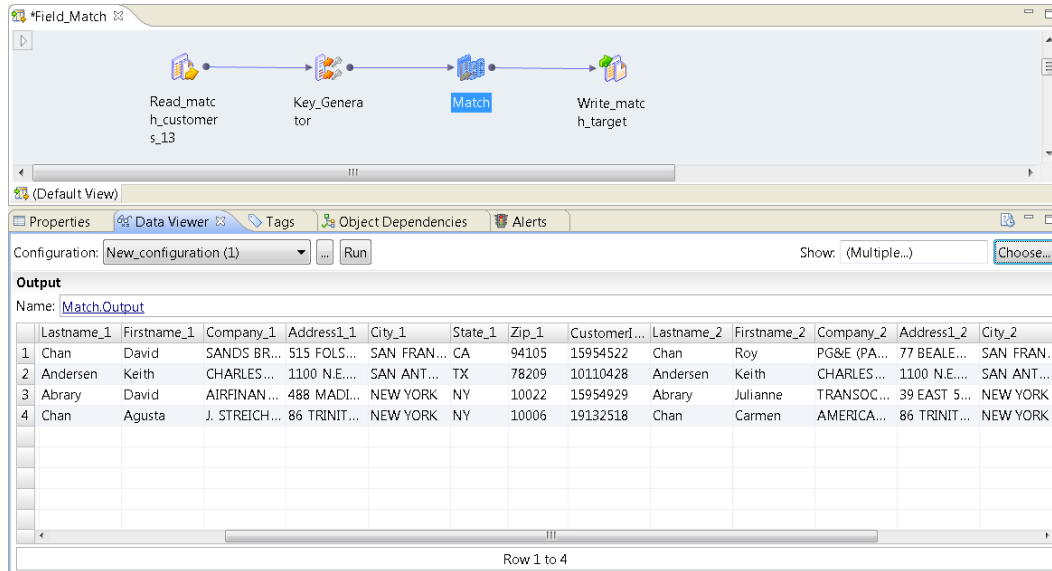
他の出力ポートを使用してレコード間のリレーションを確認することもできます。リンクポートの値とドライバポートの値は、各クラスタ内のレコード間の類似度を示します。

この例では、すべてのポートをデータターゲットに接続します。ポートで出力データを表示するには、データビューアを実行します。

## データビューアの実行

データビューアを実行すると、照合分析の結果が確認できます。デフォルトでは、一致トランスフォーメーションに対するすべての出力ポートがデータビューアに表示されます。マッピングを実行する場合は、データターゲットを出力ポートからのデータで更新します。

次の図に、データビューアに表示された出力データを示します。



データビューアでは、顧客の口座データに 1 つまたは複数の重複レコードが含まれているかを検証します。

以下のデータをデータビューアで考慮してください。

- このトランスフォーメーションで、同じ名字と住所データが同じであることから Augusta Chan と Carmen Chan のレコードに同じ情報が入っている可能性があることが検出されました。ユーザーはデータを確認するときに、データセット内でレコードが一意であるかを判断します。ここで、両方のレコードの顧客 ID が同じ値になっていることに注意します。顧客 ID のカラムはデータセットの中のプライマリーキーですから、この場合ニューヨークのオフィスに連絡します。ニューヨークのオフィスでこのエラーを解決します。
- このトランスフォーメーションで、Keith Anderson のレコードに同じ情報が入っている可能性があることが検出されました。ユーザーはレコードを確認するときに、2 つのレコードが同じ口座を表しているか検証します。ここで、これらのレコードの顧客 ID の値が異なっていることに注意します。顧客の口座の ID は単一の値になる必要があるため、この場合サンアントニオのオフィスに連絡します。サンアントニオのオフィスでこのエラーを解決します。

## まとめ

照合分析の結果が、顧客の口座データのセットに少なくとも 1 つの重複レコードペアがあることを示しています。確認する必要のある口座を管理している支店に連絡します。データセット内の他のすべてのレコードが一意で顧客の口座を特定することを確認します。

## 第 30 章

# ID 分析での一致トランスフォーメーション

この章では、以下の項目について説明します。

- [ID 照合分析, 480 ページ](#)
- [ID 照合分析の処理フロー, 481 ページ](#)
- [ID 一致タイプのプロパティ, 481 ページ](#)
- [ID 照合ストラテジ, 485 ページ](#)
- [ID 照合の出力オプション, 487 ページ](#)
- [ID 照合の詳細プロパティ, 490 ページ](#)
- [永続インデックスのケーススタディ, 491 ページ](#)
- [ID 照合分析の例, 493 ページ](#)

## ID 照合分析

単一のデータセット内または 2 つデータセット間で重複している ID や類似 ID をを見つけるには、ID 照合分析を実行します。

2 つ以上のレコードの中で類似する ID が、レコードの重複を示す可能性があります。あるいは、類似する ID（共有されている家族 ID や従業員 ID など）がレコード間の接続を示している可能性があります。

ID 照合分析のための一致トランスフォーメーションを設定するときは、以下のビューに関するオプションを設定します。

- 照合タイプ
- ストラテジ
- 照合出力

任意で、パラメータビューと詳細ビューのオプションを設定します。

一致トランスフォーメーションを ID 照合分析用に設定する場合、ID ストラテジ内のプライマリ必須フィールドすべてに入力ポートを接続します。ほとんどの ID ストラテジにはプライマリ必須フィールドが含まれます。一部のストラテジにもセカンダリ必須フィールドが含まれています。少なくとも 1 つのセカンダリ必須フィールドに入力ポートを接続します。

# ID 照合分析の処理フロー

次の処理フローは、ID 照合分析用に一致トランスフォーメーションを設定するために行う手順をまとめたものです。一致トランスフォーメーションを単独で使用するプロセス、または一致トランスフォーメーションや他のトランスフォーメーションを使用するプロセスを定義することができます。

一致トランスフォーメーションをアップストリームのデータオブジェクトに接続する前に、対象レコードに一意のシーケンス識別子の値が含まれているか確認します。キージェネレータトランスフォーメーションを使用して値を作成することができます。ID 照合分析を実行するときは、任意で入力データをグループに分けて整理することができます。

一致トランスフォーメーションで次の手順を実行します。

1. 照合タイプとして ID 分析を指定し、データソースの数を指定します。  
2 つのデータセットを分析するようにトランスフォーメーションを設定する場合、マスターデータセットを選択します。  
[照合タイプ] ビューを使用してデータソースのタイプと数を設定します。
2. インデックスデータを保存する場所を特定します。トランスフォーメーションは、インデックスデータを一時ファイルに書き込むことや、インデックスデータをデータベーステーブルに保存することができます。  
[照合タイプ] ビューを使用してインデックスデータストアを指定します。
3. 照合分析ストラテジを定義します。ポピュレーションと比較アルゴリズムを選択し、カラムのペアをアルゴリズムに割り当てます。  
ポピュレーションは、選択するカラムペアを示します。  
[ストラテジ] ビューを使用してストラテジを定義します。
4. トランスフォーメーションが照合分析の結果の生成に使用するメソッドを指定します。
5. 一致しきい値を設定します。一致しきい値とは、2 つのレコードが互いの重複であると識別するための最小スコアです。  
[照合出力] ビューを使用して出力メソッドと一致しきい値を選択します。  
**注:** 一致トランスフォーメーションまたは加重平均トランスフォーメーションの一致しきい値を設定することができます。一致マップレットを作成する場合は加重平均トランスフォーメーションを使用します。

## ID 一致タイプのプロパティ

[一致タイプ] ビューを使用して、一致トランスフォーメーションが実行する分析のタイプの指定や、分析を定義するプロパティの設定を行います。単一ソースでの分析またはデュアルソースでの分析を指定できます。また、ID インデックスデータの永続データストアも指定できます。

設定するプロパティは、選択した分析のタイプによって異なります。オプションの多くはどのタイプの分析でも共通です。

### 共通プロパティ

次のプロパティはどのタイプの分析でも共通です。

#### ポピュレーション

トランスフォーメーションが使用するポピュレーションファイルを指定します。ポピュレーションファイルには、インデックスキーを生成するキー構築アルゴリズムが含まれます。

## キーレベル

ID 照合アルゴリズムが生成するキーの数を決定します。デフォルトの設定は[標準]です。[限定]設定を選択すると、キーの数が減って精度が高くなりますが、処理時間は長くなります。[拡張]設定を選択すると、キーの数が増えて精度は低くなりますが、処理時間は短くなります。

## キータイプ

キーフィールドに含まれる情報のタイプを記述します。ID 照合分析では、個人名、組織、および住所のキーを生成できます。[キーフィールド] プロパティに選択するカラムを最も適切に表すキータイプを選択します。

## 検索レベル

トランスフォーメーションが照合分析に適用する検索の深さと速度のバランスを示します。検索の深さは、返される一致数と逆の相関があります。例えば、[高] オプションが返す一致は少なくなります。

## キーフィールド

一致トランスフォーメーションがインデックスキーデータの生成に使用するカラムを指定します。選択したカラムに [キータイプ] プロパティで指定した種類の情報が含まれることを確認します。

## インデックスディレクトリ

データ統合サービスが現在のトランスフォーメーションのインデックスキーデータを書き込むディレクトリを指定します。デフォルトでは、プロパティは空白です。インデックスディレクトリを指定しない場合、データ統合サービスはコンテンツ管理サービスに設定されている場所を使用します。

ディレクトリへのパスを入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。

## キャッシュディレクトリ

データ統合サービスが、ID 照合分析のインデックス作成ステージ中に一時データを書き込むディレクトリを指定します。現在のトランスフォーメーションからのデータの場所を指定するプロパティを更新します。デフォルトでは、プロパティは空白です。キャッシュディレクトリを指定しない場合、データ統合サービスはコンテンツ管理サービスに設定されている場所を使用します。

ディレクトリへのパスを入力するか、またはパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。

## キャッシュサイズ

データ統合サービスが ID インデックスの作成に割り当てるシステムメモリの量を決定します。デフォルト値は 400,000 バイトです。

インデックス作成操作によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュディレクトリに書き込みます。この操作にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

**注:** 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

## デュアルソースのプロパティ

デュアルソースでの分析のトランスフォーメーションを設定するときは、共通プロパティのほかに次のプロパティも設定します。

## マスターデータセット

マスターデータを含むデータソースを指定します。デュアルソース分析のマスターデータセットを指定します。

## 永続データ記憶領域プロパティ

永続インデックスデータストアを使用するトランスフォーメーションを設定するときは、共通プロパティのほかに次のプロパティも設定します。

### 永続方法

トランスフォーメーションがマッピングデータソースからのインデックスデータを使用して、現在のインデックステーブルを更新するかどうかを指定します。次のいずれかのオプションを選択します。

- 新しい ID を保有するデータベースを更新します。  
トランスフォーメーションが、インデックスデータのシーケンス識別子に重複が生じないすべての行をインデックスデータに追加します。トランスフォーメーションがインデックスの現在の行を更新することはありません。

デフォルトでは、このオプションを選択した場合、トランスフォーメーションが照合分析を実施します。照合プロセスのオプションを使用して、照合分析を有効または無効にできます。

- データベースを更新しません。  
トランスフォーメーションは、マッピングデータソースからのインデックスデータを使用してインデックステーブルを更新しません。

このオプションを選択した場合、トランスフォーメーションが照合分析を実行します。

- データベースから ID を削除する  
行がマッピングソースデータとシーケンス識別子を共有するときは、トランスフォーメーションがインデックステーブルから行を削除します。

このオプションを選択した場合は、トランスフォーメーションは照合分析を実行しません。

- データベースの現在の ID を更新する  
行がシーケンス識別子を共有するときは、トランスフォーメーションがインデックステーブルの行をマッピングソースデータの行に置換します。トランスフォーメーションは行をインデックスに追加しません。

デフォルトでは、このオプションを選択した場合、トランスフォーメーションが照合分析を実施します。照合プロセスのオプションを使用して、照合分析を有効または無効にできます。

デフォルトの永続方法は、**[データベースを新しい ID で更新する]** です。

### 照合プロセス

現在のトランスフォーメーションが ID 分析を実行するかどうかを決定します。

[永続方法] プロパティで選択したオプションによって [照合プロセス] プロパティのオプションが決定されます。

### DB 接続

インデックステーブルが入ったデータベースを特定します。

### 永続ストア

指定したデータベース内でインデックステーブルを特定します。

## 関連項目：

- [「永続インデックスのケーススタディ」 \(ページ 491\)](#)
- [「永続方法パラメータ」 \(ページ 485\)](#)

## インデックスディレクトリとキャッシュディレクトリのプロパティ

インデックスディレクトリとキャッシュディレクトリには、一致トランスフォーメーションが ID 分析中に生成する一時データが格納されます。

インデックスデータをデータベーステーブルに書き込むようにトランスフォーメーションを設定しない場合、データ統合サービスはインデックスディレクトリにデータを書き込みます。ID 分析がキャッシュサイズに指定されている容量を超える量のメモリを必要とする場合、データ統合サービスはキャッシュディレクトリにデータを書き込みます。デフォルトでは、[照合タイプ] ビューのプロパティは空白です。インデックスディレクトリまたはキャッシュディレクトリを指定しない場合、データ統合サービスはコンテンツ管理サービスからディレクトリパスを読み取ります。

[照合タイプ] ビューでインデックスディレクトリまたはキャッシュディレクトリを指定する場合は、データ統合サービスマシンのローカルパスを入力します。完全修飾パスまたは相対パスを入力できます。相対パスを入力する場合は、パスをピリオドで始めます。相対パスは、データ統合サービスマシンの tomcat/bin ディレクトリに対して相対的です。

以下の表に、キャッシュディレクトリまたはインデックスディレクトリのプロパティの相対パスおよびそのプロパティが表す完全修飾パスを示します。

| 相対パス | 完全修飾パス                                             |
|------|----------------------------------------------------|
| ./ch | [Informatica_installation_directory]/tomcat/bin/ch |

インデックスディレクトリとキャッシュディレクトリのプロパティが同じディレクトリを指定するように設定することもできます。それぞれのプロパティに同じディレクトリを指定すると、データ統合サービスは指定されたディレクトリ内に両方のディレクトリを作成します。データ統合サービスは、インデックスデータ用に index ディレクトリを作成し、キャッシュデータ用に cache ディレクトリを作成します。それぞれのプロパティに異なるディレクトリを指定すると、データ統合サービスは指定されたディレクトリにデータを書き込みます。

データ統合サービスは、マッピングを実行した後にディレクトリからインデックスファイルとキャッシュファイルを削除します。データ統合サービスはディレクトリを削除しません。一致トランスフォーメーションがディレクトリを識別する場合、データ統合サービスは他のマッピングでディレクトリを再利用できます。

### コンテンツ管理サービスのプロパティ

コンテンツ管理サービスは、インデックスディレクトリとキャッシュディレクトリのために以下のデフォルトの場所を指定します。

- ./identityIndex。ID インデックスデータのデフォルトディレクトリ。
- ./identityCache。ID キャッシュデータのデフォルトディレクトリ。

一致トランスフォーメーションに対してプロパティを設定しない場合、データ統合サービスは ID 照合マッピングが実行される際にデフォルトのディレクトリを作成します。データ統合サービスは、tomcat/bin ディレクトリ内にディレクトリを作成します。



## 永続方法パラメータ

ID 照合分析で永続データインデックスを選択すると、パラメータを使用して永続方法を特定できます。文字列を使用してパラメータ値を定義します。

次の表に、**【一致タイプ】** タブで選択可能な永続方法とその方法で設定可能なパラメータ値のリストを示します。

| 永続方法                     | パラメータ   |
|--------------------------|---------|
| データベースを新しい ID で更新します     | 無視      |
| データベースを更新しないでください        | addNone |
| データベースからの ID の削除         | 削除      |
| データベース内の現在の ID を更新してください | 更新      |

パラメータ値は大文字と小文字が区別されます。

### 関連項目：

- [「ID 一致タイプのプロパティ」 \(ページ 481\)](#)

## ID 照合ストラテジ

「ストラテジ」ビューには、ID データに対して定義したストラテジが一覧表示されます。ストラテジによって、ID インデックス内のデータ間の類似性と差異をトランスフォーメーションがどのように測定するかが決まります。

## ID 照合のアルゴリズム

一致トランスフォーメーションには定義済み ID アルゴリズムが含まれており、これを使って ID インデックスの中のデータ値を比較しています。データセット内の ID データのタイプに最も適したアルゴリズムを選択します。

各アルゴリズムの内容と、それぞれで選択する入力を次の表に示します。

| ID アルゴリズム | 説明                                                                                                                                                 |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 住所        | 住所を共有するレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- 住所<br>このアルゴリズムにセカンダリ入力はありません。                                                                 |
| 担当者       | 1 つの組織の場所にいる担当者を共有するレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>- Organization_Name<br>- Address_Part1<br>このアルゴリズムにセカンダリ入力はありません。 |

| ID アルゴリズム | 説明                                                                                                                                                              |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 法人        | 法人 ID データが同じレコードを特定します。住所と電話番号のデータを分析する必要がある場合は、このアルゴリズムを選択します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Organization_Name<br>このアルゴリズムにセカンダリ入力は必要ありません。                |
| 除算        | 社内で事務所の場所が同じレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Organization_Name<br>- Address_Part1<br>このアルゴリズムにセカンダリ入力は必要ありません。                                     |
| 家族        | 同じ家族に属する個人を特定します。名前、住所、電話番号のデータを分析します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>このアルゴリズムには次のセカンダリ入力のいずれかが必要です。<br>- Address_Part1<br>- Telephone_Number |
| フィールド     | 選択したポートに同じデータがあるレコードを特定します。<br>このアルゴリズムは必須入力を指定しません。重複した ID データを含む可能性のあるポートを 1 つまたは複数選択します。                                                                     |
| 世帯        | 同じ世帯に属する個人を特定します。名前データと住所データを分析します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>- Address_Part1                                                            |
| 個人        | 重複している個人を特定します。名前、誕生日、身分証明書のデータ（社会保障番号、口座番号、車両登録番号など）を分析します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>このアルゴリズムには次のセカンダリ入力のいずれかが必要です。<br>- Date<br>- ID  |
| 組織        | 組織データを共有するレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Organization_Name<br>このアルゴリズムにセカンダリ入力は必要ありません。                                                          |
| 個人名       | 個人についての情報が同じレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>このアルゴリズムにセカンダリ入力は必要ありません。                                                              |

| ID アルゴリズム | 説明                                                                                                                                                   |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 住居        | ある住所の重複する個人を特定します。必要があれば、このストラテジを個人特定データを分析するように設定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>- Address_Part1<br>このアルゴリズムにセカンダリ入力は必要ありません。 |
| 担当者（広域）   | ある組織の担当者を共有するレコードを特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Person_Name<br>- Organization_Name<br>このアルゴリズムにセカンダリ入力は必要ありません。                           |
| 世帯（広域）    | 同じ世帯に属する個人を特定します。<br>このアルゴリズムには次のプライマリ入力が必要です。<br>- Address_Part1<br>このアルゴリズムにセカンダリ入力は必要ありません。                                                       |

## ID 照合ストラテジのプロパティ

各 ID ストラテジのプロパティを設定します。

ID ストラテジの設定時には、以下のストラテジプロパティを設定できます。

### ポピュレーション

ID 分析に適用するポピュレーションを決定します。ポピュレーションには、特定のロケールと言語のキー構築アルゴリズムが含まれます。

### 照合レベル

検索品質と検索速度のバランスを決定します。検査速度は、返される一致数に反比例します。[ルーズ] 設定が使用される検索が返す一致は少ないのに対して、[保守的] 設定が使用される検索が返す一致は多くなります。

## ID 照合の出力オプション

[照合出力] ビューには、出力データの形式を指定するオプションがあります。レコードをクラスタ内または一致ペア内に書き込むようにトランスフォーメーションを設定することができます。また、永続インデックスデータストアに対して ID 照合分析を実行するときに異なる ID カテゴリを含めるまたは除外するように、このトランスフォーメーションを設定することもできます。

[照合出力のタイプ] 領域と [プロパティ] 領域のオプションを設定します。

### 照合出力のタイプ

[照合出力] ビューには、出力データの形式を指定するオプションがあります。レコードをクラスタ内または一致ペア内に書き込むようにトランスフォーメーションを設定することができます。

以下のいずれかの照合出力のタイプを選択します。

## 最良の一致

マスタデータセット内の各レコードを、第2データセット内でマッチ率が最高のレコードを使って書き込みます。照合操作では、第2データセットでマスタレコードとのマッチ率が最も高いレコードが選択されます。2つ以上のレコードが最高のマッチ率を返す場合、照合操作で第2データセット内の最初のレコードが選択されます。最良の一致では、レコードの各ペアを単一の行に書き込みます。

トランスフォーメーションをデュアルソースでの分析に設定する場合に【**最良の一致**】を選択することができます。

## クラスタ - 最良の一致

同一データセット内または2つのデータセット間で、レコード間で最良の一致を示すクラスタを書き込みます。2つのレコード間のマッチ率は、一致しきい値を満足している必要があります。1つのレコードが複数のレコードとの最良の一致を表す場合、もっとも一致するクラスタには3つ以上のレコードを含めることができます。

任意のタイプの ID 分析で、【**クラスタ - なるべく多くに一致**】を選択できます。

**注:** 選択するインデックスデータのストレージ方法は、【**クラスタ - 最良の一致**】モードのクラスタ出力のコンテンツに影響することがあります。テーブルにインデックスを付けるために接続するトランスフォーメーションは、一時ファイルの同じレコードにインデックスデータを保存するトランスフォーメーションと異なるクラスタを作成できます。インデックスデータのストレージ方法は、トランスフォーメーションがレコードペアに対して生成するマッチ率には影響しません。

## クラスタ - すべてに一致

一致しきい値を満足するマッチ率を示すレコードのクラスタを書き込みます。各レコードは、クラスタ内の他の1つ以上のレコードと一致する必要があります。

任意のタイプの ID 分析で、【**クラスタ - すべてに一致**】を選択できます。

## 一致ペア

一致しきい値を満足するマッチ率で相互に一致するレコードの全ペアを書き込みます。トランスフォーメーションは各ペアを1行に書き込み、各ペアのマッチ率を各行に追加します。レコードが他の複数のレコードと一致する場合、トランスフォーメーションはレコードペアごとに1行を書き込みます。

任意のタイプの ID 分析で、【**一致ペア**】を選択できます。

# 照合出力のプロパティ

【照合出力】ビューには、キャッシュメモリの動作とマッチ率のしきい値を指定するプロパティが含まれます。これらのプロパティを使用すると、トランスフォーメーションが分析のためにデータストアのレコードを選択する方法およびデータストアのレコードを出力として書き込む方法を決定することもできます。

照合出力タイプの選択後に、以下のプロパティを設定します。

## キャッシュディレクトリ

データ統合サービスが ID 照合分析中に一時データを書き込むディレクトリを指定します。照合分析が生成するデータ量が利用可能なシステムメモリ量より大きい場合、データ統合サービスは一時ファイルをディレクトリに書き込みます。データ統合サービスは、マッピングを実行した後に一時ファイルを削除します。

ディレクトリパスは、プロパティに入力するか、またはシステムパラメータを使用してディレクトリを指定できます。データ統合サービスのマシン上のローカルパスを指定します。データ統合サービスは、このディレクトリへの書き込みができる必要があります。デフォルト値は、CacheDir システムパラメータです。

## キャッシュサイズ

データ統合サービスが ID 照合分析に割り当てるシステムメモリの量を決定します。デフォルト値は 400,000 バイトです。

照合分析によって、これを超える量のデータが生成される場合、データ統合サービスは残りのデータをキャッシュディレクトリに書き込みます。照合分析にシステムメモリとファイルストレージが提供できる量を超えるメモリが必要な場合、マッピングは失敗します。

**注:** 65536 またはそれより高い値を入力した場合、トランスフォーメーションは値をバイト単位で読み取ります。それより低い値を入力すると、トランスフォーメーションは値をメガバイト単位で読み取ります。

## 一致

トランスフォーメーションがインデックスデータをデータベーステーブルから読み込むときに、分析対象のレコードを特定します。**[照合タイプ]** ビューのオプションを使用してインデックステーブルを識別します。

デフォルトでは、トランスフォーメーションはデータソースとインデックスデータベーステーブルのすべてのレコードを分析します。照合プロパティを設定し、重複分析のレコードのサブセットを指定します。

## 出力

インデックスデータベーステーブルを読み込むようにトランスフォーメーションを設定した場合に、トランスフォーメーションが出力として書き込むレコードをフィルタします。**[照合タイプ]** ビューのオプションを使用してインデックステーブルを識別します。

デフォルトでは、一致トランスフォーメーションがデータソースとインデックスデータベーステーブルからのすべてのレコードを出力として書き込みます。入力データのすべてのレコードを確認する必要がない場合は、**出力**プロパティを設定します。

## しきい値

2つのレコードを互いの重複候補とみなす最小マッチ率を設定します。

パラメータをしきい値に割り当てることができます。0 から 1 の範囲で 10 進値を設定します。

## 照合プロパティの設定

**[照合出力]** ビューの **[照合]** プロパティを使用することで、分析用の入力データをトランスフォーメーションがどのように選択するかを指定することができます。インデックスデータの永続ストアを読み取る一致トランスフォーメーションを設定するときにプロパティを設定します。**[照合]** プロパティにより、**[照合タイプ]** ビューで設定するオプションが絞り込まれます。

**[照合]** プロパティを設定することで、次のタイプの分析を実行することができます。

### データソースのレコードをインデックスデータのレコードと比較する

データソースとインデックスデータテーブルの間の重複レコードを検出する場合に、**[占有]** を選択します。

**[占有]** オプションを選択すると、一致トランスフォーメーションはデータソースのレコードをインデックスデータストアと比較します。トランスフォーメーションはデータソース内の重複レコードもデータストア内の重複レコードも分析しません。

インデックスデータストアに重複レコードがなく、データソースにも重複レコードがないことが分っている場合は、**[占有]** を選択します。

### データソースのレコードをインデックスデータのレコードと比較し、さらにデータソースのレコード同士を比較する

データソース内の重複だけでなく、データソースとインデックステーブル間の重複も検出する場合は、**[部分]** を選択します。

トランスフォーメーションがデータソースのレコードをインデックスのデータストアと比較します。また、トランスフォーメーションはデータソース内のレコードの相互比較も行います。

インデックスデータストア内に重複レコードがないことは分かっているが、データソースに対して重複分析を行っていない場合は、**[部分]** を選択します。

### データソース内のすべてのレコードとインデックステーブルを単一のデータセットとして比較する

データソースとインデックステーブル間の重複を検出する場合、およびデータソース内部の重複とインデックステーブル内部の重複を検出する場合は、**【完全】**を選択します。デフォルトのオプションは**【完全】**です。

トランスフォーメーションは、データソースとデータストアを単一のデータセットとして分析し、データセット内のすべてのレコードのデータを比較します。

どちらのデータセットにも重複レコードが1つもないことが確認できない場合は、**【完全】**を選択します。

## 出力プロパティの設定

**【照合出力】** ビューの**【出力】** プロパティを使用すると、トランスフォーメーションが出力として書き込むレコードをフィルタすることができます。このプロパティは、インデックスデータテーブルを指定しクラスタ化された出力形式を選択するときに設定します。レコードをフィルタリングして、データソースからの1つまたは複数のレコードが入ったクラスタに出力を制限します。

出力データは次の方法でフィルタリングすることができます。

### データソースまたはインデックステーブルからのレコードが入ったクラスタをすべて書き込みます。

**【すべての行】** を選択します。トランスフォーメーションは、データソースまたはインデックスデータストアのいずれかからのレコードを少なくとも1つ含むクラスタをすべて書き込みます。デフォルトは**【すべての行】** です。

クラスタには単一レコードを入れることができるため、この出力にはすべてのレコードが入ります。

### データソースからのレコードが入っているクラスタをすべて書き込む

**【新規および関連行】** を選択します。トランスフォーメーションは、データソースからのレコードが少なくとも1つ入っているクラスタをすべて書き込みます。

クラスタに単一のレコードが入っている可能性があるため、出力にはデータソースのすべてのレコードが入っています。クラスタには、インデックステーブルからのレコードを含めることもできます。

### データソースからのクラスタをすべて書き込む

**【新規行のみ】** を選択します。トランスフォーメーションは、データソースからのレコードを含むクラスタを書き込みます。出力にはインデックステーブルからのレコードが入っていません。

# ID 照合の詳細プロパティ

トランスフォーメーションには、実行インスタンスの数、トランスフォーメーションが同じ行を分析するかどうか、およびログデータのトレースレベルを決定する詳細プロパティが含まれます。

以下の詳細プロパティを設定できます。

### 実行インスタンス数

データ統合サービスが、現在のトランスフォーメーションで実行時に作成するスレッドの数を指定します。データ統合サービスは、トランスフォーメーションを含むマッピングの**【最大並行処理】** ランタイムプロパティをオーバーライドする場合に、この実行インスタンス数の値を考慮します。デフォルトの実行インスタンス数の値は**【自動】** です。

データ統合サービスでは、複数の要因を考慮して、トランスフォーメーションに割り当てるスレッド数を決定します。主要な要因は、実行インスタンス数の値と、マッピングおよびドメイン内の関連アプリケーションサービスの各値です。

マッピングを実行するデータ統合サービスは、以下の各値を読み取って、トランスフォーメーションに使用するスレッドの数を決定します。

- データ統合サービスの [最大並行処理] 値。デフォルトは 1 です。
- マッピングレベルで設定された [最大並行処理] 値。デフォルトは [自動] です。
- トランスフォーメーションの [実行インスタンス数] 値。デフォルトは [自動] です。

マッピングレベルの [最大並行処理] 値をオーバーライドする場合、データ統合サービスは、上記の各プロパティの最低値を使用してスレッド数を決定します。

マッピングレベルでデフォルトの [最大並行処理] 値を使用している場合、データ統合サービスは実行インスタンス数の値を無視します。

実行インスタンスの数を設定する際は、以下のルールとガイドラインを考慮します。

- 複数のユーザーがデータ統合サービスに同時にマッピングを実行することがあります。正しいスレッド数を計算するには、データ統合サービスがアクセスできる CPU 数を同時実行マッピング数で除算します。
- デフォルトの [実行インスタンス数] 値およびデフォルトの [最大並行処理] 値を使用する場合、トランスフォーメーション操作はパーティション化できません。

### 完全一致のフィルタ

トランスフォーメーションが、照合ストラテジで入力データの同じレコードのペアに比較アルゴリズムを適用するかどうかを決定します。

トランスフォーメーションが同じレコードのペアを検出すると、アルゴリズムはレコード間の類似性のレベルを分析する必要がありません。トランスフォーメーションは、追加の分析を行わずにレコードを出力ステージに直接渡すことができます。同じレコードを出力ステージに直接渡すようにトランスフォーメーションを設定するには、[完全一致のフィルタ] を選択します。入力データに多くの同じ行が含まれる場合、比較アルゴリズムが実行する計算が少なくなり、マッピングが高速になります。

入力データに多くの同じ行が含まれる場合は、このオプションを選択します。入力データに多くの同じ行が含まれない場合は、トランスフォーメーションの実行速度が低下することがあるので、このオプションを選択しないでください。

**注:** オプションを選択するか、または選択を取り消しても、トランスフォーメーションの出力には同じレコードデータが含まれます。オプションを選択するか、または選択を取り消すと、トランスフォーメーションは異なるリンクスコアとドライバスコアを出力レコードに割り当てることがあります。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## 永続インデックスのケーススタディ

数か所に支店のある小売銀行のデータスチュワードとして、全支店の顧客口座レコードのマスターセットを管理しているとします。一連のインデックスデータベーステーブルを使用して、顧客口座データベースに冗長レコードや重複レコードがないことを確認します。

インデックスデータストアを作成して管理するために、次の操作を実行します。

- データストアを作成する。
- 各支店から送信された最新データに基づいてデータストアを更新する。

必要に応じて、口座データのデータストアへの追加や、データストアの現在のデータの更新を行う。



- 古くなったレコードをデータストアから削除する。

上記の各操作によってデータストアに重複レコードが作成される可能性があることがわかっています。そこで、データをマスターデータストアに追加する前に支店データを分析するポリシーを作成することにします。ID 照合分析を使用して、支店データを分析し、そのデータによってデータストアに重複 ID が作成されないことを確認します。支店データとデータストアを分析する一致トランスフォーメーションに永続インデックスオプションを設定します。

### 永続インデックスデータ管理のポリシーの作成

データスチュワードとして、顧客口座データストア内の重複 ID を禁じるビジネスルールを定義します。データをデータストアに追加する前にステージングデータベースの支店データを分析する ID 照合マッピングを設計します。

次に該当する場合は、支店データをデータストアに追加する操作で重複 ID が作成される可能性があります。

- 支店データに重複 ID が含まれる場合
- 支店データに、インデックスに既存の ID がある場合
- 支店データにデータストアにある ID の新しいバージョンが含まれ、その新しいバージョンがインデックス内の別の ID と一致する場合

ステージングデータベースをデータストアと比較するときに、支店データの重複レコードステータスを反映する永続インデックスオプションを選択します。データストアを更新する前に、支店データをインデックスデータと比較することもできます。

**注:** 一定のオプションに対する照合分析を有効または無効にできます。照合分析を有効にして、マッピングデータの分析や、インデックスデータストアとマッピングデータの比較などを行います。データを比較する必要がある場合は照合分析を無効にします。また、[照合出力] タブの [照合] プロパティを使用して、照合分析のデータを包含または除外できます。

### マッピングデータソースとインデックスデータストアの比較

マッピング入力データをインデックスデータストアと比較し、データストアに変更を行わない場合は、次のオプションを選択します。

- データベースを更新しない

マッピングで入力データをインデックスデータストアと比較します。マッピングではインデックスデータストアのデータが追加、削除、更新されません。

このオプションを選択したときは、ID 照合分析を無効にできません。

インデックスデータを更新しないため、ストアに重複する行を作成することはできません。[照合出力] タブの [照合] プロパティから、データプロジェクトの現在のニーズに応じたオプションを選択します。例えば、**[完全]** オプションを選択します。**[完全]** オプションは、マッピングデータに重複がないことと、マッピングデータによってデータストアに重複が追加されないことを確認します。

**注:** データストアを更新する前にマッピングデータとデータストアを比較する場合は、このオプションを使用します。マッピング出力でマッピングデータによってデータストアに重複が追加されないことが示されたら、マッピングを再度実行します。マッピングを再度実行するときに、データベースを更新するオプションを選択します。

### データストアの作成およびデータストアへの行の追加

データストアを作成するか、またはマッピングデータの行をデータストアに追加するには、次のオプションを選択します。

- データベースを新しい ID で更新する

行がデータストア内の行とシーケンス識別子を共有していない場合は、マッピングでデータストアに行が追加されます。マッピングでインデックステーブルの行が上書きされることはありません。空のデータベーステーブルを指定すると、マッピングですべてのマッピングインデックスデータがテーブルに書き込まれます。



このオプションを選択した場合は、ID 照合分析を有効または無効にできます。デフォルトで、このオプションは照合分析を有効にします。

インデックスの行を更新しないため、[照合出力] タブの [照合] プロパティから [占有] オプションまたは [部分的] オプションを選択します。事前のプロセスでマッピングデータの行の一意性を確認している場合は [占有] オプションを使用します。

### データストアの行の更新

マッピングデータに基づいてデータストアの現在の行を更新するには、次のオプションを選択します。

- データベースの現在の ID を更新する

レコードがマッピングデータ内のレコードとシーケンス識別子を共有している場合は、マッピングでデータストアの現在のレコードが更新されます。マッピングで行がインデックステーブルに追加されることはありません。

このオプションを選択した場合は、ID 照合分析を有効または無効にできます。デフォルトで、このオプションは照合分析を無効にします。

インデックスの行をインデックステーブルに追加しないため、[照合出力] タブの [照合] プロパティから [完全] オプションを選択します。

**注:** データストアの行を更新するときに、マッピングのソースデータとデータストア間に重複が見つかることがあります。ストアに追加する ID データがストアの現在のデータと一致していないことを確認するには、[完全] オプションを選択します。

### データストアから行の削除

データストアから行を削除するには、次のオプションを選択します。

- データベースから ID を削除する

行がマッピングデータ内のレコードとシーケンス識別子を共有している場合は、マッピングでデータストアから行が削除されます。

このオプションを選択した場合は、ID 照合分析を有効または無効にできます。デフォルトで、このオプションは照合分析を無効にします。

**注:** データストアからデータを削除するときは、ストア内の行間のリレーションを変更します。ストアに重複 ID がある場合、クラスタ内のドライバレコードまたはリンクされたレコードのデータを削除できます。あるいは、一致するペアの最良の一致のデータを削除することもできます。マッピングを再度実行するときに、異なるクラスタまたは重複するペアが生成されることがあります。データストアから重複レコードのない行を削除する場合は、レコードの重複ステータスを変更できません。行を削除した後にマッピングを実行すると、データセットに残っている ID に対して同じマッチ率が生成されます。

## ID 照合分析の例

各都市に開発拠点のあるソフトウェア組織の人事責任者であるとします。同社では従業員の人事記録を本社のデータベースに保存しています。開発拠点では定期的に従業員を採用して、採用した従業員の人事データを本社に送信します。

本社の人事責任者は人事記録をスプレッドシートファイルに追加して、そのファイルのデータを基に従業員データベースを更新します。現在のファイルに重複する ID が含まれているのではないかと懸念があります。

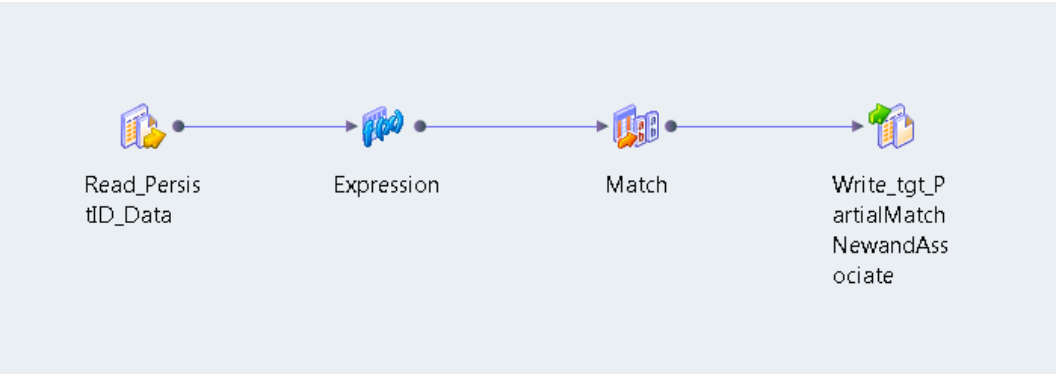
そこで、従業員記録に対して ID 照合分析を実行するマッピングを設計することにします。スプレッドシートファイルの重複 ID を検索する一致トランスフォーメーションを設定します。また、ファイルのデータを追加する際にマスターデータベースの従業員データに重複が生じないことも確認する必要があります。ファイルのデータを組織の従業員記録を保存するマスターデータと比較する一致トランスフォーメーションを設定します。

このデータベースはマスターデータセットのため、従業員記録のインデックスデータを永続データストアに保存します。

## マッピングの作成

重複した ID を検出するマッピングを作成します。このマッピングでは、データソースを読み取り、シーケンス ID をソースレコードに追加し、ID 照合分析を実行し、結果をデータターゲットに書き込みます。

次の図に、Developer ツールのマッピングを示します。



作成するマッピングには、以下のオブジェクトが含まれます。

| オブジェクト名                               | 説明                                          |
|---------------------------------------|---------------------------------------------|
| Read_PersistID_Data                   | データソース。<br>社員の名前と詳細が入ります。                   |
| 式                                     | 式トランスフォーメーション。<br>シーケンス ID 値をソースデータに追加します。  |
| 一致                                    | 一致トランスフォーメーション。<br>ソースデータの ID の重複レベルを分析します。 |
| Write_tgt_PartialMatchNewandAssociate | データターゲット。<br>ID 照合分析の結果が入ります。               |

**注:** このマッピングはキージェネレータトランスフォーメーションを使用しません。ID 照合分析では、キージェネレータトランスフォーメーションは任意です。

## 入力データサンプル

この職員ファイルには、社員の名前、社員の働く町の名前、社員の役職が入っています。モデルリポジトリで職員ファイルからデータソースを作成します。データソースをマッピングに追加します。

次のデータの抜粋は、職員ファイル内の社員データの例です。

| 名前       | 市区町村     | 役職   |
|----------|----------|------|
| Chaithra | バンガロール   | SE   |
| Ramanan  | チェンナイ    | SSE  |
| Ramesh   | チェンナイ    | SSE  |
| Ramesh   | チェンナイ    | リーダー |
| Sunil    | バンガロール   | 主任   |
| Venu     | ハイデラーバード | 主任   |
| Harish   | バンガロール   | SE   |
| Sachin   | バンガロール   | SSE  |

## 式トランスフォーメーションの設定

式トランスフォーメーションを設定するときは、マッピング出力に含める必要のあるすべてのデータソースポートを接続します。これらのポートは、パススルーポートとして接続します。ポートにシーケンス ID 値を追加する式を作成します。

次の式で、変数 *Init3* が作成され、整数値 1267 が各シーケンス ID に追加されます。

*Init3+1267*

次の表で、社員のデータソースを読み取る式トランスフォーメーションのポートについて説明します。

| 名前          | ポートタイプ  | ポートグループ |
|-------------|---------|---------|
| SEQID       | bigint  | 出力のみ    |
| Name        | string  | パススルー   |
| City        | string  | パススルー   |
| Designation | string  | パススルー   |
| Init3       | integer | 変数      |

## 一致トランスフォーメーションの設定

再利用不可能な一致トランスフォーメーションをマッピングに追加して、ID 照合分析を実行します。

以下のタスクを実行して、トランスフォーメーションを設定してください。

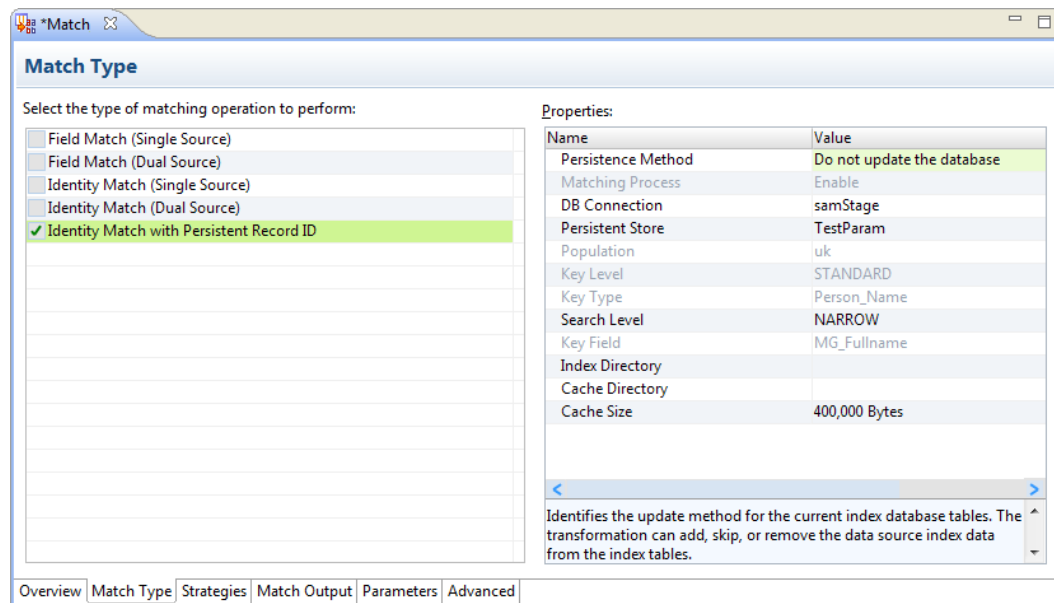
1. 実行する照合分析のタイプを選択する。

2. 入力ポートをトランスフォーメーションに接続する。
3. レコードデータを比較するようにストラテジを設定する。
4. トランスフォーメーションで作成する照合出力データのタイプを選択する。
5. 出力ポートをデータターゲットに接続する。

## 照合操作のタイプの選択

[一致タイプ] ビューのオプションを使用して照合操作を選択することができます。

次の図に [一致タイプ] ビューを示します。



データソースからのインデックスデータをマスターデータセットからのインデックスデータと比較するには、**[永続的レコード ID での ID 照合]** を選択します。照合分析によってインデックステーブルにデータが追加されないように永続方法を更新します。マッピングの結果を確認後、インデックステーブルを更新するかどうかを決定できます。

**[DB 接続]** オプションを使用して、インデックスデータを含むデータベースを特定します。**[永続ストア]** オプションを使用してインデックステーブルを選択します。

**注:** 一致トランスフォーメーションが、インデックスデータベーステーブルのメタデータから ID ポピュレーション、キーレベル、キータイプ、キーフィールドのプロパティ値を読み取ります。これらの値は、インデックスデータストアを作成したトランスフォーメーションの対応するプロパティと一致します。

## 入力ポートの接続

入力データポートをトランスフォーメーションに接続します。ポート名、ポートの順序、データ型、および精度が、データストアを作成したトランスフォーメーションのポート設定に一致していることを確認します。

一致トランスフォーメーションは、設定済み入力ポートを使用してレコードを処理する順序を決めます。このトランスフォーメーションでは、シーケンス識別子を使用して入力ポートからのレコードを、出力として書き込む一致ペアまたはクラスタまで追跡します。このトランスフォーメーションではグループキーを使用して、処理するレコードを並び替えます。

式トランスフォーメーションの以下のポートに、設定済みポートを接続します。

- SequenceID。式トランスフォーメーションから SEQID ポートに接続します。

- GroupKey。式トランスフォーメーションから City ポートに接続します。

## ID 照合分析のストラテジの設定

[ストラテジ] ビューのオプションを使用してストラテジを設定します。トランスフォーメーションがレコードデータに対して実行する分析のタイプは、ストラテジによって決まります。

レコードデータに対して Person\_Name というアルゴリズムを選択します。Name という入力ポートを分析対象に選択します。このトランスフォーメーションではポートデータのコピーを作成するため、ポートには Name\_1 と Name\_2 を選択してください。

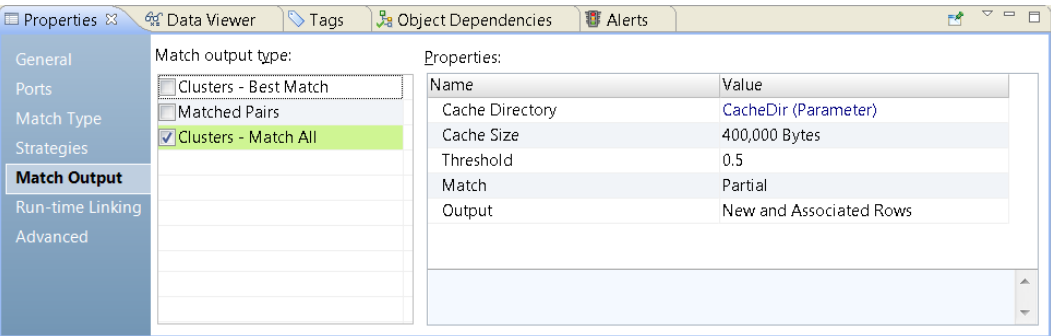
### 関連項目：

- [「ID 照合のアルゴリズム」 \(ページ 485\)](#)

## 照合出力のタイプの選択

[照合出力] ビューのオプションを使用して、照合分析の結果を出力するときの形式を定義します。

次の図に、単一ソースでの ID 照合分析の場合の [照合出力] ビューを示します。



このトランスフォーメーションを、クラスタ内の出力レコードを整理するように設定します。各クラスタには、指定した照合プロパティに基づいた分析で少なくとも他の 1 つのレコードと一致するレコードがすべて入っています。??トランスフォーメーションがデータソースのレコードをインデックスレコードと比較する方法は、照合プロパティによって決まります。

次の表で、ユーザーが職員記録のデータを分析するときに指定する照合プロパティのオプションについて説明します。

| 照合プロパティ | オプション    | オプションの説明                                                                                                                                                      |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 一致      | 部分一致     | トランスフォーメーションがデータソースのレコードをインデックスのデータストアと比較します。また、トランスフォーメーションはデータソース内のレコードの相互比較も行います。                                                                          |
| 出力      | 新規および関連行 | トランスフォーメーションは、データソースからのレコードが少なくとも 1 つ入っているクラスタをすべて書き込みます。クラスタには、インデックスのデータストアからのレコードが入っている可能性があります。<br>クラスタに単一のレコードが入っている可能性があるため、出力にはデータソースのすべてのレコードが入っています。 |

## 関連項目：

- [「照合プロパティの設定」 \(ページ 489\)](#)
- [「出力プロパティの設定」 \(ページ 490\)](#)

## 出力ポートの接続

一致トランスフォーメーションの出力ポートをマッピングのデータターゲットに接続します。データターゲットに書き込むレコードデータが入っているポートを選択します。

このトランスフォーメーションには、クラスタ化されたデータに対する一連の設定済みポートが入っています。レコードの重複ステータスを示す設定済みポートを選択し、各レコードを格納しているデータソースを特定します。

以下のポートに、重複レコードの検出とソースまたはレコードの判定に使用できるデータが入っています。

- **ClusterSize** ポートは、クラスタ内のレコードの数を示しています。レコードがクラスタサイズが 1 より大きいクラスタに属する場合、トランスフォーメーションはこのレコードを他のレコードの重複であるとみまします。
- **ClusterID** ポートは、レコードが属しているクラスタを特定します。現在のレコードの重複であるレコードを、ClusterID データを使用して見つけます。
- **PersistenceStatus** ポートは、コード値を使用して、マッピングソースからのインデックスデータとデータストア内のインデックスデータ間のリレーションを説明します。
- **PersistenceStatusDesc** ポートは、PersistenceStatus ポートコードの値の説明をテキストで返します。

他のポートを使用してクラスタレコード間のリレーションを確認することもできます。リンクポートの値とドライバポートの値は、各クラスタ内のレコード間の類似度を示します。

この例では、すべてのポートをデータターゲットに接続します。ポートで出力データを表示するには、データビューアを実行します。

## 関連項目：

- [「持続ステータスコードと持続ステータス記述」 \(ページ 462\)](#)
- [「ステータスコード値とステータス記述値」 \(ページ 462\)](#)

## データビューアの実行

データビューアを実行すると、照合分析の結果が確認できます。デフォルトでは、一致トランスフォーメーションに対するすべての出力ポートがデータビューアに表示されます。マッピングを実行する場合は、データターゲットを出力ポートからのデータで更新します。

次の図に、データビューアに表示された出力データを示します。

The screenshot shows a workflow window titled '\*PartialMatchNewwandAssociated\_MatchAll'. The workflow consists of four steps: 'Read\_IncreP', 'Expression', 'Match', and 'Write\_tgtLP'. Below the workflow, the 'Data Viewer' tab is active, displaying a table of output data. The table has columns: Name, ClusterSize, PersistenceStatusDesc, Designation, GroupKey, LinkId, ClusterId, LinkScore, and PersistenceSt... (truncated). The table contains 16 rows of data, with the first row being a header and the subsequent 15 rows representing individual records. The 'Match' step is highlighted in blue.

|    | Name   | ClusterSize | PersistenceStatusDesc  | Designation | GroupKey  | LinkId    | ClusterId | LinkScore | PersistenceSt... |
|----|--------|-------------|------------------------|-------------|-----------|-----------|-----------|-----------|------------------|
| 1  | Chaith | 2           | Input, Exists, Ignored | SE          | Bangalore | S - 1     | 000000001 | 1         | IEI00000         |
| 2  | Chaith | 2           | Store, -, No change    | SE          | Bangalore | 1 - 1267  | 000000001 | 1         | SON00000         |
| 3  | Ramana | 2           | Input, Exists, Ignored | SSE         | Chennai   | S - 2     | 000000002 | 1         | IEI00000         |
| 4  | Ramana | 2           | Store, -, No change    | SSE         | Chennai   | 1 - 2534  | 000000002 | 1         | SON00000         |
| 5  | Ramesh | 4           | Input, Exists, Ignored | SSE         | Chennai   | S - 3     | 000000003 | 1         | IEI00000         |
| 6  | Ramesh | 4           | Input, Exists, Ignored | Lead        | Chennai   | S - 3     | 000000003 | 1         | IEI00000         |
| 7  | Ramesh | 4           | Store, -, No change    | SSE         | Chennai   | 1 - 3801  | 000000003 | 1         | SON00000         |
| 8  | Ramesh | 4           | Store, -, No change    | Lead        | Chennai   | 1 - 3801  | 000000003 | 1         | SON00000         |
| 9  | Sunil  | 2           | Input, Exists, Ignored | Principal   | Bangalore | S - 5     | 000000004 | 1         | IEI00000         |
| 10 | Sunil  | 2           | Store, -, No change    | Principal   | Bangalore | 1 - 6335  | 000000004 | 1         | SON00000         |
| 11 | Venu   | 2           | Input, Exists, Ignored | Principal   | Hydrabad  | S - 6     | 000000005 | 1         | IEI00000         |
| 12 | Venu   | 2           | Store, -, No change    | Principal   | Hydrabad  | 1 - 7602  | 000000005 | 1         | SON00000         |
| 13 | Harish | 2           | Input, Exists, Ignored | SE          | Bangalore | S - 7     | 000000006 | 1         | IEI00000         |
| 14 | Harish | 2           | Store, -, No change    | SE          | Bangalore | 1 - 8869  | 000000006 | 1         | SON00000         |
| 15 | Sachin | 2           | Input, Exists, Ignored | SSE         | Bangalore | S - 8     | 000000007 | 1         | IEI00000         |
| 16 | Sachin | 2           | Store, -, No change    | SSE         | Bangalore | 1 - 10136 | 000000007 | 1         | SON00000         |

Row 1 to 16

マスターデータセット内のデータと重複しているレコードがそのファイルに入っているかをデータビューアで確認します。

以下のデータ値をデータビューアで考慮してください。

- データセット内の各レコードが、2つ以上のレコードが入っているクラスタに属しています。したがって、各レコードが少なくとも他の1つのレコードと重複しています。このトランスフォーメーションで、一意のレコードにクラスタサイズ1が割り当てられます。マスターデータセットに入っていないレコードは、データソースにも入っていません。
- PersistenceStatusDesc** データは、そのレコードの起源を特定し、さらに一致トランスフォーメーションがレコードをインデックステーブルに追加するかどうかを示します。このカラムは、各入力レコードがマスターデータセット内に存在することを示します。トランスフォーメーションは、マスターデータインデックスにデータを追加しません。

## まとめ

照合分析の結果は、マスターデータセットに入っていないレコードは職員記録ファイルにも入っていないことを示しています。持続ステータスの説明は、マッピングがインデックステーブルをデータソースからのデータで更新しないことを示しています。職員記録ファイルを破棄します。

地域オフィスから別の更新が送られてきたら、もう 1 つファイルを作成してそれをマスターデータセットと比較することができます。マッピングとインデックステーブルは再利用できます。マスターデータセットのインデックスデータをデータベーステーブルに保存するため、インデックスデータを再生成する必要はありません。



## 第 31 章

# ノーマライザトランスフォーメーション

この章では、以下の項目について説明します。

- [ノーマライザトランスフォーメーションの概要, 501 ページ](#)
- [複数出現フィールド, 502 ページ](#)
- [複数出現レコード, 503 ページ](#)
- [入力階層の定義, 504 ページ](#)
- [ノーマライザトランスフォーメーションの出力グループとポート, 512 ページ](#)
- [出力グループのキーの生成, 515 ページ](#)
- [ノーマライザトランスフォーメーションの詳細プロパティ, 515 ページ](#)
- [ノーマライザトランスフォーメーションの作成, 516 ページ](#)
- [アップストリームのソースからのノーマライザトランスフォーメーションの作成, 517 ページ](#)
- [ノーマライザマッピングの例, 517 ページ](#)
- [非ネイティブ環境でのノーマライザトランスフォーメーション, 521 ページ](#)

## ノーマライザトランスフォーメーションの概要

ノーマライザトランスフォーメーションは、単一のソース行を複数のターゲット行に変換するアクティブなトランスフォーメーションです。ノーマライザトランスフォーメーションが複数出現データを含む行を受け取ると、複数出現データの出現ごとに行を返します。

ノーマライザトランスフォーメーションは、複数出現データを解析し、結果を個別の出力行に生成します。例えば、リレーショナルソース行に 4 つの四半期の売上が含まれることがあります。ノーマライザトランスフォーメーションは、売上の各出現に対して個別の出力行を生成します。

ノーマライザトランスフォーメーションを定義する場合は、ソースデータ構造を示す入力階層を設定します。必要な場合は、トランスフォーメーション入力階層にレコードを定義できます。レコードは、フィールドのグループのためのコンテナです。ソースデータにフィールドのグループが複数回出現する場合は、レコードを定義します。入力階層は、トランスフォーメーション出力グループを設定する方法を決定します。

ノーマライザトランスフォーメーションは、リレーショナルテーブルまたはフラットファイルソースからデータを変換します。

## 複数出現フィールド

ソースデータ内で 1 つのフィールドが複数回出現する場合は、入力行階層でこのフィールドを複数出現フィールドとして定義できます。ノーマライゼイトランスフォーメーションは、ソースにおける複数出現フィールドまたはフィールドのグループの各出現に対して個別の行を返すことができます。

次に示すように、ソース行には 4 つの四半期の店舗別売上が含まれる可能性があります。

| Store  | Sales(1) | Sales(2) | Sales(3) | Sales(4) |
|--------|----------|----------|----------|----------|
| Store1 | 100      | 300      | 500      | 700      |
| Store2 | 250      | 450      | 650      | 850      |

ノーマライゼイトランスフォーメーションを定義すると、4 つの売上フィールドを結合し、1 つの複数出現フィールドとしてまとめることができます。Qtr\_Sales のようなフィールド名を定義し、ソース内で 4 回発生するようにこのフィールドを設定します。

出力グループに店舗データと売上データが含まれる場合、ノーマライゼイトランスフォーメーションは、Store と Qtr\_Sales のそれぞれの組み合わせに対して行を返します。出力行には、出力行にある Qtr\_Sales のそれぞれのインスタンスを識別するインデックスが含まれます。

トランスフォーメーションは次の行を返します。

| Store  | Qtr_Sales | Qtr (GCID) |
|--------|-----------|------------|
| Store1 | 100       | 1          |
| Store1 | 300       | 2          |
| Store1 | 500       | 3          |
| Store1 | 700       | 4          |
| Store2 | 250       | 1          |
| Store2 | 450       | 2          |
| Store2 | 650       | 3          |
| Store2 | 850       | 4          |

出力グループに単独出現カラムと複数出現カラムが含まれる場合、ノーマライゼイトランスフォーメーションは各出力行に単独出現カラムの重複データを返します。例えば、Store1 と Store2 は Qtr\_Sales のそれぞれのインスタンスに対して繰り返されます。

ソース行には、複数レベルの複数出現データが含まれることがあります。ノーマライゼイトランスフォーメーションは、入力階層の定義方法に基づいて、各レベルで個別の行を返すように設定できます。

## 生成カラム ID

ノーマライゼイトランスフォーメーションは、複数出現フィールドのインスタンスごとに生成カラム ID (GCID) の出力ポートを返します。

生成カラム ID のポートは、複数出現データのインスタンス用のインデックスです。例えば、ソースレコード内にフィールドが 4 回出現する場合、Developer ツールは生成カラム ID ポートの中で、複数出現データのどのインスタンスが行内に出現したかに基づいて 1、2、3、または 4 の値を返します。

# 複数出現レコード

ノーマライズトランスフォーメーションのソースデータでは、複数出現レコードを定義できます。レコードは、フィールドのグループです。複数出現のソースフィールドのグループを定義する必要がある場合は、ノーマライズトランスフォーメーションでレコードを定義します。

## 複数出現レコードの例

次の Customer 行には、自宅住所と勤務先住所の顧客情報が含まれます。

```
CustomerID
FirstName
LastName
Home_Street
Home_City
Home_State
Home_Country
Business_Street
Business_City
Business_State
Business_Country
```

ノーマライズトランスフォーメーションを設定する場合は、顧客フィールドと複数出現住所レコードを含む入力構造を定義できます。住所レコードは、2 回出現します。ノーマライズトランスフォーメーションの出力グループを設定する場合は、Address レコードを CustomerID、FirstName、および LastName の各フィールドとは異なるターゲットに返すことができます。

次の例に、複数出現住所レコードを含む入力構造を示します。

```
CustomerID
FirstName
LastName
Address (occurs twice)
 Street
 City
 State
 Country
```

サブレコードは、レコード内のレコードです。レコードとサブレコードを定義する際は、ソース行にフィールドの入力階層を定義します。各レコードは、トランスフォーメーションの出力を定義する際に参照できる階層内のノードです。

例えば、ソース行に、各住所タイプに対する複数の電話番号が含まれることがあります。

```
CustomerID
FirstName
LastName
Home_Street
Home_City
Home_State
Home_Country
Telephone_No
Cell_Phone_No
Alternate_Phone_No
Business_Street
Business_City
Business_State
Business_Country
Business_Telephone_No
Business_Cell_Phone_No
Business-Alternate_Phone1
```

Address が Phone の親になっている入力階層を定義します。ノーマライズトランスフォーメーションの出力を定義する場合、住所と電話番号を顧客情報とは異なるターゲットに返すことができます。

次の例のようにして、入力階層を定義します。

```
CustomerID
FirstName
LastName
Address (occurs twice)
 Street
 City
 State
 Country
 Phone
 Telephone_No (occurs three times)
```

## 入力階層の定義

ノーマライザトランスフォーメーションを作成する際は、ソースのレコードとフィールドを記述する入力階層を定義します。入力階層は、トランスフォーメーションの【**ノーマライザ**】ビューで定義します。

Developer tool は、入力階層で定義されたフィールドに基づいてトランスフォーメーションの入力ポートを作成します。トランスフォーメーションの出力グループを定義する前に、入力グループの構造を定義します。

入力階層を定義する場合は、ソースデータの構造に対応する入力構造を定義する必要があります。ソースデータには、複数出現フィールドのグループが1つ以上含まれることがあります。構造を定義する場合は、ソースの別のレコードと同じレベルで出現するレコードを設定できます。または、他のレコード内で出現するレコードを定義できます。

### 入力階層の例

次のソース行には、顧客フィールドおよび2回出現する住所レコードが含まれます。

```
CustomerID
 FirstName
 LastName
 Address
 Street
 City
 State
 Country
 Address1
 Street1
 City1
 State1
 Country1
```

【**ノーマライザ**】ビューで入力構造を定義する場合は、フィールドとして CustomerID、FirstName、および LastName を追加できます。Address レコードを定義し、住所に Street、City、State、および Country の各フィールドを含めます。Address の【発生数】値を2に変更します。

次の図に、【**ノーマライザ**】ビューの入力階層を示します。

| Normalizer |       |        |        |           |       |  |
|------------|-------|--------|--------|-----------|-------|--|
| Name       | Level | Occurs | Type   | Precision | Scale |  |
| CustomerID | 1     | 1      | string | 10        | 0     |  |
| FirstName  | 1     | 1      | string | 10        | 0     |  |
| LastName   | 1     | 1      | string | 10        | 0     |  |
| Address    | 1     | 2      |        |           |       |  |
| Street     | 2     | 1      | string | 10        | 0     |  |
| City       | 2     | 1      | string | 10        | 0     |  |
| State      | 2     | 1      | string | 10        | 0     |  |
| Country    | 2     | 1      | string | 10        | 0     |  |

【**ノーマライザ**】ビューの【**発生数**】カラムには、ソース行におけるフィールドまたはレコードのインスタンスの数が示されます。複数出現フィールドまたはレコードの【**発生数**】カラムの値を変更します。この例では、顧客のフィールドは1回出現し、Address レコードは2回出現します。

【**ノーマライザ**】ビューの【**レベル**】カラムには、フィールドまたはレコードを入力階層のどこに表示するかを示します。顧客のフィールドは、階層のレベル1にあります。Address レコードもレベル1です。

## ノーマライザトランスフォーメーションの入力ポート

Developer ツールは、【**ノーマライザ**】ビューで入力階層を定義する際に、ノーマライザトランスフォーメーションの入力ポートを作成します。入力階層のフィールドを変更すると、Developer ツールは入力ポートを変更します。

【**概要**】ビューに、ノーマライザトランスフォーメーションの入力ポートを表示します。【**概要**】ビューでは、入力ポートの順序を変更できます。入力ポートを変更する場合は、【**ノーマライザ**】ビューで入力階層を更新します。

入力階層でフィールドを複数出現として定義すると、Developer ツールは複数出現フィールドの各インスタンスに対して1つの入力ポートを作成します。レコードが複数出現の場合、Developer ツールはレコードのフィールドの各インスタンスに対して入力ポートを作成します。

### 入力ポートの例

次の図に、Developer ツールが顧客データと複数出現アドレスデータに対して作成する入力ポートを示します。

| Ports |            |        |           |       |                 |
|-------|------------|--------|-----------|-------|-----------------|
|       | Name       | Type   | Precision | Scale | Location        |
|       | Input      |        |           |       |                 |
| 1     | CustomerID | string | 10        | 0     | CustomerID      |
| 2     | FirstName  | string | 10        | 0     | FirstName       |
| 3     | LastName   | string | 10        | 0     | LastName        |
| 4     | Street     | string | 10        | 0     | Address.Street  |
| 5     | Street1    | string | 10        | 0     | Address.Street  |
| 6     | City       | string | 10        | 0     | Address.City    |
| 7     | City1      | string | 10        | 0     | Address.City    |
| 8     | State      | string | 10        | 0     | Address.State   |
| 9     | State1     | string | 10        | 0     | Address.State   |
| 10    | Country    | string | 10        | 0     | Address.Country |
| 11    | Country1   | string | 10        | 0     | Address.Country |

## フィールドのマージ

類似するデータのフィールドは、[ノーマライザ] ビューで単一の複数出現フィールドにマージできます。別のオブジェクトからポートをドラッグして、マッピングでノーマライザトランスフォーメーションを作成する際には、フィールドのマージが必要になることがあります。

ソース行には、Base\_Salary、Bonus\_Pay、Sales\_Commissions などの異なるタイプの給与データを含む複数のフィールドが含まれる場合があります。フィールドをマージすると、3 回出現するフィールドから単一の Salary フィールドを作成できます。

次の図に、[ノーマライザ] ビューで 3 つのタイプの給与が選択されている従業員行を示します。

| Normalizer        |       |        |         |          |       |
|-------------------|-------|--------|---------|----------|-------|
| Name              | Level | Occurs | Type    | Preci... | Scale |
| EmployeeID        | 1     | 1      | string  | 10       | 0     |
| Base_Salary       | 1     | 1      | decimal | 10       | 0     |
| Bonus_Pay         | 1     | 1      | decimal | 10       | 0     |
| Sales Commissions | 1     | 1      | decimal | 10       | 0     |

3 つのタイプの給与データは、3 回出現する Salary フィールドにマージできます。

次の図に、Salary フィールドを示します。

| Normalizer |       |        |         |          |       |
|------------|-------|--------|---------|----------|-------|
| Name       | Level | Occurs | Type    | Preci... | Scale |
| EmployeeID | 1     | 1      | string  | 10       | 0     |
| Salary     | 1     | 3      | decimal | 10       | 0     |

## フィールドのマージ

[ノーマライザ] ビューで、同じタイプのフィールドを単一の複数出現フィールドにマージします。

1. [ノーマライザ] ビューをクリックします。
2. マージするフィールドを選択します。
3. [マージ] ボタンをクリックします。  
[フィールドのマージ] ダイアログボックスが表示されます。
4. マージフィールドの名前、複数出現フィールドのタイプ、精度、スケール、および出現数を入力します。
5. [OK] をクリックします。

## フィールドのフラット化

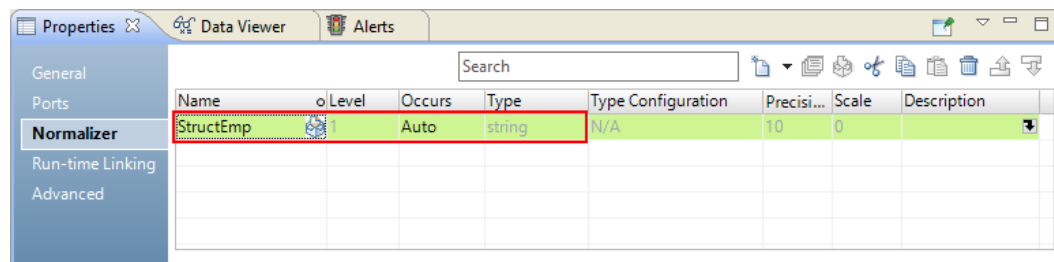
Spark エンジン上で実行するマッピングで複合データ型のフィールドをフラット化できます。[ノーマライザ] ビューでフィールドをフラット化し、複合ポートをパススルーする階層データを変更します。

フラット化アクションの出力は、複合データ型に依存します。array または struct データ型をフラット化すると、ノーマライザトランスフォーメーションは各要素の行を複合データ型で作成します。map データ型をフラット化すると、ノーマライザトランスフォーメーションはマップキー要素とマップ値要素の 2 つの列を作成します。

ネストされたデータ型に対するフラット化アクションは、第 1 レベルの要素を抽出します。すべてのレベルでネストされたデータ型をフラット化するには、Developer tool の [複合ポートのフラット化] 階層変換ウィザードを使用します。[すべてフラット化] オプションは、各レベルの要素を抽出し、プリミティブデータ型のリレーショナルデータを返します。階層変換ウィザードの詳細については、『*Data Engineering Integration ユーザーガイド*』を参照してください。

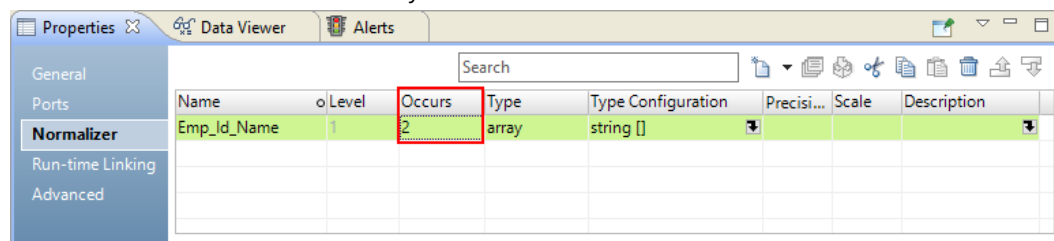
フラット化アクションによって、[ノーマライザ] ビューの [発生数] カラムの値が [自動] に変更され、フラット化されたフィールドの横にフラット化アイコンが追加されます。値 [自動] は、複合データ型のすべての要素がフラット化されることを示します。

次の図では、struct が string フィールドにフラット化され、その横にフラット化アイコンがあり、[発生数] 値が [自動] に設定されています。



複数出現フィールドはフラット化できません。例えば、[発生数] 値が 2 である array フィールドをフラット化できません。

次の画像は、フラット化できない array データ型の複数出現フィールドを示しています。



## 配列のフラット化

ノーマライズトランスフォーメーションは、1次元配列をプリミティブデータ型に、n次元配列を(n-1)次元配列にフラット化します。トランスフォーメーションが作成する行の数は、配列のサイズと同じです。

例えば、string 要素 10 個から成る配列ポートをフラット化すると、出力は 10 個の string ポートを返します。3次元配列をフラット化すると、出力は 2次元配列を返します。

次のテーブルには string ポート Name と array ポート Phones があります。array ポートをフラット化するとします。テーブルには以下の値があります。

| Name  | Phones                                     |
|-------|--------------------------------------------|
| Adams | [205-128-6478, 722-515-2889, 650-213-4020] |
| Jane  | [650-321-4506]                             |

array ポートをフラット化すると、出力は次のようになります。

| Name  | Phones       | GCID_Phones |
|-------|--------------|-------------|
| Adams | 205-128-6478 | 1           |
| Adams | 722-515-2889 | 2           |
| Adams | 650-213-4020 | 3           |
| Jane  | 650-321-4506 | 1           |

フラット化されたフィールドの「発生数」値を編集して、配列内の要素を特定の数だけ抽出できます。値は 1 より大きい正の整数にする必要があります。この値によって、抽出する要素の数が決まります。例えば、「発生数」の値を 2 に変更して、配列の最初の 2 つの要素を抽出できます。出力は次のようになります。

| Name  | Phones       | GCID_Phones |
|-------|--------------|-------------|
| Adams | 205-128-6478 | 1           |
| Adams | 722-515-2889 | 2           |
| Jane  | 650-321-4506 | 1           |

## Struct のフラット化

トランスフォーメーションは、struct を struct 内の要素のデータ型によるフィールドにフラット化します。struct データ型をフラット化するには、すべての struct 要素が同じデータ型である必要があります。トランスフォーメーションは、struct データ型の各要素の行を作成します。

例えば、次の struct フィールドをフラット化するとします。

```
customer_address{
 city : string
 state : string
 zip : string
}
```



テーブルには以下の値があります。

| Name  | customer_address                  |
|-------|-----------------------------------|
| Clara | {<br>New York<br>NY<br>10032<br>} |

struct ポートをフラット化すると、出力は次のようになります。

| Name  | customer_address | GCID_customer_address |
|-------|------------------|-----------------------|
| Clara | New York         | 1                     |
| Clara | NY               | 2                     |
| Clara | 10032            | 3                     |

struct 要素のデータ型が同じではないものの、少なくとも最初の 2 つの要素のデータ型が同じである場合、同じデータ型で連続する要素について struct データをフラット化できます。同じデータ型で連続する struct 要素を抽出するには、[発生数] 値を編集します。値は 1 より大きい正の整数にする必要があります。例えば、struct emp\_address に次の要素が含まれます。

```
emp_address{
 city : string
 state : string
 zip : int
 country : string
}
```

[発生数] の値を 2 に定義して、city および state の struct 要素を抽出できます。この値を 3 または 4 に定義すると、マッピングの検証は失敗します。

## マップのフラット化

トランスフォーメーションでは、マップがマップのキー要素と値要素の 2 つのフィールドにフラット化されます。フラット化した map フィールドの場合、[発生数] の値を [自動] から integer 値に変更できません。

例えば、string キーと integer 値の配列から成る次の map フィールド emp\_sal をフラット化するとします。

```
<emp_name -> [base_sal, bonus, commision]>
```

次の図は、[ノーマライザ] ビューでフラット化する map フィールドを示しています。

| Name    | Level | Occurs | Type   | Type Configuration    | Precisi... | Scale | Description |
|---------|-------|--------|--------|-----------------------|------------|-------|-------------|
| emp_id  | 1     | 1      | string | N/A                   | 10         | 0     |             |
| emp_sal | 1     | 1      | map    | < string, string [] > |            |       |             |

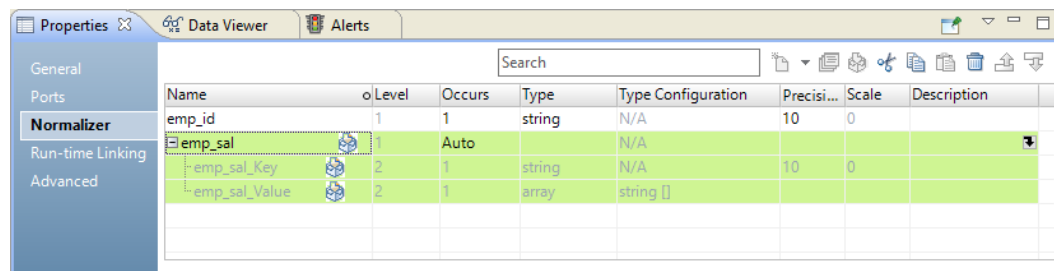
テーブルには以下の値があります。

| emp_id | emp_sal                          |
|--------|----------------------------------|
| 12200  | <Greg -> [4000, 1000, 500]>      |
| 12201  | <Patricia -> [3800, 1500, 1000]> |

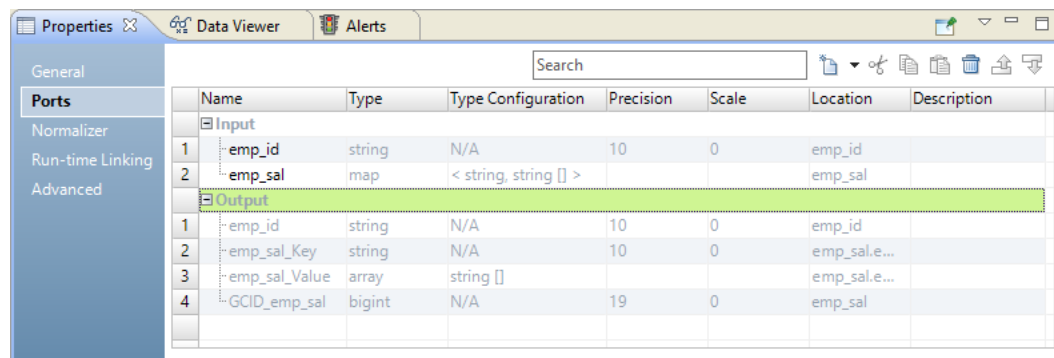
map ポートをフラット化すると、出力はマップキーの string フィールドとマップ値の array フィールドを次のように返します。

| emp_id | emp_sal_Key | emp_sal_Value      | GCID_emp_salary |
|--------|-------------|--------------------|-----------------|
| 12200  | Greg        | [4000, 1000, 500]  | 1               |
| 12201  | Patricia    | [3800, 1500, 1000] | 1               |

次の図は、[ノーマライザ] ビューで map キーフィールドと array 値フィールドにフラット化される map フィールドを示しています。



次の図は、[ポート] ビューの [出力] グループを示しています。



## フィールドのフラット化

複合データ型のフィールドをフラット化して、階層データを変更したり、リレーショナルデータに変換したりします。

1. [ノーマライザ] ビューをクリックします。
2. 複合データ型のフィールドを選択します。

次の図は、string 要素から成る array 型のフィールドを示しています。

| Name           | Level | Occurs | Type   | Type Configuration | Precision | Scale | Description |
|----------------|-------|--------|--------|--------------------|-----------|-------|-------------|
| EmpName        | 1     | 1      | string | N/A                | 100       | 0     |             |
| Salary         | 1     | 1      | double | N/A                | 15        | 0     |             |
| arr_Dep_Emp_ID | 1     | 1      | array  | string []          |           |       |             |

3. [非階層化] ボタンをクリックします。

次の図に、[非階層化] ボタンを示します。

| Name           | Level | Occurs | Type   | Type Configuration | Precision | Scale | Description |
|----------------|-------|--------|--------|--------------------|-----------|-------|-------------|
| EmpName        | 1     | 1      | string | N/A                | 100       | 0     |             |
| Salary         | 1     | 1      | double | N/A                | 15        | 0     |             |
| arr_Dep_Emp_ID | 1     | 1      | array  | string []          |           |       |             |

フラット化アクションによって、複合データ型のフィールドがフラット化されたフィールドに置換され、[発生数] の値が [自動] に変更されます。フラット化されたフィールドのデータ型は、フラット化する複合データ型によって異なります。

次の図は、string 型のフラット化されたフィールドを示しています。

| Name           | Level | Occurs | Type   | Type Configuration | Precision | Scale | Description |
|----------------|-------|--------|--------|--------------------|-----------|-------|-------------|
| EmpName        | 1     | 1      | string | N/A                | 100       | 0     |             |
| Salary         | 1     | 1      | double | N/A                | 15        | 0     |             |
| arr_Dep_Emp_ID | 1     | Auto   | string | N/A                | 10        | 0     |             |

次の図は、[ポート] ビューでフラット化された string 出力ポートおよび [GCID] 出力ポートを示しています。

| Name                  | Type   | Type Configuration | Precision | Scale | Location     | Description |
|-----------------------|--------|--------------------|-----------|-------|--------------|-------------|
| <b>Input</b>          |        |                    |           |       |              |             |
| 1 EmpName             | string | N/A                | 100       | 0     | EmpName      |             |
| 2 Salary              | double | N/A                | 15        | 0     | Salary       |             |
| 3 arr_Dep_Emp_ID      | array  | string []          |           |       | arr_Dep_E... |             |
| <b>Output</b>         |        |                    |           |       |              |             |
| 1 EmpName             | string | N/A                | 100       | 0     | EmpName      |             |
| 2 Salary              | double | N/A                | 15        | 0     | Salary       |             |
| 3 arr_Dep_Emp_ID      | string | N/A                | 10        | 0     | arr_Dep_E... |             |
| 4 GCID_arr_Dep_Emp_ID | bigint | N/A                | 19        | 0     | arr_Dep_E... |             |

# ノーマライザトランスフォーメーションの出力グループとポート

ノーマライザトランスフォーメーションの「概要」ビューで、出力グループとポートを定義します。出力グループは、トランスフォーメーションの入力階層を定義した後に定義できます。

Developer ツールは、デフォルトで少なくとも 1 つの出力グループを生成します。この出力グループには、入力ポートのすべてのレベル 1 フィールドと最初の複数出現フィールドが含まれます。「ノーマライザ」ビューで 2 つ以上の複数出現フィールドを定義すると、Developer ツールは 2 つ目以降の複数出現フィールドごとに出力グループを作成します。

次のソース行には、顧客データ、複数出現売上フィールド、および複数出現電話フィールドが含まれます。

```
CustomerID
LastName
FirstName
Sales (occurs 4 times)
Phone (occurs 3 times)
```

Developer ツールは、入力構造から 2 つの出力グループを作成します。

```
Output
CustomerID
LastName
FirstName
Sales
```

```
Output1
Phone
GCID_Phone
```

ソースデータに 2 つ以上の複数出現フィールドが含まれるため、Developer ツールは Output1 グループを 1 つ作成します。レコード内で定義されるフィールドについては、Developer ツールはグループを作成しません。レコードを定義する場合は、レコード内のフィールドを含める出力グループを定義する必要があります。

次のソース行には、顧客フィールドおよび 2 回出現する住所レコードが含まれます。

```
CustomerID
 FirstName
 LastName
 Address
 Street
 City
 State
 Country
 Address1
 Street1
 City1
 State1
 Country1
```

Developer ツールは、次のフィールドを含める出力グループを作成します。

```
CustomerID
FirstName
LastName
```

Developer ツールは、レベル 1 の顧客フィールドのデフォルト出力グループを作成します。デフォルトの出力グループは住所レコードは受け入れません。出力内に住所データをどのように返すかを設定する必要があります。

出力グループは、出力行をどのように構成するかに基づいて作成します。ソース行に顧客データと住所データが含まれる場合は、顧客フィールド用の出力グループを作成できます。住所フィールドには別の出力グループを作成できます。または、デフォルトの出力グループを更新して住所フィールドを追加することもできます。次に、出力グループの設定によって異なる出力結果の例を示します。

# 出力グループの作成

ノーマライゼーションの【概要】ビューで、出力グループを作成します。

ノーマライゼーションの【概要】ビューを開くと、Developer ツールが入力階層から作成したデフォルトのグループが表示されます。

新しい出力グループを作成する場合は、ダイアログボックスに入力階層のフィールドとレコードのリストが表示されます。グループに含めるフィールドまたはレコードを選択します。

## 出力グループの例

次の図に、【新しい出力グループ】ダイアログボックスを示します。

New Output Group

Select the Normalizer fields to add to the output group.

| Name                                        | Occurs | Type   | Precision | Scale |  |
|---------------------------------------------|--------|--------|-----------|-------|--|
| <input type="checkbox"/> CustomerID         | 1      | string | 10        | 0     |  |
| <input type="checkbox"/> FirstName          | 1      | string | 10        | 0     |  |
| <input type="checkbox"/> LastName           | 1      | string | 10        | 0     |  |
| <input checked="" type="checkbox"/> Address | 1      |        |           |       |  |
| <input checked="" type="checkbox"/> Street  | 1      | string | 10        | 0     |  |
| <input checked="" type="checkbox"/> City    | 1      | string | 10        | 0     |  |
| <input checked="" type="checkbox"/> State   | 1      | string | 10        | 0     |  |
| <input checked="" type="checkbox"/> Country | 1      | string | 10        | 0     |  |

?

OK

Cancel

Address レコードを選択すると、Developer ツールにより Address レコードのフィールドに対応する出力ポートのグループが作成されます。Output1 グループには、Street、City、State、および Country の各ポートが含まれます。出力グループのポートは変更できます。

次の図に、【概要】ビューの Output グループと Output1 グループを示します。

|   |              |        |    |   |                 |
|---|--------------|--------|----|---|-----------------|
|   | Output       |        |    |   |                 |
| 1 | CustomerID   | string | 10 | 0 | CustomerID      |
| 2 | FirstName    | string | 10 | 0 | FirstName       |
| 3 | LastName     | string | 10 | 0 | LastName        |
|   | Output1      |        |    |   |                 |
| 1 | Street       | string | 10 | 0 | Address.Street  |
| 2 | State        | string | 10 | 0 | Address.State   |
| 3 | Country      | string | 10 | 0 | Address.Country |
| 4 | GCID_Address | bigint | 19 | 0 | Address         |

Output グループから Customer テーブルに行を返すようにノーマライゼーションを設定することもできます。

Customer テーブルは、次のような行データを受け取ります。

100, Robert, Bold  
200, James, Cowan

Output1 グループから Address テーブルに行を返すこともできます。Address テーブルは、Street、City、State、Country、および GCID を受け取ります。

Address テーブルは、次のような行データを受け取ります。

100 Summit Dr, Redwood City, CA, United States,1  
41 Industrial Way, San Carlos, CA, United States,2  
85 McNulty Way, Los Angeles, CA, United States,1  
55 Factory Street, Los Vegas, NV, United States,2

GCID は、出力行にある顧客アドレスのインスタンスを識別します。この例では、ノーマライゼイトランスフォーメーションは Address レコードの 2 つのインスタンスを返します。それぞれの出力行には、1 または 2 の GCID 値が含まれます。

## 出力グループの更新

ノーマライゼイトランスフォーメーションの出力グループは更新できます。グループのフィールドは追加または削除できます。

デフォルトでは、Developer ツールは入力階層を定義する際にレベル 1 の出力グループを作成します。Developer ツールは、グループにレコードを含めません。デフォルトの出力グループは更新でき、それにレコードを追加できます。

出力グループを更新し、グループ名を強調表示し、**[新規] > [グループの更新]** をクリックします。**[出力グループの編集]** ダイアログボックスに入力階層のフィールドが表示されます。グループに含めるフィールドを選択します。

### 出力グループの更新の例

前の例では、Developer ツールは CustomerID、FirstName、および LastName の各フィールドを含むデフォルトの出力グループを作成しました。

次に図に、デフォルトの出力グループを示します。

|   | Output     |        |    |   |            |
|---|------------|--------|----|---|------------|
| 1 | CustomerID | string | 10 | 0 | CustomerID |
| 2 | FirstName  | string | 10 | 0 | FirstName  |
| 3 | LastName   | string | 10 | 0 | LastName   |

デフォルトの出力グループを更新し、Address レコードを追加できます。

次の図に、**[出力グループの編集]** ダイアログボックスを示します。

**Edit Output Group**  
Select the Normalizer fields to include in the output group.

| Name                                           | Occurs | Type   | Precision | Scale |
|------------------------------------------------|--------|--------|-----------|-------|
| <input checked="" type="checkbox"/> CustomerID | 1      | string | 10        | 0     |
| <input checked="" type="checkbox"/> FirstName  | 1      | string | 10        | 0     |
| <input checked="" type="checkbox"/> LastName   | 1      | string | 10        | 0     |
| <input checked="" type="checkbox"/> Address    | 2      |        |           |       |
| <input checked="" type="checkbox"/> Street     | 1      | string | 10        | 0     |
| <input checked="" type="checkbox"/> State      | 1      | string | 10        | 0     |
| <input checked="" type="checkbox"/> Country    | 1      | string | 10        | 0     |
|                                                |        |        |           |       |

この例では、レベル 1 のノードに CustomerID、FirstName、および LastName があります。Address レコードもレベル 1 のノードです。ノーマライゼーションは、Address を顧客データと同じ行で返すことができます。Address は複数出現なので、Developer ツールは出力グループに GCID\_Address インデックスを追加します。

次の図に、出力グループのポートを示します。

|   | Output       |        |    |   |                 |
|---|--------------|--------|----|---|-----------------|
| 1 | CustomerID   | string | 10 | 0 | CustomerID      |
| 2 | FirstName    | string | 10 | 0 | FirstName       |
| 3 | LastName     | string | 10 | 0 | LastName        |
| 4 | Street       | string | 10 | 0 | Address.Street  |
| 5 | State        | string | 10 | 0 | Address.State   |
| 6 | Country      | string | 10 | 0 | Address.Country |
| 7 | GCID_Address | bigint | 19 | 0 | Address         |

出力グループに顧客フィールドと複数出現のアドレスフィールドがある場合、ノーマライゼーションはアドレスデータの各インスタンスに対して同じ顧客フィールドを返します。

次の例に、ノーマライゼーションが出力グループから生成する行を示します。

```
100, Robert, Bold, 100 Summit Dr, Redwood City, CA, United States,1
100, Robert, Bold, 41 Industrial Way, San Carlos, CA, United States,2
200, James, Cowan, 85 McNulty Way, Los Angeles, CA, United States,1
200, James, Cowan, 55 Factory Street, Los Vegas, NV, United States,2
```

GCID ポートには Address のインスタンス番号が含まれます。GCID 値は 1 または 2 です。

## 出力グループのキーの生成

シーケンスジェネレーター変換を設定すると、ノーマライゼーションが同じソース行から返す各出力行をリンクするキーを生成できます。

マッピングの中では、シーケンスジェネレーター変換は、ノーマライゼーションの前に追加できます。シーケンスジェネレーター変換は、各ソース行にシーケンス番号を追加します。ノーマライゼーションが同じソース行から複数の出力グループまたは行を返す場合、各出力行は同じシーケンス番号を受け取ります。この番号は、ターゲットテーブル間のプライマリキーと外部キーのリレーションのキーとして使用できます。

例えば、ノーマライゼーションが、出力グループに顧客情報を返し、別の出力グループに注文情報を返すとして。シーケンス番号は、1 つのテーブルの顧客情報を別のテーブルの注文情報にリンクするために使用できます。

## ノーマライゼーションの詳細プロパティ

【詳細】 タブで、ノーマライゼーションのプロパティを設定します。

【詳細】 タブで、以下のプロパティを設定します。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## 第 1 レベルの出力グループの生成

Developer ツールは、デフォルトでノーマライザトランスフォーメーションの第 1 レベルの出力グループを生成します。ノーマライザビューで 2 つ以上の複数出現フィールドを定義すると、Developer ツールは追加の出力グループを生成します。

Developer ツールは、レベル 1 のすべての単独出現フィールドと最初のレベル 1 の複数出現フィールドを含む出力グループを作成します。トランスフォーメーションに 2 つ以上の複数出現フィールドが含まれていると、Developer ツールは追加の出力グループを作成します。

Developer ツールは、レベル 1 のレコードの出力グループを作成しません。レベル 1 のレコードを定義する場合は、出力内に配置する場所を設定する必要があります。

Developer ツールで複数出現フィールドの出力グループを作成しないようにするには、**[第 1 レベルの出力グループの自動生成]** 詳細プロパティを無効にします。

## ノーマライザトランスフォーメーションの作成

ノーマライザトランスフォーメーションには再利用可能なものと再利用不可能なものがあり、いずれかを作成することができます。再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。再利用不可能なトランスフォーメーションは、単一のマッピングで使用されます。

1. トランスフォーメーションを作成するには、次のいずれかの方法を使用します。

| オプション | 説明                                                                                                                                                    |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 再利用可能 | Object Explorer ビューで、プロジェクトまたはフォルダーを選択します。 <b>[ファイル]</b> > <b>[新規]</b> > <b>[トランスフォーメーション]</b> をクリックします。ノーマライザトランスフォーメーションを選択し、 <b>[次へ]</b> をクリックします。 |
| 再利用不可 | マッピングまたはマップレットで、ノーマライザトランスフォーメーションを <b>[トランスフォーメーション]</b> パレットからエディタにドラッグします。ソースデータオブジェクトまたはマッピングのトランスフォーメーションからポートをドラッグして、トランスフォーメーションを定義できます。       |

**[新しいノーマライザトランスフォーメーション]** ウィザードが表示されます。

2. トランスフォーメーションの名前を入力します。
3. **[次へ]** をクリックします。  
**[ノーマライザの定義]** ページが表示されます。
4. レコードを追加するには、**[新規]** ボタンをクリックしてから、**[レコード]** を選択します。
5. フィールドを追加するには、**[新規]** ボタンをクリックしてから、**[フィールド]** を選択します。  
フィールドをレコードに追加するには、フィールドを追加する前にレコードを選択する必要があります。
6. 必要に応じて、**[出現]** カラムの中の値をダブルクリックして、フィールドまたはレコードの出現を変更します。



7. **【次へ】** をクリックします。  
**【ノーマライザのポート】** ページが表示されます。
8. 出力グループを追加するには、**【新規】** ボタンをクリックし、**【新しい出力グループ】** を選択します。
9. 出力グループを編集するには、編集する必要のある出力グループを選択します。**【新規】** ボタンをクリックしてから、**【出力グループの編集】** を選択します。
10. **【完了】** をクリックします。  
トランスフォーメーションがエディタに表示されます。

## アップストリームのソースからのノーマライザトランスフォーメーションの作成

入力とポートを作成するには、空のノーマライザトランスフォーメーションを作成し、ソースデータオブジェクトまたはトランスフォーメーションからノーマライザトランスフォーメーションにポートをドラッグします。

1. ソースデータをノーマライザトランスフォーメーションに渡すためのソースまたはトランスフォーメーションを含むマッピングを作成します。
2. ノーマライザトランスフォーメーションを作成するには、トランスフォーメーションパレットからノーマライザトランスフォーメーションを選択し、トランスフォーメーションをエディタにドラッグします。  
**【ノーマライザ】** ダイアログボックスが表示されます。
3. **【完了】** をクリックして、空のトランスフォーメーションを作成します。
4. マッピングのソースまたはトランスフォーメーションからポートを選択してノーマライザトランスフォーメーションにドラッグします。  
ノーマライザトランスフォーメーションに入出力ポートが表示されます。Developer ツールは、1 つの入力グループと 1 つの出力グループを作成します。
5. **【ノーマライザ】** ビューを開き、デフォルトグループを更新し、必要に応じてフィールドをレコードに編成します。
6. 複数のフィールドを単一の複数出現フィールドにマージするには、**【ノーマライザ】** ビューでフィールドを選択し、**【マージ】** オプションをクリックします。  
複数出現フィールドの名前を選択します。

## ノーマライザマッピングの例

小売企業は、自社の店舗の総売上高を受け取ります。企業は、店舗情報と 4 つの売上高を含む行データを受け取ります。各売上高は、その年の単一四半期の総売上高を表します。

次の例では、ノーマライザトランスフォーメーションを定義して、Store ターゲットと Sales ターゲットに売上高データを返す方法を示します。Store ターゲットは、店舗ごとに 1 行を受け取ります。Sales ターゲットは、店舗ごとに 4 行を受け取ります。各行には、単一四半期の売上データが含まれます。

シーケンスジェネレータトランスフォーメーションは、各店舗に一意の ID を生成します。ノーマライザトランスフォーメーションは、各出力行で StoreID を返します。

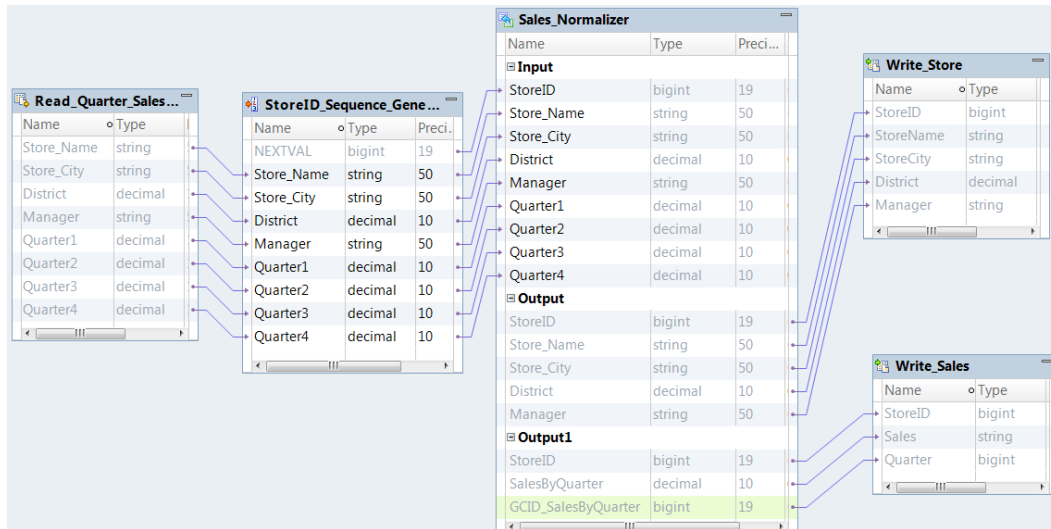
読み取りトランスフォーメーション、シーケンスジェネレータトランスフォーメーション、ノーマライザトランスフォーメーション、および 2 つの書き込みトランスフォーメーションを使用してマッピングを作成します。

## ノーマライザのマッピング例

フラットファイルソースから複数出現する四半期販売データを正規化するためのノーマライザトランスフォーメーションを含むマッピングを作成します。

ノーマライザトランスフォーメーションは、四半期ごとの売上にも個別の出力行を生成し、正規化された売上高を Sales ターゲットに書き込みます。ノーマライザトランスフォーメーションは、Store 情報を Store ターゲットに書き込みます。

次の図に、ノーマライザトランスフォーメーションのマッピングを示します。



このマッピングには次のオブジェクトが含まれています。

**Read\_STORE**

複数出現のフィールドが入ったデータソースです。

**StoreID シーケンスジェネレータトランスフォーメーション**

Store テーブルを Sales テーブルにリンクするための storeID キーを生成するシーケンスジェネレータトランスフォーメーションです。

**Sales\_Normalizer**

複数出現する売上データを正規化するノーマライザトランスフォーメーションです。

**Write\_Store**

ノーマライザトランスフォーメーションから店舗情報を受け取るターゲットです。

**Write\_Sales**

ノーマライザトランスフォーメーションから売上数値を受け取るターゲットです。

## ノーマライザの定義例

ソースは、店舗情報と四半期販売データが含まれているフラットファイルです。【ノーマライザ】ビューで、ソースデータの構造を定義します。

STORE フラットファイルには、以下のソースデータが入っています。

| StoreID | Store_Name | Store_City | 地域   | マネージャ   | Quarter1 | Quarter2 | Quarter3 | Quarter4 |
|---------|------------|------------|------|---------|----------|----------|----------|----------|
| 1       | BigStore   | New York   | East | Robert  | 100      | 300      | 500      | 700      |
| 2       | SmallStore | Phoenix    | West | Radhika | 250      | 450      | 650      | 850      |

フラットファイルを読み取りトランスフォーメーションとしてマッピングに追加してから、空のノーマライザトランスフォーメーションを作成します。ポートを Read\_STORE データオブジェクトからノーマライザトランスフォーメーションにドラッグして、ノーマライザ定義を作成します。

【ノーマライザ】ビューには、Store\_Name、Store\_City、District、および Manager の各フィールドのインスタンスが 1 つずつ含まれます。【ノーマライザ】ビューには、QUARTER というフィールドの 4 つのインスタンスがあります。QUARTER のフィールドをマージし、4 回出現する単一の複数出現フィールドを作成します。

次の図に、マージされた Quarter フィールドを含むノーマライザの定義を示します。

| Name           | Level | Occurs | Type    | Preci... | Scale |
|----------------|-------|--------|---------|----------|-------|
| StoreID        | 1     | 1      | bigint  | 19       | 0     |
| Store_Name     | 1     | 1      | string  | 50       | 0     |
| Store_City     | 1     | 1      | string  | 50       | 0     |
| District       | 1     | 1      | decimal | 10       | 0     |
| Manager        | 1     | 1      | string  | 50       | 0     |
| SalesByQuarter | 1     | 4      | decimal | 10       | 0     |

## ノーマライザの入力グループおよび出力グループ例

入力階層を変更した後のノーマライザトランスフォーメーションには、1 つの入力グループと 1 つのデフォルト出力グループがあります。出力ポートは、2 つのグループに再編成する必要があります。Store の情報を格納する 1 つのグループと Sales の情報を格納する 1 つのグループが必要です。

入力グループには、ソース内の各フィールドに対してポートが入っています。出力グループには、Store のフィールドのためのポートと複数出現 SalesByQuarter フィールドのためのポートがあります。出力グループには、複数出現 SalesByQuarter フィールドに対応する生成カラム ID (GCID\_SalesByQuarter) もあります。

四半期売上を別のターゲットに返す場合は、【概要】ビューで新しいグループを作成します。Output1 グループに次のフィールドを追加します。

StoreID  
SalesByQuarter  
GCID\_SalesByQuarter

デフォルトの出力グループを更新します。次のフィールドを削除します。

SalesByQuarter  
GCID\_SalesByQuarter

次の図に、[概要] ビューの入力グループと出力グループを示します。

|                | Name                | Type    | Precision | Scale | Location   |
|----------------|---------------------|---------|-----------|-------|------------|
| <b>Input</b>   |                     |         |           |       |            |
| 1              | StoreID             | bigint  | 19        | 0     | StoreID    |
| 2              | Store_Name          | string  | 50        | 0     | Store_N... |
| 3              | Store_City          | string  | 50        | 0     | Store_City |
| 4              | District            | decimal | 10        | 0     | District   |
| 5              | Manager             | string  | 50        | 0     | Manager    |
| 6              | Quarter1            | decimal | 10        | 0     | SalesBy... |
| 7              | Quarter2            | decimal | 10        | 0     | SalesBy... |
| 8              | Quarter3            | decimal | 10        | 0     | SalesBy... |
| 9              | Quarter4            | decimal | 10        | 0     | SalesBy... |
| <b>Output</b>  |                     |         |           |       |            |
| 1              | StoreID             | bigint  | 19        | 0     | StoreID    |
| 2              | Store_Name          | string  | 50        | 0     | Store_N... |
| 3              | Store_City          | string  | 50        | 0     | Store_City |
| 4              | District            | decimal | 10        | 0     | District   |
| 5              | Manager             | string  | 50        | 0     | Manager    |
| <b>Output1</b> |                     |         |           |       |            |
| 1              | StoreID             | bigint  | 19        | 0     | StoreID    |
| 2              | SalesByQuarter      | decimal | 10        | 0     | SalesBy... |
| 3              | GCID_SalesByQuarter | bigint  | 19        | 0     | SalesBy... |

StoreID は、Store 情報を Sales 情報にリンクする生成キーです。両方の出力グループが StoreID を返すことを確認します。

## ノーマライザのマッピング出力例

マッピングに書き込みトランスフォーメーションを追加し、データオブジェクトにノーマライザトランスフォーメーションの出力ポートを接続します。

マッピングを実行すると、ノーマライザトランスフォーメーションは次の行を Store ターゲットに書き込みます。

| StoreID | Store_Name | Store_City | 地域   | マネージャ   |
|---------|------------|------------|------|---------|
| 1       | BigStore   | New York   | East | Robert  |
| 2       | SmallStore | Phoenix    | West | Radhika |

ノーマライザトランスフォーメーションは、次の行を Sales ターゲットに書き込みます。

| StoreID | SalesByQuarter | GCID_SalesByQuarter |
|---------|----------------|---------------------|
| 1       | 100            | 1                   |
| 1       | 300            | 2                   |
| 1       | 500            | 3                   |
| 1       | 700            | 4                   |
| 2       | 250            | 1                   |
| 2       | 450            | 2                   |
| 2       | 650            | 3                   |
| 2       | 850            | 4                   |

# 非ネイティブ環境でのノーマライザトランスフォーメーション

非ネイティブ環境でのノーマライザトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

## 第 32 章

# マージトランスフォーメーション

この章では、以下の項目について説明します。

- [マージトランスフォーメーションの概要, 522 ページ](#)
- [マージストラテジの設定, 522 ページ](#)
- [マージトランスフォーメーションの詳細プロパティ, 523 ページ](#)
- [非ネイティブ環境でのマージトランスフォーメーション, 523 ページ](#)

## マージトランスフォーメーションの概要

マージトランスフォーメーションは、複数の入力カラムからデータ値を読み取り、1 つの出力カラムを作成するパッシブトランスフォーメーションです。

マージトランスフォーメーションを使用して、望ましい形式でデータを作成します。例えば、Customer\_Firstname フィールドと Customer\_Surname フィールドを結合して、Customer\_FullName というフィールドを作成できます。

マージトランスフォーメーション内では、複数の統合ストラテジを作成できます。マージトランスフォーメーションには、ストラテジの作成に使用するウィザードが用意されています。

## マージストラテジの設定

マージストラテジを設定するには、マージトランスフォーメーションの **【ストラテジ】** ビューで設定を編集します。

1. **【ストラテジ】** ビューを選択します。
2. **【新規】** をクリックします。  
新しいストラテジ ウィザードが開きます。
3. **【入力】** フィールドをクリックして、ストラテジの入力ポートを選択します。
4. マージされた項目間に配置するマージ文字を定義するには、**【選択】** をクリックします。マージ文字を選択しなかった場合は、デフォルトでスペース文字が使用されます。
5. 必要に応じて、**【マージされた出力に空の文字列を含める】** を選択して、出力に空の入力文字列を含めます。
6. **【完了】** をクリックします。

# マージトランスフォーメーションの詳細プロパティ

Data Integration Service でマージトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

**【詳細】** タブでは、以下のプロパティを設定します。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

# 非ネイティブ環境でのマージトランスフォーメーション

非ネイティブ環境でのマージトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## 第 33 章

# パーサートランスフォーメーション

この章では、以下の項目について説明します。

- [パーサートランスフォーメーションの概要, 524 ページ](#)
- [パーサートランスフォーメーションのモード, 525 ページ](#)
- [パーサートランスフォーメーションを使用するとき, 525 ページ](#)
- [パーサートランスフォーメーションでの参照データの使用, 526 ページ](#)
- [トークン解析操作, 528 ページ](#)
- [トークン解析ポート, 529 ページ](#)
- [トークン解析のプロパティ, 529 ページ](#)
- [パターンベースの解析モード, 532 ページ](#)
- [トークン解析ストラテジの設定, 533 ページ](#)
- [パターン解析ストラテジの設定, 533 ページ](#)
- [パーサートランスフォーメーションの詳細プロパティ, 534 ページ](#)
- [非ネイティブ環境でのパーサートランスフォーメーション, 534 ページ](#)

## パーサートランスフォーメーションの概要

パーサートランスフォーメーションは、入力データ値を新しいポートに解析するパッシブなトランスフォーメーションです。パーサートランスフォーメーションは、値に含まれる情報のタイプと入力文字列内の値の位置に従って、新しいポートに値を書き込みます。

パーサートランスフォーメーションは、データセットの構造を変更するときに使用します。パーサートランスフォーメーションはデータセットにカラムを追加し、新しいカラムにデータ値を書き込みます。パーサートランスフォーメーションは、データカラムで単一のカラムに複数の値が含まれ、カラムに含まれる情報のタイプに基づいて個々のカラムにデータ値を書き込むときに使用します。

パーサートランスフォーメーションは、データ値を定義された出力ポートに解析します。パーサートランスフォーメーションは入力データ値を特定できるが、定義された出力ポートが使用できない場合、オーバーフローポートに書き込まれます。パーサートランスフォーメーションが入力データ値を特定できない場合は、未解析データポートに書き込まれます。



# パーサートランスフォーメーションのモード

パーサートランスフォーメーションを作成するときは、トークン解析モードまたはパターンベースの解析モードのいずれかを選択します。

次のいずれかのモードを選択します。

- トークン解析モード: トークンセット、正規表現、確率モデル、参照テーブルなど、参照データオブジェクト内の値に一致する入力値を解析するには、このモードを使用します。 トランスフォーメーションでは、複数のトークン解析ストラテジを使用できます。
- パターンベースの解析モード: パターンセット内の値に一致する入力値を解析するには、このモードを使用します。

## パーサートランスフォーメーションを使用するとき

パーサートランスフォーメーションは、カラム内のデータフィールドに複数のタイプの情報が含まれ、フィールド値を新しいカラムに移動するときに使用します。 パーサートランスフォーメーションを使用すると、データセット内の情報のタイプごとに新しいカラムを作成できます。

以下の例に、パーサートランスフォーメーションで実行できるいくつかのタイプの構造変更を示します。

### 連絡先データの新しいカラムの作成

名前データを単一のカラムから複数のカラムに解析するデータ構造を作成することができます。例えば、敬称、名前、ミドルネーム、姓などのカラムを作成できます。

入力ポートで人名の構造を表す確率モデルのあるパーサートランスフォーメーションを設定します。 モデルを定義するには、入力ポートデータのサンプルを使用します。

確率モデルを入力ポートに適用し、名前の値を新しいカラムに書き込む、トークン解析ストラテジを作成します。 パーサートランスフォーメーションは、入力文字列の各値の位置と値が表す名前のタイプに基づいて、名前の値を新しいカラムに書き込みます。

**注:** パターンベースの解析ストラテジを使用して連絡先データを解析することもできます。 パターンベースの解析ストラテジを設定するときは、入力ポートで名前の構造を表すパターンを定義します。

### 住所カラムの作成

住所データの単一のカラムを配達可能な住所を示す複数のカラムに解析するデータ構造を作成することができます。

郵便番号、州名、市区町村名などの認識可能な住所要素が含まれる参照テーブルのあるパーサートランスフォーメーションを設定します。 各住所要素を新しいポートに書き込むトークン解析ストラテジを作成します。

町名と番地のデータは参照テーブルでキャプチャするには一般的すぎるため、参照テーブルを使用して入力文字列から所在地住所データを解析することはできません。 ただし、オーバーフローポートを使用してこのデータをキャプチャすることは可能です。 すべての市区町村、州、および郵便番号データを住所から解析すると、残りのデータに町名と番地の情報が含まれます。

例えば、トークン解析ストラテジを使用して次の住所を住所要素に分割します。

123 MAIN ST NW STE 12 ANYTOWN NY 12345

解析ストラテジは、住所要素を次のカラムに書き込むことができます。

| カラム名    | データ                   |
|---------|-----------------------|
| オーバーフロー | 123 MAIN ST NW STE 12 |
| City    | ANYTOWN               |
| State   | NY                    |
| ZIP     | 12345                 |

#### 製品データカラムの作成

製品データの単一のカラムを製品在庫の詳細を示す複数のカラムに解析するデータ構造を作成することができます。

寸法、色、重量などの在庫要素が含まれるトークンセットのあるトランスフォーメーションを設定します。各在庫要素を新しいポートに書き込むトークン解析ストラテジを作成します。

例えば、トークン解析ストラテジを使用して次の塗料の説明を別個の在庫要素に分割します。

500ML Red Matt Exterior

解析ストラテジは、住所要素を次のカラムに書き込むことができます。

| カラム名 | データ   |
|------|-------|
| サイズ  | 500ML |
| 色    | Red   |
| スタイル | Matt  |
| 外装   | Y     |

## パーサートランスフォーメーションでの参照データの使用

Informatica Developer は、パーサートランスフォーメーションで使用可能な複数の参照データオブジェクトとともにインストールされます。Developer ツールで参照データを作成することもできます。

パーサートランスフォーメーションに参照データオブジェクトを追加すると、指定した新しいカラムにオブジェクト内の値に一致する文字列が書き込まれます。

次の表に、使用できる参照データのタイプを示します。

| 参照データのタイプ | 説明                                                                                                                          |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| パターンセット   | 文字列内の各値の相対的な位置に基づいて、データ値を特定します。                                                                                             |
| 確率的なモデル   | トークン解析操作にあいまい一致機能を追加します。パーサートランスフォーメーションは、確率モデルを使用して、文字列内の情報のタイプを推測することができます。あいまい一致機能を有効にするには、Developer ツールで確率モデルをコンパイルします。 |
| 参照テーブル    | データベーステーブル内のエントリに一致する文字列を検索します。                                                                                             |
| 正規表現      | 定義した条件に一致する文字列を特定します。正規表現を使用して、大きな文字列内の文字列を検索することができます。                                                                     |
| トークンセット   | 文字列に含まれる情報のタイプに基づいて、文字列を特定します。<br>Informatica は、単語、電話番号、郵便番号、製品コードの定義など、トークンセットのさまざまなタイプのトークン定義とともにインストールされます。              |

## パターンセット

パターンセットには、トークンラベル適用操作の出力のデータパターンを識別する式が含まれます。パターンセットを使用すると、トークン化されたデータの出力ポートを分析し、一致する文字列を 1 つ以上の出力ポートに書き込むことができます。パターンセットは、パターン解析モードを使用するパーサートランスフォーメーションで使用します。

例えば、名前と頭文字を識別するパターンセットを使用するようにパーサートランスフォーメーションを設定することができます。このトランスフォーメーションでは、パターンセットを使用して、トークンラベル適用モードのラベラトランスフォーメーションの出力を分析します。出力に含まれる名前と頭文字を別のポートに書き込むようにパーサートランスフォーメーションを設定できます。

## 確率モデル

確率モデルは、トークンに含まれている情報のタイプと入力文字列内のトークンの位置によってトークンを特定します。

確率モデルには、参照データ値とラベル値が含まれます。参照データ値は、トランスフォーメーションに接続する入力ポートのデータを表します。ラベル値は、参照データ値に含まれる情報のタイプを説明します。モデルの各参照データ値にラベルを割り当てます。

確率モデルのラベルに参照データ値をリンクするには、モデルをコンパイルします。コンパイル処理では、データ値とラベル間の、一連の論理的な関連付けが生成されます。モデルを読み取るマッピングを実行すると、データ統合サービスはモデルのロジックをトランスフォーメーション入力データに適用します。データ統合サービスは、入力データ値を最も正確に描写しているラベルを返します。

Developer tool で確率モデルを作成します。モデルリポジトリは、確率モデルオブジェクトを格納します。Developer ツールは、データ値、ラベル、およびコンパイルデータを、Informatica ディレクトリ構造内のファイルに書き込みます。

**注:** トークン解析操作に確率モデルを追加し、確率モデルのラベル設定を編集する場合は、この操作を無効にします。確率モデルのラベル設定を更新するときは、そのモデルが使用される解析操作をすべて再作成します。

## 参照テーブル

参照テーブルは、少なくとも2つのカラムが含まれるデータベーステーブルです。一方のカラムにはデータ値の標準バージョンまたは必要なバージョンが含まれ、もう一方のカラムにはデータ値の代替バージョンが含まれます。参照テーブルをトランスフォーメーションに追加すると、テーブルに存在する値が入力ポートデータで検索されます。作業するデータプロジェクトに役立つデータを含むテーブルを作成できます。

## 正規表現

解析操作における正規表現とは、入力データ内の1つ以上の文字列を特定するために使用できる式です。特定された文字列が1つ以上の出力ポートに書き込まれます。正規表現は、トークン解析モードを使用するパーサートランスフォーメーションで使用できます。

パーサートランスフォーメーションでは、正規表現を使用して入力データのパターンを一致させ、一致するすべての文字列を1つ以上の出力に解析します。例えば、正規表現を使用して入力データに含まれるすべての電子メールアドレスを識別し、電子メールアドレスの構成要素ごとに異なる出力に解析することができます。

## トークンセット

トークンセットには、特定のトークンを識別する式が含まれます。トークンセットは、トークン解析モードを使用するパーサートランスフォーメーションで使用できます。

トークンセットを使用して、解析操作の一部として特定のトークンを特定します。例えば、トークンセットを使用して、「AccountName@DomainName」形式が使用されるすべての電子メールアドレスを解析することができます。

# トークン解析操作

トークン解析モードでは、トークンセットのデータ、正規表現、確率モデル、または参照テーブルのエントリに一致する文字列が解析されます。

トークン解析を実行するには、トランスフォーメーションの【**ストラテジ**】ビューでストラテジを追加します。ストラテジごとに1つ以上の操作を追加できます。トランスフォーメーションには、ストラテジの作成に使用するウィザードが用意されています。

トークン解析ストラテジには、以下のタイプの操作を追加できます。

### トークンセットを使用した解析

組み込みまたはユーザー定義のトークンセットを使用して、入力データを解析します。トークンセット操作では、1つ以上の出力にデータを書き込むカスタム正規表現を使用できます。

確率モデルを使用して、入力データ値の識別と解析を行うこともできます。

### 参照テーブルを使用した解析

参照テーブルを使用して、入力データを解析します。

操作はストラテジに示された順序で実行されます。

# トークン解析ポート

トークン解析ポートをデータに適した設定で構成します。

トークン解析モードのパーサートランスフォーメーションには、以下のポートタイプがあります。

## 入力

パーサートランスフォーメーションに渡すデータが含まれます。【**ストラテジ**】タブで指定された【**結合文字の入力**】を使用して、すべての入力ポートがマージされて1つのデータ文字列に結合されます。入力の結合文字を指定しなかった場合は、デフォルトでスペース文字が使用されます。

## 解析された出力ポート

正常に解析された文字列が含まれるユーザー定義の出力ポート。複数の解析文字列で同じ出力が使用されている場合は、【**ストラテジ**】タブで指定された【**結合文字の出力**】を使用して、組み合わせられたデータ文字列に出力がマージされます。結合文字の出力を指定しなかった場合は、デフォルトでスペース文字が使用されます。

## オーバーフロー

トランスフォーメーションで定義された出力数に収まらない、正常に解析された文字列が含まれます。例えば、トランスフォーメーションに2つの「WORD」出力のみがある場合、文字列「John James Smith」は「Smith」というオーバーフロー出力になります。追加するストラテジごとにオーバーフローポートが作成されます。

【詳細なオーバーフロー】オプションを選択すると、モデルにラベルごとのオーバーフローポートが作成されます。

## 未解析

トランスフォーメーションで正常に解析できない文字列が含まれます。追加するストラテジごとに未解析ポートが作成されます。

## 確率的な一致の出力ポート

確率的な一致方法を使用するように解析ストラテジを設定すると、パーサートランスフォーメーションはポートを追加して出力ポートごとにマッチ率を格納します。

以下の表に、ポートのタイプを示します。

| ポートタイプ        | 確率的な一致で作成されるポート                      |
|---------------|--------------------------------------|
| 解析された出力ポート    | [ラベル名]出力<br>[ラベル名]スコア出力              |
| オーバーフローデータポート | [オーバーフローデータ]出力<br>[[オーバーフローデータ]スコア出力 |
| 未解析データポート     | [未解析データ]出力<br>[未解析データ]スコア出力          |

# トークン解析のプロパティ

トークン解析操作のプロパティは、パーサートランスフォーメーションの【**ストラテジ**】ビューで設定します。

## 全般プロパティ

全般プロパティは、ストラテジで定義するすべてのトークン解析操作に適用されます。ストラテジに名前を付けたり、入出力ポートを指定したり、ストラテジで確率的な一致方法を有効にするかどうかを指定するには、全般プロパティを使用します。

以下の表に、全般プロパティを示します。

| プロパティ         | 説明                                                                       |
|---------------|--------------------------------------------------------------------------|
| 名前            | ストラテジの名前を入力します。                                                          |
| 入力            | ストラテジ操作で読み取ることができる入力ポートを特定します。                                           |
| 出力            | ストラテジ操作で書き込むことができる出力ポートを特定します。                                           |
| 説明            | ストラテジを説明します。このプロパティはオプションです。                                             |
| 確率的な一致方法を使用   | ストラテジが確率モデルを使用してトークンを識別できるように指定します。                                      |
| 結合文字の入力       | 入力データポートの結合に使用される文字を指定します。すべての入力ポートがマージされて 1 つのデータ文字列に結合され、その文字列が解析されます。 |
| 結合文字の出力       | 複数の解析操作で同じ出力が使用されているときに出力データの値の結合に使用される文字を指定します。                         |
| 反転有効          | 右から左にデータを解析するようにストラテジを設定します。このプロパティは、確率的な一致に対して無効になっています。                |
| オーバーフローの反転有効  | 右から左にオーバーフローデータを解析するようにストラテジを設定します。このプロパティは、確率的な一致に対して無効になっています。         |
| 詳細なオーバーフローが有効 | 解析操作ごとに一意のオーバーフローフィールドを作成します。                                            |
| 区切り文字         | 入力データを別々のトークンに区切る区切り文字を指定します。デフォルトはスペースです。                               |

## 確率モデルのプロパティ

トークン解析ストラテジを設定するときは、トークンセットの代わりに確率モデルを選択できます。するには、**【トークンセットを使用した解析】** 操作を選択し、確率的な一致方法を使用するオプションを選択します。

以下の表に、確率モデルのプロパティを示します。

| プロパティ    | 説明                                                          |
|----------|-------------------------------------------------------------|
| 名前       | 操作の名前を入力します。                                                |
| フィルタテキスト | 入力する文字またはワイルドカードを使用して、トークンセット、確率モデル、または正規表現のリストをフィルタリングします。 |
| 確率モデル    | 選択する確率モデルを特定します。                                            |

## 参照テーブルのプロパティ

参照テーブルプロパティは、参照テーブルを使用するためにラベル適用操作を設定するときに適用されます。

以下の表に、参照テーブルのプロパティを示します。

| プロパティ      | 説明                                               |
|------------|--------------------------------------------------|
| 名前         | 操作の名前を入力します。                                     |
| 参照テーブル     | 入力値の解析に使用する参照テーブルを指定します。                         |
| 大文字小文字の区別  | 入力文字列を参照テーブルのエントリと照合するときに大文字と小文字を区別するかどうかを指定します。 |
| 有効な値で一致を置換 | 解析されたデータを参照テーブルの [有効] カラムのデータに置き換えます。            |
| 出力         | 解析されたデータの出力ポートを指定します。                            |

## トークンセットのプロパティ

トークンセットプロパティは、トークンセットを使用するように解析操作を設定するときに適用されます。

トークンセットを使用して入力を解析するには、**【トークンセットを使用した解析】** 操作を選択します。確率的な一致方法を使用するには、このオプションの選択を取り消します。

以下の表に、トークン設定プロパティを示します。

| プロパティ             | 説明                                                                |
|-------------------|-------------------------------------------------------------------|
| 名前                | 操作の名前を入力します。                                                      |
| トークンセット（単一出力のみ）   | データの解析に使用されるトークンセットを指定します。データは単一のポートに書き込まれます。                     |
| 正規表現（単一出力または複数出力） | データの解析に使用される正規表現を指定します。入力フィールドに複数の文字列が検出された場合、データは複数のポートに書き込まれます。 |
| 出力                | 書き込まれる出力ポートを特定します。                                                |

トークンセットまたは正規表現を追加、編集、または削除できます。トークンセットのリストをフィルタリングすることもできます。

以下の表に、タスクの実行に使用するプロパティを示します。

| プロパティ    | 説明                                                           |
|----------|--------------------------------------------------------------|
| フィルタテキスト | トークンセットまたは正規表現のリストをフィルタリングします。フィルタとしてテキスト文字とワイルドカード文字を使用します。 |
| 追加       | カスタムのトークンセットまたは正規表現を定義するために使用します。                            |
| 編集       | カスタムのトークンセットの内容を編集します。                                       |

| プロパティ | 説明                                                                                                              |
|-------|-----------------------------------------------------------------------------------------------------------------|
| インポート | モデルリポジトリのフォルダーからトークンセットまたは正規表現の再利用不可能なコピーをインポートします。トークンセットまたは正規表現のソースオブジェクトが更新されても、データ統合サービスは再利用不可能なコピーを更新しません。 |
| 削除    | カスタムのトークンセットまたは正規表現を削除します。                                                                                      |

## パターンベースの解析モード

パターンベースの解析モードでは、複数の文字列からなるパターンが解析されます。

次の方法でパターンベースの解析モードでパターンを定義します。

- 参照テーブルに定義されたパターンを使用して入力データを解析します。パターン参照テーブルは、トークンラベル適用モードを使用するラベラートランスフォーメーションのプロファイルされた出力から作成できます。
- 定義するパターンを使用して入力データを解析します。
- モデルリポジトリで設定された再利用可能なパターンからインポートしたパターンを使用して、入力データを解析します。再利用可能なパターンセットを変更しても、パーサートランスフォーメーションに追加するデータには反映されません。

「+」と「\*」のワイルドカードを使用して、パターンを定義することができます。任意の文字列に一致させるには「\*」文字を、前にある文字列の 1 つ以上のインスタンスに一致させるには「+」文字を使用します。例えば、word トークンの連続する複数のインスタンスを検索するには「WORD+」を使用し、word トークンとそれに続く任意のタイプの 1 つ以上のトークンを検索するには「WORD \*」を使用します。

パーサートランスフォーメーション内では、これらの方法の複数のインスタンスを使用できます。各インスタンスは、**[設定]** ビューで示された順序で使用されます。

**注:** パターンベースの解析モードでは、トークンラベル適用モードが使用されるラベラートランスフォーメーションの出力が必要です。パターンベースの解析モードを使用するパーサートランスフォーメーションを作成する前に、ラベラートランスフォーメーションを作成および設定します。

## パターンベースの解析ポート

パターンベースの解析ポートをデータに適した設定で構成します。

パターンベースの解析モードが使用されるパーサートランスフォーメーションには、以下のポートタイプがあります。

### Label\_Data

このポートを、トークンラベル適用モードを使用するラベラートランスフォーメーションの Labeled\_Output ポートに接続します。

### Tokenized\_Data

このポートを、トークンラベル適用モードを使用するラベラートランスフォーメーションの Tokenized\_Data ポートに接続します。

### Parse\_Status

入力パターンに対して一致が見つかった場合、このポートは値 Matched を出力します。一致が見つからなかった場合は、Unmatched を出力します。



## オーバーフロー

トランスフォーメーションで定義された出力数に収まらない、正常に解析された文字列。例えば、2つの「WORD」出力のみが定義されている場合、文字列「John James Smith」はデフォルトで「Smith」というオーバーフロー出力になります。

Parsed

ユーザー定義のポート内の正常に解析された文字列。

# トークン解析ストラテジの設定

トークン解析ストラテジを設定するには、トークン解析モードでパーサートランスフォーメーションを開き、**【ストラテジ】** ビューを選択します。

1. **【ストラテジ】** ビューを選択します。

2. **【新規】** をクリックします。

新しいストラテジ ウィザードが開きます。

3. **【入力】** フィールドをクリックして、ストラテジのポートを選択します。

4. ストラテジのプロパティを設定し、**【次へ】** をクリックします。

5. 操作を選択し、**【次へ】** をクリックします。

6. 操作プロパティを設定し、正常に解析されたデータの出力ポートを選択します。

7. 必要に応じて、**【次へ】** をクリックして、その他の操作をストラテジに追加します。

8. ストラテジにすべての操作を追加したら、**【完了】** をクリックします。

9. 必要に応じて、その他のストラテジをトランスフォーメーションに追加します。

10. 必要に応じて、ストラテジと操作が処理される順序を変更します。ストラテジまたは操作を選択し、**【上に移動】** または **【下に移動】** をクリックします。

# パターン解析ストラテジの設定

パターン解析ストラテジを設定するには、パターン解析モードでパーサートランスフォーメーションを開き、**【パターン】** ビューを選択します。

パターンを解析するようにパーサートランスフォーメーションを設定する前に、**【パターン】** ビューに必要な出力ポート名が表示されていることを確認します。パーサートランスフォーメーションは、選択された出力ポートにトークンを解析します。必要に応じて、出力ポートを追加作成します。

1. **【パターン】** ビューを選択します。

2. 1つ以上のパターンをストラテジに追加します。パターンは次の方法で追加できます。

- データ値を入力してパターンを作成します。 **【新規】** をクリックし、**【新しいパターン】** を選択します。

**【新しいパターン】** を選択した場合は、**【ここにパターンを入力】** をクリックし、1つ以上のトークンタイプを入力します。入力するトークンは、入力データフィールドのトークン構造と一致する必要があります。入力ポートのトークン構造を示すために必要なパターンを追加します。

- 参照テーブルからデータ値をインポートします。【新規】をクリックし、【新しい参照テーブル】を選択します。

【新しい参照テーブル】を選択した場合は、モデルリポジトリを参照し、トークン構造のリストを含む参照テーブルを選択します。参照テーブルには2つの列が含まれている必要があります。参照テーブルの2番目の列には数値が含まれている必要があります。

- パターンセットからデータ値をインポートします。【インポート】をクリックし、モデルリポジトリで再利用可能なパターンセットを選択します。

【インポート】を選択した場合は、モデルリポジトリ内のコンテンツセットを参照し、再利用可能なパターンセットを選択します。

**注:** 【フィルタテキスト】フィールドを使用して、参照テーブルとパターンセットのリストをフィルタリングします。

[パターン] カラムには、パターンセットと参照テーブルを一緒に指定できます。

3. [パターン] カラムの各トークンを出力ポートに割り当てます。

- トークンを出力ポートに割り当てるには、ポートカラムをダブルクリックし、メニューからトークン名を選択します。
- 複数のトークンを単一の出力に解析するには、ポートカラムをダブルクリックし、【カスタム】を選択します。トークンをポートに割り当て、使用する区切り文字を選択します。

パターンの各行のトークンを1つ以上の出力ポートに割り当てます。

4. トランスフォーメーションを保存します。

## パーサートランスフォーメーションの詳細プロパティ

Data Integration Service でパーサートランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## 非ネイティブ環境でのパーサートランスフォーメーション

非ネイティブ環境でのパーサートランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。

- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

## 第 34 章

# Python トランスフォーメーション

Python トランスフォーメーションはパッシブなトランスフォーメーションで、Python コードを使用してトランスフォーメーション機能に対するインタフェースを提供します。Python コード、および Python コードで使用するリソースファイルは、Python トランスフォーメーション内で参照します。

Python トランスフォーメーションを使用すると、トランスフォーメーションに渡されるデータでマシンモデルを実装できます。例えば、Python トランスフォーメーションを使用し、トレーニング済みのモデルを読み込む Python コードを作成できます。事前トレーニング済みモデルを使用すると、入力データを分類したり予測を作成したりできます。

Spark エンジンで Python トランスフォーメーションを使用するには、データ統合サービスマシンに Python をインストールし、Hadoop 接続で Spark の対応する詳細プロパティを設定する必要があります。

Python のインストールの詳細については、『*Data Engineering Integration ガイド*』を参照してください。

Python トランスフォーメーションは、Spark エンジンまたは Databricks Spark エンジンでのみ実行できます。ネイティブ環境で Python トランスフォーメーションを実行することはできません。

バージョン 10.4.0 から、Python トランスフォーメーションは Databricks Spark エンジンのテクニカルプレビューでサポートされるようになりました。

評価目的でのテクニカルプレビュー機能はサポートされていますが、保証対象外で本番環境には対応していません。非本番環境でのみ使用することをお勧めします。Informatica では、本番環境用に次のリリースでプレビュー機能を導入するつもりですが、市場や技術的な状況の変化に応じて導入しない場合もあります。詳細については、Informatica グローバルカスタマサポートにお問い合わせください。

Python トランスフォーメーションの詳細については、『*Data Engineering Integration ユーザーガイド*』を参照してください。

## 第 35 章

# ランクトランスフォーメーション

この章では、以下の項目について説明します。

- [ランクトランスフォーメーションの概要, 537 ページ](#)
- [動的マッピングでのランクトランスフォーメーション, 538 ページ](#)
- [ランクトランスフォーメーションのポート, 538 ページ](#)
- [ランクポート, 540 ページ](#)
- [グループ化ポートの定義, 540 ページ](#)
- [ランクキャッシュ, 541 ページ](#)
- [ランクトランスフォーメーションの詳細プロパティ, 542 ページ](#)
- [非ネイティブ環境でのランクトランスフォーメーション, 543 ページ](#)

## ランクトランスフォーメーションの概要

ランクトランスフォーメーションは、レコードを上限または下限に制限するアクティブなトランスフォーメーションです。ランクトランスフォーメーションを使用して、ポートまたはグループ内で最大または最小の数値を返すことができます。または、ランクトランスフォーメーションを使用して、マッピングのソート順の最上位または最下位の文字列を返します。

マッピング実行中に、Data Integration Service はランク計算を実行できるまで入力データをキャッシュに格納します。

ランクトランスフォーメーションは、トランスフォーメーション関数 MAX や MIN とは異なります。ランクトランスフォーメーションは、1 つの値だけではなく、最上位または最下位の値のグループを返します。たとえば、ランクトランスフォーメーションを使用して、指定された区域内での上位 10 人の販売員を選択できます。あるいは財務レポートを生成する場合に、ランクトランスフォーメーションを使用して、給与や経費の支出が最も少ない 3 つの部門を調べることができます。SQL 言語ではデータグループを取り扱う多くの関数が提供されていますが、標準 SQL 関数を使用して行セット内の最上位または最下位の層を特定することは不可能です。

トランスフォーメーションには、同じ行セットを表すすべてのポートを接続します。ランクトランスフォーメーションは、トランスフォーメーションを設定するときに指定した基準に基づいて、当該ランクに収まる行を通過させます。

ランクトランスフォーメーションはアクティブトランスフォーメーションであるため、通過する行の数を変更してしまう可能性があります。ランクトランスフォーメーションに渡すことができるのは 100 行ですが、ランク付けのために選択できるのは上位 10 行だけです。上位 10 行が、ランクトランスフォーメーションから別のトランスフォーメーションに渡されます。

ランクトランスフォーメーションへは、1 つのトランスフォーメーションからのポートを接続できます。また、ローカル変数を作成して非集計式を書き込むこともできます。

## 文字列値のランク付け

文字列ポートの最上位または最下位の値を返すようにランクトランスフォーメーションを設定することができます。Data Integration Service は、デプロイされたマッピングに対して選択されたソート順に基づいて、文字列をソートします。

マッピングが含まれるアプリケーションを設定するときは、Data Integration Service でマッピングを実行するために使用されるソート順を選択します。バイナリまたはフランス語やドイツ語などの特定の言語を選択できます。バイナリを選択した場合、Data Integration Service は各文字列のバイナリ値を計算し、そのバイナリ値を使用して文字列をソートします。言語を選択した場合、Data Integration Service はその言語のソート順を使用してアルファベット順に文字列をソートします。

## ランクトランスフォーメーションのプロパティ

ランクトランスフォーメーションを作成するときに、次のプロパティを設定することができます。

- キャッシュディレクトリを入力する。
- 最上位または最下位のランクを選択する。
- ランクを判定するための値を含む入出力ポートを選択する。ランクを定義するためのポートは 1 つだけ選択できます。
- ランク付けする行の数を選択する。
- ランクのグループを定義する（たとえば各製造業者で最も安い 10 個の製品といったグループ）。

## 動的マッピングでのランクトランスフォーメーション

動的マッピングでランクトランスフォーメーションを使用できます。アグリゲータトランスフォーメーションで動的ポートを設定し、生成されたポートを参照することができます。

生成されたポートをランクトランスフォーメーションで参照している場合に、そのポートが実行時に存在しないと、マッピングは失敗します。

動的ポートをランクポートに指定する場合、動的ポートは生成されたポートを 1 つだけ持つことができます。

動的ポートをグループ化ポートに指定すると、データ統合サービスはすべての生成されたポートをグループ化ポートと見なします。生成されたポートをグループ化ポートに指定し、親動的ポートをランクポートまたはグループ化ポートに指定する場合、マッピングは無効です。

ランクポートおよびグループ化ポートはパラメータ化できます。ランクポートにはポートタイプパラメータを使用します。グループ化ポートにはポートリストタイプのパラメータを使用します。

## ランクトランスフォーメーションのポート

ランクトランスフォーメーションには、マッピング内の別のトランスフォーメーションに接続されている入力ポート、入出力ポート、または出力ポートが含まれています。トランスフォーメーションには、パススルーポートおよび変数ポートも含まれています。

ランクトランスフォーメーションには、以下のポートタイプがあります。

## 入力

アップストリームトランスフォーメーションからデータを受信します。入力ポートを入出力ポートとして指定することができます。トランスフォーメーションには少なくとも 1 つの入力ポートが必要です。

## 動的ポート

複数のカラムを受け取って、動的な数の生成済みポートを作成することができるポート。生成されたポートは、単一のカラムを表す動的ポート内のポートです。入力、出力、および変数の動的ポートを作成できます。

## 出力

ダウンストリームトランスフォーメーションにデータを渡します。出力ポートを入出力ポートとして指定することができます。トランスフォーメーションには少なくとも 1 つの出力ポートが必要です。

## パススルー

変更せずにそのままデータを渡します。

## 変数

ローカル変数に使用されます。変数ポートを使用して、式で使用する値または計算を格納することができます。変数ポートは入力ポートまたは出力ポートであってはなりません。変数ポートはトランスフォーメーション内でデータを渡します。

# ランクインデックス

Developer ツールは、それぞれのランクトランスフォーメーションに対して RANKINDEX ポートを作成します。Data Integration Service は Rank Index ポートを使用して、グループ内における各行のランキング位置を格納します。

たとえば、会社内で給料の高い 50 人の従業員を調べるために、ランクトランスフォーメーションを作成します。ランク付けの基準とする入出力ポートとして SALARY カラムを指定し、上位 50 位以外のすべての行をフィルタで除外するようにトランスフォーメーションを設定します。

ランクトランスフォーメーションは、最上位または最下位のランクに属する行をすべて識別したあと、ランクインデックス値を割り当てます。給料を基準に上位 50 人の従業員を識別する場合、最高額の給料が支払われている従業員に対し、ランクインデックスとして 1 が与えられます。次に給料が多い従業員には、ランクインデックスとして 2 が与えられ、以下同じようにランクインデックスが割り当てられます。たとえば目録内で値段の安い 10 個の製品といった最下位ランクを求める場合、ランクトランスフォーメーションは最下位から最上位の順にランクインデックスを割り当てます。したがって、最も値段の安い商品のランクインデックスには 1 が与えられます。

2 つのランク値が一致すると、それらは同じランクインデックス内で同じ値を受け取り、トランスフォーメーションはその次の値をスキップします。たとえば国内で最上位の 5 つの小売り店を調べたときに 2 つの小売り店が同じ売り上げである場合、返されるデータは次のようになります。

| RANKINDEX | SALES | STORE       |
|-----------|-------|-------------|
| 1         | 10000 | Orange      |
| 1         | 10000 | Brea        |
| 3         | 90000 | Los Angeles |
| 4         | 80000 | Ventura     |

RANKINDEX は出力ポートのみです。ランクインデックスは、マッピング内の別のトランスフォーメーションへ渡すか、直接ターゲットへ渡すことができます。

# ランクポート

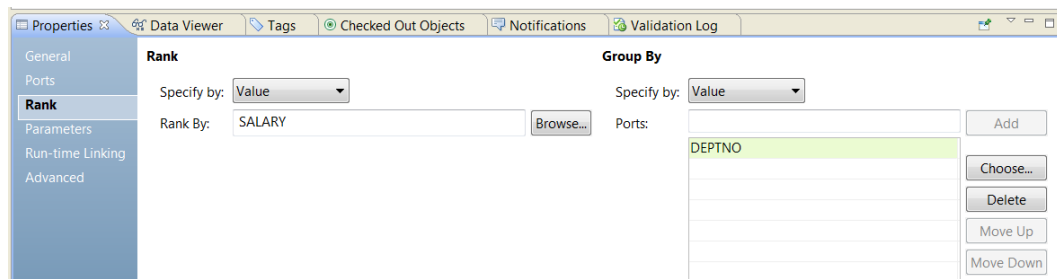
ランクポートは、値のランク付けに使用するカラムを決定します。

1つの入出力ポートまたは出力ポートをランクポートとして指定する必要があります。例えば、各部門で給与支給額に基づいて上位5名の社員をランク付けするランクトランスフォーメーションを作成できます。Salaryポートには、各社員の給与が含まれています。Salary入出力ポートをランクポートとして指定します。

【プロパティ】ビューの【ランク】タブでランクポートを選択します。ランクポートにはパラメータを使用できます。パラメータを使用するには、[指定元:] プロパティで【パラメータ】を選択します。ポートパラメータを参照するか、ポートパラメータを作成します。このパラメータのデフォルト値は、ポートまたは生成されたポートの名前です。

ランクポートはほかのトランスフォーメーションにリンクする必要があります。

次の図は、【ランク】タブを示しています。



注: ランクポートはバイナリデータ型をサポートしていません。

## グループ化ポートの定義

ランク付けされた行のグループを作成するようにランクトランスフォーメーションを設定できます。

例えば製造業者別に最も高価な商品を10個選択したい場合は、まずそれぞれの製造業者についてグループを定義します。【ランク】タブの【グループ化】パネルで、入力、入力/出力、または出力ポートのいずれかをグループ化ポートとして設定できます。

グループポート内の一意的値それぞれに対して、トランスフォーメーションは、ランク定義（最上位または最下位、および各ランク内の特定の順位）に該当する行のグループを作成します。

ランクトランスフォーメーションは行の数を2つの方法で変更します。1つは、最上位または最下位のランクに収まる行を除いたすべての行をフィルタリングして除外することにより、トランスフォーメーションを通過する行の数を減らします。もう1つはグループを定義することにより、各グループでランク付けされた行のセットを1つ作成します。

例えば、四半期ごとに上位5人の販売員をランク付けするランクトランスフォーメーションを作成した場合、ランクインデックスは四半期ごとに販売員に1から5までの数字を付けます。

| RANKINDEX | SALES_PERSON | SALES  | QUARTER |
|-----------|--------------|--------|---------|
| 1         | Sam          | 10,000 | 1       |
| 2         | Mary         | 9,000  | 1       |
| 3         | Alice        | 8,000  | 1       |



| RANKINDEX | SALES_PERSON | SALES | QUARTER |
|-----------|--------------|-------|---------|
| 4         | Ron          | 7,000 | 1       |
| 5         | Alex         | 6,000 | 1       |

【プロパティ】ビューの【詳細】タブでランキングに含まれる行数を設定します。

## グループ化パラメータ

グループに含める1つ以上のポートを含んだ、ポートリストパラメータを設定することができます。トランスフォーメーションのポートのリストからポートを選択して、ポートリストパラメータを作成します。

次の図は、パラメータを使用してグループ内のポートを特定する場合の【グループ化】タブを示しています。

ポートリストパラメータを参照するか、【新規作成】をクリックしてポートリストパラメータを作成できます。ポートリストパラメータの作成を選択すると、トランスフォーメーションのポートのリストからポートを選択できます。

## ランクキャッシュ

ランクトランスフォーメーションを使用するマッピングを実行すると、データ統合サービスはメモリにインデックスキャッシュおよびデータキャッシュを作成してトランスフォーメーションを実行します。メモリキャッシュ内の使用可能スペースよりも多くのスペースが必要な場合、データ統合サービスはオーバーフローしたデータをキャッシュファイルに格納します。

ランクトランスフォーメーションを使用するマッピングを実行すると、データ統合サービスは入力行をデータキャッシュ内の行と比較します。キャッシュに格納されている行よりも入力行の方がランクが高い場合、データ統合サービスはキャッシュに格納されている行を入力行で置き換えます。ランクトランスフォーメーションをグループ行に設定すると、データ統合サービスは各グループ内の行をランク付けします。

データ統合サービスでは、ランクトランスフォーメーションに対して以下のキャッシュが作成されます。

- Group By ポートの設定に従ってグループ値を格納するインデックスキャッシュ。
- Group By ポートに基づいて情報を格納するデータキャッシュ。

## ランクトランスフォーメーションの詳細プロパティ

Data Integration Service でランクトランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

【詳細】 タブで、以下のプロパティを設定します。

### 上/下

カラムの最上位または最下位のどちらのランクを使用するかを指定します。

### ランク数

最上位または最下位のランキングに含まれる行の数。

### 大文字小文字を区別した文字列比較

Data Integration Service で文字列をランク付けする際に大文字小文字を区別した文字列比較が使用されるかどうかを指定します。Data Integration Service で文字列の大文字小文字が無視されるようにするには、このオプションの選択を取り消します。デフォルトでは、このオプションは選択されています。

### キャッシュディレクトリ

データ統合サービスがインデックスキャッシュファイルとデータキャッシュファイルを作成するディレクトリ。このディレクトリが存在し、キャッシュファイルを格納するのに十分なディスク容量を備えていることを確認します。

キャッシュのパーティション化時のパフォーマンスを向上させるには、セミコロンで区切って複数のディレクトリを入力します。キャッシュのパーティション化により、トランスフォーメーションを処理する各パーティションに個別のキャッシュが作成されます。

デフォルトは CacheDir システムパラメータです。このプロパティには、別のシステムパラメータまたはユーザー定義のパラメータを設定できます。

### ランクデータキャッシュのサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってデータキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

### ランクインデックスキャッシュのサイズ

マッピングの実行開始時に、トランスフォーメーション用にデータ統合サービスによってインデックスキャッシュに割り当てられるメモリ量。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

関連項目：

- [「キャッシュサイズ」 \(ページ 72\)](#)

## 非ネイティブ環境でのランクトランスフォーメーション

非ネイティブ環境でのランクトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

### Blaze エンジンでのランクトランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

ランクトランスフォーメーションのデータキャッシュは、可変長を使用して、ランクトランスフォーメーションを介して渡されるバイナリデータ型と文字列データ型を格納するように最適化されます。この最適化は、8 MB までのレコードに対して有効化されます。レコードサイズが 8 MB を超える場合、可変長の最適化は無効になります。

可変長が、ランクトランスフォーメーションを介して渡されるデータのデータキャッシュ内の格納に使用される場合、ランクトランスフォーメーションは、ランタイムマッピング内にランクトランスフォーメーション前に挿入されるソートされた入力とパススルーのソータートランスフォーメーションを使用するように最適化されます。ソータートランスフォーメーションを表示するには、最適化されたマッピングを表示するか、Blaze 検証環境内の実行プランを表示します。

データキャッシュの最適化中は、ランクトランスフォーメーション用のそのデータキャッシュとインデックスキャッシュは「自動」に設定されます。ソータートランスフォーメーションのソーターキャッシュは、ランクトランスフォーメーションのデータキャッシュと同じサイズに設定されます。ソーターキャッシュを設定するには、ランクトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

### Spark エンジンでのランクトランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

#### マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。

#### データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

## ストリーミングマッピングでのランクトランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

### マッピングの検査

マッピング検証は、次の場合に失敗します。

- ランクトランスフォーメーションが、不等式ルックアップ条件を使用して設定されたルックアップトランスフォーメーションと同じストリーミングパイプライン内にある。
- ランクトランスフォーメーションがジョイナトランスフォーメーションからのアップストリームである。
- ストリーミングパイプラインに複数のランクトランスフォーメーションが含まれている。
- ストリーミングパイプラインにアグリゲータトランスフォーメーションとランクトランスフォーメーションが含まれている。

## Databricks Spark エンジンでのランクトランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### マッピング検証

マッピング検証は、次の場合に失敗します。

- 大文字小文字の区別が無効。

### データキャッシュの最適化

トランスフォーメーションのデータキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

## 第 36 章

# 読み取りトランスフォーメーション

この章では、以下の項目について説明します。

- [読み取りトランスフォーメーションの概要, 545 ページ](#)
- [読み取りトランスフォーメーションのプロパティ, 546 ページ](#)
- [リレーショナルデータオブジェクトの同期, 549 ページ](#)
- [ソースデータオブジェクトの変更, 549 ページ](#)
- [読み取りトランスフォーメーションのパラメータ, 551 ページ](#)
- [制約, 552 ページ](#)
- [読み取りトランスフォーメーションの作成, 553 ページ](#)

## 読み取りトランスフォーメーションの概要

読み取りトランスフォーメーションは、ソースからデータを読み取るパッシブトランスフォーメーションです。読み取りトランスフォーメーションは再利用不可能です。

読み取りトランスフォーメーションは、物理データオブジェクトまたは論理データオブジェクトから作成できます。PowerExchange®アダプタソースからインポートした物理データオブジェクトから読み取りトランスフォーメーションを作成する場合は、データオブジェクトから読み取りトランスフォーメーションを作成する前に読み取り操作を指定するように求めるメッセージがマッピングエディタに表示される場合があります。

トランスフォーメーションの作成で使用したデータオブジェクトのタイプに応じて、設定できる読み取りトランスフォーメーションのプロパティが異なります。例えば、リレーショナルデータオブジェクトから読み取りトランスフォーメーションを作成する場合、SQL オーバーライドの設定と制約の定義ができます。設定できるプロパティは、トランスフォーメーションのパラメータを設定したかどうかによっても異なります。

読み取りトランスフォーメーションに動的ソースを含めることができます。読み取りトランスフォーメーションのポート、メタデータ、および他のプロパティを動的に更新するように、読み取りトランスフォーメーションを設定できます。動的ソースの設定方法については、『*Informatica Developer マッピングガイド*』の「動的マッピング」の章を参照してください。

# 読み取りトランスフォーメーションのプロパティ

読み取りトランスフォーメーションを作成したら、トランスフォーメーションのプロパティを設定できます。

プロパティタブで、読み取りトランスフォーメーションのプロパティを設定します。使用できるタブは、読み取りトランスフォーメーションで表しているソースのタイプによって異なります。

以下の表で、各プロパティタブについて説明し、各タブの対象のソースタイプを示します。

| プロパティタブ        | 説明                                                                                                     | ソースタイプ                                                                                                     |
|----------------|--------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| 全般             | トランスフォーメーションのプロパティと動作を指定します。<br>リレーショナルデータオブジェクトおよびカスタマイズデータオブジェクトのソースでは、トランスフォーメーションの入力ポートをソースと同期します。 | すべて                                                                                                        |
| データオブジェクト      | トランスフォーメーションのデータソースを指定します。                                                                             | <ul style="list-style-type: none"><li>- リレーショナル</li><li>- フラットファイル</li><li>- カスタマイズデータオブジェクト</li></ul>     |
| ポート            | 関連付けられたデータオブジェクト別にポートの定義を設定します。                                                                        | <ul style="list-style-type: none"><li>- リレーショナル</li><li>- カスタマイズデータオブジェクト</li><li>- 論理データオブジェクト</li></ul>  |
| 形式             | フラットファイルデータソースの入力設定                                                                                    | フラットファイル                                                                                                   |
| クエリ            | ソースに対するクエリを指定します。                                                                                      | <ul style="list-style-type: none"><li>- リレーショナル</li><li>- カスタマイズデータオブジェクト</li><li>- 論理データオブジェクト</li></ul>  |
| ランタイム          | 実行時の動作を定義します。                                                                                          | <ul style="list-style-type: none"><li>- リレーショナル</li><li>- フラットファイル</li><li>- カスタマイズデータオブジェクト</li></ul>     |
| ソース            | ソーステーブルを選択し、ソースの詳細を設定します。                                                                              | <ul style="list-style-type: none"><li>- リレーショナル</li><li>- カスタマイズデータオブジェクト</li></ul>                        |
| データオブジェクトパラメータ | パラメータのプロパティを設定します。                                                                                     | <ul style="list-style-type: none"><li>- フラットファイル</li><li>- カスタマイズデータオブジェクト</li><li>- 論理データオブジェクト</li></ul> |
| ランタイムリンク       | トランスフォーメーション間のグループからグループへのリンクを設定し、パラメータ、リンクポリシー、またはその両方を使用して実行時にリンクするポートを決定します。                        | すべて                                                                                                        |
| 詳細             | トレースレベルと行順序を設定します。<br>リレーショナルソースの場合、実行時にターゲットテーブルを作成するまたは置き換えるようにオプションを設定します。                          | すべて                                                                                                        |

## 全般プロパティ

読み取りトランスフォーメーションの名前および説明を設定できます。また、以下のプロパティを設定することができます。

### カラムのメタデータが変わったとき

リレーショナルソースの場合に使用できます。次のいずれかのオプションを選択します。

- 出力ポートの同期。Developer tool は、モデルリポジトリに格納されている、データオブジェクトのメタデータの変更内容を使用して、読み取りトランスフォーメーションの出力ポートを更新します。
- 同期しない。読み取りトランスフォーメーションは、データオブジェクトのメタデータの変更内容を表示しません。

### 物理データオブジェクト

フラットファイルソースおよびカスタマイズソースの場合に使用できます。トランスフォーメーションの作成で使ったオブジェクト。

データオブジェクト名を選択して、そのプロパティを設定できます。

## データオブジェクトのプロパティ

［データオブジェクト］タブで、読み取りトランスフォーメーションソースを指定または変更して、リレーショナルデータオブジェクト、フラットファイルデータオブジェクト、カスタマイズデータオブジェクトのソースを動的に作成できます。

以下のプロパティを設定することができます。

### 指定元

読み取りトランスフォーメーションのソースカラムとメタデータを指定するには、以下のいずれかのオプションを選択します。

- 値。読み取りトランスフォーメーションは、関連付けられたデータオブジェクトを使用して、ソースカラムとメタデータを指定します。
- パラメータ。読み取りトランスフォーメーションは、パラメータを使用して、ソースカラムとメタデータを指定します。

### データオブジェクト

既存のデータオブジェクトから読み取りトランスフォーメーションを作成した場合、このフィールドにはオブジェクトの名前が表示されます。読み取りトランスフォーメーションに関連付けるデータオブジェクトを変更するには、**【参照】** をクリックします。

### 実行時に、データソースからデータオブジェクトのカラムを取得します

このオプションを有効にした場合、データ統合サービスは、メタデータとデータ定義の変更内容をソーステーブルから読み取ってトランスフォーメーションに取り込みます。

## クエリのプロパティ

リレーショナルリソースまたはカスタマイズデータオブジェクトに対する SQL クエリを設定します。

**【クエリ】** タブでプロパティを設定する場合、単純プロパティまたは詳細プロパティのどちらを設定するかを選択します。

**単純プロパティ**のビューでは、Define Distinct 文としてデフォルトの SQL 文を設定し、文のヒントと、結合、フィルタ、ソートの各条件を編集します。

**詳細プロパティ**のビューでは、カスタム SQL クエリを定義できます。関連付けられたデータオブジェクト内のカラムまたはパラメータを選択したり、データオブジェクトを表す新しいパラメータを作成したりできます。

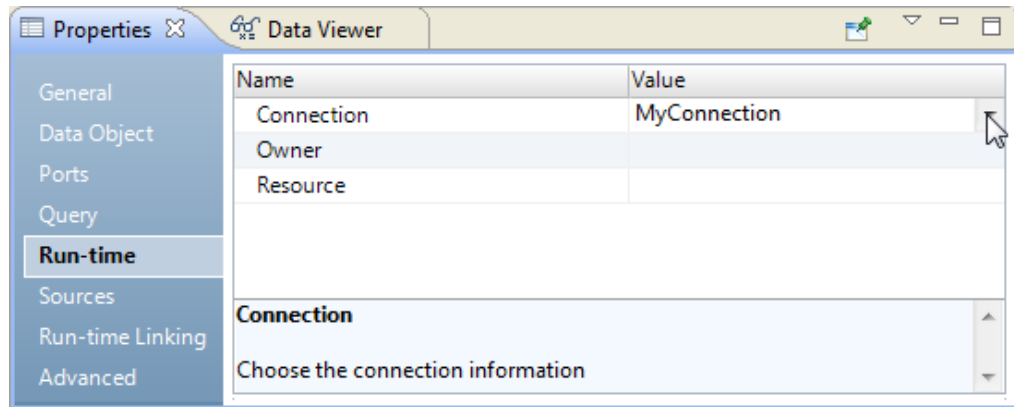
## ランタイムプロパティ

【ランタイム】タブで、以下の読み取りトランスフォーメーションのプロパティを設定できます。

### 接続

リレーショナルソースの場合に使用できます。トランスフォーメーションで使用する接続。接続を変更するには、フィールドの右側をクリックします。

次の図は、クリックするドロップダウンボタンの場所を示しています。



## ソースのプロパティ

リレーショナルソースおよびカスタマイズデータオブジェクトのソースの詳細を設定します。リポジトリにリレーショナルデータオブジェクトをインポートした後、その定義を変更できます。ポートの追加と削除や、プライマリキーの定義ができるほか、リポジトリ内の複数のリレーショナルデータオブジェクト間のリレーションを設定できます。

【ソース】タブでは、以下の設定を行うことができます。

### すべてのソース

【追加】ボタンまたは【削除】ボタンを押して、トランスフォーメーションのソースを追加または削除します。

### 【全般】タブ

選択したソースの名前および説明を変更します。他の詳細を変更するには、ソース名をクリックします。

### 【キー】タブ

リソースカラムをキーとして指定します。

### 【リレーション】タブ

複数のリレーショナルソース間のリレーションを追加または削除します。

## 詳細プロパティ

データ統合サービスでの読み取りトランスフォーメーションのデータの処理方法を指定するには、詳細プロパティを設定します。

【詳細】タブで、以下のプロパティを設定します。

### トレースレベル

マッピングログファイル内の詳細度を制御します。



## PreSQL

ソースを読み取る前にソースデータベースに対してデータ統合サービスが実行する SQL コマンド。

Developer tool では、SQL は検証されません。

## PostSQL

ターゲットに書き込んだ後にソースデータベースに対してデータ統合サービスが実行する SQL コマンド。

Developer tool では、SQL は検証されません。

## 制約

テーブルレベルの参照の整合性制約の SQL 文。リレーショナルソースのみに適用されます。

# リレーショナルデータオブジェクトの同期

物理データオブジェクトのソースが変更されたときに、同期させることができます。物理データオブジェクトの同期化では、選択したソースからオブジェクトメタデータが再インポートされます。

すべての物理データオブジェクトを同期できます。リレーショナルデータオブジェクトまたはカスタマイズデータオブジェクトを同期する際、Developer tool で定義したキーリレーションを保持または上書きすることができます。

マッピングオブジェクトの同期方法を以下から選択します。

### リレーショナルソースを同期する。

物理データオブジェクトを同期するには、[Object Explorer] ビューで右クリックして、[同期] を選択します。

### トランスフォーメーションポートを物理データオブジェクトと同期する。

トランスフォーメーションの [データオブジェクト] タブで、[実行時に、データソースからデータオブジェクトのカラムを取得します] オプションを選択します。

実行時に、データ統合サービスはデータソースからメタデータとデータ定義の変更を取得しモデルリポジトリ内のデータオブジェクト定義を更新します。

データ統合サービスがメタデータとデータ定義の変更を取得する方法をプレビューするには、解決済みパラメータとのマッピングを表示します。

### メタデータの変更時にポートを同期する

トランスフォーメーションの [全般] タブで、ポートを同期するためのオプションを選択します。このオプションの正確なラベルは、設定するトランスフォーメーションのタイプによって異なります。例えば、読み取りトランスフォーメーションの場合、このオプションは [メタデータの変更時に出力ポートを変更します] です。

マッピングが実行されると、データ統合サービスはトランスフォーメーション内のカラムメタデータをデータソース内のメタデータと同期します。

# ソースデータオブジェクトの変更

読み取りトランスフォーメーションは、モデルリポジトリ内の物理データオブジェクトまたは論理データオブジェクトに基づきます。読み取りトランスフォーメーションを設定するときに、データオブジェクトを変更で

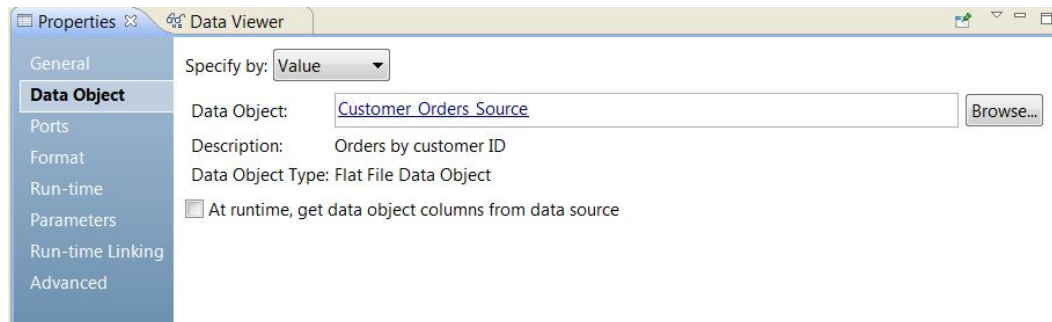
きます。データオブジェクトをパラメータ化すると、実行時にデータオブジェクトを変更することができます。例えば、本番のマッピング実行用のソースファイルではなく別のソースファイルを使用してマッピングをテストすることができます。

物理データオブジェクトからトランスフォーメーションを作成する場合、トランスフォーメーションプロパティの【データオブジェクト】タブに、データオブジェクトに関する情報が表示されます。データオブジェクト名をクリックすると、モデルリポジトリにある物理データオブジェクトの定義を表示することができます。

モデルリポジトリ内の別の物理データオブジェクトを参照することで、トランスフォーメーションのデータオブジェクトを変更できます。データオブジェクトを変更する場合、トランスフォーメーションは、選択されたデータオブジェクトのランタイムプロパティと詳細プロパティを使用します。

データソースの変更内容に基づいて、データオブジェクトの構造を実行時に更新できます。データソースは、データオブジェクトで表される物理ファイルまたはデータベーステーブルです。データ統合サービスでデータカラムをデータソースから取得することを有効にした場合、データ統合サービスはデータソースの構造を調べます。データ統合サービスは、データソースに基づいてトランスフォーメーションインスタンス内のデータオブジェクトポートを更新します。データ統合サービスは、モデルリポジトリ内にある物理データオブジェクトの定義を変更しません。

次の図は、【データオブジェクト】タブを示しています。



【データオブジェクト】タブには以下のフィールドがあります。

#### 指定元

特定のデータオブジェクト名を入力するには、【値】を選択します。データオブジェクトをパラメータ化するには、【パラメータ】を選択します。

#### データオブジェクト

モデルリポジトリ内のデータオブジェクトの名前。【データオブジェクト】リンクをクリックすると、リポジトリにあるデータオブジェクトの定義を開くことができます。モデルリポジトリ内の別のデータオブジェクトを参照することもできます。

#### 説明

リポジトリ内のデータオブジェクトの説明。読み取り専用。

#### データオブジェクトタイプ

フラットファイルデータオブジェクト、リレーショナルデータオブジェクト、カスタマイズデータオブジェクトなど、データオブジェクトのタイプを示します。

#### 実行時に、データソースからデータオブジェクトのカラムを取得します

データ統合サービスは実行時に、データオブジェクトが参照するテーブルのデータファイルからメタデータとデータ定義の変更を取得し、トランスフォーメーションインスタンスのデータオブジェクトの構造を更新します。

データ統合サービスが実行時にメタデータとデータ定義の変更を取得する方法をプレビューするには、解決済みパラメータとのマッピングを表示します。

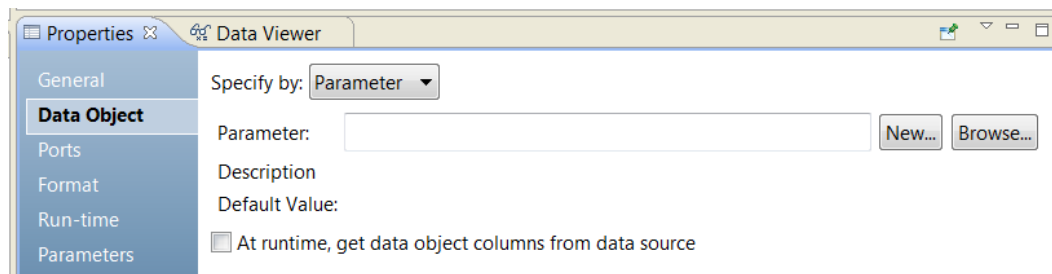
## 読み取りトランスフォーメーションのパラメータ化

読み取りトランスフォーメーションをパラメータ化すると、実行時にデータオブジェクトを変更することができます。

データオブジェクトをパラメータ化するには、**【データオブジェクト】** タブの **【指定元:]** で **【パラメータ】** を選択します。**【データオブジェクト】** タブのプロパティが変化します。

データオブジェクトをパラメータ化するには、リソースタイプパラメータを作成するか、作成済みのリソースパラメータを参照します。パラメータのデフォルト値は、モデルリポジトリ内の物理データオブジェクトの名前です。デフォルトのパラメータ値を作成する場合、リポジトリ内のデータオブジェクトのリストから物理データオブジェクト名を選択します。

次の図は、パラメータでデータオブジェクトを指定した場合の **【データオブジェクト】** タブを示しています。



**【データオブジェクト】** タブの **【指定元: パラメータ】** には、以下のオプションがあります。

### パラメータ

データオブジェクトとして設定したリソースパラメータの名前。読み取り専用。

### 説明

パラメータの説明。読み取り専用。

### 新規

リソースパラメータを作成します。パラメータのデフォルト値としてモデルリポジトリ内のデータオブジェクトを参照し、選択します。

### 参照

リソースパラメータを参照し、パラメータを選択します。

### デフォルト値

データオブジェクトに設定したリソースパラメータのデフォルト値。デフォルト値は、モデルリポジトリ内の物理データオブジェクト名とオブジェクトへのパスです。読み取り専用。

## 読み取りトランスフォーメーションのパラメータ

読み取りトランスフォーメーションの一部のプロパティをパラメータ化できます。また、読み取りトランスフォーメーションの作成元である再利用可能な物理データオブジェクトの一部のプロパティをパラメータ化できます。

物理データオブジェクトを作成する場合、読み取りプロパティと書き込みプロパティを設定します。物理データオブジェクトの読み取りプロパティに対して設定したパラメータは、データオブジェクトをマッピングに追加すると、読み取りトランスフォーメーションの **【データオブジェクトパラメータ】** タブに表示されます。

物理データオブジェクトの以下の読み取りプロパティに対してパラメータを設定できます。

- 制御ファイルディレクトリ
- 制御ファイル名
- デフォルトのスケール
- 区切り文字
- フラットファイルの区切り文字
- マージファイルディレクトリ
- ソースファイル名
- ソースファイルディレクトリ

物理データオブジェクトをマッピングに追加すると、読み取りトランスフォーメーションの【データオブジェクトパラメータ】タブにパラメータを表示できます。このパラメータをマッピングパラメータとして公開し、パラメータ値を実行時に上書きできます。

**注:** パラメータ化ソース内にユーザー定義パラメータをネストすることはできません。ソースデータオブジェクトがパラメータ化されていると、ユーザー定義のパラメータをマッピングパラメータとして公開してパラメータ値を実行時に上書きすることはできません。代わりに、マッピングではデフォルト値が使用されます。

読み取りトランスフォーメーションでは、以下のマッピングパラメータを設定できます。

- 接続（リレーショナル）
- データオブジェクト
- リンクの解決順序
- リソース名（リレーショナル）
- テーブル所有者名（リレーショナル）

これらのパラメータは、マッピングの【データオブジェクトパラメータ】タブに表示できます。

## 制約

制約は、データ行上の値が合致する必要がある条件式です。

制約を設定する場合は、各データ行に関して TRUE に評価される式を入力します。

データ統合サービスは、リレーショナルソース、論理データオブジェクト、物理データオブジェクト、または仮想テーブルから制約を読み取ることができます。再利用可能な物理データオブジェクトに制約を設定するには、カスタマイズされたデータオブジェクトを作成します。

制約を読み取るときにデータ統合サービスは、適用されている最適化方式に基づき、データ行に関して TRUE に評価されない行を削除することがあります。

制約を設定する前に、その制約で設定される条件をソースデータが満たすことを確認する必要があります。例えば、「AGE < 70」の行を含んでいると思われる AGE カラムがソースデータベースに存在するとします。このソースデータベースに、「AGE < 70」という式で制約を設定できます。データ統合サービスは、ソースデータベースから制約「AGE < 70」に合致するレコードを読み取ります。「AGE >= 70」のレコードを読み取る場合、データ統合サービスはそれらを削除することがあります。

データベースでは、SQL コマンドを使用して、データベースに接続する際のデータベース環境に制約を設定できます。データ統合サービスは、データベースに接続するたびに、接続環境 SQL を実行します。

# 読み取りトランスフォーメーションの作成

読み取りトランスフォーメーションを作成する場合、トランスフォーメーションの作成元のリソースに基づいて以下のいずれかの方法を選択します。

## モデルリポジトリ内のデータオブジェクトからトランスフォーメーションを作成する。

以下の手順を実行し、モデルリポジトリ内のデータオブジェクトから読み取りトランスフォーメーションを作成します。

1. エディタでマッピングを開きます。
2. **【オブジェクトエクスプローラ】** から、データオブジェクトをエディタビューにドラッグします。
3. **【読み取り】** を選択し、**【OK】** をクリックします。  
マッピング内の読み取りトランスフォーメーションには、データオブジェクトのポートとプロパティが含まれます。

## マッピングエディタを使用してトランスフォーメーションを作成します。

読み取りトランスフォーメーションの詳細な設定を行う場合、この方法を使用します。また、パラメータに基づいて読み取りトランスフォーメーションを作成する場合も、この方法を使用します。

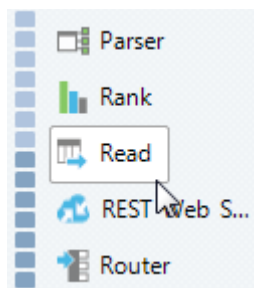
マッピングエディタで読み取りトランスフォーメーションを作成するには、[「マッピングエディタでの読み取りトランスフォーメーションの作成」](#) (ページ 553)を参照してください。

# マッピングエディタでの読み取りトランスフォーメーションの作成

読み取りトランスフォーメーションを作成して、マッピング内のデータのソースとカラムのメタデータおよびプロパティを表すことができます。

以下の手順を実行します。

1. 以下のいずれかの方法を洗濯して、読み取りトランスフォーメーションを作成します。
  - マッピングエディタ内で右クリックし、**【トランスフォーメーションの追加】** を選択します。  
**【トランスフォーメーションの追加】** ダイアログボックスが開きます。  
読み取りトランスフォーメーションを選択し、**【次へ】** をクリックします。
  - マッピングパレット内を下にスクロールして、読み取りトランスフォーメーションアイコンを見つけてダブルクリックします。  
次の図は、読み取りトランスフォーメーションアイコンを示しています。



**【新しい読み取りトランスフォーメーション】** ダイアログボックスが開きます。

2. フラットファイル、リレーショナルリソース、またはカスタマイズデータオブジェクトをソースとして使用するには、以下の手順を実行します。
  - a. データオブジェクトタイプとして **【物理データオブジェクト】** を選択します。

- b. **【参照】** をクリックして、フラットファイル、リレーショナルリソース、またはカスタマイズデータオブジェクトを選択します。  
**【データオブジェクトの選択】** ウィンドウが開きます。
  - c. データオブジェクトを選択し、**【OK】** をクリックします。
  - d. 必要に応じて、実行時にソースからデータオブジェクトカラムを取得するようにトランスフォーメーションを設定します。**【実行時に、データソースからデータオブジェクトのカラムを取得します】** を選択します。  
データ統合サービスは、マッピングの実行時に読み取りトランスフォーメーションのカラムメタデータを更新します。
3. パラメータをソースとして使用するには、以下の手順を実行します。
  - a. **【パラメータを使用して作成】** を選択します。
  - b. **【新規作成】** をクリックして新しいパラメータを作成するか、**【参照】** をクリックして既存のパラメータを選択します。
  - c. パラメータを選択して、**【OK】** をクリックします。
  - d. 必要に応じて、実行時にソースからデータオブジェクトカラムを取得するようにトランスフォーメーションを設定します。**【実行時に、データソースからデータオブジェクトのカラムを取得します】** を選択します。  
データ統合サービスは、マッピングの実行時に読み取りトランスフォーメーションのカラムメタデータを更新します。
4. 論理データオブジェクトをソースとして使用するには、以下の手順を実行します。
  - a. データオブジェクトタイプとして **【論理データオブジェクト】** を選択します。
  - b. **【参照】** をクリックしてデータオブジェクトを選択した後、**【OK】** をクリックします。
5. 必要に応じて、読み取りトランスフォーメーションの名前を入力します。
6. **【完了】** をクリックします。

## 第 37 章

# リレーショナルから階層型へのトランスフォーメーション

この章では、以下の項目について説明します。

- [リレーショナルから階層型へのトランスフォーメーションの概要, 555 ページ](#)
- [例 - リレーショナルから階層型へのトランスフォーメーション, 556 ページ](#)
- [入力リレーショナルポートと \[概要\] ビュー, 558 ページ](#)
- [リレーショナルから階層型へのトランスフォーメーションのポート, 558 ページ](#)
- [スキーマ参照, 559 ページ](#)
- [リレーショナルから階層型へのトランスフォーメーションの開発, 559 ページ](#)

## リレーショナルから階層型へのトランスフォーメーションの概要

リレーショナルから階層型へのトランスフォーメーションでは、リレーショナル入力を処理して階層出力に変換します。リレーショナルから階層型へのトランスフォーメーションでは、入力ポートからリレーショナル入力を読み取り、そのデータをトランスフォーメーション出力ポートで階層出力に変換します。リレーショナル入力を階層出力に変換するには、スキーマオブジェクトを使用して階層構造を定義します。

リレーショナルから階層型へのトランスフォーメーションウィザードを使用して、リレーショナル入力ポートを反映した階層構造を作成できます。トランスフォーメーション [概要] ビューで階層出力ポートのマッピングを表示できます。

トランスフォーメーションを作成すると、マッピングで階層出力ポートから別のトランスフォーメーションへデータが渡せます。

**注:** リレーショナルから階層型へのトランスフォーメーションは、1 つの.xsd ファイルで最大 10,000 個のスキーマ要素を処理できます。10,000 個を超える要素を処理するには、データを複数のファイルに分割します。

リレーショナルモデルでは、各テーブルスキーマによってプライマリキーというカラムが識別され、各行が一意に識別されます。テーブル内の各行と別のテーブルの行の間のリレーションは、外部キーによって識別します。ウィザードではトランスフォーメーション作成時にキーが生成されます。自動生成トランスフォーメーションを変更し、ポートの追加、編集、削除ができます。

1 つの入力リレーショナルポートは、階層内の 1 つのノードにリンクできます。階層内の関連する要素または属性から入力のリレーショナルグループにプライマリキーをリンクします。プライマリキーは、リレーショナルテーブルの各行を識別します。

階層内の関連する要素または属性から入力のリレーショナルグループに外部キーをリンクします。リレーショナル入力の外部キーは、1つのテーブルのカラムで、別のテーブルのプライマリキーをポイントします。

入力リレーショナルポートと階層ノードのデータ型には互換性が必要です。

## 例 - リレーショナルから階層型へのトランスフォーメーション

Electronics Superstore 社の財務部門は、従業員の給与を処理します。リレーショナルデータベースに保存された従業員データを給与システムで処理可能な階層形式に変換する必要があります。

マッピングで、リレーショナルから階層型へのトランスフォーメーションを使用し、従業員名、従業員 ID、従業員住所、従業員銀行口座などの従業員詳細を入力し、使用可能な階層形式の詳細を出力する必要があります。

リレーショナル入力では、Bank\_ID 要素は従業員テーブルのプライマリキーであり、銀行テーブルの外部キーです。

| Employee_ID | Last_Name | First_Name | Address                | Bank_ID | Bank_Account |
|-------------|-----------|------------|------------------------|---------|--------------|
| 9173327437  | Sandrine  | Jacques    | 74 Mobile Avenue       | 74845   | 8723487234   |
| 9174562342  | Race      | Tom        | 266 Crouse St.         | 9234734 | 45324734     |
| 8484526471  | Jones     | Charles    | 3815 LaValle Boulevard | 389236  | 234638437    |
| 7023847265  | Smith     | Delilah    | 193 Short Drive        | 74845   | 8723463432   |
| 9174596725  | Frederick | George     | 17 Serenity Road       | 9234734 | 6342636699   |

| Bank_ID | Bank_Name          | SWIFT_Code |
|---------|--------------------|------------|
| 74845   | National Bank      | 9173327    |
| 9234734 | International Bank | 9174562    |
| 389236  | Star National Bank | 8484526    |

階層形式の給与出力では、要素はテーブルから結合されます。

```
<banks>
 <bank name="National Bank" SWIFT="9173327">
 <account id="8723487234">
 <employee_id>9173327437</employee_id>
 <fname>Sandrine</fname>
 <lname>Jacques</lname>
 <address>74 Mobile Avenue</address>
 </account>
 <account id="8723463432">
 <employee_id>9082745558</employee_id>
 <fname>Delilah</fname>
 <lname>Smith</lname>
 <address>193 Short Drive</address>
 </account>
 </bank>
 <bank name="International Bank" SWIFT="9174562">
 <accounts>
 <account id="45324734">
 <employee_id>5534398889</employee_id>
 <fname>Race</fname>
 <lname>Tom</lname>
 <address>266 Crouse St.</address>
 </account>
 <account id="6342636699">
 <employee_id>9174596725</employee_id>
 <fname>Frederick</fname>
 <lname>George</lname>
 <address>17 Serenity Road</address>
 </account>
 </accounts>
 </bank>
</banks>
```

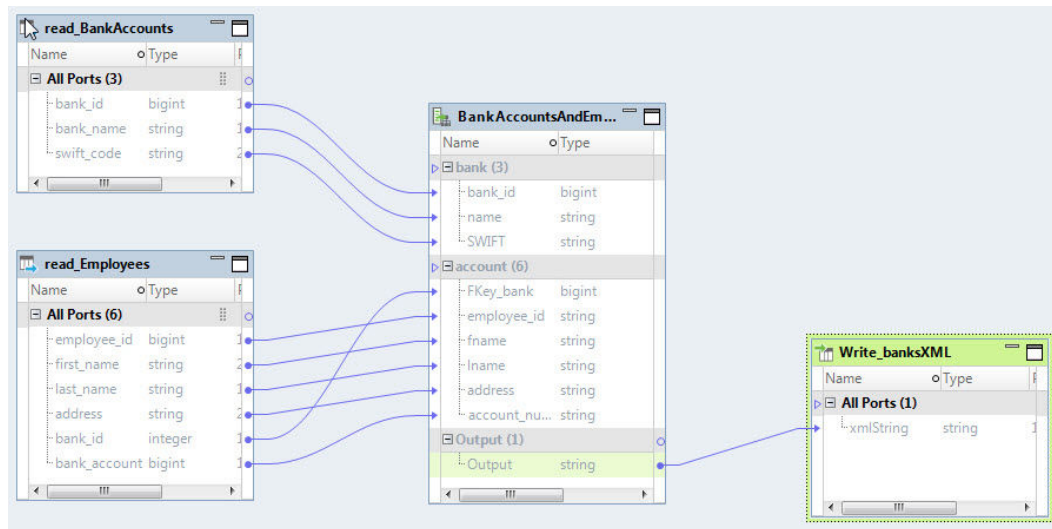


```

</bank>
<bank name="Star National Bank" SWIFT="8484526">
 <accounts>
 <account id="234638437">
 <employee_id>8484526471</employee_id>
 <fname>Jones</fname>
 <lname>Charles</lname>
 <address>3815 LaValle Boulevard</address>
 </account>
 </accounts>
</bank>
</banks>

```

次の図は、この例のマッピングを示します。



マッピングには次のオブジェクトが含まれます。

**Read\_BankAccounts**

銀行データを含むソース。

**Read\_Employees**

従業員データを含むソース。

**BankAccountsAndEmployees\_To\_PaymentsSystemXML**

従業員および銀行口座情報を含むリレーショナル入力を給与システムがコンシュームする XML 形式に変換する、リレーショナルから階層型へのトランスフォーメーション。

**Write\_BanksXML**

マッピングの実行のたびに変換済みデータを保存するファイルのターゲットパス。

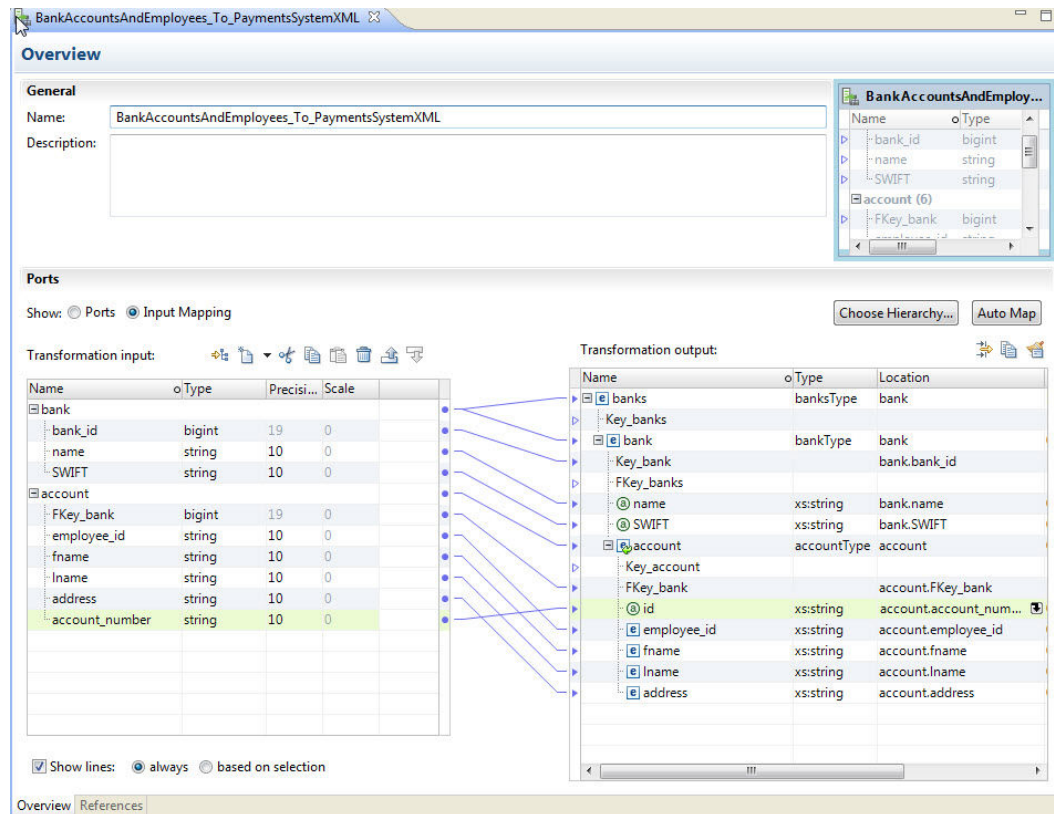
マッピングでは、**Read\_BankAccount** と **Read\_Employees** ファイルを使用して、リレーショナル入力を行います。**BankAccountsAndEmployees\_To\_PaymentsSystemXML** トランスフォーメーションでデータが処理され、変換されます。その出力は **Write\_BanksXML** フラットファイルにリストされたターゲットパスに保存されます。

# 入力リレーショナルポートと【概要】ビュー

リレーショナルデータを階層データに変換するには、ウィザードでリレーショナル入力ポートを反映する階層構造を作成します。【概要】ビューを使用して、リレーショナルポートを階層ポートにリンクできます。

リレーショナル入力と階層出力間のリンクを表示するには、【概要】ビューを使用します。【入力マッピング】を選択します。【概要】ビューに【ポート】パネルが表示されます。

以下の図は、【ポート】パネルを示しています。



ポートパネルの左側に、リレーショナル要素とグループを表示する【トランスフォーメーション入力】領域があります。右側には、階層スキーマノードを表示する【トランスフォーメーション出力】領域があります。

【トランスフォーメーション入力】領域でポートを作成し、リレーショナル要素をスキーマノードにリンクできます。スキーマのノードから【トランスフォーメーション入力】領域の空のフィールドにポインタをドラッグしてポートを作成することもできます。リレーショナルポートをスキーマノードに接続すると、Developer tool はそれらの間にリンクを表示します。

## リレーショナルから階層型へのトランスフォーメーションのポート

リレーショナルから階層型へのトランスフォーメーションのポートは、トランスフォーメーションの【概要】ビューで定義します。

リレーショナルから階層型へのトランスフォーメーションでは、バッファからリレーショナルデータ出力を読み込みます。出力ポートはバッファに階層データを返します。

## スキーマ参照

リレーショナルから階層型へのトランスフォーメーションには、トランスフォーメーションで出力階層を定義する階層スキーマが必要です。トランスフォーメーションでスキーマを使用するには、スキーマ参照を定義します。

トランスフォーメーションのスキーマ参照は、トランスフォーメーションの【参照】ビューで定義します。

リレーショナルから階層型へのトランスフォーメーションでは、モデルリポジトリのスキーマオブジェクトが参照されます。このスキーマオブジェクトは、トランスフォーメーションを作成する前のリポジトリに存在することができます。

スキーマは別のスキーマを参照できます。【参照】ビューには、リレーショナルから階層型へのトランスフォーメーションが参照する各スキーマの名前空間とプレフィックスが表示されます。

## リレーショナルから階層型へのトランスフォーメーションの開発

新規トランスフォーメーションウィザードを使用してリレーショナルから階層型へのトランスフォーメーションを自動生成します。スキーマまたは階層サンプルファイルを選択して、出力階層を定義します。

### リレーショナルから階層型へのトランスフォーメーションの作成

1. Developer tool で、【ファイル】 > 【新規】 > 【トランスフォーメーション】 をクリックします。
2. リレーショナルから階層型へのトランスフォーメーションを選択し、【次へ】 をクリックします。
3. トランスフォーメーションの名前を入力し、モデルリポジトリでトランスフォーメーションを配置する場所を参照して、【次へ】 をクリックします。
4. スキーマを選択するには、次の方法のいずれかを選択します。
  - モデルリポジトリのスキーマを使用して出力階層を定義するには、【スキーマオブジェクト】 フィールド付近で、リポジトリのスキーマファイルを参照して選択します。
  - スキーマファイルをインポートするには、【新規スキーマオブジェクトの作成】 をクリックします。【新規スキーマオブジェクト】 ウィンドウで、スキーマファイルを参照し選択するか、サンプル階層ファイルからのスキーマ作成を選択できます。
5. 出力階層のルートを選択します。【階層のルート】 ダイアログボックスで、出力階層ファイルのルート要素となっている、スキーマ内の要素を選択します。ルートオブジェクトの選択をしやすいするため、サンプル階層ファイルを追加できます。サンプルファイルを追加するには、【サンプルファイル】 フィールド付近でファイルシステムからファイルを参照し選択します。
6. 【完了】 をクリックします。

ウィザードにより、リポジトリにトランスフォーメーションが作成されます。

### ポートの作成

【概要】 ビューでポートを設定します。

1. マッピングを表示するには、【概要】 ビューの【ポート】 領域で、【入力マッピング】 を選択します。
2. 入力ポートタイプ、精度、およびスケールを選択します。

3. **【ポート】** グリッドでツリーを展開します。左の**【トランスフォーメーション入力】** パネルにリレーショナル入力が表示され、右には**【トランスフォーメーション出力】** パネルに階層出力の期待値が表示されます。
4. ノードをルートとして定義するには、**【階層の選択】** をクリックします。  
Developer tool では、**【トランスフォーメーション入力】** 領域にルートレベルのノードとルートレベルより下のノードだけが表示されます。
5. ポートを階層ノードと関連付ける行を表示するには、**【行の表示】** をクリックします。すべての関連付けられた行を表示するか、選択したポートの行だけを表示するかを選択します。
6. **【トランスフォーメーション入力】** 領域に入力グループまたは入力ポートを追加するには、次のいずれかの方法を使用します。
  - **【トランスフォーメーション出力】** 領域の単純要素または複合要素を、**【トランスフォーメーション入力】** 領域の空のカラムにドラッグします。グループノードの場合は、Developer tool によってポートのないリレーショナルグループが追加されます。
  - リレーショナルグループを追加するには、行を選択して右クリックし、**【新規作成】** > **【グループ】** を選択します。
  - リレーショナルポートを追加するには、右クリックして **【新規作成】** > **【フィールド】** を選択します。
7. ポートの場所の階層ノード設定をクリアするには、次のいずれかの方法を使用します。
  - **【トランスフォーメーション出力】** 領域で 1 つ以上のノードを選択して右クリックし、**【クリア】** を選択します。
  - リレーショナル入力ポートと階層ノードを結ぶ線を 1 つ以上選択してから、右クリックして **【削除】** を選択します。
8. 出力ポートを階層形式で表示するには、**【階層で表示】** をクリックします。各子グループは親グループの下に表示されます。

## 第 38 章

# REST Web サービスコンシューマ トランスフォーメーション

この章では、以下の項目について説明します。

- [REST Web サービスコンシューマトランスフォーメーションの概要, 561 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションの設定, 563 ページ](#)
- [HTTP メソッド, 565 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションのポート, 568 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションの入力マッピング, 570 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションの出力マッピング, 573 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションの詳細プロパティ, 575 ページ](#)
- [REST Web サービスコンシューマトランスフォーメーションの作成, 576 ページ](#)
- [配列を含む JSON 応答メッセージの解析, 577 ページ](#)

## REST Web サービスコンシューマトランスフォーメーションの概要

REST Web サービスコンシューマトランスフォーメーションは、データアクセスまたはデータ変換する Web サービスクライアントとして REST Web サービスに接続するアクティブなトランスフォーメーションです。REST Web サービスコンシューマトランスフォーメーションを使用して、REST Web サービスに接続します。REST Web サービスコンシューマトランスフォーメーションは、リクエストを REST Web サービスに送ることと、REST Web サービスから応答を受け取ることができます。

REST Web サービスコンシューマトランスフォーメーションは、トランスフォーメーションまたは HTTP 接続で定義する URL を通じて、Web サービスに接続します。HTTPS 接続を使用することもできます。REST Web サービスコンシューマトランスフォーメーションでは、TLS 1.2、TLS 1.1、TLS 1.0 を使用できます。

REST Web サービスには、Web サービスがサポートするアクションごとに HTTP メソッドが 1 つあります。データ統合サービスが REST Web サービスに接続されている場合、データの Get、Post、Put、Delete を実行するためのリクエストが送信できます。このリクエストの対象は、個別のリソースでもリソースの集合でもかまいません。データ統合サービスがリクエストメッセージを送った後に、Web サービスから応答メッセージを受け取ります。

リクエストメッセージと応答メッセージには、階層を形成できる要素を使った XML または JSON のデータが入っています。リクエストメッセージまたは応答メッセージに複数回出現する要素が入っている場合、要素のグ

ループが XML または JSON の階層内にレベルを形成します。あるレベルが別のレベルの入れ子になっている場合、グループは関連しています。

REST Web サービスコンシューマトランスフォーメーションでは、メソッド入力とメソッド出力でリクエストメッセージと応答メッセージの構造を定義します。メソッド入力とメソッド出力にはマッピングが含まれ、メッセージ要素を入力ポートと出力ポートにマップする方法を定義しています。

REST Web サービスコンシューマトランスフォーメーションはプロキシサーバーをサポートします。REST Web サービスコンシューマトランスフォーメーションを使って Microsoft SharePoint アプリケーションに接続することもできます。

## 例

オンラインストアは、製品データベースに対しリソースを定義します。データベースは各製品を部品番号で識別します。

Web サービスクライアントは REST Web サービスを通じて製品の詳細にアクセスします。Web サービスは次の URL を使用します。

<http://www.HypoStores.com/products/ProductDetails>

具体的な製品についての詳細（説明や単価など）を取得し、詳細をマッピング内のダウンストリームのトランスフォーメーションに渡す必要があります。製品についての詳細を取得して別のトランスフォーメーションに渡すための REST Web サービスコンシューマトランスフォーメーションを作成します。

次のテーブルに、ユーザーが設定するトランスフォーメーションの詳細を示します。

トランスフォーメーションの詳細	値
HTTP メソッド	取得
ベース URL	<a href="http://www.HypoStores.com/products/ProductDetails">http://www.HypoStores.com/products/ProductDetails</a>
入力引数ポート	Part_No
出力ポート	Description、Unit_Price
メソッド出力	<応答メッセージの構造。>

メソッド出力は、出力マッピングを含み、応答メッセージ内の要素を出力ポートにマップする方法を定義しています。

データ統合サービスがリクエストを Web サービスに送るときに、引数ポート内の値をベース URL に付加します。例えば、部品 0716 についての詳細を取得するために、データ統合サービスは次の URL を使用します。

[http://www.HypoStores.com/products/ProductDetails?Part\\_No=0716](http://www.HypoStores.com/products/ProductDetails?Part_No=0716)

データ統合サービスは応答を受け取ると、応答メッセージ内の製品の説明と単価を出力ポートのデータに変換します。

Part\_No をパラメータとして渡して、マッピングを実行している最中に値を置き換えることもできます。

## REST Web サービスコンシューマトランスフォーメーションの処理

REST Web サービスコンシューマトランスフォーメーションは、メソッド入力と入力ポート内のデータに基づいてリクエストメッセージを作成します。応答メッセージ内の要素をメソッド出力に基づいて出力ポートのデータに変換します。

REST Web サービスコンシューマトランスフォーメーションの入力ポートには、マッピング内のアップストリームのトランスフォーメーションからのリレーショナルデータがあります。データ統合サービスは、メソッド入力を使用して、データを入力ポートからリクエストメッセージ内の要素に変換します。

Web サービスに接続するときに、データ統合サービスは、トランスフォーメーションプロパティまたは HTTP 接続で設定したベース URL を読み込みます。URL ポートまたは引数ポートからの値をベース URL に付加することによって、Get、Post、Put、Delete を実行する必要があるリソースを識別します。

データ統合サービスが応答を受け取ると、応答メッセージ内のデータをトランスフォーメーションの出力ポートに渡します。データ統合サービスは、メソッド出力を設定する方法に基づいてデータを渡します。出力ポートにはリレーショナルデータが入っています。データ統合サービスは、出力ポート内のデータをマッピング内のダウストリームのトランスフォーメーションに、またはターゲットに送ります。

## REST Web サービスコンシューマトランスフォーメーションの設定

REST Web サービスコンシューマトランスフォーメーションを作成する場合は、HTTP メソッドを選択し、メソッド入力とメソッド出力を定義します。Get メソッドを選択する場合は、メソッド入力を定義しません。

HTTP リクエストメッセージ内の入力要素は、入力ポートにマップします。HTTP 応答メッセージ内の出力要素は、出力ポートにマップします。Developer ツールは、最初のレベルの要素にポートを作成します。

トランスフォーメーションを設定するときは、以下のタスクをすべて行ってください。

1. HTTP メソッドを選択します。
2. リクエストメッセージと応答メッセージのヘッダと本文の要素を代表するポートを設定します。
3. 入力マッピングを設定します。
4. 出力マッピングを設定します。
5. 詳細プロパティ（Web サービスのベース URL や接続など）を設定します。

REST Web サービスに認証が必要な場合、HTTP 接続オブジェクトを作成します。

## メッセージの設定

データ統合サービスがリクエストメッセージを生成し、メソッド入力とメソッド出力に基づいて、さらに REST Web サービスコンシューマトランスフォーメーションで設定するポートに基づいて、応答メッセージを解釈します。

入力ポートは、リクエストメッセージのさまざまな部分を表します。取得または変更するリソースを特定する入力ポートを追加できます。リクエストメッセージ内の HTTP ヘッダ、クッキー情報、および要素を表す入力ポートを追加することもできます。

出力ポートは、ダウストリームのトランスフォーメーションやマッピング内のターゲットに送信する応答メッセージの要素を表します。応答メッセージ内の HTTP ヘッダ、クッキー情報、応答コード、および要素を表す出力ポートを追加できます。



## リソースの識別

HTTP リクエストでリソースを識別できるように、データ統合サービスがベース URL に特定の入力ポートの値を付加します。ベース URL は、HTTP 接続またはトランスフォーメーションプロパティで定義します。URL または引数ポートを使用して特定のリソースを識別します。

Web サービスが一意の文字列を使ってリソースを識別するときに、URL ポートを使用します。

例えば、HypoStores という REST Web サービスは、次の URL を通じて部品番号で部品を識別します。

```
http://www.HypoStores.com/products/ProductDetails/<Part_No>
```

部品を識別するには、次のトランスフォーメーションの詳細を定義します。

1. ベース URL を、次の URL に設定します。

```
http://www.HypoStores.com/products/ProductDetails
```

2. URL ポートを定義し、その URL ポートを介して部品番号をトランスフォーメーションに渡します。

マッピングが部品番号 500 を URL ポートに渡す場合、データ統合サービスはリクエストメッセージで次の URL を使用します。

```
http://www.HypoStores.com/products/ProductDetails/500
```

Web サービスがリソースの場所を引数で識別する場合に、引数ポートを使用します。

例えば、部品番号を HypoStores という REST Web サービスに「Part\_No」という引数で渡す必要があるとします。

部品を識別するには、次のトランスフォーメーションの詳細を定義します。

1. ベース URL を、次の URL に設定します。

```
http://www.HypoStores.com/products/ProductDetails
```

2. 引数名「Part\_No」を使って引数ポートを作成し、その引数ポートから部品番号をトランスフォーメーションに渡します。

マッピングが部品番号 600 を引数ポートに渡す場合、データ統合サービスはリクエストメッセージで次の URL を使用します。

```
http://www.HypoStores.com/products/ProductDetails?Part_No=600
```

複数の引数を定義するには複数の引数ポートを作成します。データ統合サービスでは、アンパーサンド文字 (&) で引数と引数を分割します。

例えば、従業員の詳細を REST Web サービスから取得し、従業員の名と姓をそれぞれ「First\_Name」と「Last\_Name」という引数を使って渡します。「First\_Name」と「Last\_Name」という引数名の引数ポートを作成します。マッピングが「John Smith」という名前をトランスフォーメーションに渡す場合、データ統合サービスはリクエストメッセージで次のように URL を使用します。

```
http://www.HypoStores.com/employees/EmpDetails?First_Name=John&Last_Name=Smith
```

URL または引数ポートを指定しない場合、データ統合サービスは、HTTP 接続またはトランスフォーメーションプロパティのベース URL を使用してリソースを識別します。HTTP 接続内のベース URL は、トランスフォーメーション内のベース URL より優先されます。



# HTTP メソッド

REST Web サービスコンシューマトランスフォーメーションを作成するときは、データ統合サービスがリクエストメッセージで使用する HTTP メソッドを選択します。トランスフォーメーションを作成した後は、HTTP メソッドを変更することはできません。

以下の HTTP メソッドのうちいずれかを使用するようにトランスフォーメーションを設定します。

## Get

Web サービスからリソースを個別にあるいはまとめて取得します。例えば、複数製品を表形式で取得することも、1 製品についての情報を取得することもできます。

## Post

Web サービスにデータを送信します。Post メソッドを使用すると、リソースを個別に、またはまとめて作成できます。例えば、新しい店の取引の詳細を追加できます。

## Put

リソースを個別に、あるいはまとめて置き換えます。データが存在しない場合、Put メソッドでデータを送信します。例えば、発送先となる顧客の住所を更新できます。

## Delete

リソースを個別に、あるいはまとめて削除します。例えば、すでに退職している社員のレコードを削除できます。

## HTTP Get メソッド

データ統合サービスは、HTTP Get メソッドを使用して REST Web サービスからデータを取得します。Get メソッドを使用してリソースを個別に、またはまとめて取得します。

Get メソッドを使用するように REST Web サービスコンシューマトランスフォーメーションを設定する場合、入力ポート、メソッド出力、および出力ポートを設定します。メソッド入力はありません。

### 例

HypoStores という製品のデータベースから部品番号 500 に関する説明と価格を取得する必要があるとします。この Web サービスでは、次の URL を使用して部品を特定しています。

`http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>`

次の URL を入力します。

`http://www.HypoStores.com/products/ProductDetails`

定義する可能性のある入力ポートを下表に示します。

ポートタイプ	引数名	入力値
引数	Part_No	500

定義する可能性のある出力ポートを下表に示します。

ポートタイプ	ポート名	戻り値
出力	Part_Desc	…<desc>ACME ボールペン、12-pk、黒、0.7 mm</desc>…
出力	Price_USD	…<price>9.89</price>…

## HTTP Post メソッド

データ統合サービスは、HTTP Post メソッドを使用して REST Web サービスにデータを送信します。Web サービスは、Post メソッドが実行する実際の機能を決定します。リソースを個別またはまとめて作成する際に、Post メソッドを使用する場合があります。

Post メソッドを使用するように REST Web サービスコンシューマトランスフォーメーションを設定する場合、入力ポート、メソッド入力、メソッド出力、および出力ポートを設定します。

### 例

HypoStores という製品のデータベースに新しい部品 501 を掲載する必要があるとします。Web サービスは部品 501 に、次の URL を使用しています。

<http://www.HypoStores.com/products/ProductDetails/501>

次の URL を入力します。

<http://www.HypoStores.com/products/ProductDetails>

定義する可能性のある入力ポートを下表に示します。

ポートタイプ	ポート名	入力値
URL	URL_Part_No	501
入力	Part_Desc	ACME ボールペン、12-pk、黒、0.5 mm
入力	Price_USD	9.89

定義する可能性のある出力ポートを下表に示します。

ポートタイプ	ポート名	戻り値
出力	応答	<Web サービスが戻す応答>

## HTTP Put メソッド

データ統合サービスは、HTTP Put メソッドを使用して REST Web サービスによるデータの更新を行います。Put メソッドを使用してリソースを個別に、またはまとめて更新します。データが存在しない場合、データ統合サービスがリソースを個別にあるいはまとめて作成します。

Put メソッドを使用するように REST Web サービスコンシューマトランスフォーメーションを設定する場合、入力ポート、メソッド入力、メソッド出力、および出力ポートを設定します。

## 例

HypoStores という製品のデータベースで、部品 501 の単価を更新する必要があるとします。Web サービスは部品 501 に、次の URL を使用しています。

`http://www.HypoStores.com/products/ProductDetails/501`

次の URL を入力します。

`http://www.HypoStores.com/products/ProductDetails`

定義する可能性のある入力ポートを下表に示します。

ポートタイプ	ポート名	入力値
URL	URL_Part_No	501
入力	Price_USD	9.99

定義する可能性のある出力ポートを下表に示します。

ポートタイプ	ポート名	戻り値
出力	応答	<Web サービスが戻す応答>

## HTTP Delete メソッド

データ統合サービスは、HTTP Delete メソッドを使用して REST Web サービスによるデータ削除を行います。Delete メソッドを使用してリソースを個別に、またはまとめて削除します。

Delete メソッドを使用するように REST Web サービスコンシューマトランスフォーメーションを設定する場合、入力ポート、メソッド入力、メソッド出力、および出力ポートを設定します。

## 例

HypoStores という製品のデータベースから部品番号 502 を削除する必要があるとします。この Web サービスでは、次の URL を使用して部品を特定しています。

`http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>`

次の URL を入力します。

`http://www.HypoStores.com/products/ProductDetails`

定義する可能性のある入力ポートを下表に示します。

ポートタイプ	引数名	入力値
引数	Part_No	502

定義する可能性のある出力ポートを下表に示します。

ポートタイプ	ポート名	戻り値
出力	応答	<Web サービスが戻す応答>

# REST Web サービスコンシューマトランスフォーメーションのポート

REST Web サービスコンシューマトランスフォーメーションに、複数の入力ポートと複数の出力ポートを設定することができます。XML または JSON の階層の構造に基づいてポートをグループに分けて作成します。

トランスフォーメーションのポートを表示するときに、XML または JSON 階層を確認する必要がない場合はポートを表示します。ポートを表示しているときは、グループの定義、ポートの定義、およびメソッド入力/出力から入力/出力ポートへの要素のマッピングを行うことができます。

REST Web サービスコンシューマトランスフォーメーションには、複数の入力グループと複数の出力グループを設定できます。ポートを作成するときは、グループを作成してそのグループにポートを追加します。ポートは、XML または JSON 内の入力階層または出力階層の構造に基づいたグループ階層で定義します。キーを追加して、子グループと親グループを関連付けます。

プライマリキーは、最下位のグループを除く階層内のすべてのグループに必要です。外部キーは、ルートグループを除く階層内のすべてのグループに必要です。

このトランスフォーメーションには、RequestInput という名前のルート入力グループがあります。プライマリキーをルートの入力グループに追加する必要があります。キーは、string 型、bigint 型、integer 型のいずれかである必要があります。ルートの入力グループに任意のポートをパススルーポートとして設定できます。

要素をポートにマップするには、**[場所]** カラム内のフィールドをクリックし、**[場所の選択]** ダイアログボックスで階層を展開します。その後で階層から要素を選択します。

## 入力ポート

入力ポートは、アップストリームのトランスフォーメーションからのデータ、または Web サービスに渡す必要のあるソースを表します。複数の入力ポートを設定できます。各入力ポートは、リクエストメッセージ内の要素に割り付けられています。

入力ポートを追加するには、入力グループを選択してから **[新規]** ボタンの横の矢印をクリックし、**[フィールド]** を選択します。

## 出力ポート

出力ポートは、ダウンストリームのトランスフォーメーションまたはターゲットに渡す予定の応答メッセージ内の要素を表します。複数の出力ポートを設定できます。各出力ポートが応答メッセージ内の要素にマップされます。

出力ポートを追加するには、出力グループを選択し、**[新規]** ボタンの横の矢印をクリックして、**[フィールド]** を選択します。

## パススルーポート

パススルーポートは、データを変更せずにトランスフォーメーションを介してデータを渡します。ルートの入力グループに任意のポートをパススルーポートとして設定できます。

パススルーポートを追加するには、ポートをルートの入力グループに追加します。ポートを右クリックして、**[マップ]** を選択します。

## 引数ポート

リソースの URL が引数を取るときに引数ポートでリソースを識別することができます。引数ポートは入力グループのルートに追加します。

引数ポートにはポート名と引数名があります。ポート名に許可されていない文字が引数名に含まれている場合は、ポート名と異なる引数名を入力してください。例えば、引数「Cust-ID」を Web サービスに渡そうとしても、ポート名の中のダッシュ (-) をデータ統合サービスが許可しません。この場合は「Cust-ID」を引数名として入力し、「CustID」をポート名として入力してください。

データ統合サービスは、各引数ポートの引数名と値を、「名前=値」の形でペアにしてベース URL に付加します。引数ポートは複数設定することができます。その場合はデータ統合サービスが、アンパーサンド (&) 文字を使ってリクエスト内の複数の引数を分割します。

以下に例を示します。

```
http://www.HypoStores.com/customers/CustDetails?Last_Name=Jones&First_Name=Mary
```

トランスフォーメーション内で引数ポートと URL ポートを定義する場合は、データ統合サービスは URL ポートの値をベース URL に付加し、その後に引数名と値を続けます。

引数ポートを追加するには、ルートの入力グループを右クリックし、**[新規]** > **[引数ポート]** を選択します。引数名とポート名を入力します。

## URL ポート

URL ポートでは、静的 URL を通じてリソースを識別できます。リソースを識別するために、データ統合サービスが URL ポートの値をベース URL に付加します。

以下に例を示します。

```
http://www.HypoStores.com/products/ProductDetails/<URL_port_value>
```

ルートの入力グループに URL ポートを追加します。

複数の URL ポートを設定することができます。データ統合サービスは、各 URL ポート内の値をスラッシュ文字 (/) で分割します。トランスフォーメーションの URL ポートと引数ポートを定義する場合、データ統合サービスはベース URL の後で引数名と値の前に URL ポート値を付加します

URL ポートを追加するには、ルートの入力グループを右クリックして、**[新規]** > **[URL ポート]** を選択します。

## HTTP ヘッダポート

HTTP ヘッダポートは、要求メッセージの中の HTTP のヘッダを表します。複数の HTTP ヘッダポートを設定できます。

リクエストでヘッダ情報を Web サービスに渡すには、ポートをルートの入力グループに追加します。ルートの入力グループに 1 つの HTTP ヘッダポートを設定できます。HTTP ヘッダをルート入力グループに追加する場合、それをパススルーポートとして設定できます。

HTTP ヘッダポートにはポート名と HTTP ヘッダ名があります。ポート名には許可されていない文字が HTTP ヘッダ名に含まれている場合は、ポート名と異なる HTTP ヘッダ名を入力してください。例えば、ヘッダ名「Content-Type」を Web サービスに渡そうとしても、ポート名の中のダッシュ (-) をデータ統合サービスが許可しません。「Content-Type」を HTTP ヘッダ名として入力し、「ContentType」をポート名に入力します。

HTTP ヘッダポートを追加するには、ルートの入力グループを右クリックし、**[新規]** > **[HTTP ヘッダ]** を設定します。ヘッダ名とポート名を入力します。

## クッキーポート

クッキー認証を使用するよう REST Web サービスコンシューマトランスフォーメーションを設定することができます。リモートの Web サーバーが、クッキーに基づいて、Web サービスコンシューマのユーザーを追跡します。マッピングが Web サービスを繰り返し呼び出す場合のパフォーマンスが向上します。

リクエストでクッキー情報を Web サービスに渡すには、ポートをルートの入力グループに追加します。ルートの入力グループに 1 つのクッキーポートを設定できます。クッキーポートをルートの入力グループに追加する場合、それをパススルーポートとして設定できます。

クッキー情報を応答から抽出するには、クッキーポートを出力グループに追加します。各出力グループに 1 つのクッキーポートを設定できます。

クッキーポートを Web サービス要求メッセージに射影する場合、Web サービスプロバイダは応答メッセージでクッキー値を返します。クッキー値は、マッピング内のダウンストリームにある別のトランスフォーメーションに渡すことも、ファイル内に保存することもできます。クッキー値をファイルに保存する場合、そのクッキー値を REST Web サービスコンシューマトランスフォーメーションへの入力に設定できます。

クッキーポートを追加するには、ルートの入力グループを右クリックして、**[新規]** > **[その他のポート]** を選択します。その次に、**[クッキーポート]** **[OK]** をクリックします。

## 出力 XML ポート

出力 XML ポートは Web サービスからの応答を表します。出力 XML ポートは文字列ポートです。

出力 XML ポートを出力グループに追加します。出力ポートごとに出力 XML ポートを 1 つ設定することができます。

ルートの入力グループを右クリックし、**[新規]** > **[その他のポート]** を選択します。その次に **[出力 XML]** を選択し、**[OK]** をクリックします。

## 応答コードポート

応答コードのポートは、Web サービスからの HTTP 応答コードを代表しています。応答コードポートは整数のポートです。

応答コードポートを出力グループに追加します。出力グループごとに応答コードポートを 1 つ設定することができます。

応答コードのポートを追加するには、出力グループを選択して、ルートの入力グループを右クリックして、**[新規]** > **[その他のポート]** を選択します。それから **[応答コード]** を選択して **[OK]** をクリックします。

# REST Web サービスコンシューマトランスフォーメーションの入力マッピング

トランスフォーメーションのポートを確認するときは、入力マッピングを表示してメソッド入力階層を表示します。入力マッピングを表示しているときは、入力グループの定義、入力ポートの定義、およびメソッド入力要素への入力ポートのマッピングを行うことができます。

入力マッピングは以下の領域で構成されています。

### ポート

**[ポート]** 領域では、トランスフォーメーションの入力グループと入力ポートを作成します。

## メソッド入力

【メソッド入力】領域には、REST Web サービスコンシューマトランスフォーメーションから Web サービスに送られるリクエストメッセージ内の要素が表示されます。スキーマオブジェクトを使用してトランスフォーメーションを作成する場合、スキーマオブジェクトがメソッド入力の階層を定義します。

入力ポートを作成したら、その入力ポートを【ポート】領域から【メソッド入力】領域の要素にマップします。入力ポートをメソッド入力内の要素にマップすると、【メソッド入力】領域の【場所】カラムにポートの場所が表示されます。

入力階層の第 1 レベルをマップするよう選択すると、Developer ツールによってメソッド入力の第 1 レベルの要素が入力ポートにマップされます。また、Developer ツールはマッピングを実行するためのポートも作成します。階層の第 1 レベルに複数回出現する親要素が含まれ、その子要素が 1 つまたは複数回出現する場合、Developer ツールは階層の第 1 レベルをマップしません。

入力ポートをメソッド入力内の要素に結ぶ線を表示するように選択できます。

## 入力ポートを要素にマップするためのルールとガイドライン

入力ポートをメソッド入力階層内の要素にマップする場合には、以下のルールを確認します。

- 階層内の 1 個の要素に対し 1 個の入力ポートをマップできます。同じポートを、階層内の任意の数のキーにマップすることができます。
- 入力ポートおよび要素にデータ型の互換性があることが必要です。
- 1 つの入力グループ内のポートを、メソッド入力内の複数の階層レベルにマップできます。
- メソッド入力のキーに入力ポートをマップする必要があります。キーにマップするポートのデータ型は、string、integer、または bigint であることが必要です。メソッド入力内の、リクエストメッセージに含める階層レベルより上位にある全レベルのキーに、データをマップします。マップするレベルと、その上位にある全レベルの外部キーを含めます。

**注:** メソッド入力階層の最下位レベルだけをマップする場合は、入力ポートをキーにマップする必要はありません。

- RequestInput のルート要素を、メソッド入力定義の Rest\_Consumer\_input グループの子要素にマップする必要があります。
- string 型、bigint 型、または integer 型の複数の入力ポートを【メソッド入力】領域のキーにマップして、複合キーを作成できます。複合キーの【場所】フィールドをクリックして、入力ポートの順序を変更したり、いずれかのポートを削除したりすることができます。
- Web サービスが JSON 文書を生成する場合、xmlRoot が応答階層の中の最初のノードであることを確認します。xmlRoot が Web サービスの JSON 応答を使った最初のノードではない場合、NULL 値が表示される可能性があります。

## メソッド入力への入力ポートのマッピング

メソッド入力要素が表示されているときに、入力グループと入力ポートを定義でき、さらに入力ポートをメソッド入力要素にマップすることができます。

1. REST Web サービスコンシューマトランスフォーメーションを開きます。
2. 【ポート】ビューで、入力マッピングを表示します。
3. ルート入力グループのプライマリキーを定義します。

4. 次のいずれかの方法を使用して、入力グループまたは入力ポートを【ポート】領域に追加します。

方法	説明
要素をドラッグ。	【メソッド入力】領域からグループまたは子要素を【ポート】領域の空のカラムにドラッグします。【ポート】領域にグループをドラッグする場合、Developer ツールはポートを持たないグループを追加します。
グループまたはポートを手動で追加。	グループを追加するには、【新規】ボタンの隣の矢印をクリックし、【グループ】をクリックします。ポートを追加するには、【新規】ボタンの隣の矢印をクリックし、【フィールド】をクリックします。
別のトランスフォーメーションからポートをドラッグ。	エディタ内で、ポートを別のトランスフォーメーションから REST Web サービスコンシューマトランスフォーメーションにドラッグします。
ポートをコピー。	別のトランスフォーメーションからポートを選択し、【ポート】領域へコピーします。ポートをコピーするときは、キーボードショートカットを使用するか、Developer ツールの【コピー】ボタンと【貼り付け】ボタンを使用します。
【階層の第 1 レベルのマッピング】を選択。	Developer ツールが、メソッド入力の第 1 レベルの要素を入力ポートと入力グループにマッピングします。また、Developer ツールはマッピング実行のための入力ポートと入力グループも作成します。

5. ポートを手動で作成するか、別のトランスフォーメーションからポートをコピーするときは、【メソッド入力】領域の【場所】カラムをクリックし、リストからポートを選択します。
6. 入力ポートを複合キーにマッピングするには、以下のいずれかの方法を使用します。

方法	説明
入力ポートをドラッグする。	2 つ以上の入力ポートを選択してから、メソッド入力階層内のキーにドラッグします。
【場所の選択】ダイアログボックスから入力ポートを選択する。	メソッド入力階層でキーの【場所】カラムをクリックし、入力ポートを選択します。

7. ポートの場所をクリアするには、以下のいずれかの方法を使用します。

方法	説明
【クリア】をクリックする。	【メソッド入力】領域で要素を 1 つか複数選択し、【クリア】をクリックします。
ポートと要素を結ぶ線を削除。	メソッド入力領域で入力ポートと要素を結ぶ線を 1 つか複数選択してから、キーボードの Delete キーを押します。



# REST Web サービスコンシューマトランスフォーメーションの出力マッピング

トランスフォーメーションのポートを確認するときは、出力マッピングを表示してメソッド出力階層を確認します。出力マッピングを表示している場合は、出力グループの定義、出力ポートの定義、出力ポートへのメソッド出力要素のマッピングを行うことができます。

出力マッピングは以下の領域で構成されています。

## メソッド出力

**【操作出力】** 領域には、Web サービスから REST Web サービスコンシューマトランスフォーメーションに返される応答メッセージ内の要素が表示されます。スキーマオブジェクトを使用してトランスフォーメーションを作成する場合、スキーマオブジェクトがメソッド出力の階層を定義します。

## ポート

**【ポート】** 領域では、トランスフォーメーションの出力グループと出力ポートを作成します。

出力ポートを作成したら、要素を **【メソッド出力】** 領域から **【ポート】** 領域のポートにマップします。要素をメソッド出力から出力ポートにマップすると、要素の場所が **【ポート】** 領域の **【場所】** カラムに表示されます。

出力階層の第 1 レベルをマップするよう選択すると、Developer ツールによってメソッド出力の第 1 レベルの要素が出力ポートにマップされます。また、Developer ツールはマッピングを実行するためのポートも作成します。階層の第 1 レベルに複数回出現する親要素が含まれ、その子要素が 1 つまたは複数回出現する場合、Developer ツールは階層の第 1 レベルをマップしません。

出力ポートを階層構造で表示するように選択することもできます。それぞれの子グループが親グループの下に表示されます。メソッド出力内の要素を出力ポートと結ぶ線を表示するように選択できます。

関連付けられたスキーマオブジェクトがリポジトリから削除された場合、Developer ツールはメソッド要素の場所を出力マッピングに保持します。出力マッピングを表示すると、**【ポート】** 領域で出力ポートの **【場所】** カラムにメソッド要素の場所が表示されます。別のスキーマをトランスフォーメーションに関連付けると、Developer ツールがそれぞれの場所が有効であるかどうかを調べます。場所が有効でなくなると、Developer ツールは出力マッピングの **【ポート】** 領域にあるメソッド要素の場所をクリアします。

## 要素を出力ポートにマップするためのルールとガイドライン

メソッド出力階層内の要素を出力ポートにマップする際は、以下のルールを確認してください。

- メソッド出力要素および出力ポートにはデータ型の互換性が必要です。
- 1 つの要素を同じグループ内の複数の出力ポートにマップすることはできません。
- パススルーポートを除く各出力ポートに、有効な場所を指定する必要があります。
- 複数出現の子要素を空の出力ポートにドラッグした場合、グループを他の出力グループに関連付ける必要があります。グループを選択すると、Developer ツールがグループ間に関連付けるキーを作成します。
- 複数回出現する要素を親要素が含まれているグループにドラッグする場合、含める子要素の出現回数を設定することができます。または、親グループを、トランスフォーメーション出力内の複数出現子グループで置き替えることもできます。
- Web サービスが JSON 文書を生成する場合、xmlRoot が応答階層の中の最初のノードであることを確認します。xmlRoot が Web サービスの JSON 応答を使った最初のノードではない場合、NULL 値が出力ポートに現れる可能性があります。

# ビューのカスタマイズのオプション

メソッド出力階層を、クッキーポート、パススルーポート、およびキーが **【メソッド出力】** 領域内に表示されるように変更することができます。要素の順序を定義するグループ化構造を表示することもできます。

**【メソッド出力】** 領域で、**【ビューのカスタマイズ】** をクリックします。以下のいずれかのオプションを有効にします。

Sequence、Choice、および All

要素定義が sequence か choice か all かを、線で表示します。

sequence グループ内の要素は、階層内に定義された順序で並んでいる必要があります。

choice グループ内の少なくとも 1 つの要素が、応答メッセージに表示される必要があります。

全グループのすべての要素を応答メッセージ内に含める必要があります。

## キー

**【メソッド出力】** 領域にキーを表示します。**【メソッド出力】** 領域には、各グループのキーが含まれていますが、**【ポート】** 領域内の出力ポートに、キーを追加できます。

## パススルーポート

**【メソッド出力】** 領域には、パススルーポートが表示されます。パススルーポートとは、トランスフォーメーションを通過させるだけで変更せずにデータを渡すポートのことです。パススルーポートをメソッド出力から REST Web サービスコンシューマトランスフォーメーションの出力グループのいずれかに射影することができます。パススルーポートは、データを 1 回のみ受け取るため、応答メッセージ内のルートレベルにあります。

# 出力ポートへのメソッド出力のマッピング

トランスフォーメーションの出力マッピングが表示されているときに、出力グループと出力ポートが定義でき、出力ポートにメソッド出力要素をマップすることができます。

- REST Web サービスコンシューマトランスフォーメーションを開きます。
- 【ポート】** ビューで、出力マッピングを表示します。
- 以下のいずれかの方法を使用して、出力グループまたは出力ポートを **【ポート】** 領域に追加します。

方法	説明
要素をドラッグ。	<b>【メソッド出力】</b> 領域のグループまたは子要素を、 <b>【ポート】</b> 領域の空のカラムにドラッグします。 <b>【ポート】</b> 領域にグループをドラッグする場合、Developer ツールはポートを持たないグループを追加します。
グループまたはポートを手動で追加。	グループを追加するには、 <b>【新規】</b> ボタンの隣の矢印をクリックし、 <b>【グループ】</b> をクリックします。ポートを追加するには、 <b>【新規】</b> ボタンの隣の矢印をクリックし、 <b>【フィールド】</b> をクリックします。
別のトランスフォーメーションからポートをドラッグ。	エディタ内で、ポートを別のトランスフォーメーションから REST Web サービスコンシューマトランスフォーメーションにドラッグします。
ポートをコピー。	別のトランスフォーメーションからポートを選択し、 <b>【ポート】</b> 領域へコピーします。ポートをコピーするときは、キーボードショートカットを使用するか、Developer ツールの <b>【コピー】</b> ボタンと <b>【貼り付け】</b> ボタンを使用します。

4. ポートを手動で作成するか、別のトランスフォーメーションからポートをコピーするときは、**【ポート】** 領域の **【場所】** カラムをクリックし、リストから要素を選択します。
5. ポートの場所をクリアするには、以下のいずれかの方法を使用します。

方法	説明
<b>【クリア】</b> をクリック。	<b>【ポート】</b> 領域でポートを 1 つまたは複数選択し、 <b>【クリア】</b> をクリックします。
要素とポートを結ぶ線を削除。	メソッド出力の要素と出力ポートを結ぶ線を 1 つまたは複数選択し、キーボードの Delete キーを押します。

## REST Web サービスコンシューマトランスフォーメーションの詳細プロパティ

REST Web サービスコンシューマトランスフォーメーションのデータがデータ統合サービスでどのように処理されるかを決定するためのプロパティを設定します。

**【詳細】** タブで、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

### 接続

Web サービスに接続する HTTP 接続オブジェクトを特定します。Developer ツールで HTTP 接続を作成および編集します。HTTP 接続を設定するときは、ベース URL、Web サービスに必要なセキュリティのタイプ、および接続タイムアウトの期間を設定します。

REST Web サービスコンシューマトランスフォーメーションは、URL を使用して Web サービスに接続します。URL はトランスフォーメーションプロパティまたは HTTP 接続で定義できます。

以下のような状況で HTTP 接続を設定してください。

- URL の入力ポートを使用しない。
- Web サービスには HTTP 認証または SSL 証明書が必要です。
- デフォルトの接続タイムアウト期間を変更したほうがよい。

### XML スキーマ検証

実行時間に応答メッセージを検証します。**【無効な XML でのエラー】** または **【検証なし】** が選択できます。

### ソート済み入力

入力データを必ずしもすべて処理しなくても、データ統合サービスが出力を生成できるようにします。入力データが XML 入力階層内のキーでソートされる場合に、ソート済み入力を有効にします。

### URL

REST Web サービスのベース URL。HTTP 接続内のベース URL は、この値より優先されます。

### フォーマット

Web サービスの応答の形式。Web サービスの応答によって、**XML** または **JSON** を選択します。

# REST Web サービスコンシューマトランスフォーメーションの作成

REST Web サービスコンシューマトランスフォーメーションは、再利用可能なものと再利用不可能なものが作成できます。再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。再利用不可能なトランスフォーメーションは、単一のマッピングで使用されます。

REST Web サービスコンシューマトランスフォーメーションを作成する場合、要素と XML 階層を手動で定義するか、スキーマオブジェクトから要素と階層をインポートすることができます。このスキーマオブジェクトは、XML ファイルの場合もあれば、テキストファイルの場合もあります。

## REST Web サービスコンシューマトランスフォーメーションの作成

REST Web サービスコンシューマトランスフォーメーションを作成するときは、メソッドを選択し、選択したメソッドに基づいてメソッドの入力と出力を定義します。

1. REST Web サービスコンシューマトランスフォーメーションを作成するには、以下のメソッドのいずれかを使用します。

メソッド	説明
再利用可能	[Object Explorer] ビューで、プロジェクトまたはフォルダを選択します。【ファイル】 > 【新規】 > 【トランスフォーメーション】 をクリックします。REST Web サービスコンシューマトランスフォーメーションを選択し、【次へ】 をクリックします。
再利用不可	マッピングまたはマップレットで、REST Web サービスコンシューマフォーメーションをトランスフォーメーションパレットからマッピングまたはマップレットエディタにドラッグします。

2. トランスフォーメーション名を入力し、場所と HTTP メソッドを選択します。
3. 【次へ】 をクリックします。
4. メソッド入力を定義するために、以下のメソッドのいずれかを使用します。

メソッド	説明
空として作成	XML の要素と階層を手動で定義します。
スキーマオブジェクト内の要素から作成	XML の要素と階層をスキーマオブジェクトからインポートします。

【メソッド入力の定義】 領域には、トランスフォーメーションの入力グループと入力ポートが表示されます。【マッピング入力】 領域には、要求メッセージ階層が表示されます。

5. 入力グループと入力ポートを定義し、入力要素に入力ポートをマップします。
6. 【次へ】 をクリックします。
7. メソッド出力を定義するには、【空として作成】 または【スキーマオブジェクト内の要素から作成】 を選択します。

【メソッド出力の定義】 領域には、トランスフォーメーションの入力グループと入力ポートが表示されます。【マッピング出力】 領域には、要求メッセージ階層が表示されます。

8. 出力グループと出力ポートを定義し、要素を出力ポートにマップします。
9. 【完了】 をクリックします。

# 配列を含む JSON 応答メッセージの解析

要素が複合型の子要素であり、この要素の最大オカレンスが無制限である場合、スキーマは無効です。JSON パーサーは、要素の複数のインスタンスを抽出することを制限します。

複合型の子要素の最大オカレンスは、スキーマ内の複合型に対して 0 または 1 にし、順序インジケータを choice にする必要があります。最大オカレンスを 1 に変更してスキーマを検証する場合、一度に抽出できる要素のインスタンスは 1 つです。

スキーマ内の複合型の choice 順序インジケータでは、最大オカレンスを無制限として使用できます。

## JSON 応答メッセージの例

次のスキーマは、複合型要素 xmlRoot に最大オカレンスが無制限である要素名 Likes が含まれています。

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="xmlRoot">
 <xs:complexType>
 <xs:all>
 <xs:element type="xs:byte" name="Age"/>
 <xs:element type="xs:string" name="FirstName"/>
 <xs:element type="xs:string" name="Likes" maxOccurs="unbounded" minOccurs="0"/>
 <xs:element type="xs:string" name="FamilyName"/>
 </xs:all>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

JSON 応答は、次の形式で変更できます。

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="xmlRoot">
 <xs:complexType>
 <xs:choice maxOccurs="unbounded">
 <xs:element type="xs:byte" name="Age"/>
 <xs:element type="xs:string" name="FirstName"/>
 <xs:element type="xs:string" name="Likes"/>
 <xs:element type="xs:string" name="FamilyName"/>
 </xs:choice>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

<xs:choice maxOccurs="unbounded">を使用すると、コンテンツを任意の順序で 1 回以上繰り返すことができます。

## 応答メッセージ内の名前なし配列

REST Web サービスコンシューマ変換では、要求メッセージではなく応答メッセージのみで名前なし配列をサポートします。メソッド出力定義で定義された名前なし配列スキーマを解析するには、複合型または単純型の配列要素の親要素が xmlRoot という名前である必要があります。










REST Web サービスコンシューマトランスフォーメーションでは、最大発生数が「バインドなし」に設定された xmlRoot 要素の子要素として xmlRoot を定義し、名前なし配列の要素を xmlRoot 要素の子要素として定義する必要があります。

次の図は、名前なし配列の定義済みメソッド出力を示しています。

☐ Ports   ☐ Method input   ☒ Method output

Show: ☒ Method output definition   ☐ Output mapping

Method output definition

	Name	Type	Min...	Ma...	Description	>>
	 Rest_Consume...	(Rest_Cons...				
	  xmlRoot	(xmlRoot)	1	1		
	  xmlRoot	(xmlRoot)	1	Un...		
	  emp	xs:string	1	1		
	  empid	xs:string	1	1		

## 第 39 章

# ルータトランスフォーメーション

この章では、以下の項目について説明します。

- [ルータトランスフォーメーションの概要, 579 ページ](#)
- [動的マッピングでのルータトランスフォーメーション, 580 ページ](#)
- [グループに関する作業, 581 ページ](#)
- [ポートに関する作業, 584 ページ](#)
- [マッピング内のルータトランスフォーメーションの接続, 585 ページ](#)
- [ルータトランスフォーメーションの詳細プロパティ, 585 ページ](#)
- [非ネイティブ環境でのルータトランスフォーメーション, 586 ページ](#)

## ルータトランスフォーメーションの概要

ルータトランスフォーメーションは、1 つ以上の条件に基づいてデータを複数の出力グループにルーティングするアクティブなトランスフォーメーションです。マッピングで出力グループを異なるトランスフォーメーションまたは異なるターゲットにルーティングします。

ルータトランスフォーメーションはフィルタトランスフォーメーションに似ており、両方とも条件を使用してデータをテストします。ただし、フィルタトランスフォーメーションは 1 つの条件についてデータをテストし、条件を満たさない他のデータ行は削除します。ルータトランスフォーメーションは 1 つ以上の条件についてデータをテストし、どの条件も満たさないデータ行をデフォルト出力グループにルーティングできます。

複数の条件に基づいて同じ入力データをテストする必要がある場合、同じ作業を実行する複数のフィルタトランスフォーメーションを作成する代わりに、マッピング内でルータトランスフォーメーションを使用します。ルータトランスフォーメーションの方が効率的です。例えば、3 つの条件に基づいてデータをテストするために、3 つのフィルタトランスフォーメーションの代わりに 1 つのルータトランスフォーメーションを使用できます。マッピング内でルータトランスフォーメーションを使用すると、データ統合サービスは入力データを一度処理します。マッピング内で複数のフィルタトランスフォーメーションを使用すると、データ統合サービスは各トランスフォーメーションについて入力データを処理します。

ルータトランスフォーメーションは、入力および出力グループ、入力および出力ポート、グループフィルタの条件、Developer ツールで設定される詳細プロパティによって構成されています。

次の図に、ルータトランスフォーメーションおよびそのコンポーネントの例を示します。

名前	タイプ	長さ/精度
<b>Input (4)</b>		
COUNTRY	string	10
CUSTOMER_NO	string	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>Default (4)</b>		
COUNTRY	string	10
CUSTOMER_NO	string	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>France (4)</b>		
COUNTRY	string	10
CUSTOMER_NO	string	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>Japan (4)</b>		
COUNTRY	string	10
CUSTOMER_NO	string	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>USA (4)</b>		
COUNTRY	string	10
CUSTOMER_NO	string	10
FIRSTNAME	string	10
LASTNAME	string	10

1. 入力グループ
2. 入力ポート
3. デフォルト出力グループ
4. ユーザー定義出力グループ

## 動的マッピングでのルータトランスフォーメーション

動的マッピングでルータトランスフォーメーションを使用できます。トランスフォーメーションに動的ポートを設定して、生成されたポートをグループフィルタ条件で参照できます。

グループフィルタ条件で動的ポートを使用する場合は、動的ポートに複数の生成されたポートを含めることができます。グループフィルタ条件は、生成された各ポートが含まれるように展開されます。生成された各ポートは、式に対して有効なタイプである必要があります。

グループフィルタ条件はパラメータ化できます。式タイプパラメータを使用してフィルタを指定します。



# グループに関する作業

ルーフトランスフォーメーションには、以下の種類のグループがあります。

- 入力
- 出力

## 入力グループ

ランクトランスフォーメーションには入力グループが 1 つ含まれます。入力グループは、トランスフォーメーションに追加するすべての入力ポートが含まれます。

## 出力グループ

ランクトランスフォーメーションには以下の種類の出力グループが含まれます。

### ユーザ定義グループ

ユーザー定義グループを作成して、到着するデータに応じて条件をテストします。ユーザー定義グループは、出力ポートおよびグループフィルタ条件で構成されています。Developer ツールの **【グループ】** ビューで、ユーザー定義グループを作成および編集できます。指定したい各条件について、それぞれ 1 つのユーザー定義グループを作成します。

Data Integration Service は条件を使用して、入力データの各行を評価します。各ユーザー定義グループの条件をテストしてから、デフォルトグループを処理します。Data Integration Service は、接続された出力グループの順序に基づいて、各条件の評価順を決定します。Data Integration Service は、マッピング内のトランスフォーメーションまたはターゲットに接続されたユーザー定義グループを処理します。

行が複数のグループフィルタ条件を満たす場合、Data Integration Service はこの行を複数回渡します。

### デフォルトグループ

Developer ツールがデフォルトグループを作成するのは、ユーザーがユーザー定義グループを 1 つ作成した後です。Developer ツールでは、デフォルトグループを編集または削除することはできません。このグループに関連付けられたグループフィルタ条件はありません。すべてのグループ条件が FALSE と評価された場合、Data Integration Service は行をデフォルトグループに渡します。Data Integration Service でデフォルトグループの行をすべて削除する場合は、デフォルトグループをマッピング内のトランスフォーメーションまたはターゲットに接続しないようにします。

リストから最後のユーザー定義グループを削除すると、Developer ツールはデフォルトグループを削除します。

Developer ツールは入力グループの入力ポートからプロパティ情報をコピーし、各出力グループに対する一連の出力ポートを作成します。出力ポートやそのプロパティを変更または削除することはできません。

## グループフィルタ条件の使用

1 つ以上のグループフィルタ条件に基づいてデータをテストできます。式のエディタを使用して、**【グループ】** タブでグループフィルタ条件を作成します。

単一の値を返す任意の式を入力することができます。条件に定数を指定することもできます。グループフィルタ条件は、トランスフォーメーションを通過する行ごとに、その行が指定の条件を満たすかどうかを基準にして、TRUE または FALSE を返します。ゼロ (0) は FALSE として、ゼロでない値はすべて TRUE として評価されます。単一の数値型ポートをフィルタ条件として使用できます。データ統合サービスは、TRUE と評価されたデータ行を、各ユーザー定義グループに関連付けられたそれぞれのトランスフォーメーションまたはターゲットに渡します。

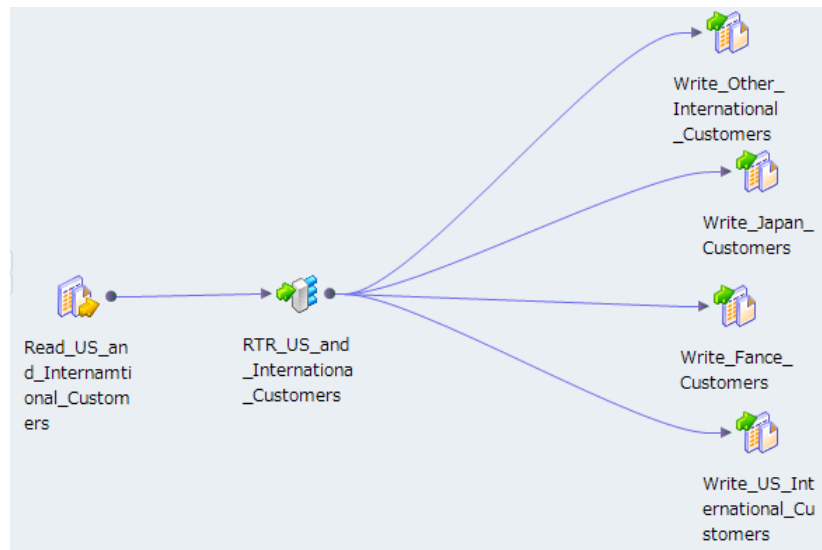
**注:** 単一の動的ポートを使用してブール値を返すことはできません。

たとえば、9ヶ国の顧客について、その内の3ヶ国のデータに対して種々の計算を行うとします。マッピング内でルータトランスフォーメーションを使用し、このデータをフィルタリングして3つの異なる式トランスフォーメーションに渡すことができます。

グループフィルタ条件の要素としてパラメータを使用できます。システムパラメータかユーザー定義パラメータを使用します。式エディタでパラメータを作成して式に追加できます。

デフォルトグループにはグループフィルタ条件はありません。それでも、残りの6ヶ国のデータに基づいて計算を行う式トランスフォーメーションを作成することができます。

以下の図は、複数の条件に基づいてデータのフィルタリングを行うルータトランスフォーメーションによるマッピングを示しています。



3つの異なる国から得たデータに基づいて複数の計算を行うには、[グループ] タブでユーザー定義グループを3つ作成し、グループフィルタ条件を3つ指定します。

以下の図に、顧客データをフィルタリングするグループフィルタ条件を示します。

出力グループ:		
グループ名	グループフィルタ条件	
Default		
France	COUNTRY='France'=TRUE	
Japan	COUNTRY='Japan'=TRUE	
USA	COUNTRY='USA'=TRUE	

説明:

以下の表に、顧客データをフィルタリングするグループフィルタ条件を示します。

グループ名	グループフィルタの条件
フランス	customer_name='France'=TRUE
日本	customer_name='Japan'=TRUE
USA	customer_name='USA'=TRUE

データ統合サービスは、マッピング実行中に、各ユーザー定義グループ（Japan、France、USA など）に関連付けられた各トランスフォーメーションまたはターゲットに、TRUE と評価されたデータ行を渡します。すべての条件が FALSE と評価された場合、データ統合サービスは行をデフォルトグループに渡します。データ統合サービスは他の 6 ヶ国のデータを、デフォルトグループに関連付けられたトランスフォーメーションまたはターゲットに渡します。データ統合サービスでデフォルトグループの行をすべて削除する場合は、デフォルトグループをマッピング内のトランスフォーメーションまたはターゲットに接続しないようにします。

ルータトランスフォーメーションは、条件を満たす各グループを通じてデータを渡します。データが 3 つの出力グループ条件を満たす場合、ルータトランスフォーメーションは 3 つの出力グループを通じてデータを渡します。

例えば、ルータトランスフォーメーションで以下のグループ条件を設定したとします。

グループ名	グループフィルタの条件
出力グループ 1	employee_salary > 1000
出力グループ 2	employee_salary > 2000

ルータトランスフォーメーションが入力行のデータを employee\_salary=3000 で処理する場合、Output Group 1 および 2 を通じてデータがルーティングされます。

## グループフィルタ条件の動的ポート

グループフィルタ条件では動的ポートを使用できます。データ統合サービスは、フィルタ条件を動的ポートで生成された各ポートに適用します。

例えば、動的ポート MyDynamicPort に次の 3 つの 10 進ポートが含まれているとします。

Salary  
Bonus  
Stock

このとき、次のグループフィルタ条件を設定する可能性があります。

MyDynamicPort > 100

グループフィルタ条件は次の式に展開されます。

Salary > 100 AND Bonus > 100 AND Stock > 100

生成された各ポートは、式に対して有効なタイプである必要があります。

## グループフィルタのパラメータ化

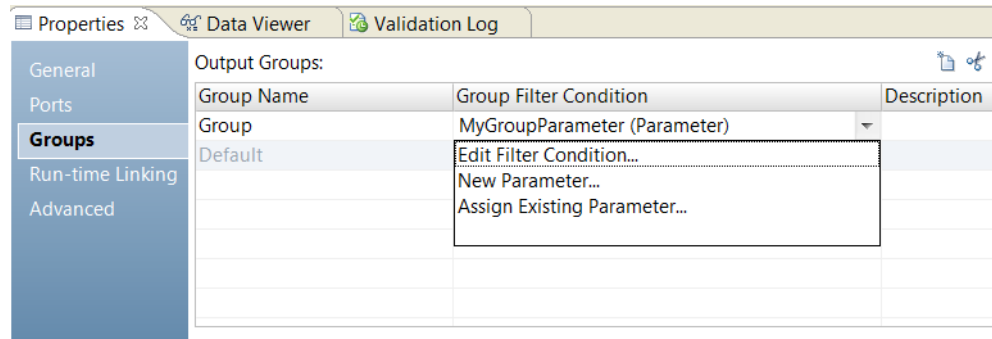
式パラメータを使用してグループフィルタを定義できます。式パラメータとは、完全な式が含まれるパラメータです。

例えば、式パラメータに次の式を含めることができます。

Employee\_Salary > 1000

グループフィルタをパラメータ化するには、[グループ] タブの [グループフィルタ条件] フィールドで [新しいパラメータ] を選択します。パラメータを定義し、式エディタで式を入力します。式パラメータに他のパラメータを含めることはできません。

次の図は、トランスフォーメーションプロパティの [グループ] タブを示しています。



式パラメータを使用した場合、Developer tool は式を検証できません。実行時に式が有効でない場合、マッピングが失敗する可能性があります。

## グループの追加

グループを追加すると、Developer ツールによってプロパティ情報が入力ポートから出力ポートにコピーされます。

1. [グループ] タブをクリックします。
2. [新規] ボタンをクリックします。
3. 新しいグループの名前を、[グループ名] セクションに入力します。
4. [グループフィルタ条件] フィールドをクリックして [式エディタ] を開きます。
5. グループフィルタ条件を入力します。
6. [検証] をクリックして条件の構文をチェックします。
7. [OK] をクリックします。

## ポートに関する作業

ルータトランスフォーメーションには、入力ポートと出力ポートがあります。入力ポートは入力グループにあり、出力ポートは出力グループにあります。

入力ポートは、別のトランスフォーメーションからコピーして作成するか、または [ポート] タブで手動で作成することができます。

Developer ツールは、入力ポートから以下のプロパティをコピーすることによって出力ポートを作成します。

- ポート名
- データタイプ
- 精度

- スケール
- デフォルト値

入力ポートに変更を加えると、Developer ツールは出力ポートを更新して変更を反映させます。出力ポートを編集または削除することはできません。

Developer ツールは、入力ポート名に基づいて出力ポート名を作成します。Developer ツールは各入力ポートに対して、各出力グループ内に対応する出力ポートを作成します。

## マッピング内のルータトランスフォーメーションの接続

トランスフォーメーションをマッピング内のルータトランスフォーメーションに接続する場合、以下の規則に留意してください。

- 1つのグループは、1つのトランスフォーメーションまたはターゲットに接続することができます。
- グループ内の1つの出力ポートは、複数のトランスフォーメーションまたはターゲットに接続することができます。
- 1つのグループ内の複数の出力ポートは、複数のトランスフォーメーションまたはターゲットに接続することができます。
- 複数のグループを1つのターゲットまたは1つの入力グループトランスフォーメーションに接続することはできません。
- 各出力グループを異なる入力グループに接続する場合は、ジョイナトランスフォーメーションを除いて、1つのグループを複数の入力グループトランスフォーメーションに接続することができます。

## ルータトランスフォーメーションの詳細プロパティ

Data Integration Service でのルータトランスフォーメーションのデータの処理方法を指定するには、詳細プロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

# 非ネイティブ環境でのルータートランスフォーメーション

非ネイティブ環境でのルータートランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

## 第 40 章

# シーケンスジェネレータトランスフォーメーション

この章では、以下の項目について説明します。

- [シーケンスジェネレータトランスフォーメーションの概要, 587 ページ](#)
- [シーケンスジェネレータのポート, 587 ページ](#)
- [シーケンスジェネレータトランスフォーメーションのプロパティ, 590 ページ](#)
- [シーケンスデータオブジェクト, 592 ページ](#)
- [シーケンスジェネレータトランスフォーメーションの作成, 595 ページ](#)
- [FAQ（よくある質問）, 596 ページ](#)
- [シーケンスジェネレータトランスフォーメーション非ネイティブ環境で, 597 ページ](#)

## シーケンスジェネレータトランスフォーメーションの概要

シーケンスジェネレータトランスフォーメーションは数値を生成するパッシブトランスフォーメーションです。一意なプライマリー値の作成、欠落しているプライマリー値の置き換え、連続した数値のサイクル動作を実行するには、シーケンスジェネレータトランスフォーメーションを使用します。

シーケンスジェネレータには、パススルーポートと出力ポートがあります。NEXTVAL ポートを他のトランスフォーメーションの入力ポートに接続します。統合サービスはマッピングが実行されるときにシーケンスを増分します。

シーケンスジェネレータトランスフォーメーションは、新しいシーケンスまたはシーケンスデータオブジェクトに基づいて作成できます。シーケンスデータオブジェクトは、値のシーケンスを作成して維持するオブジェクトです。

## シーケンスジェネレータのポート

シーケンスジェネレータトランスフォーメーションにはパススルーポートと 1 つの出力ポート（NEXTVAL）があります。出力ポートを編集または削除することはできません。

## パススルーポート

シーケンスジェネレータトランスフォーメーションには、ポートをパススルーポートとして追加できます。パススルーポートは、入力データを受信し、同じデータを変更せずにマッピングに返す入力ポートです。

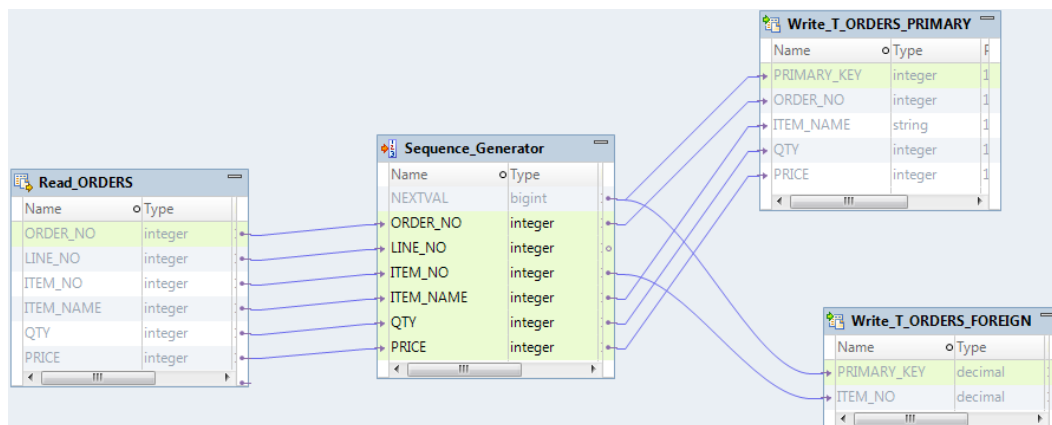
NEXTVAL 出力ポートをターゲットにリンクする前に、少なくとも 1 つの入力ポートをトランスフォーメーションに追加してアップストリームのソースまたはトランスフォーメーションに接続する必要があります。パススルーポートをトランスフォーメーションに追加するには、ポートを、マッピングでアップストリームのソースまたはトランスフォーメーションからシーケンスジェネレータトランスフォーメーションにドラッグします。

## NEXTVAL ポート

NEXTVAL をトランスフォーメーションに接続し、トランスフォーメーションの各行に対して一意の値を生成することができます。NEXTVAL ポートをダウンストリームのトランスフォーメーションまたはターゲットに接続し、番号のシーケンスを生成します。NEXTVAL を複数のトランスフォーメーションに接続した場合、統合サービスは各トランスフォーメーションに対して同じ番号のシーケンスを生成します。

NEXTVAL ポートを後続のトランスフォーメーションに接続し、[開始値] プロパティおよび [増分値] プロパティに基づいてシーケンスを生成します。シーケンスジェネレータがシーケンスでサイクル動作を実行するように設定されていない場合、NEXTVAL ポートは設定された終了値までキーシーケンス番号を生成します。

次の図に、ソースと 2 つのターゲットに接続され、プライマリキー値と外部キー値を生成するシーケンスジェネレータトランスフォーメーションの NEXTVAL ポートを含むマッピングを示します。



シーケンスジェネレータトランスフォーメーションを開始値 = 1、増分値 = 1 で設定すると、統合サービスは T\_ORDERS\_PRIMARY ターゲットテーブルと T\_ORDERS\_FOREIGN ターゲットテーブルに同じプライマリキー値を生成します。

## キーの作成

シーケンスジェネレータトランスフォーメーションでプライマリキーまたは外部キーの値を作成するには、NEXTVAL ポートをターゲットまたはダウンストリームのトランスフォーメーションに接続します。1～9,223,372,036,854,775,807 の範囲の値を、最小間隔 1 で使用することができます。複合キーを作成してテーブルの各行を識別することもできます。

複合キーを作成するには、値の小さなセットでサイクル動作を実行するように統合サービスを設定します。例えば、3 つの店舗が注文番号を生成する場合は、シーケンスジェネレータトランスフォーメーションが 1 から 3 まで 1 ずつ増やしてサイクル動作するように設定します。ORDER\_NO ポートをシーケンスジェネレータトランスフォーメーションに接続する場合は、生成される値によって一意の複合キーが作成されます。



次の例に、複合キーと注文番号を示します。

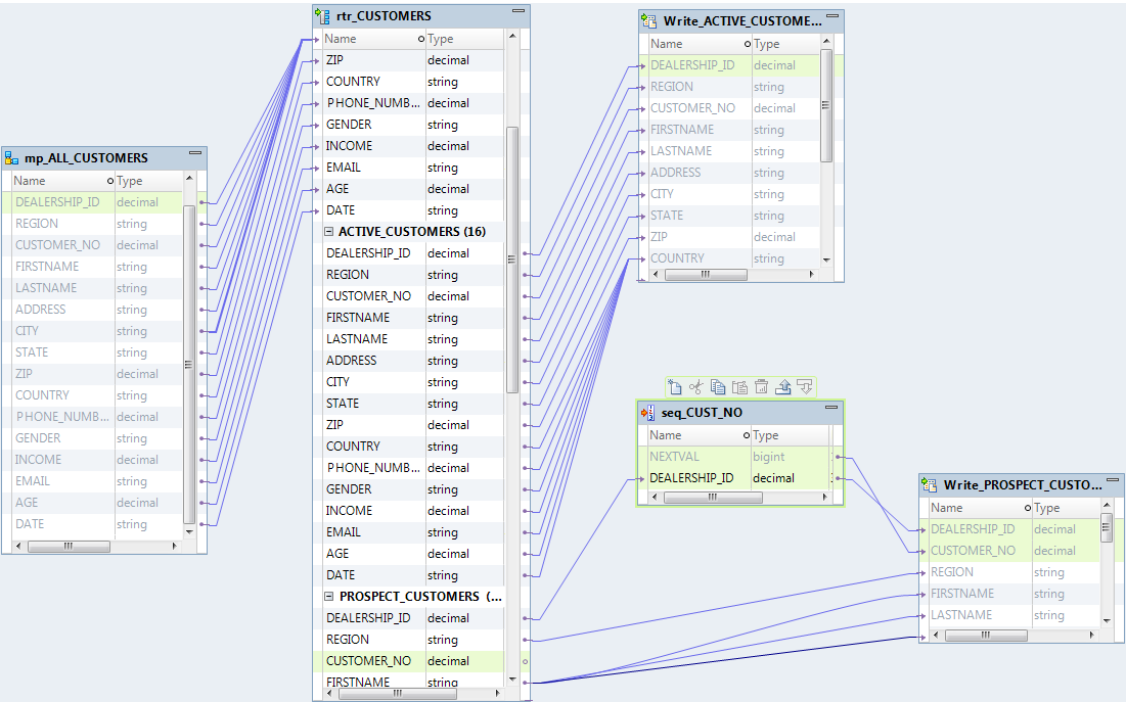
COMPOSITE_KEY	ORDER_NO
1	12345
2	12345
3	12345
1	12346
2	12346
3	12346

欠落値の置き換え

シーケンスジェネレータトランスフォーメーションを使用して欠落したキーを置き換える場合は、ルータトランスフォーメーションを使用して、値を割り当てたカラムから NULL 値のカラムをフィルタリングすることもできます。ルータトランスフォーメーションをシーケンスジェネレータトランスフォーメーションに接続し、NEXTVAL を使用して数値のシーケンスを生成して NULL 値を入力します。

例えば、CUSTOMER\_NO カラムの NULL 値を置き換えるには、顧客データを含むソースを持つマッピングを作成します。ルータトランスフォーメーションを追加し、NULL 値の顧客から顧客番号が割り当てられている顧客をフィルタリングします。シーケンスジェネレータトランスフォーメーションを追加し、一意の CUSTOMER\_NO 値を生成します。顧客ターゲットを追加し、データを書き込みます。

次の図に、CUSTOMER\_NO カラムの NULL 値を置き換えるマッピングを示します。



# シーケンスジェネレータトランスフォーメーションのプロパティ

統合サービスが連続した値を生成するのに使用するトランスフォーメーションプロパティを設定します。

以下の表に、シーケンスデータオブジェクトと新しいシーケンスに対して構成するプロパティを示しています。

プロパティ	説明
開始値	[サイクル] オプションを選択する場合に、統合サービスが使用する生成シーケンスの開始値。 [サイクル] を選択すると、統合サービスのサイクルは終了値に到達したときにこの値に戻ります。 デフォルトは 0 です。 最大値は 9,223,372,036,854,775,806 です。
終了値	統合サービスが生成する最大値。統合サービスがセッション中にこの値に到達し、シーケンスのサイクル動作が設定されていない場合、セッションは失敗します。 最大値は 9,223,372,036,854,775,807 です。
増分値	NEXTVAL ポートから出力される連続した 2 つの値の差。 デフォルトは 1 です。 正の整数にする必要があります。 最大値は 2,147,483,647 です。
サイクル	有効にすると、統合サービスはシーケンスの範囲でサイクル動作し、開始値を使用して再開します。 無効にすると、統合サービスは設定されている終了値でシーケンスを停止します。統合サービスが終了値に到達した時点で処理する必要がある行が残っていると、オーバーフローエラーによってセッションが失敗します。

## 開始値

[サイクル] を使用すると、月に対応する 1 から 12 の番号など、繰り返しのシーケンスを生成できます。

1. 統合サービスが開始値として使用するシーケンスの最小値を入力します。
2. [End Value] として使用する最大値を入力します。
3. [Cycle] を選択します。

サイクル動作の実行中にシーケンスで設定した終了値に到達すると、統合サービスは設定した開始値から再びサイクル動作を実行します。

## 終了値

終了値は、統合サービスが生成する最大値です。統合サービスが終了値に到達し、シーケンスジェネレータがシーケンスをサイクル動作するように設定されていない場合、マッピングの実行はオーバーフローエラーで失敗します。

End Value は、1~9,233,372,036,854,775,807 の任意の整数に設定できます。NEXTVAL ポートをダウンストリームの整数ポートに接続する場合は、[終了値] を最大整数値ほどの大きさの値に設定します。たとえば、NEXTVAL ポートを小整数ポートに接続する場合は、[終了値] を最大 32,767 に設定します。NEXTVAL がそのダウンストリームポートのデータタイプの最大値を超えると、マッピングは失敗します。

## 増分値

統合サービスは、シーケンスジェネレータトランスフォーメーションの [現在の値] および [増分] プロパティに基づいて、NEXTVAL ポートにシーケンスを生成します。

[現在の値] プロパティは、統合サービスが各セッション用シーケンスの作成を開始する際に使用する値です。[増分] は、統合サービスがシーケンスで新しい値を生成するために既存の値に追加する整数値です。デフォルトでは、[現在の値] と [増分] が共に 1 に設定されます。

たとえば [Current Value] を 1000 に、[Increment By] を 10 に設定したシーケンスジェネレータトランスフォーメーションを作成したとします。マッピングを介して 3 つの行を渡した場合は、統合サービスによって次の値のセットが生成されます。

```
1000
1010
1020
```

## 値の範囲内でのサイクル

シーケンスジェネレータトランスフォーメーションに値の範囲を設定できます。サイクルオプションを使用する場合、シーケンスジェネレータトランスフォーメーションは終了値に到達すると範囲を繰り返します。

例えば、シーケンス範囲を 10 で開始し 50 で終了するように設定し、増分値を 10 に設定すると、シーケンスジェネレータトランスフォーメーションは 10、20、30、40、50 の値を作成します。シーケンスは、10 から再び開始されます。

## リセット

[リセット] プロパティを使用するように再利用不可能なシーケンスジェネレータトランスフォーメーションを設定すると、統合サービスはマッピングを実行するたびに元の開始値を使用します。使用しないように設定すると、統合サービスは現在の値をインクリメントし、次のマッピングの実行にその値を使用します。

例えば、1 から 1,000 までの値を増分値 1、開始値 1 で作成するようにシーケンスジェネレータトランスフォーメーションを設定し、リセットを選択します。最初のマッピングの実行中、統合サービスは 1~234 の値を生成します。統合サービスは、その後にマッピングを実行するたびに開始値 1 から始まる数字を再度生成します。

リセットしない場合、統合サービスは最初の実行の終了時に現在の値を 235 に更新します。次回そのシーケンスジェネレータトランスフォーメーションを使用するとき、最初に生成される値は 235 になります。

**注:** [Reset] は、再利用可能なシーケンスジェネレータトランスフォーメーションに対しては無効になっています。

## 行順序の保持

トランスフォーメーションへの入力データの行順序を保持します。統合サービスが行順序を変更する可能性がある最適化を実行しないようにする場合に、このオプションを選択します。

統合サービスが最適化を実行すると、以前のマッピングで確立された順序が失われる場合があります。順序は、マッピングでソート済みのフラットファイルソース、ソート済みのリレーショナルソース、またはソータートランスフォーメーションのいずれかを使用することで確立できます。行順序を保持するようにトランスフォーメーションを設定すると、統合サービスではマッピングの最適化が実行される際に、この設定が考慮されます。統合サービスは、行順序を保持できる場合には、トランスフォーメーションの最適化を実行します。最適化により行順序が変更される可能性がある場合には、統合サービスはトランスフォーメーションの最適化を実行しません。

# シーケンスデータオブジェクト

シーケンスデータオブジェクトは、数値のシーケンスを作成して維持します。シーケンスジェネレータトランスフォーメーションは、シーケンスデータオブジェクトを使用してトランスフォーメーションの値を生成します。

再利用可能なシーケンスデータオブジェクトは、複数のシーケンスジェネレータトランスフォーメーションで使用できます。同じ統合サービスで実行される場合、同じシーケンスデータオブジェクトを使用するすべてのシーケンスジェネレータトランスフォーメーションは、同じ値シーケンスを使用します。再利用可能なシーケンスデータオブジェクトは、再利用不可能なシーケンスジェネレータトランスフォーメーションでも使用できます。再利用不可能なシーケンスデータオブジェクトは、再利用不可能なシーケンスジェネレータトランスフォーメーションで使用できます。

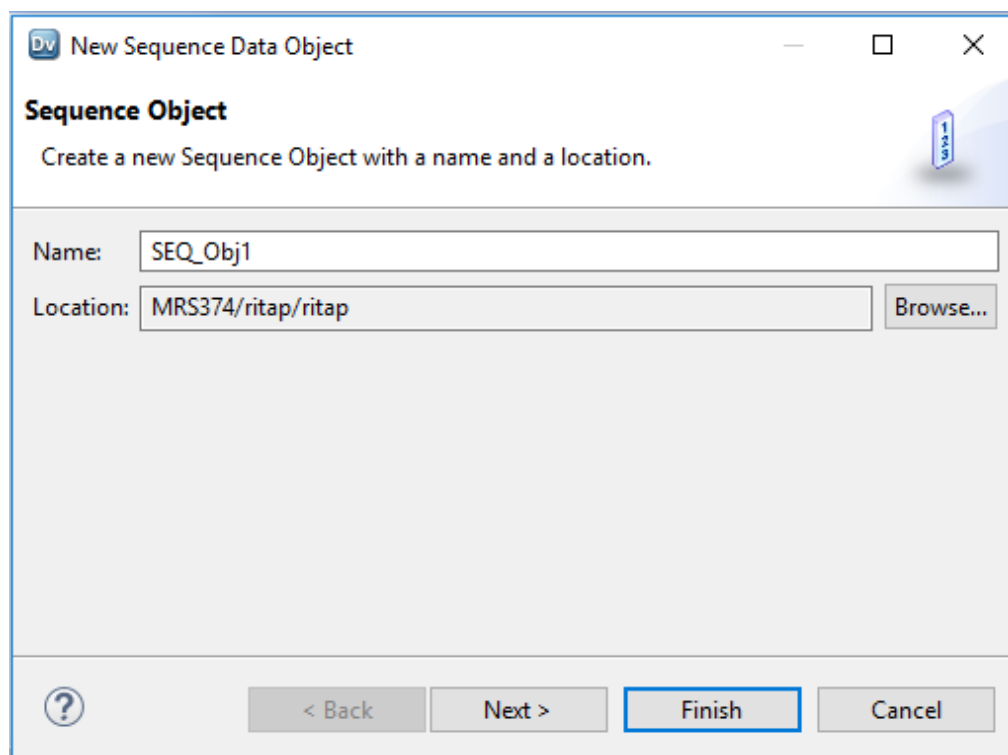
例えば、リレーショナルテーブルの同じプライマリキーフィールドに書き込む複数のマッピングを作成するとします。各マッピングは同じ再利用可能なシーケンスジェネレータトランスフォーメーションを使用し、そのトランスフォーメーションは同じ統合サービスで実行される同じ再利用可能なシーケンスデータオブジェクトを使用します。各マッピングは、プライマリキーフィールドに一意の値を書き込みます。

## シーケンスデータオブジェクトの作成

シーケンスデータオブジェクトを使用してシーケンスジェネレータトランスフォーメーションを作成するには、シーケンスデータオブジェクトを作成し、オブジェクトのプロパティを構成し、[シーケンスジェネレータトランスフォーメーション] ダイアログボックスでオブジェクトを選択します。

1. マッピングエディタで、マッピングパレットを下にスクロールして、シーケンスジェネレータトランスフォーメーションを見つけ、マッピングにドラッグします。  
**新しいトランスフォーメーションウィザード**が開きます。
2. **【新しいシーケンスデータオブジェクト】**をクリックします。

新しいデータオブジェクトウィザードが開きます。



**New Sequence Data Object**

**Sequence Object**

Create a new Sequence Object with a name and a location.

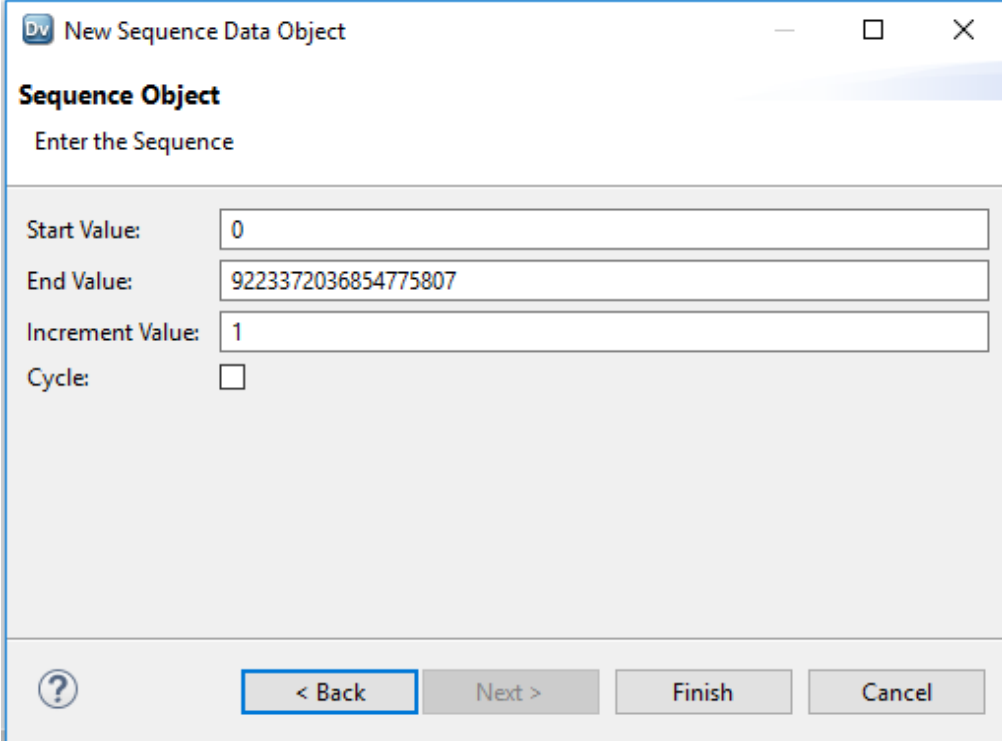
Name: SEQ\_Obj1

Location: MRS374/ritap/ritap Browse...

? < Back Next > Finish Cancel

3. シーケンスデータオブジェクトの名前を入力します。  
シーケンスデータオブジェクトの命名規則は、「SEQ\_<データオブジェクト名>」です。
4. **【次へ】** をクリックして、シーケンスデータオブジェクトのプロパティを構成します。

このオブジェクトからシーケンスジェネレータ変換を作成する場合、変換ではデータオブジェクトに対して入力したプロパティを使用します。以下の図に、設定可能なプロパティを示します。



5. データオブジェクトのプロパティを構成した後、シーケンスデータオブジェクトを使用してシーケンスジェネレータ変換を作成できます。変換を作成するときは、シーケンスジェネレータ変換に名前を付け、**【既存のシーケンスオブジェクトを選択してください。】**を選択します。データオブジェクトに移動し、**【OK】** をクリックします。

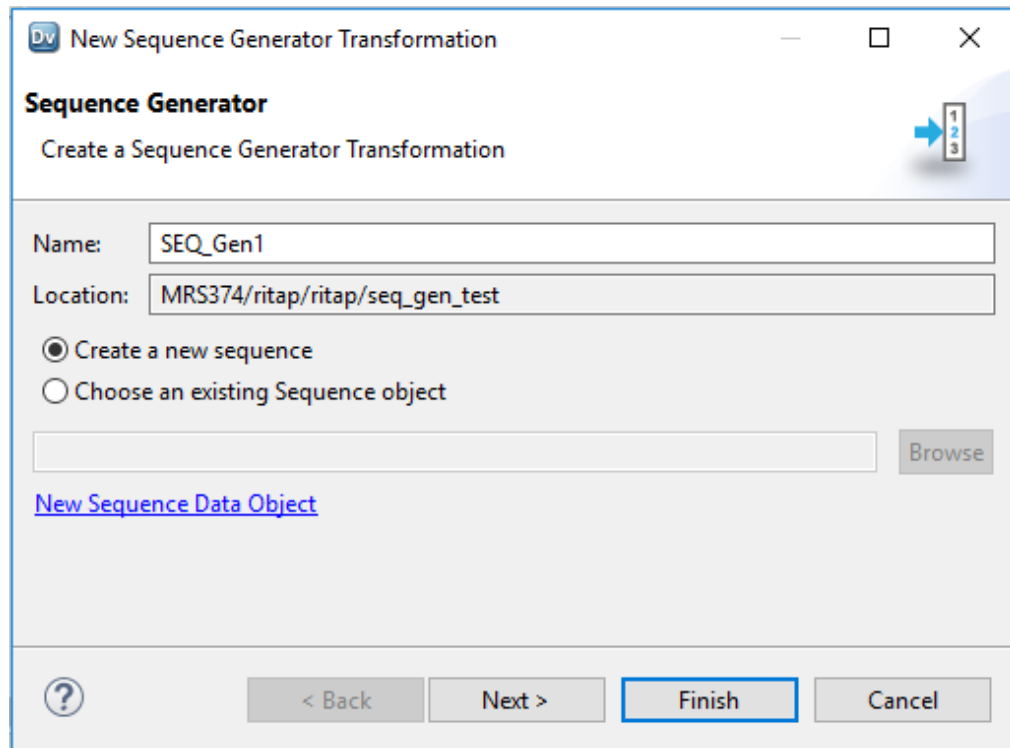
シーケンスジェネレータ変換は、NEXTVAL 出力専用ポートを持つマッピングエディタに表示されます。NEXTVAL ポートをダウンストリームの変換またはターゲットに接続し、番号のシーケンスを生成できます。

# シーケンスジェネレータトランスフォーメーションの作成

マッピングでシーケンスジェネレータトランスフォーメーションを使用するためには、まずそれをマッピングに追加します。次に、トランスフォーメーションのプロパティを設定し、NEXTVAL を 1 つまたは複数のトランスフォーメーションに接続します。

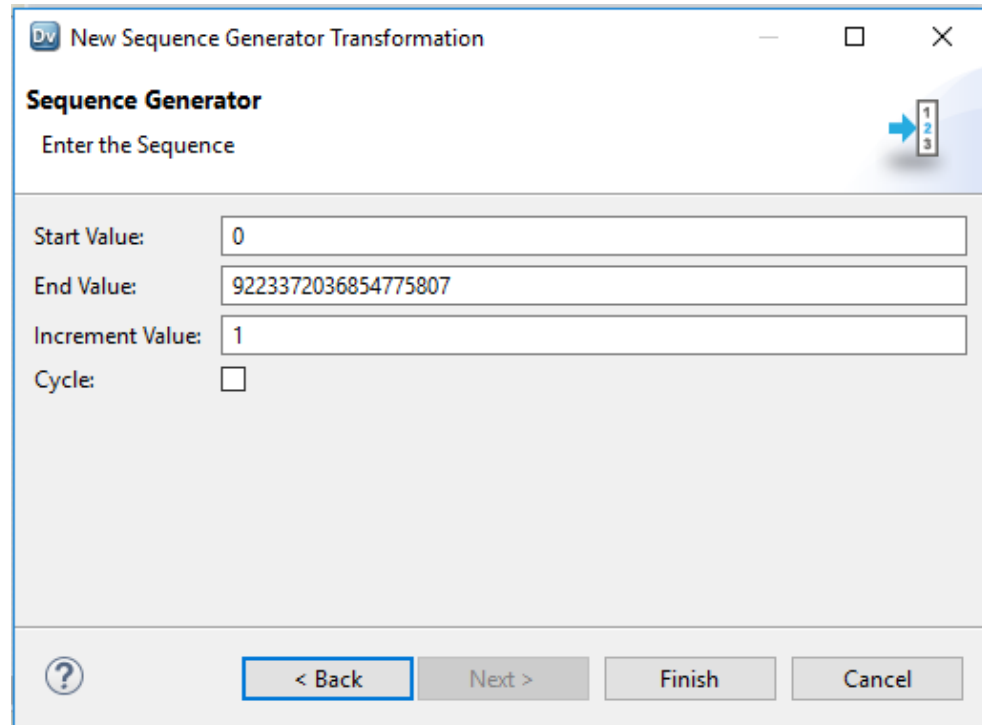
1. マッピングエディタで、マッピングパレットを下にスクロールして、シーケンスジェネレータトランスフォーメーションを見つけ、マッピングにドラッグします。

新しいトランスフォーメーションウィザードが開きます。



2. シーケンスジェネレータトランスフォーメーションの名前を入力します。  
シーケンスジェネレータトランスフォーメーションの命名規則は、「SEQ\_<トランスフォーメーション名>」です。
3. 新しいシーケンスを作成するか、既存のシーケンスオブジェクトを使用するかを選択します。

- 新しいシーケンスを作成するには、**【新しいシーケンスを作成】**を選択します。**【次へ】**をクリックして、シーケンスのプロパティを構成します。以下の図に、設定可能なプロパティを示します。



- 既存のシーケンスオブジェクトを使用するには、**【既存のシーケンスオブジェクトを選択してください。】**を選択します。シーケンスオブジェクトに移動し、**【OK】**をクリックします。

シーケンスジェネレータ変換は、NEXTVAL 出力専用ポートを持つマッピングエディタに表示されます。NEXTVAL ポートをダウンストリームの変換またはターゲットに接続し、番号のシーケンスを生成できます。

## FAQ（よくある質問）

### 再利用不可能なシーケンスジェネレータ変換を再利用可能に変更できますか？

変換を再利用可能にはできませんが、シーケンスデータオブジェクトを使用するように変換を変更することはできます。シーケンスデータオブジェクトは、シーケンスを使用する変換の数に関係なくシーケンスの整合性を維持します。

### 再利用不可能なシーケンスジェネレータ変換をマップレットに配置できますか？

できません。マップレットは再利用可能なオブジェクトであるため、マップレット内のオブジェクトはすべて再利用可能である必要があります。代わりに再利用可能なシーケンスジェネレータ変換を使用します。



# シーケンスジェネレータトランスフォーメーション 非ネイティブ環境で

非ネイティブ環境でのシーケンスジェネレータトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングで制限付きでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## Blaze エンジンでのシーケンスジェネレータトランスフォーメーション

シーケンスジェネレータトランスフォーメーションを使用したマッピングでは、次の条件が当てはまる場合に大量のリソースが消費されます。

- トランスフォーメーションの **【行順序を保持】** プロパティを *true* に設定した。
- マッピングが単一パーティションで実行されている。

## Spark エンジンでのシーケンスジェネレータトランスフォーメーション

シーケンスジェネレータトランスフォーメーションは、出力データの行順序を保持しません。トランスフォーメーションの **【行順序を保持】** プロパティを有効にした場合、データ統合サービスはこのプロパティを無視します。

## 第 41 章

# ソータートランスフォーメーション

この章では、以下の項目について説明します。

- [ソータートランスフォーメーションの概要, 598 ページ](#)
- [動的マッピングでのソータートランスフォーメーション, 599 ページ](#)
- [ソータートランスフォーメーションの開発, 599 ページ](#)
- [ソータートランスフォーメーションのポート, 600 ページ](#)
- [\[ソート\] タブ, 600 ページ](#)
- [ソートキーの設定, 600 ページ](#)
- [ソータートランスフォーメーションの詳細プロパティ, 603 ページ](#)
- [ソーターキャッシュ, 603 ページ](#)
- [ソータートランスフォーメーションの作成, 604 ページ](#)
- [ソータートランスフォーメーションの例, 605 ページ](#)
- [非ネイティブ環境でのソータートランスフォーメーション, 606 ページ](#)

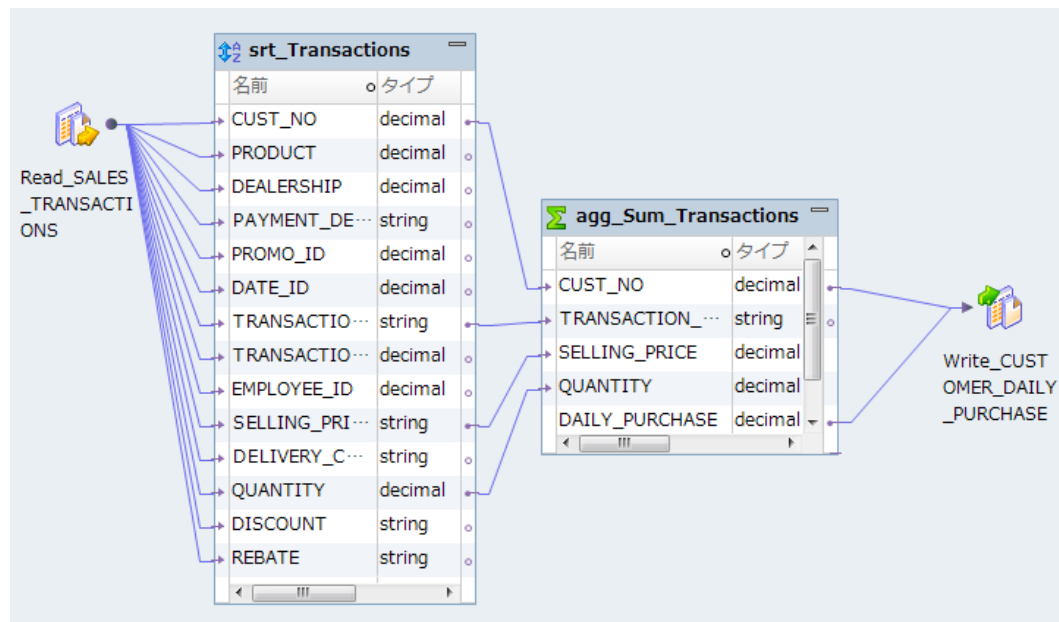
## ソータートランスフォーメーションの概要

指定されたソートキーに従って昇順または降順にデータをソートするには、ソータートランスフォーメーションを使用します。ソータートランスフォーメーションは、大文字小文字を区別してソートするように設定したり、出力が重複しないように設定したりすることができます。ソータートランスフォーメーションはアクティブなトランスフォーメーションです。

ソータートランスフォーメーションを作成する場合、ポートをソートキーとして指定し、昇順または降順でソートを行うように各ソートキーポートを設定します。ソートキーに複数のポートを指定した場合、Data Integration Service は各ポートを順番にソートします。

例えば、顧客売上の全体的な請求書を顧客データベースから作成する必要があるとします。顧客売上テーブルのソータートランスフォーメーションを使用し、顧客番号に従ってデータを降順にソートします。ソータートランスフォーメーションの結果をアグリゲータトランスフォーメーションに対する入力として使用します。[ソート済み入力] オプションを使用して、アグリゲータトランスフォーメーションのパフォーマンスを向上させることができます。

以下の図はマッピングを示しています。



## 動的マッピングでのソータートランスフォーメーション

動的マッピングでソータートランスフォーメーションを使用できます。アグリゲータトランスフォーメーションで動的ポートを設定し、生成されたポートを参照することができます。

ソータートランスフォーメーションでは、動的ポートまたは生成されたポートを参照できます。ただし、生成されたポートが実行時に存在しない場合は、マッピングが失敗します。

動的ポートをソートキーとして使用する場合、データ統合サービスは、動的ポート内の生成されたすべてのポートおよび生成されたポートの順序を考慮します。

ソートキーはパラメータ化できます。ソートキーのソートリストパラメータを使用します。

## ソータートランスフォーメーションの開発

ソータートランスフォーメーションを開発するときは、ソートキーポート、重複しない出力行、大文字小文字を区別するソート基準などの項目について検討する必要があります。

ソータートランスフォーメーションを開発するときは、以下の項目について検討してください。

- ソートキーおよびソート方向として設定するポート。
- 大文字小文字を区別してソートするかどうか。
- NULL 値をソートの優先順位として考慮するかどうか。
- 出力行を重複しないようにするかどうか。

- 設定するソーターキャッシュサイズ。

## ソータートランスフォーメーションのポート

ソータートランスフォーメーションにポートを作成する場合、デフォルトでは入力ポートおよび出力ポートを作成します。ソータートランスフォーメーションは、入力ポートと同じ出力ポートを返します。

ソータートランスフォーメーションには動的ポートを定義できます。動的ポートは、マッピングのアップストリームトランスフォーメーションからさまざまなカラムのデータを受信できます。これにより、さまざまなカラムが含まれる行をソータートランスフォーメーションがソートできます。

ソートキーは、**【プロパティ】** ビューの **【ソート】** タブで定義します。

## 【ソート】 タブ

ソータートランスフォーメーションの **【プロパティ】** ビューの **【ソート】** タブでソートキーを定義します。ソート条件として使用する 1 つ以上のポートを選択します。

データ統合サービスは、**【ソーター】** タブのポートの順番でデータをソートします。昇順または降順でデータをソートするように設定します。デフォルトは昇順です。

出力行が重複しないようにソータートランスフォーメーションを設定した場合、Developer tool はすべてのポートをソートキーの一部として設定します。データ統合サービスは、ソート操作中に重複行を破棄します。

## ソートキーの設定

トランスフォーメーションの **【プロパティ】** ビューの **【ソート】** タブでソートキーを定義します。

これが概念のスタートです。

次の図は、**【ソート】** タブを示しています。

Sort

Output:
☒ All rows
☐ Distinct rows only

Sort Keys

Specify by: Value

Ports:

Department	Ascending (A)
Employee	Ascending (A)

Add
Choose...
Delete
Move Up
Move Down

【ソート】タブには、以下のオプションがあります。

#### 出力

ソートしたすべての行を返すか、重複行を破棄するかを選択します。重複行とは、すべてのカラム値が同じである行です。

#### 指定元

【値】または【パラメータ】を選択します。ポート名を使用するには、【値】を選択します。ソートリストパラメータを使用するには、【パラメータ】を選択します。

#### 追加

手動で入力したポート名を受け入れます。【追加】をクリックする前に、有効なポート名を入力する必要があります。

#### 選択

【選択】をクリックして、ソートキーに追加するポートを選択します。Developer tool は、トランスフォーメーションからポートのリストを表示します。このリストから選択します。

#### 上に移動、下に移動

グループ内のポートの順序を変更できます。ポート名を選択し、この移動ボタンのどちらかをクリックして、ソート順の中でポート名を上下に移動します。

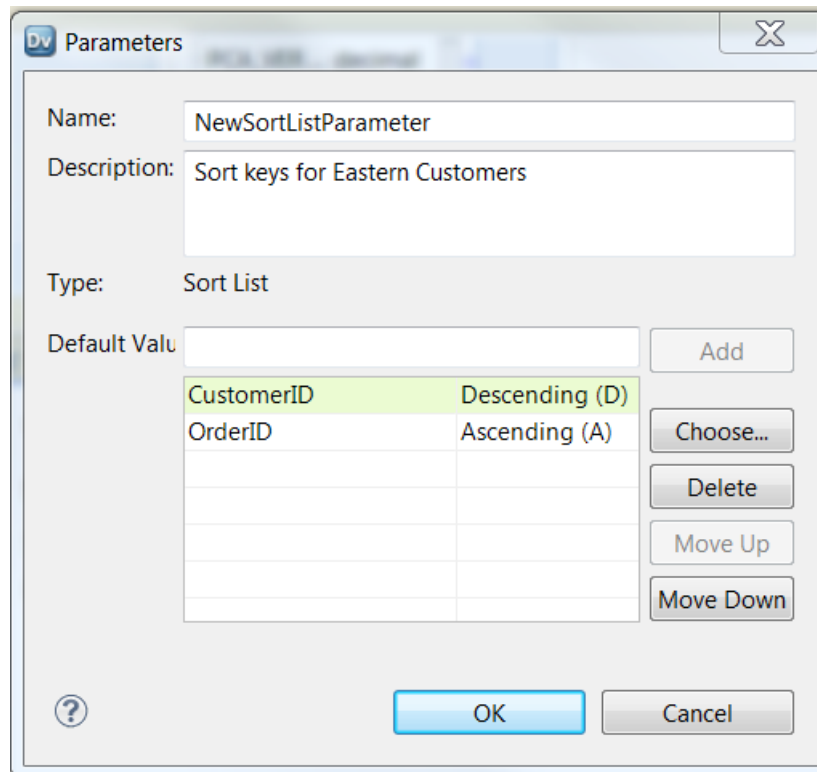
## ソートキーのパラメータ化

ソートキー用のポートのリストを含んだソートリストパラメータを作成できます。

ソータートランスフォーメーションが動的マッピング内にある場合、そのソータートランスフォーメーションには生成されたポートを含めることができます。ソートキーはパラメータ化できます。ソート基準となるポートのリストを含んだソートリストパラメータを作成します。

トランスフォーメーションプロパティの【ソート】タブで、[指定元:] プロパティの【パラメータ】を選択します。【新規】をクリックしてパラメータを作成します。

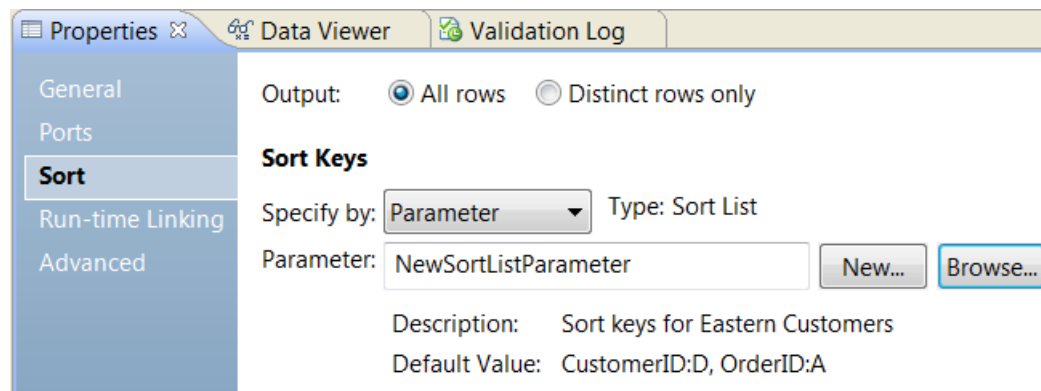
次の図は、【パラメータ】ダイアログボックスを示しています。



ソートキーのポートまたは生成されたポートを選択します。ソートタイプとして、昇順または降順を選択できます。

ポート名は手動で入力できます。【デフォルト値】フィールドにポート名を入力して、【追加】をクリックします。Developer tool によって、ソートリストにポート名が追加されます。

次の図は、ソートキーのパラメータを設定した後の【ソート】タブを示しています。



# ソータトランスフォーメーションの詳細プロパティ

ソータトランスフォーメーションの詳細プロパティで、追加のソート基準を指定することができます。データ統合サービスは、プロパティをすべてのソートキーポートに適用します。また、ソータトランスフォーメーションのプロパティは、データ統合サービスがデータのソート時に割り当てるシステムリソースも決定します。

以下に、ソータトランスフォーメーションの詳細プロパティを示します。

## 大文字小文字の区別

データ統合サービスがデータをソートする際に大文字と小文字を区別するかどうかを決定します。[大文字小文字の区別] プロパティを有効にすると、データ統合サービスはソートの際に大文字を小文字よりも上位にソートします。Developer tool では [大文字小文字の区別] がデフォルトで設定されています。

## NULL を下として扱う

NULL 値を最下位にソートします。データ統合サービスがソート操作を行う際に、NULL 値を他のすべての値より下位として処理するようにする場合は、このプロパティを有効にします。

## ソーターキャッシュサイズ

マッピングの実行開始時に、ソート操作を実行するためデータ統合サービスによって割り当てられるメモリの量。データ統合サービスは、ソート操作を行う前に、すべての入力データをソータトランスフォーメーションに渡します。[自動] を選択すると、実行時にデータ統合サービスによってメモリ要件が自動的に計算されます。キャッシュサイズを調整する場合は、固有の値をバイト単位で入力します。デフォルトは [自動] です。

## 作業ディレクトリ

入力データの量がソーターキャッシュサイズよりも大きい場合に、データ統合サービスがデータを一時的に格納するディレクトリ。データ統合サービスはデータをソートした後で、一時ファイルを削除します。

キャッシュのパーティション化時のパフォーマンスを向上させるには、セミコロンで区切って複数のディレクトリを入力します。キャッシュのパーティション化により、トランスフォーメーションを処理する各パーティションに個別のキャッシュが作成されます。

デフォルトは TempDir システムパラメータです。このフィールドには、別のシステムパラメータまたはユーザー定義のパラメータを設定できます。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## 関連項目：

- [「キャッシュサイズ」 \(ページ 72\)](#)

# ソーターキャッシュ

データ統合サービスでは、ソータトランスフォーメーションを実行するためにメモリ内にキャッシュが作成されます。データ統合サービスは、ソート操作を行う前に、すべての入力データをソータトランスフォーメーションに渡します。データ統合サービスは、メモリキャッシュで使用可能なスペースよりも多くのスペースを必要とする場合、データをソータトランスフォーメーションの作業ディレクトリに一時的に格納します。

メモリ内のすべてのデータをソートするようにキャッシュサイズを設定していない場合、データ統合サービスによってソースデータに複数のパスが作成されたことを示す警告が、セッションログに表示されます。ソートを完了するためにディスクに情報をページングする必要がある場合、データ統合サービスによってデータに複

数のパスが作成されます。データ統合サービスでデータが1回読み込まれ、ディスクにページングしないでメモリでソートが実行される場合に、このメッセージには単一パスに必要なメモリ量が表示されます。マッピングのパフォーマンスを最適化するには、データ統合サービスによってデータに1つのパスが作成されるように、キャッシュサイズを設定します。

入力されるデータの量がソーターキャッシュサイズよりも大きい場合、データ統合サービスはデータを一時的にソータートランスフォーメーションの作業ディレクトリに保存します。データを作業ディレクトリに保存する場合、データ統合サービスは最低でも入力されるデータの量の2倍のディスク領域を必要とします。

最高のパフォーマンスのため、マッピングを実行するマシン上の利用可能な物理メモリの量以下の値になるようにソーターキャッシュサイズを設定します。ソータートランスフォーメーションを使用してデータをソートするには、少なくとも16MB（16,777,216バイト）の物理メモリを割り当てます。デフォルトでは、ソーターキャッシュサイズは「自動」に設定されています。

## ソーターキャッシュの最適化

ソーターキャッシュは、ソータートランスフォーメーションをパススルーする binary および string データ型を保存するために可変長を使用するよう最適化されます。

可変長を使用すると、データ統合サービスがソーターキャッシュに保存するデータ量、およびデータ統合サービスマシン状のディスク使用量が減少します。

例えば、顧客に関する住所を保存します。顧客によって名前の長さはまちまちです。データ統合サービスで顧客名に関するデータを保存するために固定長を使用していたら、名前ごとに20文字でデータを保存する可能性があります。しかし、可変長を使用している場合は、平均10文字でデータを保存できるかもしれません。

# ソータートランスフォーメーションの作成

再利用可能なソータートランスフォーメーションまたは再利用不可能なソータートランスフォーメーションを作成できます。

## 再利用可能なソータートランスフォーメーションの作成

複数のマッピングまたはマプレットで使用する、再利用可能なソータートランスフォーメーションを作成します。

1. **【Object Explorer】** ビューで、プロジェクトまたはフォルダーを選択します。
2. **【ファイル】** > **【新規】** > **【トランスフォーメーション】** をクリックします。  
**【新規】** ダイアログボックスが表示されます。
3. ソータートランスフォーメーションを選択します。
4. **【次へ】** をクリックします。
5. トランスフォーメーションの名前を入力します。
6. **【完了】** をクリックします。  
トランスフォーメーションがエディタに表示されます。
7. **【新規】** をクリックして、トランスフォーメーションにポートを追加します。
8. ポートを編集して、名前、データ型、および精度を設定します。
9. **【ソート】** タブで、ソート基準とするポートを選択するか、ソートリストパラメータを選択します。
10. **【詳細】** ビューをクリックし、トランスフォーメーションのプロパティを編集します。



## 再利用不可能なソータートランスフォーメーションの作成

マッピングまたはマブレットで再利用不可能なソータートランスフォーメーションを作成します。

1. マッピングまたはマブレットで、トランスフォーメーションパレットからエディタにソータートランスフォーメーションをドラッグします。  
トランスフォーメーションがエディタに表示されます。
2. **【プロパティ】** ビューで、トランスフォーメーションの名前と説明を編集します。
3. **【ポート】** タブで、**【新規】** をクリックして、トランスフォーメーションにポートを追加します。
4. ポートを編集して、名前、データ型、および精度を設定します。
5. **【キー】** を選択して、ポートをソートキーとして指定します。
6. **【詳細】** タブをクリックし、トランスフォーメーションのプロパティを編集します。

## ソータートランスフォーメーションの例

顧客からのすべての注文に関する情報を含むデータベーステーブル PRODUCT\_ORDERS があります。

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
43	123456	ItemA	3	3.04
41	456789	ItemB	2	12.02
43	000246	ItemC	6	34.55
45	000468	ItemD	5	0.56
41	123456	ItemA	4	3.04
45	123456	ItemA	5	3.04
45	456789	ItemB	3	12.02

PRODUCT\_ORDERS でソータートランスフォーメーションを使用し、ORDER\_ID をソートキーに、方向を降順に指定します。

データをソートした後、Data Integration Service はソータートランスフォーメーションから以下の行を渡します。

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
45	000468	ItemD	5	0.56
45	123456	ItemA	5	3.04
45	456789	ItemB	3	12.02
43	123456	ItemA	3	3.04
43	000246	ItemC	6	34.55
41	456789	ItemB	2	12.02

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
41	123456	ItemA	4	3.04

注文ごとの合計金額と数量を特定する必要があります。ソータートランスフォーメーションの結果をアグリゲータトランスフォーメーションへの入力として使用できます。パフォーマンスを向上させるには、アグリゲータトランスフォーメーションでソート済み入力を使用します。

〔ソート済み入力〕オプションを使用しない場合、Data Integration Service は読み込みと並行して集計計算を実行します。すべての集計計算が正確に実行されるように、Data Integration Service はソース全体の読み込みが完了するまで各グループのデータを格納しておきます。〔ソート済み入力〕オプションを使用する場合、前もってデータを正しくソートしておかないと、予期しない結果が生じます。

アグリゲータトランスフォーメーションでポート別の ORDER\_ID グループがあり、ソート済み入力オプションが選択されているとします。ソータートランスフォーメーションからデータを渡すと、アグリゲータトランスフォーメーションは ORDER\_ID をグループ化して、注文ごとの合計金額を計算します。

ORDER_ID	SUM
45	54.06
43	216.42
41	36.2

## 非ネイティブ環境でのソータートランスフォーメーション

非ネイティブ環境でのソータートランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限付きのサポート。

### Blaze エンジンでのソータートランスフォーメーション

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

#### マッピング検証

マッピング検証は、次の場合に失敗します。

- ターゲットデータが行の順序を維持するように設定されており、ソータートランスフォーメーションがフラットファイルターゲットと直接接続されていない。

## 並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

## グローバルソート

Blaze エンジンでは、次の状況においてグローバルソートを実行できます。

- ソータートランスフォーメーションがフラットファイルターゲットと直接接続されている。
- ターゲットが行の順序を維持するように設定されている。
- ソートキーがバイナリデータ型ではない。

満たされない条件がある場合、Blaze エンジンはローカルソートを実行します。

## データキャッシュの最適化

アグリゲータデータキャッシュまたはリンクデータキャッシュを最適化するために、ソータートランスフォーメーションがアグリゲータトランスフォーメーションまたはリンクトランスフォーメーションの前に挿入されている場合、ソーターキャッシュのサイズは、アグリゲータトランスフォーメーションまたはリンクトランスフォーメーションのデータキャッシュと同じにする必要があります。ソーターキャッシュを設定するには、アグリゲータトランスフォーメーションまたはリンクトランスフォーメーションのデータキャッシュのサイズを設定する必要があります。

# Spark エンジンでのソータートランスフォーメーション

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

## マッピングの検証

マッピング検証は、大文字小文字の区別が無効な場合に失敗します。

データ統合サービスは、次の場合に、警告をログに記録し、ソータートランスフォーメーションを無視します。

- ターゲットとソータートランスフォーメーションのソートキーの間に型の不一致がある。
- トランスフォーメーションに、ターゲットに接続されていないソートキーが含まれる。
- 書き込みトランスフォーメーションが行の順序を維持するように設定されていない。
- トランスフォーメーションが書き込みトランスフォーメーションからの直接のアップストリームではない。

## NULL 値

データ統合サービスは、NULL 値を最高値として扱うようにトランスフォーメーションが設定されている場合でも、NULL 値を最低値として扱います。

## データキャッシュの最適化

ソーターキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

## 並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

## ストリーミングマッピングでのソータートランスフォーメーション

ストリーミングマッピングには、バッチマッピングには適用されない追加の処理ルールがあります。

### マッピングの検査

マッピング検証は、次の場合に失敗します。

- ストリーミングマッピングのソータートランスフォーメーションに、アップストリームのアグリゲータトランスフォーメーションが存在しない。

### 一般的なガイドライン

次の一般的なガイドラインを考慮してください。

- マッピングにソータートランスフォーメーションが含まれている場合は、完全出力モードでマッピングを実行している。
- グローバルなソート順を維持するため、ターゲットはシングルパーティションになっているようにする。ソースはシングルパーティションでもマルチパーティションでも構わない。

# Databricks Spark エンジンでのソータートランスフォーメーション

Databricks Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### マッピングの検証

マッピング検証は、大文字小文字の区別が無効な場合に失敗します。

データ統合サービスは、次の場合に、警告をログに記録し、ソータートランスフォーメーションを無視します。

- ターゲットとソータートランスフォーメーションのソートキーの間に型の不一致がある。
- トランスフォーメーションに、ターゲットに接続されていないソートキーが含まれる。
- 書き込みトランスフォーメーションが行の順序を維持するように設定されていない。
- トランスフォーメーションが書き込みトランスフォーメーションからの直接のアップストリームではない。

### NULL 値

データ統合サービスは、NULL 値を最高値として扱うようにトランスフォーメーションが設定されている場合でも、NULL 値を最低値として扱います。

### データキャッシュの最適化

ソーターキャッシュを、可変長を使用してデータを格納するように最適化することはできません。

## 並列ソート

データ統合サービスでは、次の制限付きで、並列ソートが有効になります。

- マッピング内のソータートランスフォーメーションとターゲットの間に別のトランスフォーメーションが含まれていない。
- ソータートランスフォーメーションとターゲットの間で、ソートキーのデータ型が変わらない。
- ソータートランスフォーメーションの各ソートキーは、ターゲットのカラムにリンクされている必要がある。

## 第 42 章

# SQL トランスフォーメーション

この章では、以下の項目について説明します。

- [SQL トランスフォーメーションの概要, 610 ページ](#)
- [SQL トランスフォーメーションのポート, 611 ページ](#)
- [SQL トランスフォーメーションの詳細プロパティ, 615 ページ](#)
- [SQL トランスフォーメーションクエリ, 617 ページ](#)
- [入力行と出力行のカーディナリティ, 619 ページ](#)
- [SQL トランスフォーメーションによるフィルタの最適化, 623 ページ](#)
- [SQL クエリを使った SQL トランスフォーメーションの例, 624 ページ](#)
- [ストアードプロシージャ, 628 ページ](#)
- [SQL トランスフォーメーションの接続, 633 ページ](#)
- [SQL トランスフォーメーションの手動による作成, 634 ページ](#)
- [ストアードプロシージャから SQL トランスフォーメーションの作成, 634 ページ](#)

## SQL トランスフォーメーションの概要

SQL トランスフォーメーションは、マッピングの中間地点で SQL クエリを処理します。SQL トランスフォーメーションから SQL クエリを実行するか、SQL トランスフォーメーションを設定してデータベースからストアードプロシージャを実行することができます。

入力ポートの値をクエリまたはストアードプロシージャのパラメータに渡すことができます。トランスフォーメーションは、データベースに行を挿入したり、データベースの行を削除、更新、および取得したりできます。SQL DDL 文を実行して、マッピングの途中でテーブルを作成したり、削除したりできます。SQL トランスフォーメーションはアクティブなトランスフォーメーションです。このトランスフォーメーションは入力行ごとに複数の行を返すことができます。

ストアードプロシージャをデータベースから SQL トランスフォーメーションにインポートすることができます。ストアードプロシージャをインポートすると、Developer ツールはストアードプロシージャ内のパラメータに対応するトランスフォーメーションポートを作成できます。Developer ツールは、ストアードプロシージャの呼び出しも作成します。

SQL トランスフォーメーションをストアードプロシージャを実行するように設定するには、以下のタスクを実行します。

1. 接続先のデータベースタイプなどのトランスフォーメーションプロパティを定義します。

2. ストアドプロシージャをインポートして、ポートを定義し、ストアドプロシージャの呼び出しを作成します。
3. 追加で実行する必要があるストアドプロシージャまたは結果セットのポートを手動で定義します。
4. 追加のストアドプロシージャ呼び出しを SQL エディタで追加します。

トランスフォーメーション SQL エディタで SQL クエリを設定できます。SQL トランスフォーメーションを実行すると、トランスフォーメーションはクエリを処理し、行を返し、データベースエラーを返します。

SQL トランスフォーメーションをクエリを実行するように設定するには、以下のタスクを実行します。

1. 接続先のデータベースタイプなどのトランスフォーメーションプロパティを定義します。
2. 入力ポートと出力ポートを定義します。
3. SQL エディタで SQL クエリを作成します。

トランスフォーメーションを設定したら、マッピング内に SQL トランスフォーメーションを設定し、アップストリームのポートを接続します。データをプレビューして結果を確認します。

## SQL トランスフォーメーションのポート

SQL トランスフォーメーションを作成すると、Developer ツールはデフォルトで `SQL_Error` ポートを作成します。【ポート】ビューで入力ポート、出力ポート、およびパススルーポートを追加します。

SQL トランスフォーメーションには、以下のタイプのポートがあります。

### 入力

SQL クエリで使用できるソースデータを受け取ります。

### 出力

SQL SELECT クエリからのデータベースデータを返します。

### パススルー

トランスフォーメーションを介してソースデータをそのままの状態ですす入出力ポートです。

### SQL\_Error

データベースからの SQL エラーを返します。エラーが発生しなかった場合は、NULL を返します。

### NumRowsAffected

入力行の INSERT、DELETE、および UPDATE クエリ文の影響を受けたデータベース行の合計数を返します。このポートは、出力行に統計の更新を含めることを選択したときに、Developer ツールによって作成されます。

### 戻り値

戻り値をストアドプロシージャから受け取ります。

## 入力ポート

どのようなタイプの SQL 文またはストアドプロシージャでもパラメータのバインドを使って SQL トランスフォーメーションの入力ポートを参照することができます。出力ポートに渡さないデータに対して SQL トランスフォーメーションに入力ポートを作成することができます。

入力パラメータのある SQL クエリを設定している場合は、ポートを手動で追加する必要があります。SQL トランスフォーメーションにストアドプロシージャをインポートすると、SQL トランスフォーメーションが入力ポ

ートを作成します。パススルーポートを追加して、データを変更せずにトランスフォーメーションを介してデータを渡すことができます。

**【概要】** ビューにポートを追加できます。ポートを追加するときは、そのポートのネイティブのデータ型を入力します。ネイティブデータ型は、接続先にするデータベースで有効なデータ型です。ネイティブのデータ型を設定すると、トランスフォーメーションのデータ型が表示されます。行を SQL トランスフォーメーションにドラッグすると、接続しようとしているデータベースにとって有効なデータ型に基づいて Developer ツールがネイティブのデータ型を設定します。クエリ内で使用するカラムのデータ型がデータベース内のカラムと同じデータ型であることを確認してください。

次の図に、再利用可能な SQL トランスフォーメーションの **CreationDate** という入力ポートを示します。

**概要**

**全般**

名前: SQLTransformation

説明:

**SQLTransformation**

名前	タイプ
Input (1)	
CreationDate	date/time
Output (1)	
SQL Error	string

**ポート**

名前	タイプ	ネイティブ	精度	スケール	デフォルト値	説明	出力...
Input (1)							
1 CreationDate	date/time	timestamp	29	9			
Output (1)							
1 SQL Error	string		4096	0			

概要 SQL 詳細

入力ポートを追加するには、**【ポート】** パネルで **【入力】** をクリックします。**【新規】** をクリックします。

**注:** ポートに対して **【出力にコピー】** を選択すると、入力ポートはパススルーポートになります。パススルーポートは、**【入力】** セクションおよび **【出力】** セクション（**【ポート】** ビュー内）に表示されます。

## 出力ポート

SQL トランスフォーメーション出力ポートは、クエリ文からまたはストアードプロシージャから値を返します。

SQL トランスフォーメーションを手動で設定する場合、出力ポートを定義する必要があります。ストアードプロシージャの出力パラメータごとに、または SELECT 文が返すポートごとに、出力ポートを定義します。

ストアードプロシージャをインポートすると、Developer ツールは、プロシージャが返す出力パラメータごとに出力ポートを作成します。プロシージャが結果セットを返す場合、結果セットに出力ポートを手動で定義する必要があります。ストアードプロシージャは、結果セットを返すことができ、同じ実行からの結果セットの一部ではない出力パラメータを返すこともできます。結果セットフィールドおよび出力パラメータに出力ポートを定義する必要があります。

出力ポートを設定するときは、そのポートのネイティブデータ型を選択します。出力ポートのネイティブのデータ型は、データベース内でそれに対応するカラムのデータ型と一致する必要があります。ネイティブデータ型が設定されると、Developer ツールはポートのトランスフォーメーションデータ型を定義します。



例えば、SQL トランスフォーメーションに、Oracle データベースに対する以下の SQL クエリが含まれるとします。

```
SELECT FirstName, LastName, Age FROM EMPLOYEES
```

この場合、SQL トランスフォーメーションに以下の出力ポートとネイティブデータ型を設定できます。

出力ポート	ネイティブデータ型	トランスフォーメーションデータ型
FirstNm	varchar2	string
LastNm	varchar2	string
Age	number	double

出力ポートの数および順序は、クエリまたはストアドプロシージャが返すカラムの数および順序と一致する必要があります。出力ポートの数がクエリまたはストアドプロシージャ内のカラム数よりも多い場合、余分なポートには NULL 値が返ります。出力ポートの数が SQL 内のカラム数よりも少ない場合、データ統合サービスは行エラーを生成します。

トランスフォーメーションの接続先のデータベースタイプを変更すると、Developer ツールによって出力ポートのネイティブタイプが変更されます。Developer ツールは、すべてのポートに対して正しいデータ型を選択するとはかぎりません。データベースタイプを変更した場合は、各出力ポートのネイティブデータ型がデータベース内のカラムと同じデータ型であることを確認します。例えば、Oracle データベースカラムに対して nVarchar2 が選択される可能性があります。その場合、データ型を varchar2 に変更する必要があります。

出力ポートは、SQL トランスフォーメーションの【概要】ビューで設定します。

## パススルーポート

パススルーポートは、トランスフォーメーションを介してデータをそのままの状態ですぐに出力ポートです。SQL トランスフォーメーションは、SQL クエリが行を返すかどうかにかかわらず、パススルーポートにデータを返します。

入力行に SELECT クエリ文が含まれている場合、SQL トランスフォーメーションは、データベースから返される行ごとに、パススルーポートにデータを返します。クエリ結果に複数の行が含まれている場合、SQL トランスフォーメーションは、各行についてパススルーデータの生成を繰り返します。

クエリによって行が返されなかった場合、SQL トランスフォーメーションは、出力カラムに、NULL 値を含むパススルーカラムデータを返します。たとえば、INSERT、UPDATE、および DELETE の各文を含むクエリでは、行が返されません。クエリでエラーが発生した場合、SQL トランスフォーメーションは、パススルーカラムデータ、SQLException メッセージ、および NULL 値を出力ポートに返します。

SELECT クエリからデータを返すようにパススルーポートを設定することはできません。

パススルーポートを作成するには、入力ポートを作成して【出力にコピー】を選択します。Developer ツールによって出力ポートが作成され、ポート名に「\_output」接尾語が追加されます。Developer ツールがパススルーポートに対して作成する出力ポートを変更することはできません。「\_output」接尾語が含まれる出力ポートを作成することはできません。

以下の図に、再利用可能な SQL トランスフォーメーションの Name パススルーポートを示します。

概要

全般

名前: SQLTransformation

説明:

SQLTransformation

名前

タイプ

Input (2)

Name

string

CreationDate

date/time

Output (2)

ポート

名前

タイプ

ネイティ...

精度

スケ...

デフォルト値

説明

出力...

Input (2)

1

Name

string

nvarchar2

10

0

2

CreationDate

date/time

timestamp

29

9

Output (2)

1

SQLError

string

4096

0

2

Name\_output

string

nvarchar2

10

0

概要

SQL

詳細

## SQLError ポート

SQLError ポートは、ストアドプロシージャまたは SQL クエリからのデータベースから SQL エラーを戻します。

以下の図に、再利用可能な SQL トランスフォーメーションの SQLError ポートを示します。

概要

全般

名前: SQLTransformation

説明:

SQLTransformation

名前

タイプ

Input (0)

Output (1)

SQLError

string

ポート

名前

タイプ

ネイティ...

精度

スケ...

デフォルト値

説明

出力...

1

Input (0)

Output (1)

1

SQLError

string

4096

0

概要

SQL

詳細

SQL クエリに構文エラーがある場合、SQLError ポートにはデータベースからのエラーテキストが含まれます。たとえば、以下の SQL クエリでは、Oracle データベースからの行エラーが生成されます。

```
SELECT Product_ID FROM Employees
```

Employees テーブルには Product\_ID はありません。データ統合サービスによって 1 行が生成されます。SQLError ポートに、以下のエラーテキストが 1 行で含まれます。

```
ORA-0094: "Product_ID": invalid identifier Database driver error... Function Name: Execute SQL Stmt: SELECT Product_ID from Employees Oracle Fatal Error
```

SQL クエリ内に複数のクエリ文を設定すること、または複数のストアードプロシージャを呼び出すことができます。SQL のエラー時でも処理を続行するように SQL トランスフォーメーションを設定した場合、SQL トランスフォーメーションは 1 つのクエリ文に対して行を返し、別のクエリ文に対してはデータベースエラーを返すことがあります。SQL トランスフォーメーションは、データベースエラーを別の行に返します。

## 影響を受けた行の数

NumRowsAffected 出力ポートを有効にすると、入力行ごとに INSERT、UPDATE、または DELETE クエリ文が変更する行の数が返ります。SQL クエリに NumRowsAffected 出力ポートを設定することができます。

データ統合サービスは、クエリ内のクエリ文ごとに NumRowsAffected を返します。NumRowsAffected はデフォルトで無効になっています。

NumRowsAffected を有効にし、SQL クエリに INSERT、UPDATE、または DELETE 文が含まれていない場合、各出力行の NumRowsAffected はゼロになります。

SQL クエリに複数の文が含まれる場合、データ統合サービスは文ごとに NumRowsAffected を返します。NumRowsAffected には、入力行の INSERT、UPDATE、および DELETE 文が変更する行の合計が含まれます。

たとえば、クエリに以下の文が含まれているとします。

```
DELETE from Employees WHERE Employee_ID = '101';
SELECT Employee_ID, LastName from Employees WHERE Employee_ID = '103';
INSERT into Employees (Employee_ID, LastName, Address)VALUES ('102', 'Gein', '38 Beach Rd')
```

DELETE 文は、1 行に影響を与えます。SELECT 文は、どの行にも影響を与えません。INSERT 文は、1 行に影響を与えます。

データ統合サービスは、DELETE 文から 1 行返します。NumRowsAffected は 1 になります。データ統合サービスは SELECT 文から 1 行返し、NumRowsAffected はゼロになります。データ統合サービスは INSERT 文から 1 行返し、NumRowsAffected は 1 になります。

## SQL トランスフォーメーションの詳細プロパティ

SQL トランスフォーメーションのプロパティはいつでも変更できます。デフォルトのデータベースタイプは Oracle です。接続する必要があるデータベースが別のデータベースタイプである場合は、トランスフォーメーションにポートを追加する前にデータベースタイプを変更します。

【詳細】 タブで、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [ノーマル] です。SQL トランスフォーメーションのトレースレベルを [Verbose Data] に設定した場合、データ統合サービスは準備する各 SQL クエリをマッピングログに書き込みます。

## 接続タイプ

データ統合サービスがデータベースにどのように接続するかを示します。接続タイプは静的プロパティです。データ統合サービスはデータベースに 1 回接続します。SQL トランスフォーメーション内のデータベース接続オブジェクトを選択します。読み取り専用です。

## DB タイプ

SQL トランスフォーメーションが接続するデータベースのタイプです。データベースタイプをリストから選択します。Oracle、Microsoft SQL Server、IBM DB2、または ODBC を選択することができます。データベースタイプは、[ポート] タブで割り当てることができるデータ型に影響を与えます。データベースタイプを変更すると、Developer ツールによって入力ポート、出力ポート、およびパススルーポートのポートデータ型が変更されます。

## 行内のエラー時でも処理を続行する

SQL エラーの発生後、クエリ内の残りの SQL 文の処理を継続します。

## 統計を出力として含める

NumRowsAffected 出力ポートを追加します。ポートは、入力行の INSERT、DELETE、および UPDATE クエリ文で更新されたデータベース行の合計数を返します。

## 最大出力行数

SQL トランスフォーメーションで SELECT クエリーから出力できる最大行数を定義します。行が制限されないように設定するには、[最大出力行数] をゼロに設定します。

## クエリの説明

トランスフォーメーションに定義した SQL クエリの説明です。

## SQL モード

SQL クエリが例外スクリプトであるかどうかや、クエリがトランスフォーメーションに定義されるかどうかを決定します。[SQL モード] は [クエリ] です。SQL トランスフォーメーションは、SQL エディタで定義されたクエリを実行します。読み取り専用です。

## SQL クエリ

SQL エディタで設定された SQL クエリを表示します。

## 副次作用あり

SQL トランスフォーメーションが行を返す以外にも関数を実行することを示す。SQL クエリがデータベースを更新すると、SQL トランスフォーメーションに副次作用が生じます。SQL クエリ内に CREATE、DROP、INSERT、UPDATE、GRANT、REVOKE などの文が使用されている場合は、[副次作用あり] を有効にします。

SQL トランスフォーメーションが結果を返さない SELECT 文の NULL 行を返す場合は、SQL トランスフォーメーションにも副次作用が生じます。行には、パススルーポート値、SQL エラー情報、または NUMRowsAffected フィールドが含まれます。

最適化にプッシュインまたは初期選択の最適化を可能にするには、[副次作用あり] プロパティを無効にします。デフォルトでは有効になっています。

## データベース出力のみ返す

SQL トランスフォーメーションで、0 個の結果を返す SELECT 文の行、INSERT、UPDATE、DELETE、COMMIT などの他の文の行、または NULL 行が生成されません。

## 最適化にプッシュインを有効にする

データ統合サービスが、SQL トランスフォーメーションにおける SQL へのマッピングでフィルタトランスフォーメーションからロジックをプッシュできるようにします。

### 行順序を保持

トランスフォーメーションへの入力データの行順序を保持します。データ統合サービスが行順序を変更する可能性がある最適化を実行しないようにする場合に、このオプションを選択します。

データ統合サービスが最適化を実行すると、以前のマッピングで確立された順序が失われる場合があります。ソート済みフラットファイルソース、ソート済みリレーショナルソース、またはソータトランスフォーメーションを使用したマッピングにおける順序を確立できます。行順序を保持するようにトランスフォーメーションを設定すると、データ統合サービスではマッピングの最適化が実行される際に、この設定が考慮されます。データ統合サービスは、行順序を保持できる場合には、トランスフォーメーションの最適化を実行します。最適化により行順序が変更される可能性がある場合には、データ統合サービスはトランスフォーメーションの最適化を実行しません。

### パーティション化可能

トランスフォーメーションを複数のスレッドで処理できます。データ統合サービスが1つのスレッドを使用してトランスフォーメーションを処理するようにする場合は、このオプションをクリアします。データ統合サービスは複数のスレッドを使用して残りのマッピングパイプラインステージを処理します。

SQL クエリで SQL トランスフォーメーションを1つのスレッドで処理する必要がある場合に、SQL トランスフォーメーションのパーティション化を無効にします。または、SQL トランスフォーメーションのパーティション化を無効にすることで、データベースへの接続を1つだけに限定することもできます。

## SQL トランスフォーメーションクエリ

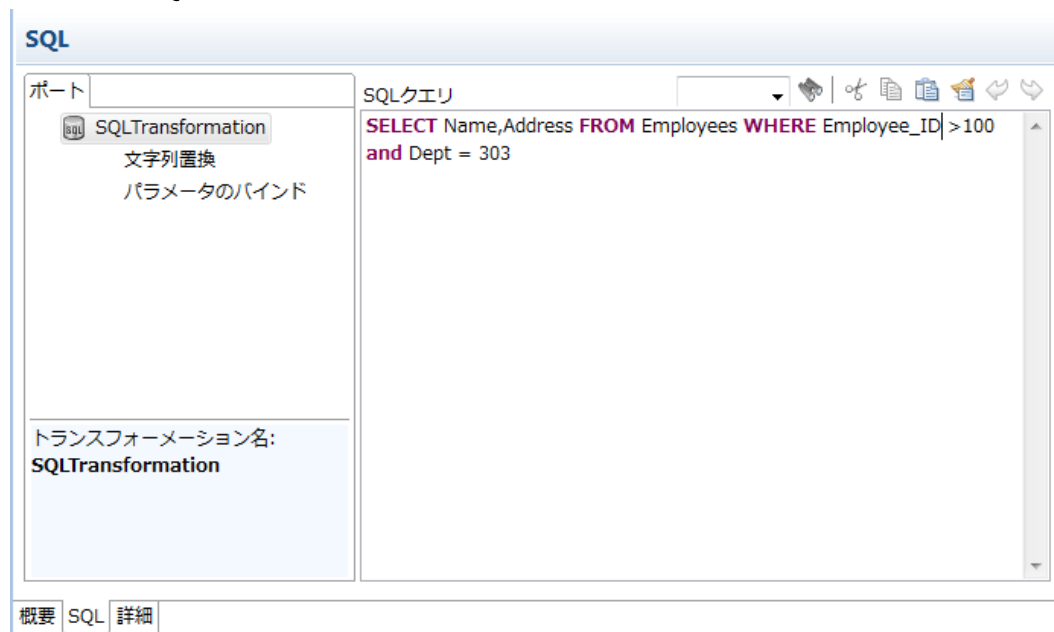
データベースから行を取得したり、データベースを更新したりするには、SQL エディタで SQL クエリを作成します。

クエリを作成するには、[SQL] ビューの SQL エディタでクエリ文を入力します。SQL クエリ文には、32767 文字まで入力できます。SQL エディタには、クエリで参照できるトランスフォーメーションポートのリストが表示されます。ポート名をダブルクリックして、クエリパラメータとして追加できます。

SQL クエリを作成すると、クエリ内のポート名が SQL エディタによって検証されます。また、文字列の置換に使用するポートが文字列データ型であるかどうかを確認されます。SQL エディタでは、SQL クエリの構文は検証されません。

SQL クエリに定数を使用できます。各文字列を一重引用符 (') で括ります。

以下の図に、SQL クエリの例を示します。



静的な SQL クエリを作成することができます。このクエリ文は変更されませんが、パラメータを含めて値を変更することはできます。データ統合サービスが入力行ごとにクエリを実行します。

## SQL クエリの定義

入力行ごとに同じクエリ文を実行する SQL クエリを定義します。クエリのカラムやテーブルを、行内の入力ポートの値に基づいて変更できます。WHERE 句の値も、入力ポートの値に基づいて変更できます。

入力行ごとの WHERE 句のデータ値を変更するには、パラメータのバインドを設定します。

入力ポートの値に基づいてクエリのカラムを変更したり、テーブルを変更したりするには、文字列の置換を使用します。

### パラメータのバインド

クエリ内のデータを変更するには、クエリパラメータを設定し、これらのクエリパラメータをトランスフォーメーションの入力ポートにバインドします。パラメータを入力ポートにバインドする場合、クエリ内のポート名を指定します。SQL エディタでは、ポート名は疑問符 (?) で囲まれます。クエリデータは、ポート内のデータの値に基づいて変化します。

以下のクエリでは、パラメータのバインドが使用されています。

```
DELETE FROM Employee WHERE Dept = ?Dept?
INSERT INTO Employee(Employee_ID, Dept) VALUES (?Employee_ID?, ?Dept?)
UPDATE Employee SET Dept = ?Dept? WHERE Employee_ID > 100
```

以下の SQL クエリには、SQL トランスフォーメーションの Employee\_ID および Dept 入力ポートにバインドするクエリパラメータがあります。

```
SELECT Name, Address FROM Employees WHERE Employee_Num =?Employee_ID? and Dept = ?Dept?
```

ソースに以下の行があるとします。

Employee_ID	Dept
100	Products

Employee_ID	Dept
123	HR
130	Accounting

Data Integration Service によって、行から以下のクエリ文が生成されます。

```
SELECT Name, Address FROM Employees WHERE Employee_ID = '100' and DEPT = 'Products'
SELECT Name, Address FROM Employees WHERE Employee_ID = '123' and DEPT = 'HR'
SELECT Name, Address FROM Employees WHERE Employee_ID = '130' and DEPT = 'Accounting'
```

## 文字列の置換

クエリ文のコンポーネントを置換するには、文字列変数を使用します。例えば、文字列変数を使用してクエリ内のテーブル名を置換することができます。また、SELECT 文内のカラム名を置換することもできます。

テーブル名を置換するには、各入力行からテーブル名を受け取るように入力ポートを設定します。SQL エディタで、ポートの【文字列置換】リストからポートを選択します。Developer ツールは、クエリ内の名前を入力ポートを識別し、名前をティルダ (~) で囲みます。

以下のクエリには、文字列変数 ~Table\_Port~ が含まれています。

```
SELECT Emp_ID, Address from ~Table_Port~ where Dept = 'HR'
```

ソースから、以下の値が **Table\_Port** カラムに渡されるとします。

### Table\_Port

Employees\_USA

Employees\_England

Employees\_Australia

Data Integration Service は、~Table\_Port~ 変数を入力ポートのテーブル名の値で置換します。

```
SELECT Emp_ID, Address from Employees_USA where Dept = 'HR'
SELECT Emp_ID, Address from Employees_England where Dept = 'HR'
SELECT Emp_ID, Address from Employees_Australia where Dept = 'HR'
```

# 入力行と出力行のカーディナリティ

Data Integration Service によって SELECT クエリが実行されると、SQL トランスフォーメーションは、取得した行ごとに 1 行を返します。クエリでデータが取得されない場合、SQL トランスフォーメーションによって、入力行ごとにゼロまたは 1 行が返されます。

## クエリ文の処理

SELECT クエリが成功した場合、SQL トランスフォーメーションによって複数の行が取得されることがあります。クエリに他の文が含まれる場合、Data Integration Service は、SQL エラーまたは影響を受けた行の数が含まれる行を生成することがあります。

## ポート設定

NumRowsAffected 出力ポートには、1 つの入力行の UPDATE、INSERT、または DELETE 文が変更する行の数が含まれます。SQL トランスフォーメーションは、クエリ内の文ごとに影響を受けた行の数を返し

ます。SQL トランスフォーメーションにパススルーポートが設定されている場合、トランスフォーメーションは各ソース行に対して最低 1 回はカラムデータを返します。

#### 最大行数の設定

「最大出力行数」は、SQL トランスフォーメーションで SELECT クエリから返される行数を制限します。

#### エラー行

Data Integration Service は、接続エラーまたは構文エラーを検出すると、行エラーを返します。SQL トランスフォーメーションには、エラーを SQLError ポートに返します。

#### SQL エラー時に続行

SQL 文にエラーがあっても処理を継続するように SQL トランスフォーメーションを設定できます。SQL トランスフォーメーションは、行エラーを生成しません。

## クエリ文の処理

SQL クエリのタイプによって、SQL トランスフォーメーションで返される行数が決定します。SQL トランスフォーメーションは、ゼロ行、1 行、または複数の行を返すことができます。クエリに SELECT 文が含まれる場合、SQL トランスフォーメーションによってデータベースの各列が出力ポートに返されます。該当するすべての行がトランスフォーメーションによって返されます。

以下の表に、クエリモードでエラーが発生しない場合に、さまざまなタイプのクエリ文について SQL トランスフォーメーションが生成する出力行を示します。

クエリー文	出力行
UPDATE、INSERT、DELETE のみ	クエリ内の文ごとに 1 行。
1 つ以上の SELECT 文	取得されたデータベース行の合計数。
CREATE、DROP、TRUNCATE などの DDL クエリー	クエリ内の文ごとに 1 行。

## ポート設定

「統計を出力として含める」を有効にすると、Developer ツールによって NumRowsAffected ポートが作成されます。Data Integration Service は、SQL クエリの文に基づいて NumRowsAffected が含まれる行を少なくとも 1 行返します。

以下の表に、NumRowsAffected を有効にした場合に、SQL トランスフォーメーションによって生成される出力行を示します。

クエリ文	出力行
UPDATE、INSERT、DELETE のみ	文ごとにその文の NumRowsAffected が含まれる 1 行。
1 つ以上の SELECT 文	取得されたデータベース行の合計数。 各行の NumRowsAffected はゼロです。
CREATE、DROP、TRUNCATE などの DDL クエリ	ゼロの NumRowsAffected が含まれる 1 行。



## 最大出力行数

SQL トランスフォーメーションが SELECT クエリに対して返す行数を制限できます。**【最大出力行数】** プロパティを設定して、行数を制限します。クエリに複数の SELECT 文が含まれる場合、SQL トランスフォーメーションはすべての SELECT 文からの合計の行数を制限します。

例えば、**【最大出力行数】** を 100 に設定したとします。クエリに以下の 2 つの SELECT 文が含まれるとします。

```
SELECT * FROM table1; SELECT * FROM table2;
```

最初の SELECT 文によって 200 行が返され、2 番目の SELECT 文によって 50 行が返される場合、SQL トランスフォーメーションによって最初の SELECT 文から 100 行が返されます。SQL トランスフォーメーションは、2 番目の文からは行を返しません。

出力行が制限されないように設定するには、**【最大出力行数】** をゼロに設定します。

## エラー行

Data Integration Service は、接続エラーまたは構文エラーを検出すると行エラーを返します。SQL トランスフォーメーションは、SQL エラーを SQLError ポートに返します。

パススルーポートまたは NumRowsAffected ポートを設定した場合、SQL トランスフォーメーションは各ソース行について最低 1 行のデータを返します。クエリから行が返されない場合、SQL トランスフォーメーションはパススルーデータと NumRowsAffected 値を返しますが、出力カラムには NULL 値を返します。NULL 値の行を削除するには、フィルタトランスフォーメーション経由で出力行を渡します。

以下の表に、UPDATE、INSERT、または DELETE のクエリ文について、SQL トランスフォーメーションが生成する行を示します。

設定されている NumRowsAffected ポート またはパススルーポート	SQLError	出力される行
いずれのポートも設定されていません	いいえ	SQLError ポートに NULL を含む 1 行。
いずれのポートも設定されていません	はい	SQLError ポートにエラーを含む 1 行
どちらかが設定済み	いいえ	クエリ文ごとに、NumRowsAffected またはパススルーのカラムデータを含む 1 行。
どちらかが設定済み	はい	SQLError ポート、NumRowsAffected ポート、またはパススルーポートのデータにエラーを含む 1 行。

以下の表に、SELECT 文について SQL トランスフォーメーションが生成する出力行の数を示します。

設定されている NumRowsAffected ポート またはパススルーポート	SQLError	出力される行
いずれのポートも設定されていません	いいえ	1 つ以上の行。各 SELECT 文から返された行によって異なります。
いずれのポートも設定されていません	はい	正常に終了した文の出力行の合計よりも大きい 1 行。最後の行には、SQLError ポートにエラーが含まれます。

設定されている NumRowsAffected ポート またはパススルーポート	SQLError	出力される行
どちらかが設定済み	いいえ	1 つ以上の行。各 SELECT 文で返された行によって異なります。 - NumRowsAffected が有効な場合、各行には値 0 の NumRowsAffected カラムが含まれます。 - パススルーポートが設定されている場合、各行にはパススルーカラムデータが含まれます。クエリによって複数の行が返された場合、パススルーカラムデータは各行について重複して生成されます。
どちらかが設定済み	はい	1 つ以上の行。各 SELECT 文で返された行によって異なります。最後の行には、SQLError ポートのエラーが含まれます。 - NumRowsAffected が有効な場合、各行には値 0 の NumRowsAffected カラムが含まれます。 - パススルーポートが設定されている場合、各行にはパススルーカラムデータが含まれます。クエリによって複数の行が返された場合、パススルーカラムデータは各行について重複して生成されます。

以下の表に、CREATE、DROP、TRUNCATE などの DDL クエリについて、SQL トランスフォーメーションが生成する出力行の数を示します。

設定されている NumRowsAffected ポート またはパススルーポート	SQLError	出力される行
いずれのポートも設定されていません	いいえ	SQLError ポートに NULL を含む 1 行。
いずれのポートも設定されていません	はい	SQLError ポートにエラーを含む 1 行。
どちらかが設定済み	いいえ	値 0 の NumRowsAffected カラムおよびパススルーカラムデータを含む 1 行
どちらかが設定済み	はい	SQLError ポートにエラー、値 0 の NumRowsAffected カラム、およびパススルーカラムデータを含む 1 行

## SQL エラー時に続行

クエリ文で発生する SQL エラーは無視することができます。【**行内のエラー時でも処理を続行する**】を有効にします。Data Integration Service は、行の残りの SQL 文の実行を続行します。

Data Integration Service は行エラーを生成しません。ただし、SQLError ポートには、失敗した SQL 文とエラーメッセージが含まれます。

たとえば、クエリに以下の文があるとして。

```
DELETE FROM Persons WHERE FirstName = 'Ed';
INSERT INTO Persons (LastName, Address) VALUES ('Gein', '38 Beach Rd')
```

DELETE 文が失敗した場合、SQL トランスフォーメーションによってデータベースからエラーメッセージが返されます。Data Integration Service は INSERT 文の処理を続行します。

データベースエラーをトラブルシューティングしたり、エラーをエラーを発生させたクエリ文に関連付けたりするには、【**行内のエラー時でも処理を続行する**】 オプションを無効にします。

# SQL トランスフォーメーションによるフィルタの最適化

フィルタ条件がパススルーポートのみを参照し、SQL トランスフォーメーションに副次作用がない場合、Data Integration Service は SQL トランスフォーメーションにフィルタの最適化を適用することができます。

SQL トランスフォーメーションには、次の状況で副次作用があります。

- SQL クエリがデータベースを更新する。SQL クエリには、CREATE、DROP、INSERT、UPDATE、GRANT、REVOKE などの文が含まれます。
- SQL トランスフォーメーションが結果を返さない SELECT 文に対して NULL 行を返す。行には、パススルーポート値、SQL エラー情報、または NUMRowsAffected フィールドが含まれます。

Data Integration Service は、SQL トランスフォーメーションに初期選択の最適化と最適化にプッシュインの方式を適用することができます。

## SQL トランスフォーメーションを使用した初期選択の最適化

フィルタ条件がパススルーポートのみを参照し、SQL トランスフォーメーションに副次作用がない場合、Data Integration Service は SQL トランスフォーメーションを使用して初期選択の最適化を実行することができます。

SQL トランスフォーメーションには、次の状況で副次作用があります。

- SQL クエリがデータベースを更新する。SQL クエリには、CREATE、DROP、INSERT、UPDATE、GRANT、REVOKE などの文が含まれます。
- SQL トランスフォーメーションが結果を返さない SELECT 文に対して NULL 行を返す。行には、パススルーポート値、SQL エラー情報、または NUMRowsAffected フィールドが含まれます。

## SQL トランスフォーメーションによる初期選択の最適化の有効化

SQL トランスフォーメーションに副次作用がない場合は、SQL トランスフォーメーションで初期選択の最適化を有効にします。

1. SQL トランスフォーメーションの [詳細プロパティ] で、[データベース出力のみ返す] オプションを有効にします。
2. SQL トランスフォーメーションの [詳細プロパティ] で、[副次作用あり] をクリアします。
3. SQL トランスフォーメーションに **NumAffectedRows** ポートがある場合は、このポートを削除します。

## SQL トランスフォーメーションによるプッシュイン最適化

最適化にプッシュインを有効にすると、SQL トランスフォーメーションにおけるクエリへのマッピングで、Data Integration Service はフィルタトランスフォーメーションからフィルタロジックをプッシュします。

SQL トランスフォーメーションで最適化にプッシュインを有効にする場合は、以下のルールおよびガイドラインに従います。

- トランスフォーメーション SQL クエリは SELECT 文を含む必要があります。
- トランスフォーメーション SQL クエリは有効なサブクエリである必要があります。
- フィルタ条件は、[SQL エラー] フィールドまたは [NumRowsAffected] フィールドを参照できません。
- 出力ポートの名前は、SQL SELECT 文内のカラムの名前と一致している必要があります。フィルタ条件内で出力ポートを参照すると、Data Integration Service は対応するポート名を SQL クエリにプッシュしま

す。クエリ内のカラムが出力ポート名と一致しない場合は、SQL にエイリアスを追加できます。次に例を示します。SELECT mycolname1 AS portname1, mycolname2 AS portname2

- トランスフォーメーションには副次作用がありません。

## SQL トランスフォーメーションによるプッシュイン最適化の例

SQL トランスフォーメーションは顧客 ID 別に注文を取得します。SQL トランスフォーメーションの後に出現するフィルタトランスフォーメーションは、注文数が 1000 を超えた行のみを返します。

Data Integration Service は、次のフィルタを SQL トランスフォーメーションの SELECT 文にプッシュします。

```
orderAmount > 1000
```

SQL クエリ内の各文は、フィルタが含まれる SELECT 文の個々のサブクエリになります。

次のクエリ文は、元のクエリ文を SELECT 文内のサブクエリとして示しています。

```
SELECT <customerID>, <orderAmount>, ... FROM (original query statements) ALIAS WHERE <orderAmount> > 1000
```

SQL クエリに複数の文がある場合、各文は個々のサブクエリに含まれます。サブクエリの構文は、WHERE 句など、同じになります。

ポート *customerID* および *orderAmount* は、SQL トランスフォーメーションの出力ポートの名前です。サブクエリには、パススルーポート、SQL エラー、または SQL 統計ポートは含まれません。複数のフィルタを SQL トランスフォーメーションにプッシュする場合、WHERE 句にすべてのフィルタが含まれます。

## SQL トランスフォーメーションによるプッシュイン最適化の有効化

最適化にプッシュインを有効にするには、SQL トランスフォーメーションの【詳細プロパティ】タブでプロパティを設定します。

1. 【副次作用あり】をクリアします。
2. 【データベース出力のみ返す】を有効にします。
3. 【最大出力行数】をゼロに設定します。
4. 【最適化にプッシュイン】を有効にします。

# SQL クエリを使った SQL トランスフォーメーションの例

あなたは Hypostores corporation の人事部の開発者です。Hypostores は、従業員の給与情報を人事の従業員データとは別のデータベースで管理しています。人事部では、複数の地域をまたいで、従業員と給与を単一ビューで照会する必要があります。

あなたは、従業員の論理データオブジェクト内の従業員データと給与データを単一ビューに表示する、論理データオブジェクトマッピングを作成する必要があります。

従業員データソースを使用して論理データオブジェクトマッピングを作成します。これに、給与データベースから給与及び入社日を取得する SQL トランスフォーメーションを追加します。

## 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マッピングには、以下のオブジェクトが含まれます。

### Employee テーブル

Human Resources データベース内の、従業員データの入力リレーショナルテーブル。

### Salary テーブル

従業員の給与および入社日が含まれる、Payroll データベース内のテーブル。データベースは Oracle データベースです。

### SQL トランスフォーメーション

各従業員行に対して入社日と給与を取得するトランスフォーメーション。このトランスフォーメーションは Payroll データベースに接続され、データベース内の Salary テーブルに対して SQL クエリを実行します。

### 論理データオブジェクト

従業員データと給与データが結合されたビューを含みます。論理データオブジェクトは、SQL トランスフォーメーションから出力を受け取ります。

### SQLErrors ファイル

SQLErrors ファイルは、データベースの SQL エラーが含まれるフラットファイルです。Data Integration Service は、入力行ごとに少なくとも 1 行を SQLErrors ファイルに書き込みます。SQL エラーが発生しなかった場合、SQLError カラムには NULL が含まれます。SQLErrors ファイルを確認して、エラーをトラブルシューティングします。

## Salary テーブル

Salary テーブルは、Payroll データベース内のリレーショナルテーブルです。このテーブルには、給与部門が管理する従業員データが含まれます。SQL トランスフォーメーションは、Salary テーブルから入社日と従業員の給与を取得します。

以下の表に、Salary テーブルのいくつかの行を示します。

Employee_Num	HireDate	Salary
10	3-May-97	232000
11	11-Sep-01	444000
12	17-Oct-89	656000
13	13-Aug-07	332100

## Employee テーブル

ソースは Human Resources データベースの Employee テーブルです。

以下の表に、Employee テーブルのサンプル行を示します。

EmpID	LastName	FirstName	DeptId	Phone
10	Smith	Martha	FIN	(415) 552-1623
11	Jones	Cynthia	ENG	(415) 552-1744
12	Russell	Cissy	SLS	(415) 552-1656
13	Goyal	Girish	FIN	(415) 552-1656

## SQL トランスフォーメーション

SQL トランスフォーメーションは、Payroll データベースの Salary テーブルから従業員の入社日と給与を取得します。Salary テーブルは Oracle データベース内にあります。

SQL トランスフォーメーションを設定するには、以下の手順に従います。

1. SQL トランスフォーメーションのプロパティを設定します。
2. ポートを定義します。
3. SQL クエリを作成します。
4. SQL トランスフォーメーションのデータベース接続を設定します。

### SQL トランスフォーメーションのプロパティを定義します。

【詳細プロパティ】ビューで、SQL トランスフォーメーションのプロパティを設定します。

以下のプロパティを設定します。

#### データベースタイプ

データベースタイプは Oracle です。ポートを定義するときに、Oracle に対して適用可能なポートデータ型を選択できます。

#### 行内のエラー時でも処理を続行する

無効にします。行で SQL エラーが発生した場合は、処理を停止します。

#### 統計を出力として含める

無効にします。NumRowsAffected 出力ポートを作成しないでください。

### ポートの定義

Employee ソーステーブルのカラムごとに、入力ポートを定義します。入力ポートをカラムのパススルーポートに変更するには、【出力にコピー】を選択します。【出力にコピー】を選択すると、Developer ツールはコピーするポートごとに対応する出力ポートを作成します。

以下の入力パススルーポートを作成します。

名前	タイプ	ネイティブタイプ	精度	スケール	出力にコピー
EmpID	decimal	number(p,2)	4	0	x
LastName	string	varchar2	30	0	x

名前	タイプ	ネイティブタイプ	精度	スケール	出力にコピー
FirstName	string	varchar2	20	0	x
DeptID	string	varchar2	4	0	x
Phone	string	varchar2	16	0	x

SQL トランスフォーメーションでの出力ポートを以下に挙げます。

名前	タイプ	ネイティブタイプ	精度	スケール
EmpID	decimal	number(p,s)	4	0
LastName	string	varchar2	30	0
FirstName	string	varchar2	20	0
DeptID	string	varchar2	4	0
Phone	string	varchar2	16	0
HireDate	date/time	timestamp	29	0
Salary	decimal	number(p,s)	8	2

【出力にコピー】を選択すると、Developer ツールは、作成する各出力行に「\_output」接尾辞を追加します。

入社日と給与のカラムに対して、出力ポートを手動で定義します。SQL トランスフォーメーションは、ポート内の Salary テーブルから入社日と給与のカラムを返します。

## SQL クエリの定義

Salary テーブルから各従業員の入社日と給与を選択する SQL クエリを作成します。

SQL トランスフォーメーションの [SQL] ビューでクエリを定義します。

SQL エディタで以下のクエリを入力します。

```
select HIREDATE,SALARY,from Salary where EMPLOYEE_NUM =?EmpID?
```

Hiredate、Salary、および Employee\_Num は、Salary テーブルのカラム名です。

?EMPID? は、EmpID ポートの値を含むパラメータです。

## データベース接続の定義

【ランタイム】ビューで、SQL トランスフォーメーションの接続先データベースのデータベース接続オブジェクトを選択します。Oracle データベースの接続オブジェクトを選択します。

## 出力

SQLException ポートと EmpID\_output ポートを SQLExceptions フラットファイルに接続します。SQL エラーが発生しなかった場合、SQLException ポートには NULL 値が含まれます。

EmpID 出力ポートと他の出力ポートを論理データオブジェクトに接続します。

SQL トランスフォーメーションは、Employee テーブルのデータが含まれる行を返し、Salary テーブルの入社日と給与を組み込みます。

以下の表に、論理データオブジェクトのいくつかの行を示します。

EmplID	LastName	FirstName	DeptID	Phone	HireDate	Salary
10	Smith	Martha	FIN	(415) 552-1623	19970303 00:00:00	2320.00
11	Jones	Cynthia	ENG	(415) 552-1744	20010911 00:00:00	4440.00
12	Russell	Cissy	SLS	(415) 552-1656	19891017 00:00:00	6560.00
13	Goyal	Girish	FIN	(415) 552-1660	20070813 00:00:00	3210.00

## ストアードプロシージャ

SQL トランスフォーメーションからストアードプロシージャを呼び出すことができます。ストアードプロシージャを使用してリレーショナルデータベースのタスクを自動化することができます。ストアードプロシージャは、ユーザー定義の変数や条件文、その他にも標準の SQL 文がサポートしていない機能を受け付けます。

SQL トランスフォーメーションはリレーショナルデータベースに接続してストアードプロシージャを実行します。SQL トランスフォーメーションは、Oracle、IBM DB2、Microsoft SQL Server、Sybase、および ODBC からストアードプロシージャを呼び出すことができます。ストアードプロシージャはデータベース内に保管されていて、データベース内で実行されます。

ODBC 接続を作成して、Sybase データベースからストアードプロシージャを呼び出します。Windows 以外のオペレーティングシステムの Microsoft SQL Server データベースからストアードプロシージャを呼び出す場合も、ODBC 接続を作成する必要があります。

ストアードプロシージャとは、Transact-SQL や PL-SQL などのデータベースプロシージャ文からなる事前コンパイルされた集合です。ストアードプロシージャの構文は、データベースに基づいて変わります。

ストアードプロシージャを使用して、以下の作業を実行できます。

- データをターゲットデータベースにロードする前にターゲットデータベースの状態をチェックする。
- データベース内に十分な領域があるかどうかを調べる。
- 特殊な計算を行う。
- 値でデータを取得する。
- インデックスを削除および再作成する。

ストアードプロシージャを使うことで、本来ならトランスフォーメーションに含めるようなクエリまたは計算を行うことができます。たとえば、十分にテストした消費税計算用ストアードプロシージャがあれば、同じ計算を式トランスフォーメーションで再作成せずに、そのストアードプロシージャを使用して計算してもかまいません。

ストアードプロシージャは、入力を受け付けてから行の結果セットを返すことができます。ストアードプロシージャは、入力を必要とせず出力を返さない DDL タスクを実行することができます。

SQL トランスフォーメーションを設定して、複数のストアードプロシージャを実行することができます。設定した各ストアードプロシージャに対して、ストアードプロシージャのパラメータを照合するようにトランスフォーメーションのポートを設定します。各ストアードプロシージャがデータを出力ポートに戻すことができます。



ストアードプロシージャを含んだデータベースにはユーザー権限があります。データベースに対してストアードプロシージャを実行するには、権限を持っている必要があります。

**注:** ストアド関数は、関数が単一の値を返すことを除いてストアードプロシージャと同じです。SQL トランスフォーメーションはストアド関数を実行できます。

## ストアードプロシージャのための SQL トランスフォーメーションのポート

SQL トランスフォーメーションの入力ポートと出力ポートは、ストアードプロシージャ内の入力パラメータと出力パラメータに対応します。

ストアードプロシージャをインポートする場合、Developer ツールはデータベース接続からデータベースタイプを決定します。ストアードプロシージャ内のパラメータから SQL トランスフォーメーションの入力ポートと出力ポートを生成します。Developer ツールは、ストアードプロシージャのパラメータから各ポートのネイティブのデータ型を決定します。

SQL トランスフォーメーションを手動で設定する場合、トランスフォーメーションの入力ポートと出力ポートを設定する必要があります。データベースタイプを設定すると、SQL トランスフォーメーションは、入力したデータベースタイプに基づいてポートごとにネイティブのデータ型を変更します。

次のタイプのデータが、SQL トランスフォーメーションとストアードプロシージャの間を通過できます。

### 入力パラメータと出力パラメータ

SQL トランスフォーメーションがパラメータをストアードプロシージャに送り、SQL トランスフォーメーションが入力ポートと出力ポート内のパラメータをストアードプロシージャから受け取ります。

### 戻り値

ストアードプロシージャが戻り値を渡す場合、Developer ツールは [戻り値] ポートを作成します。

### SQL エラー

SQL トランスフォーメーションは、SQL\_Error ポート内のストアードプロシージャからエラーを返します。

## 入力パラメータと出力パラメータ

SQL トランスフォーメーションからストアードプロシージャを呼び出すときは、呼び出し文が参照する各フィールドで、入力ポートまたは出力ポートが識別されます。ストアードプロシージャをインポートすると、Developer ツールがストアードプロシージャの呼び出し文を生成します。インポートしない場合、呼び出し文を手動で設定する必要があります。

トランスフォーメーションの **SQL** ビューで呼び出し文を編集することができます。

呼び出し文のフォーマットは次のとおりです。

```
?RETURN_VALUE? = call <stored proc name>(?Field1?, ?Field2?, . . .)
```

ポート名は疑問符で括ります。ポート名はストアードプロシージャ内のパラメータ名と一致する必要はありません。出力ポートは、SELECT クエリ内のパラメータと同じ順序である必要があります。

INOUT パラメータが入っているストアードプロシージャを使用できます。SQL トランスフォーメーションが入力ポート名で INOUT パラメータを識別します。出力ポートには output\_ というプレフィックスがあります。データ統合サービスが入力ポートと出力ポートを同じパラメータにバインドします。

SQL トランスフォーメーションを結果セットを返すように設定できます。ストアードプロシージャが結果セットを返す場合、Developer ツールは結果セット内のカラムに対して出力ポートを作成できません。ストアードプロシージャをインポートする場合、手動でポートを入力してストアードプロシージャの呼び出しを設定する必要があります。

## 戻り値

戻り値は、ストアードプロシージャのステータスを定義するコードまたはテキストの文字列です。ストアードプロシージャに戻り値がある場合、SQL トランスフォーメーションには【戻り値】ポートがあります。

ほとんどのデータベースは、ストアードプロシージャを実行したあとに戻り値を渡すことができます。戻り値には整数値を含むことができます。またはストアードプロシージャで定義する値を含むことができます。例えば、プロシージャが成功したときにストアードプロシージャが「成功」を返す場合があります。

ストアードプロシージャが単一の戻り値ではなく結果セットを返す場合、SQL トランスフォーメーションはプロシージャから最初の戻り値を受け取ります。

## ストアードプロシージャの結果セット

ストアードプロシージャから結果セットを受け取るように SQL トランスフォーメーションを設定できます。ストアードプロシージャは、結果セット内の複数の行を返します。SQL トランスフォーメーションはマッピングに各行を返すことができます。

次のストアードプロシージャが結果セットを返します。

```
CREATE OR REPLACE FUNCTION fetchEMPinfo
(p_State IN VARCHAR2)
return types.cursorType
AS
my_cursor types.cursorType;
BEGIN
OPEN my_cursor FOR SELECT EMP_ID, NAME, CITY FROM EMP WHERE STATE = p_State ORDER BY EMP_ID;
RETURN my_cursor;
END;
```

ストアードプロシージャをインポートすると、Developer ツールは次の文と同じような構文でストアードプロシージャ呼び出し文を作成します。

```
call FETCHEMPINFO (?P_STATE?)
```

入力パラメータは p\_state です。Developer ツールは出力ポートを作成してくれません。そこで、ストアードプロシージャのパラメータと同じデータ型を使って手動で出力ポートを作成する必要があります。

例えば、resultSet には、EMP\_ID、EMPNAME、および CITY というカラムが含まれています。これらのカラムに対し出力ポートを作成します。

また、SQL 呼び出しを出力カラムで手動で更新する必要もあります。次の構文を使用します。

```
(?EMP_ID?,?EMPNAME?,?CITY?) = call FETCHEMPINFO (?P_STATE?)
```

## 異なるデータベースを使った結果セット

データベースのタイプによって異なる構文を使用した結果セットを返すようにストアードプロシージャを設定します。

### Oracle

Oracle のストアード関数が結果をカーソルで返します。

```
create or replace function sp_ListEmp return types.cursorType
as
l_cursor types.cursorType;
begin
open l_cursor for select ename, empno from emp order by ename;
return l_cursor;
end;
```

Oracle はカーソルを入力パラメータとしても受け入れます。SQL トランスフォーメーションでカーソルを入力パラメータとして設定することはできません。

## Microsoft SQL Server

Microsoft SQL Server のストアドプロシージャは、プロシージャの本文の選択文を使って、またはテーブルとして明示的に宣言した戻りタイプを使って、結果セットのストアドプロシージャを返します。

```
Create PROCEDURE InOut(
@inout varchar(100) OUT
)
AS
BEGIN
set @inout = concat(@inout, '__')
select * from mytable;
END
```

## IBM DB2

IBM DB2 のストアドプロシージャは、開いたカーソルで結果セットを返すことができます。返す結果セットの数は RESULT SET 句の中で宣言されます。ストアドプロシージャがカーソルを開き、それを返します。次の例では、2 つの開いたカーソルを返しています。

```
CREATE PROCEDURE TESTMULTIRS
 (IN i_cmacct CHARACTER(5))
RESULT SETS 2
LANGUAGE SQL
BEGIN

DECLARE csnum INTEGER;

--Declare serial cursors to consume less resources
--You do not need a rollable cursor.

DECLARE getDeptNo CHAR(50); --Be careful with the estimated length.
DECLARE getDeptName CHAR(200);
DECLARE c1 CURSOR WITH RETURN FOR s1;
SET getDeptNo = 'SELECT DEPTNO FROM DEPT';
SET getDeptName = 'SELECT DEPTNAME FROM DEPT';

PREPARE s1 FROM getDeptNo;
OPEN c1;

END;
```

## Sybase

Sybase のストアドプロシージャは、プロシージャ本文の選択文を使用するか、または明示的にテーブルとして宣言された戻りタイプを使用して、結果セットのストアドプロシージャを返します。

```
CREATE PROCEDURE FETCHEMPINFO
(
 @p_State VARCHAR(5),
 @e_id INT OUTPUT,
 @e_name VARCHAR(50) OUTPUT
)
AS
BEGIN
SET NOCOUNT ON
SELECT EMP_ID, NAME FROM EMP WHERE STATE = @p_State ORDER BY EMP_ID
SET NOCOUNT OFF
SELECT @e_id AS EMP_ID, @e_name AS NAME
RETURN
END
GO

--Configure the following variables to execute the procedure.

DECLARE @p_State VARCHAR(5)
DECLARE @EMPID int
```

```

DECLARE @EMPNAME varchar(50)

SET @p_State = 'CA'
exec FETCHEMPINFO @p_State, @e_id = @EMPID, @e_name = @EMPNAME
GO

```

## 結果セットの行

一部のストアードプロシージャは、結果セット行に加えて出力パラメータを返します。SQL トランスフォーメーションは、最後の行に出力パラメータを返します。結果セット行に単回出現の出力パラメータは含めません。

例えば、従業員 ID を受け取り、出力パラメータ 1 の従業員名と出力パラメータ 2 の部門名を返すストアードプロシージャを書き込みます。ストアードプロシージャはその年の従業員の病欠の日ごとにも行を返します。その行には、日付、時間数、欠勤の理由が入っています。

結果セットには、従業員ごとに異なる数の行が入っています。結果セットの各行には空の従業員名と部門名があります。SQL トランスフォーメーションは結果セットの後に従業員名と部門名を返します。従業員名と部門名が最後の行に表示されます。

## ストアードプロシージャの例

ストアードプロシージャを呼び出して SQL トランスフォーメーションにデータを返すことができます。

次のストアードプロシージャが従業員番号を受け取って、従業員番号と従業員名の入った行を 1 つ返します。

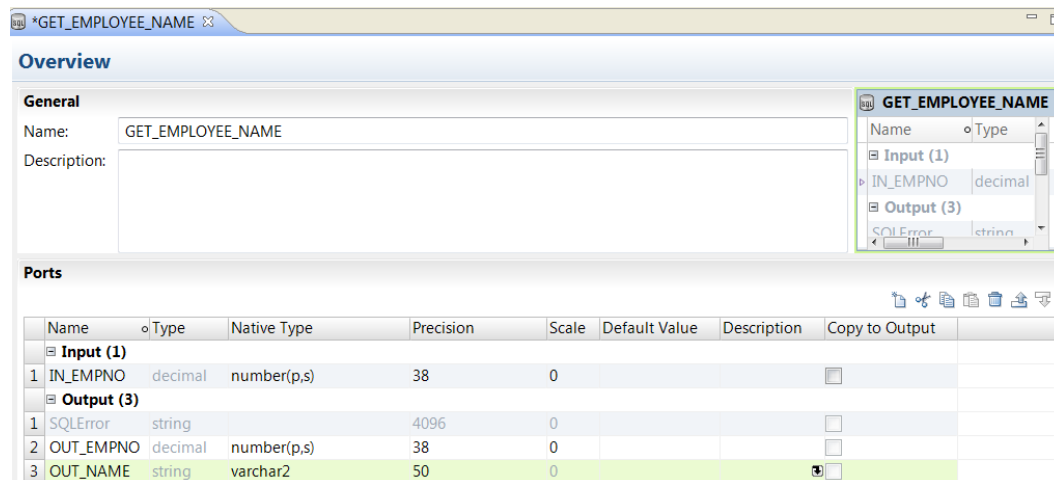
```

CREATE OR REPLACE PROCEDURE SP_GETNAME
(IN_EMPNO IN NUMBER, OUT_EMPNO NUMBER, OUT_NAME OUT STRING)
AS
BEGIN
SELECT EMP_KEY,EMP_NAME into OUT_EMPNO , OUT_NAME from EMP_TABLE where EMP_KEY=IN_EMPNO;
END;/"

```

SQL トランスフォーメーションを作成するには、ストアードプロシージャをインポートします。Developer ツールは入力ポートと出力ポートを作成します。ポート名は、ストアードプロシージャ内のパラメータ名と同じです。

次の図に、SQL トランスフォーメーションのポートを示します。



Developer ツールが従業員名を取得するためのストアードプロシージャの呼び出しを次のように作成します。

```
call SP_GETNAME (?IN_EMPNO?,?OUT_EMPNO?,?OUT_NAME?)
```

SQL エディタでストアードプロシージャ呼び出しを確認できます。SQL エラーがあれば、それが SQL\_Error ポートに表示されます。

# SQL トランスフォーメーションの接続

トランスフォーメーションのランタイムプロパティで SQL トランスフォーメーションの接続を設定します。トランスフォーメーションの作成時に接続を指定しなかった場合、ランタイム接続の設定が必要になる可能性があります。

SQL トランスフォーメーションの作成のために選択した接続とは異なる接続名を定義できます。SQL トランスフォーメーションの【詳細】プロパティのデータベースタイプと同じデータベースタイプの接続を選択する必要があります。

SQL トランスフォーメーションの接続名のパラメータを設定できます。マッピングの【パラメータ】ビューでパラメータを定義してから、ランタイム接続に割り当てる必要があります。

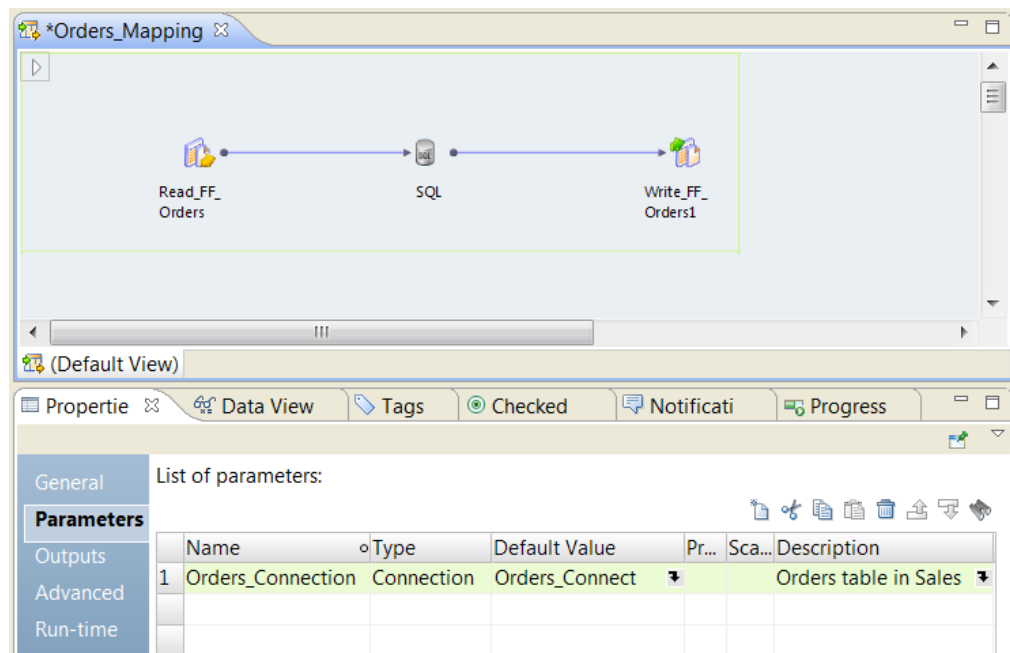
## 接続名パラメータの作成

SQL トランスフォーメーションのランタイム接続名にユーザー定義のパラメータを指定できます。Developer tool は、トランスフォーメーションパラメータの代わりに、接続のマッピングパラメータを作成します。

1. SQL トランスフォーメーションを含むマッピングを作成します。
2. SQL トランスフォーメーションの【ランタイム】タブをクリックします。
3. 【接続名】で選択矢印をクリックして、【パラメータの割り当て】を選択します。
4. 【パラメータの割り当て】ダイアログボックスで、【新規】をクリックします。
5. 【パラメータ】ダイアログボックスで、接続パラメータの名前と説明を入力します。パラメータのタイプのデフォルトは接続です。
6. 【パラメータ】ダイアログボックスおよび【パラメータの割り当て】ダイアログボックスで、【OK】をクリックします。

Developer tool によってマッピングパラメータが作成され、接続名に割り当てられます。パラメータ名はランタイムプロパティに表示されます。

7. マッピングパラメータの一覧を表示するには、エディタの内部をクリックし、【マッピングパラメータ】タブをクリックします。



# SQL トランスフォーメーションの手動による作成

SQL トランスフォーメーションは手動で作成することができます。SQL クエリを実行するトランスフォーメーションを設定するときはトランスフォーメーションを手動で作成します。また、プロシージャがインポートに利用できないときにストアードプロシージャを呼び出すトランスフォーメーションも手動で作成することがあります。手動でトランスフォーメーションを作成するときは、入力ポートと出力ポートを設定し、SQL エディタで SQL 文をタイプ入力します。

1. **[Object Explorer]** ビューで、プロジェクトまたはフォルダを選択します。
2. **[ファイル]** > **[新規]** > **[トランスフォーメーション]** をクリックします。  
**[新規]** ダイアログボックスが表示されます。
3. SQL トランスフォーメーションを選択します。
4. **[次へ]** をクリックします。
5. **[空として作成]** を選択します。
6. トランスフォーメーションの名前を入力し、トランスフォーメーションのリポジトリの場所を入力します。
7. **[完了]** をクリックします。
8. **[概要]** ビューをクリックして、ポートをトランスフォーメーションに追加します。
9. 入力ポートを追加するには、**[ポート]** パネル内の **[入力]** をクリックして、どこにポートを追加するかを示します。**[新規]** ボタンをクリックし、ポート名、ネイティブタイプ、精度を入力します。  
デフォルトのデータベースタイプは Oracle です。**[詳細]** ビューに入ってデータベースのタイプを変更しない限り、Developer ツールには Oracle データベースのネイティブのタイプが表示されます。
10. 出力ポートを追加するには、ポートを追加する前に **[ポート]** パネル内の **[出力]** をクリックします。**[新規]** ボタンをクリックし、ポート名、ネイティブタイプ、精度を入力します。  
**SQLException** ポートがデフォルトで最初の出力ポートです。
11. **[詳細]** ビューで、SQL トランスフォーメーションの接続先にするデータベースのタイプを選択します。その他のエラー処理用詳細プロパティ、およびその他の任意設定プロパティを設定します。  
データベースのタイプを選択すると、Developer ツールが **[概要]** ビューで各ポートのネイティブのデータ型を変更します。
12. **SQL** ビューで、SQL クエリまたはストアードプロシージャの呼び出しを入力します。**SQL エディタ**で、パラメータ結合または文字列置き換えのためのポートを選択します。  
ストアードプロシージャが結果セットを返す場合、次の構文に類似する構文でストアードプロシージャの呼び出しを入力する必要があります。( ?Field1?, ?Field2?, ?Field3?) = **call** Stored\_Procedure\_Name (?Input\_Parm?)

## 関連項目：

- [「SQL クエリの定義」 \(ページ 627\)](#)

# ストアードプロシージャから SQL トランスフォーメーションの作成

データベース接続からストアードプロシージャをインポートすることにより、SQL トランスフォーメーションを設定することができます。

1. **[Object Explorer]** ビューで、プロジェクトまたはフォルダを選択します。

2. **【ファイル】 > 【新規】 > 【トランスフォーメーション】** をクリックします。

**【新規】** ダイアログボックスが表示されます。

3. SQL トランスフォーメーションを選択します。
4. **【次へ】** をクリックします。
5. **【既存のストアドプロシージャから作成】** を選択します。
6. データベース接続を探して選択します。
7. インポートするストアドプロシージャを探して選択します。
8. トランスフォーメーションの名前と場所を入力します。
9. **【完了】** をクリックします。

Developer ツールは、ポートとストアドプロシージャの呼び出しを作成します。

10. ストアドプロシージャが結果セットを返したら、出力ポートを手動で追加してから、ストアドプロシージャの呼び出しを再設定する必要があります。
  - a. **【概要】** ビューで、**【ポート】** パネル内の**【出力】** をクリックします。**【新規】** ボタンをクリックし、出力名、ネイティブのタイプ、精度を入力します。
  - b. **【SQL】** ビューで、ストアドプロシージャの呼び出しを次の構文を使用するように変更します: (?Field1?,?Field2?,?Field3?) = **call** Stored\_Procedure\_Name (?Input\_Parm?)  
SQL エディタの**【パラメータのバインド】** ポートリストから、入力パラメータと出力パラメータを選択することができます。

## 第 43 章

# 標準化トランスフォーメーション

この章では、以下の項目について説明します。

- [標準化トランスフォーメーションの概要, 636 ページ](#)
- [標準化ストラテジ, 636 ページ](#)
- [標準化のプロパティ, 637 ページ](#)
- [標準化ストラテジの設定, 638 ページ](#)
- [標準化トランスフォーメーションの詳細プロパティ, 638 ページ](#)
- [非ネイティブ環境での標準化トランスフォーメーション, 639 ページ](#)

## 標準化トランスフォーメーションの概要

標準化トランスフォーメーションは、入力文字列を調べ、それらの文字列の標準化されたバージョンを作成するパッシブなトランスフォーメーションです。

標準化トランスフォーメーションでは、入力文字列の標準化されたバージョンを含むカラムを作成します。それらのカラムを作成するときに、入力データに含まれる文字列を置き換えたり削除したりできます。

例えば、標準化トランスフォーメーションを使用して、Street、St.、および STR という文字列を含むアドレスデータのカラムを調べ、それらの文字列のすべての出現箇所を St という文字列に置き換えることができます。

標準化トランスフォーメーションには、複数の標準化ストラテジを作成できます。それぞれのストラテジに複数の標準化操作を含めることができます。標準化トランスフォーメーションには、ストラテジを作成するためのウィザードが用意されています。

## 標準化ストラテジ

入力文字列の標準化されたバージョンを含むカラムを作成するには、標準化ストラテジを使用します。

標準化ストラテジを設定するときは、操作を 1 つ以上追加し、操作ごとに特定の標準化タスクを実装します。

標準化ストラテジには、以下のタイプの操作を追加できます。

### 参照テーブルの一致を有効な値で置換

参照テーブルの値に一致する文字列を、参照テーブルの"有効"値で置き換えます。

### 参照テーブルの一致をカスタム文字列で置換

参照テーブルの値に一致する文字列を、ユーザー定義の置き換え文字列で置き換えます。



### 参照テーブルの一致を削除

参照テーブルの値に一致する文字列を削除します。

### カスタム文字列の置換

ユーザー定義の文字列を、ユーザー定義の置き換え文字列で置き換えます。

### カスタム文字列の削除

ユーザー定義の文字列を削除します。

**重要:** 操作の順序は変更が可能です。各操作で前の操作の結果を読み取るため、操作の順序によってストラテジの出力が変わることがあります。

## 標準化のプロパティ

標準化のストラテジや操作のプロパティを設定するには、標準化トランスフォーメーションの **【ストラテジ】** ビューを選択します。

### ストラテジのプロパティ

ストラテジのプロパティは、ストラテジ内のすべての操作に適用されます。設定できるストラテジのプロパティを次に示します。

#### 複数のスペースの削除

連続する複数のスペースを 1 つのスペースに置き換えます。

#### 末尾および先頭のスペースの削除

データ文字列の先頭および末尾からスペースを削除します。

#### 区切り文字

検索トークンを定義する区切り文字を指定します。例えば、[セミコロン] を選択した場合、標準化トランスフォーメーションで文字列 "oranges;apples;" を検索すると、"oranges" と "apples" の各文字列が検出されます。このトランスフォーメーションは、デフォルトでスペースによるデリミタを使用します。

### 操作のプロパティ

以下のタイプの標準化操作のプロパティを設定できます。

#### 参照テーブル操作

参照テーブル操作のプロパティを次に示します。

- **参照テーブル。** データの標準化に使用する参照テーブルを指定します。 **【参照】** をクリックして参照テーブルを選択します。
- **大文字小文字の区別。** 入力文字列を参照テーブルのエントリと照合するときに大文字と小文字を区別するかどうかを指定します。
- **置換後の文字列。** 参照テーブルのエントリに一致する入力文字列を、指定したテキストに置き換えます。置換操作にのみ適用されます。
- **スコープ。** 参照テーブルの値を含む入力文字列の部分を指定します。

#### カスタム文字列操作

カスタム文字列操作のプロパティを次に示します。

- **トークンの照合。** 入力データ内で検索する検索文字列を定義します。

- **置換後の文字列。** 指定した検索文字列に一致する入力文字列を置き換えます。置換操作にのみ適用されます。
- **スコープ。** 検索する入力文字列の部分を指定します。

## 標準化ストラテジの設定

標準化ストラテジを設定するには、標準化トランスフォーメーションの **【ストラテジ】** ビューで設定を編集します。

1. **【ストラテジ】** ビューを選択します。
2. **【新規】** をクリックします。  
新しいストラテジ ウィザードが開きます。
3. **【入力】** フィールドをクリックして、ストラテジのポートを選択します。
4. ストラテジのプロパティを設定し、**【次へ】** をクリックします。
5. 操作を選択し、**【次へ】** をクリックします。
6. 操作のプロパティを設定します。
7. 必要に応じて、**【次へ】** をクリックして、その他の操作をストラテジに追加します。
8. ストラテジにすべての操作を追加したら、**【完了】** をクリックします。
9. 必要に応じて、その他のストラテジをトランスフォーメーションに追加します。
10. 必要に応じて、ストラテジや操作を処理する順序を変更します。ストラテジまたは操作を選択し、**【上に移動】** または **【下に移動】** をクリックします。

## 標準化トランスフォーメーションの詳細プロパティ

Data Integration Service で標準化トランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

**【詳細】** タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

# 非ネイティブ環境での標準化トランスフォーメーション

非ネイティブ環境での標準化トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングおよびストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。サポートしません。

## 第 44 章

# 共有体トランスフォーメーション

この章では、以下の項目について説明します。

- [共有体トランスフォーメーションの概要, 640 ページ](#)
- [グループおよびポート, 641 ページ](#)
- [共有体トランスフォーメーションの詳細プロパティ, 642 ページ](#)
- [共有体トランスフォーメーションの処理, 642 ページ](#)
- [共有体トランスフォーメーションの作成, 642 ページ](#)
- [非ネイティブ環境での共有体トランスフォーメーション, 643 ページ](#)

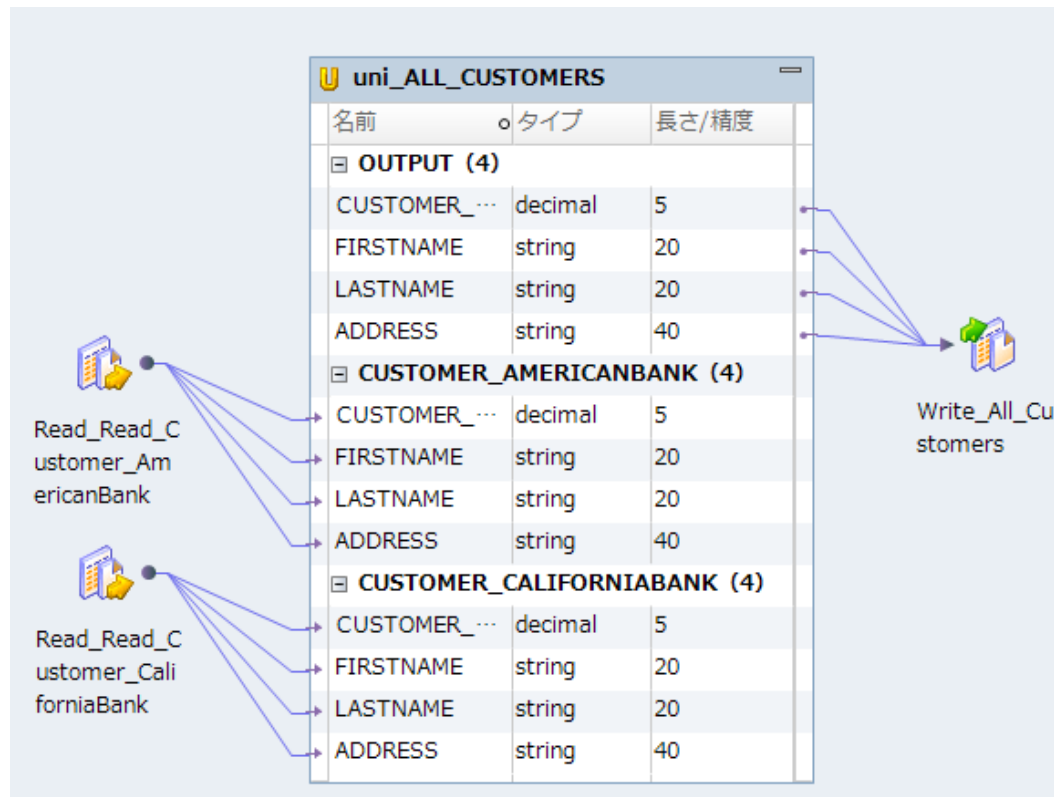
## 共有体トランスフォーメーションの概要

共有体トランスフォーメーションを使用して、複数のパイプラインまたはパイプラインブランチから、1つのパイプラインブランチにデータを統合します。

共有体トランスフォーメーションは、複数の入力グループと1つの出力グループを持つアクティブなトランスフォーメーションです。一致するポートを持つソースを統合し、入力グループと同じポート構造を持つ出力グループを通じてデータを渡します。共有体トランスフォーメーションは、重複行を削除せずに Developer ツールで複数のソースのデータを統合する場合に使用します。

たとえば、アメリカ銀行とカリフォルニア銀行の顧客口座データを組み合わせるとします。共有体トランスフォーメーションを含むマッピングを作成することで、ソースオブジェクトからのデータをマージし、ターゲットオブジェクトに書き込むことができます。

次の図に、共有体トランスフォーメーションを使ったマッピングを示します。



## グループおよびポート

共有体トランスフォーメーションには複数の入力グループと 1 つの出力グループがあります。入力グループは作成が可能で、いくつでも作成することができます。出力グループは、Developer ツールによって 1 つだけ作成されます。出力グループを作成、編集、または削除することはできません。各グループには一致するポートが必要です。

ポートを作成するには、トランスフォーメーションからコピーするか手動で作成します。ポートを作成すると、Developer ツールにより、各入力グループに入力ポートが作成され、出力グループに出力ポートが作成されます。Developer ツールでは、入力ポートと出力ポートのそれぞれに対してユーザーが指定した出力ポート名が使用されます。また、データ型、精度、位取りなどのメタデータは、ポートごとに同じものが使用されます。

入力グループは、1 つのパイプラインの異なるブランチ、あるいは異なるソースパイプラインから接続することができます。共有体トランスフォーメーションをマッピングに追加する際には、すべての入力グループで同じポートに接続していることを確認する必要があります。1 つの入力グループのポートは接続していても、別の入力グループの同じポートを接続していない場合、Data Integration Service は未接続のポートに NULL を渡します。

# 共有体トランスフォーメーションの詳細プロパティ

Data Integration Service での共有体トランスフォーメーションのログ詳細の表示方法を指定できるプロパティを設定します。

ログに表示するトレースレベルを設定できます。

【詳細】 タブでは、以下のプロパティを設定します。

## トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

# 共有体トランスフォーメーションの処理

共有体トランスフォーメーションを使用して、複数のパイプラインまたはパイプラインブランチから、1つのパイプラインブランチにデータを統合します。Data Integration Service は、すべての入力グループを並列に処理します。また、共有体トランスフォーメーションに接続されたソースを読み込むと同時に、データブロックをトランスフォーメーションの入力グループに渡します。共有体トランスフォーメーションは、Data Integration Service からブロックを受け取る順番に基づいてデータブロックを処理します。入力グループの入力データはブロックされません。

# 共有体トランスフォーメーションの作成

共有体トランスフォーメーションは、再利用可能または再利用不可能なトランスフォーメーションのどちらとしても作成できます。

## 再利用可能な共有体トランスフォーメーションの作成

複数のマッピングまたはマプレットで使用する場合は、再利用可能な共有体トランスフォーメーションを作成します。

1. **【Object Explorer】** ビューで、プロジェクトまたはフォルダを選択します。
2. **【ファイル】 > 【新規】 > 【トランスフォーメーション】** をクリックします。  
**【新規】** ダイアログボックスが表示されます。
3. 共有体トランスフォーメーションを選択します。
4. **【次へ】** をクリックします。
5. トランスフォーメーションの名前を入力します。
6. **【完了】** をクリックします。  
トランスフォーメーションがエディタに表示されます。
7. **【新規】** ボタンをクリックして、トランスフォーメーションにポートを追加します。
8. ポートを編集して、名前、データ型、および精度を設定します。
9. **【グループ】** ビューを選択します。
10. **【新規】** ボタンをクリックして、入力グループを追加します。

11. **【詳細】** ビューをクリックし、トランスフォーメーションのプロパティを編集します。

## 再利用不可能な共有体トランスフォーメーションの作成

再利用不可能なトランスフォーメーションは、特定のマッピングまたはマプレットで作成します。

1. マッピングまたはマプレットで、トランスフォーメーションパレットからエディタに共有体トランスフォーメーションをドラッグします。  
トランスフォーメーションがエディタに表示されます。
2. **【全般】** タブで、トランスフォーメーションの名前と説明を編集します。
3. アップストリームトランスフォーメーションのポートをすべて選択し、共有体トランスフォーメーションにドラッグします。ドラッグしたポートが、共有体トランスフォーメーションの入力グループおよび出力グループのポートとして表示されます。
4. **【プロパティ】** ビューの **【グループ】** タブで **【新規】** をクリックして、入力グループを追加します。  
既存の入力グループと同様のポートを含む入力グループがもう 1 つ表示されます。
5. 共有体トランスフォーメーションの出力グループのポートを選択し、マッピング内のダウンストリームトランスフォーメーションにドラッグします。

## 非ネイティブ環境での共有体トランスフォーメーション

非ネイティブ環境での共有体トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングで制限付きでサポートされます。
- Databricks Spark エンジン。制限なくサポートされます。

## ストリーミングマッピングでの共有体トランスフォーメーション

ストリーミングマッピングにはマッピング検証の制限があります。

### マッピングの検査

マッピング検証は、次の場合に失敗します。

- トランスフォーメーションが、ストリーミングパイプラインと非ストリーミングパイプラインからのデータを統合するように設定されている。

## Databricks Spark エンジンでの共有体トランスフォーメーション

## 第 45 章

# アップデートストラテジトランスフォーメーション

この章では、以下の項目について説明します。

- [アップデートストラテジトランスフォーメーションの概要, 644 ページ](#)
- [動的マッピングでのアップデートストラテジトランスフォーメーション, 645 ページ](#)
- [マッピング内の行のフラグ設定, 645 ページ](#)
- [個々のターゲットに対する更新オプションの指定, 647 ページ](#)
- [非ネイティブ環境でのアップデートストラテジトランスフォーメーション, 648 ページ](#)

## アップデートストラテジトランスフォーメーションの概要

アップデートストラテジトランスフォーメーションは、挿入、更新、または拒否のフラグを行に設定するアクティブなトランスフォーメーションです。アップデートストラテジトランスフォーメーションを使用して、適用する条件に基づいてターゲットの既存の行の変更を制御します。

アップデートストラテジトランスフォーメーションはアクティブなトランスフォーメーションであるため、通過する行の数を変更してしまう可能性があります。アップデートストラテジトランスフォーメーションは、特定の条件を満たしているかどうかを確認するために各行をテストし、それに応じて行にフラグを設定します。アップデートストラテジトランスフォーメーションは、挿入、更新、または削除のフラグを設定する行を次のトランスフォーメーションに渡します。拒否のフラグが設定された行を次のトランスフォーメーションに渡すか、拒否のフラグが設定された行を削除するか、トランスフォーメーションを設定できます。

例えば、アップデートストラテジトランスフォーメーションを用いて、全顧客の行に対して、メールアドレスが変更された際に更新フラグを設定したり、全従業員の行に対して、社員でなくなった者については拒否フラグを設定したりできます。

アップデートストラテジトランスフォーメーションを使用して、Spark エンジン上でマッピングを実行するときに、結果をリレーショナルデータベースに書き込むことができます。マッピングでは、JDBC 接続文字列を使用します。



## アップデートストラテジの設定

更新方式を定義するには、以下の手順を実行します。

1. マッピング内の行に対する挿入、更新、削除、または拒否のフラグ設定を制御する場合は、マッピングにアップデートストラテジトランスフォーメーションを追加します。アップデートストラテジトランスフォーメーションを使用して、同じターゲットに対する行に異なるデータベース操作のフラグを設定したり、行を拒否したりします。
2. マッピングの設定時に、個々のターゲットの挿入オプション、更新オプション、および削除オプションを定義します。ターゲット単位で、挿入または削除のフラグが設定されたすべての行に対して、挿入および削除を許可または禁止できます。更新のフラグが設定されたすべての行に対して更新を処理するさまざまな方法を選択できます。
3. Spark ランタイムエンジンでマッピングを実行する場合は、アップデートストラテジトランスフォーメーションの詳細プロパティで [Hive マージの使用] を選択できます。クエリで INSERT、UPDATE または DELETE の代わりに MERGE 文を使用すると、通常は処理がより効率的になります。

## 動的マッピングでのアップデートストラテジトランスフォーメーション

動的マッピングでアップデートストラテジトランスフォーメーションを使用できます。アグリゲータトランスフォーメーションで動的ポートを設定し、生成されたポートを参照することができます。

アップデートストラテジトランスフォーメーションでは、動的ポートまたは生成されたポートを参照できます。ただし、生成されたポートが実行時に存在しない場合は、マッピングが失敗します。

アップデートストラテジ式で動的ポートを使用する場合、動的ポートは生成されたポートを 1 つだけ持つことができます。

アップデートストラテジ式はパラメータ化できます。式タイプパラメータを使用して、式全体をデフォルトのパラメータ値として入力します。

## マッピング内の行のフラグ設定

個々の行に挿入、更新、削除、または拒否のフラグを設定するには、アップデートストラテジトランスフォーメーションを使用します。

個々の行が特定の条件を満たしているかどうかを見るために、各行をテストするアップデートストラテジトランスフォーメーションを使用します。次に、各行に数値コードを割り当て、特定のデータベース操作に関して行にフラグを設定します。

以下の表に、各データベース操作を表す定数とその数値を示します。

操作	定数	数値
挿入	DD_INSERT	0
更新	DD_UPDATE	1

操作	定数	数値
削除	DD_DELETE	2
Reject	DD_REJECT	3

Data Integration Service は、上記以外の値をすべて挿入として扱います。

## アップデートストラテジ式

式エディタでアップデートストラテジ式を入力します。

アップデートストラテジ式は、トランスフォーメーション言語の IIF または DECODE 関数を使用して、各行をテストします。たとえば次の IIF 文は、入力日付が適用日付よりあとの場合は行に拒否のフラグを設定します。そうでない場合、この文は行に更新のフラグを設定します。

```
IIF((ENTRY_DATE > APPLY_DATE), DD_REJECT, DD_UPDATE)
```

アップデートストラテジ式内でパラメータを設定できます。式エディタでパラメータを作成するか、パラメータを参照します。

トランスフォーメーションが動的マッピング内にある場合、トランスフォーメーションで生成されたフィールドが変わる可能性があります。完全なアップデートストラテジ式をパラメータ化できます。パラメータを使用して式を定義した場合、Developer tool は式を検証できません。式パラメータに別のパラメータを含めることはできません。

## アップデートストラテジトランスフォーメーションの詳細プロパティ

Data Integration Service でのアップデートストラテジトランスフォーメーションのデータの処理方法を指定するには、詳細プロパティを設定します。

[詳細] タブで、アップデートストラテジトランスフォーメーションの以下の詳細プロパティを定義できます。

### 拒否された行の転送

アップデートストラテジトランスフォーメーションが、拒否された行を次のトランスフォーメーションに渡すか、拒否された行を削除するかを決定します。デフォルトでは、Data Integration Service は拒否された行を次のトランスフォーメーションに転送します。Data Integration Service は、行に拒否のフラグを設定し、拒否ファイルに書き込みます。[拒否された行の転送] を選択しなかった場合、Data Integration Service は拒否された行を削除してマッピングログファイルに書き込みます。

### Hive マージの使用

アップデートストラテジトランスフォーメーションが Hive マージを使用して、Spark でマッピングを実行するときに Hive ターゲット上でアップデートを実行するかどうかを決定します。クエリで INSERT、UPDATE または DELETE の代わりに MERGE 文を使用すると、処理がより効率的になります。

マッピングは Hive MERGE オプションを無視し、データ統合サービスは INSERT、UPDATE および DELETE を使用して、次のシナリオで操作を実行します。

- マッピングは Blaze または Hive で実行されている。
- MERGE が、特定の Hadoop ディストリビューションの Hive 実装によって制限されているシナリオ。

マッピングのログには、制限事項が結果に影響したかどうかを含む、操作の結果が含まれます。

パーティション化カラムまたはバケット化カラムに更新が影響した場合、カラムへの更新は省略されます。

**注:** Developer tool とデータ統合サービスは、この制限に対して検証を行いません。アップデートストラテジの式がこれらの制限事項に違反した場合、マッピングが予期しない結果になることがあります。

#### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは [Normal] です。

## アグリゲータトランスフォーメーションとアップデートストラテジトランスフォーメーション

同じパイプライン内でアグリゲータトランスフォーメーションとアップデートストラテジトランスフォーメーションを連結する場合はアグリゲータを配置してからアップデートストラテジトランスフォーメーションを配置します。この順番では、Data Integration Service は集計計算を実行し、計算結果を含む行に、挿入、更新、削除、または拒否のフラグを設定します。

アグリゲータトランスフォーメーションの前にアップデートストラテジトランスフォーメーションを配置する場合は、アグリゲータトランスフォーメーションが、異なる演算子をフラグ付された行をどのように処理するかを考慮する必要があります。この順番では、Data Integration Service は挿入、更新、削除、または拒否のフラグを行に設定してから集計計算を実行します。行にどのようにフラグを設定するかによって、アグリゲータトランスフォーメーションが計算で使用する行の値をどのように扱うかが決まります。例えば、行に削除のフラグを設定した後で、その行を使用して合計を計算すると、Data Integration Service はこの行の値を合計から除算します。行に拒否のフラグを設定した後で、その行を使用して合計を計算すると、Data Integration Service はこの行の値を合計には加えません。行に挿入または更新のフラグを設定した後で、その行を使用して合計を計算すると、Data Integration Service はこの行の値を合計に加えます。

## 個々のターゲットに対する更新オプションの指定

アップデートストラテジトランスフォーメーションを使用して特定のデータベース操作の行ごとにフラグを設定した後、マッピング内のターゲットごとに挿入、更新、および削除オプションを定義します。挿入または削除のフラグが設定された行に対して、挿入または削除を禁止できます。更新のフラグが設定されたすべての行に対して更新を処理するさまざまな方法を選択できます。

マッピング内のターゲットデータオブジェクトの詳細プロパティでアップデートストラテジオプションを定義します。次のアップデートストラテジオプションを設定できます。

#### 挿入

ターゲットに挿入するようにフラグが設定されたすべての行を挿入します。デフォルトでは有効になっています。

#### 削除

ターゲットから削除するようにフラグが設定されたすべての行を削除します。デフォルトでは有効になっています。

#### アップデートストラテジ

既存の行のアップデートストラテジ。次のいずれかのストラテジを選択します。

- **[更新時に更新]**。更新のフラグが設定されたすべての行を更新します。これがデフォルト値です。
- **[挿入時に更新]**。更新のフラグが設定されたすべての行を挿入します。
- **[更新しない場合は挿入]**。更新のフラグが設定されたすべての行がターゲット内に存在している場合、これらの行を更新した後、挿入のマークが付いている残りの行を挿入します。

## テーブルの切り詰め

データをロードする前にターゲットを切り詰めます。デフォルトでは無効になっています。

# 非ネイティブ環境でのアップデートストラテジトランスフォーメーション

非ネイティブ環境でのアップデートストラテジトランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限付きのサポート。
- Spark エンジン。バッチマッピングで制限付きでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

**注:** アップデートストラテジトランスフォーメーションは、Hive ACID をサポートする Hadoop ディストリビューションでのみサポートされます。

## Blaze エンジンでのアップデートストラテジトランスフォーメーション

Hive ACID をサポートする Hadoop ディストリビューションで、アップデートストラテジトランスフォーメーションを使用できます。

Blaze エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### 一般的な制限

アップデートストラテジトランスフォーメーションが、同じプライマリキー値に対して複数の行の更新を受け取った場合、トランスフォーメーションは 1 つのランダムな行を選択してターゲットを更新します。

複数のアップデートストラテジトランスフォーメーションが、同じターゲットの異なるインスタンスに書き込む場合、ターゲットデータが予測できなくなる場合があります。

Blaze エンジンは、削除、更新、挿入の順番で操作を実行します。アップデートストラテジトランスフォーメーションが受け取った順番では行を処理しません。

Hive ターゲットは、更新を常に更新操作として実行します。Hive ターゲットは、[更新しない場合は挿入] または [挿入時に更新] をサポートしません。

### マッピング検証とコンパイル検証

マッピング検証は、次の場合に失敗します。

- アップデートストラテジトランスフォーメーションが、複数のターゲットに接続されている。
- アップデートストラテジトランスフォーメーションが、ターゲットの直前に配置されていない。
- アップデートストラテジターゲットが Hive ターゲットではない。
- アップデートストラテジトランスフォーメーションのターゲットが外部の ACID テーブルである。
- ターゲットにプライマリキーが含まれていない。

- 実行時にターゲットテーブルを切り詰めるための Hive ターゲットプロパティが有効になっている。
- 実行時にターゲットテーブルを作成または置き換えるための Hive ターゲットプロパティが有効になっている。

マッピングは、次の場合に失敗します。

- ターゲットが ORC パケット化されていない。
- 変更後の Hive ターゲットで、実際のテーブルより行が少なくなっている。

次の場合に、コンパイル検証エラーが発生し、マッピングの実行が停止されます。

- Hive バージョンが 0.14 より古い。
- ターゲットテーブルがトランザクションに対して有効化されていない。

## Hive ターゲットテーブルの使用

アップデートストラテジトランスフォーメーションと一緒に Hive ターゲットテーブルを使用するには、Hive のデータ定義言語の句 TBLPROPERTIES ("transactional"="true")を使用して、Hive ターゲットテーブルを作成する必要があります。

Hive ターゲットと一緒にアップデートストラテジトランスフォーメーションを使用するには、Hadoop 接続に関連付けられた hive-site.xml 設定セットに次のプロパティが設定されていることを確認します。

```
hive.support.concurrency true
hive.enforce.bucketing true
hive.exec.dynamic.partition.mode nonstrict
hive.txn.manager org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on true
hive.compactor.worker.threads 1
```

## Spark エンジンでのアップデートストラテジトランスフォーメーション

Hive ACID をサポートする Hadoop ディストリビューションで、アップデートストラテジトランスフォーメーションを使用できます。

アップデートストラテジトランスフォーメーションを使用して、マッピングの結果を JDBC 準拠のリレーショナルターゲットに書き込むこともできます。

Spark エンジンの処理ルールには、データ統合サービスの処理ルールと異なるものがあります。

### Hive ターゲットの一般的な制限

アップデートストラテジトランスフォーメーションは、ターゲットが Hive テーブルまたは JDBC 準拠のテーブルであるときは、拒否された行を次のトランスフォーメーションに転送しません。

アップデートストラテジトランスフォーメーションが、同じプライマリーキー値に対して複数の行の更新を受け取った場合、トランスフォーメーションは 1 つのランダムな行を選択してターゲットを更新します。

複数のアップデートストラテジトランスフォーメーションが、同じターゲットの異なるインスタンスに書き込む場合、ターゲットデータが予測できなくなる場合があります。

Spark エンジンでマッピングを実行する場合、Hive Merge を使用するオプションを選択できます。このオプションには、次の制限事項があります。

- 削除または更新対象の単一行は、ターゲットの複数行と一致させることはできない。マッピングがこの制限に違反すると、マッピングはランタイムエラーで失敗する。
- アップデートストラテジトランスフォーメーションでパーティション化カラムまたはパケット化カラムを更新するよう設定すると、マッピングで Hive MERGE オプションが無視され、カラムは更新されない。

**注:** Developer tool とデータ統合サービスは、これらの制限に対して検証を行いません。式またはマッピングがこれらの制限事項に違反している場合、マッピングが実行される可能性はありますが、期待どおりの結果にはなりません。

Hive ターゲットは、更新を常に更新操作として実行します。Hive ターゲットは、[更新しない場合は挿入] または [挿入時に更新] をサポートしません。

## マッピング検証

マッピング検証は、次の場合に失敗します。

- アップデートストラテジトランスフォーメーションが、複数のターゲットに接続されている。
- アップデートストラテジトランスフォーメーションが、ターゲットの直前に配置されていない。
- アップデートストラテジトランスフォーメーションのターゲットが外部の ACID テーブルである。
- ターゲットに接続されたプライマリキーが含まれていない。
- 実行時の Hive またはリレーショナルターゲットテーブルの切り詰めを有効にするプロパティが選択されている。
- 実行時の Hive またはリレーショナルターゲットテーブルについて次のターゲットストラテジの 1 つが選択されている。
  - ターゲットテーブルの作成または置換
  - ApplyNewColumns
  - ApplyNewSchema
  - 失敗

ターゲットが Hive ターゲットのとき、マッピングは次の場合に失敗します。

- ターゲットテーブルがトランザクションに対して有効化されていない。
- ターゲットが ORC バケット化されていない。

## Hive ターゲットテーブルの使用

アップデートストラテジトランスフォーメーションと一緒に Hive ターゲットテーブルを使用するには、Hive のデータ定義言語の句 TBLPROPERTIES ("transactional"="true") を使用して、Hive ターゲットテーブルを作成する必要があります。

Hive ターゲットと一緒にアップデートストラテジトランスフォーメーションを使用するには、Hadoop 接続に関連付けられた hive-site.xml 設定セットに次のプロパティが設定されていることを確認します。

```
hive.support.concurrency true
hive.enforce.bucketing true
hive.exec.dynamic.partition.mode nonstrict
hive.txn.manager org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on true
hive.compactor.worker.threads 1
```

## 第 46 章

# Web サービスコンシューマトランスフォーメーション

この章では、以下の項目について説明します。

- [Web サービスコンシューマトランスフォーメーションの概要, 651 ページ](#)
- [WSDL の選択, 654 ページ](#)
- [Web サービスコンシューマトランスフォーメーションのポート, 655 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの入力マッピング, 657 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの出力マッピング, 660 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの詳細プロパティ, 663 ページ](#)
- [フィルタの最適化, 667 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの作成, 669 ページ](#)
- [Web サービスコンシューマトランスフォーメーションの例, 671 ページ](#)

## Web サービスコンシューマトランスフォーメーションの概要

Web サービスコンシューマトランスフォーメーションは、Web サービスクライアントとして Web サービスに接続し、データへのアクセスまたはデータの変換を行います。Web サービスコンシューマトランスフォーメーションは複数のグループのトランスフォーメーションです。

Web サービスは、SOAP、WSDL、XML などのオープンスタンダードを使用します。SOAP は、Web サービス用の通信プロトコルです。Web サービスクライアントの要求および Web サービスの応答は、SOAP メッセージです。WSDL は、Web サービス操作のプロトコル、形式、およびシグネチャを記述する XML スキーマです。

Web サービス操作には、情報の要求、データ更新の要求、タスク実行の要求などがあります。例えば、`getCustomerOrders` という Web サービス操作を実行するための SOAP リクエストを Web サービスコンシューマトランスフォーメーションが送信するとします。トランスフォーメーションは、リクエストで顧客 ID を渡します。Web サービスは、顧客情報と注文情報を取得し、その情報を SOAP レスポンスでトランスフォーメーションに返します。

Web サービスコンシューマトランスフォーメーションは、WSDL、Web サービス接続、またはエンドポイント URL 入力ポートで定義されたエンドポイント URL を使用して Web サービスに接続します。Web サービス接続で Web サービスのセキュリティを有効にします。



## SOAP メッセージ

Web サービスコンシューマトランスフォーメーションは、Web サービスプロバイダと情報をやり取りしたり、Web サービスを要求したりするために、Simple Object Access Protocol (SOAP) を使用します。SOAP は、Web サービスのリクエストメッセージとレスポンスメッセージの形式を定義します。

Web サービスコンシューマトランスフォーメーションでデータを変換すると、トランスフォーメーションによって SOAP リクエストが生成され、Web サービスへの接続が行われます。トランスフォーメーションは、WSDL オブジェクト、Web サービス接続、またはエンドポイント URL 入力ポートで定義されたエンドポイント URL を使用して Web サービスに接続します。SOAP リクエストには、要求された操作を実行するために Web サービス側で必要となる情報が含まれています。その Web サービス操作から、SOAP 応答としてデータがトランスフォーメーションに返されます。トランスフォーメーションは、SOAP レスポンスからデータをマップし、出力ポートでデータを返します。

Web サービスコンシューマトランスフォーメーションは、SOAP メッセージヘッダを ISO-8859-1 でエンコードします。

トランスフォーメーションは、ドキュメント/リテラルのエンコーディングを使用する SOAP メッセージを処理できます。ドキュメント/リテラルのスタイルには SOAP メッセージを記述する XML スキーマが必要です。SOAP メッセージは XML から形成されます。SOAP メッセージに複数出現要素が含まれている場合は、要素のグループによって XML 階層のレベルが形成されます。あるレベルが別のレベル内にネストされている場合、グループは関連しています。

SOAP リクエストメッセージには、階層データを含めることができます。例えば、販売データベースに顧客の注文を追加する要求を Web サービスコンシューマトランスフォーメーションが送信するとします。トランスフォーメーションは、2 つのデータグループを SOAP リクエストメッセージで渡します。一方のグループには顧客の ID と名前が含まれ、もう一方のグループには注文情報が含まれています。注文情報は複数回出現します。

SOAP レスポンスメッセージには、階層データを含めることができます。例えば、Web サービスコンシューマトランスフォーメーションが顧客注文に対する SOAP リクエストを生成するとします。Web サービスは、注文ヘッダーと複数出現の注文詳細要素を、SOAP レスポンスで返します。

## WSDL ファイル

WSDL ファイルには、送信側と受信側が交換するデータを認識できるように、Web サービスに渡されるデータが記述されています。Web サービスコンシューマトランスフォーメーションを作成する前に、WSDL ファイルをリポジトリにインポートする必要があります。

WSDL ファイルには、Web サービスコンシューマが正しい形式で要求メッセージを送信できるよう、データに対して実行する操作と、プロトコルやトランスポートへのバインドについて記述されています。WSDL ファイルには、Web サービスに接続するネットワークアドレスが記述されています。

WSDL には、SOAP リクエストメッセージと SOAP レスポンスメッセージのエンコード方法に関する情報が含まれています。SOAP エンコーディングにより SOAP メッセージ本体の形式が決まります。SOAP エンコーディングには、Web サービスが Web サービスコンシューマとの通信に使用する要求と応答のメッセージの形式が記述されています。Web サービス開発者は、さまざまなツールキットを使用して Web サービスを作成できます。ツールキットによって、サポートされる SOAP メッセージのエンコーディング方法が異なります。

Web サービスコンシューマトランスフォーメーションは、ドキュメント/リテラル SOAP エンコーディングスタイルをサポートします。Web サービスコンシューマトランスフォーメーションでは、開発者は WSDL 1.1 を使用できます。MIME、DIME、MTOM メッセージなど、WSDL アタッチメントは使用できません。



## 操作

Web サービスには、その Web サービスがサポートする各アクションの操作が含まれています。

例えば、顧客名を受信して顧客の詳細を応答で返す `getcustomerid` という操作を Web サービスに含めることができます。操作入力には、顧客名の要素が含まれています。操作出力には、顧客名に基づいた顧客詳細の要素が含まれています。

Web サービスコンシューマトランスフォーメーションを設定するときに、トランスフォーメーションが操作入力にデータをマップする方法と、トランスフォーメーションが操作出力からデータをマップする方法を定義します。トランスフォーメーションでは以下の情報を設定します。

### 入力マッピング

トランスフォーメーションの入力ポートを Web サービスの操作入力ノードにマップする方法を定義します。操作入力では、SOAP リクエスト内の操作の要素を定義します。

### 出力マッピング

Web サービスの操作出力ノードをトランスフォーメーションの出力ポートにマップする方法を定義します。操作出力では、SOAP レスポンス内の操作の要素を定義します。

## Web サービスのセキュリティ

Web サービス接続で Web サービスのセキュリティを有効にします。次のタイプのセキュリティを設定できます。

### Web サービスのセキュリティ

データ統合サービスは、Web サービスプロバイダに SOAP リクエストを送信するときに Web サービスセキュリティヘッダーを含めることができます。Web サービスセキュリティヘッダーには、Web サービスプロバイダがデータ統合サービスを認証できるように、認証情報が含まれています。

Web サービスコンシューマトランスフォーメーションによってユーザー名トークンが提供されます。データ統合サービスは、SOAP リクエスト内に個別のセキュリティ SOAP ヘッダーを作成し、その SOAP リクエストを Web サービスプロバイダに渡します。

Web サービス接続では、以下のタイプの Web サービスセキュリティを使用できます。

- PasswordText。データ統合サービスは WS-Security SOAP ヘッダー内のパスワードを変更しません。
- PasswordDigest。データ統合サービスが、ナンス値とタイムスタンプをパスワードに組み合わせます。データ統合サービスでは、そのパスワードに SHA ハッシュを適用して base64 エンコーディングでエンコードし、エンコードしたパスワードを SOAP ヘッダー内で使用します。

### トランスポートレイヤセキュリティ

Secure Sockets Layer (SSL) を使用して TCP/IP のトランスポートレイヤ (TCP レイヤ) の上に実装されるセキュリティ。Web サービスは、Web アドレスとして Hypertext Transfer Protocol over SSL (HTTPS) を使用し、メッセージをセキュリティ保護して転送します。Web サービスコンシューマトランスフォーメーションでは、TLS 1.2、TLS 1.1、TLS 1.0 を使用できます。トランスポートレイヤセキュリティに、HTTP 認証、プロキシサーバー認証、SSL 証明書を使用できます。

### SSL 認証

HTTPS プロトコルを介して接続する場合は、SSL 認証を使用できます。

使用できる SSL 認証には、次の種類があります。

- 1 方向 SSL 認証
- 2 方向 SSL 認証

## HTTP 認証

HTTP プロトコルを介して接続する場合は、HTTP 認証を使用できます。

使用できる HTTP 認証メソッドは、次のとおりです。

- 基本認証
- ダイジェスト認証
- NT LAN Manager (NTLM) 認証

# WSDL の選択

Web サービスコンシューマトランスフォーメーションを作成する前に、WSDL ファイルをモデルリポジトリにインポートする必要があります。ドキュメント/リテラルの SOAP バインディングスタイルで定義された WSDL ファイルのみをインポートできます。

WSDL ファイルは、実行する Web サービスの操作シグネチャを定義します。WSDL ファイルをインポートすると、Developer ツールによって、他のトランスフォーメーションで再利用可能な物理データオブジェクトが作成されます。

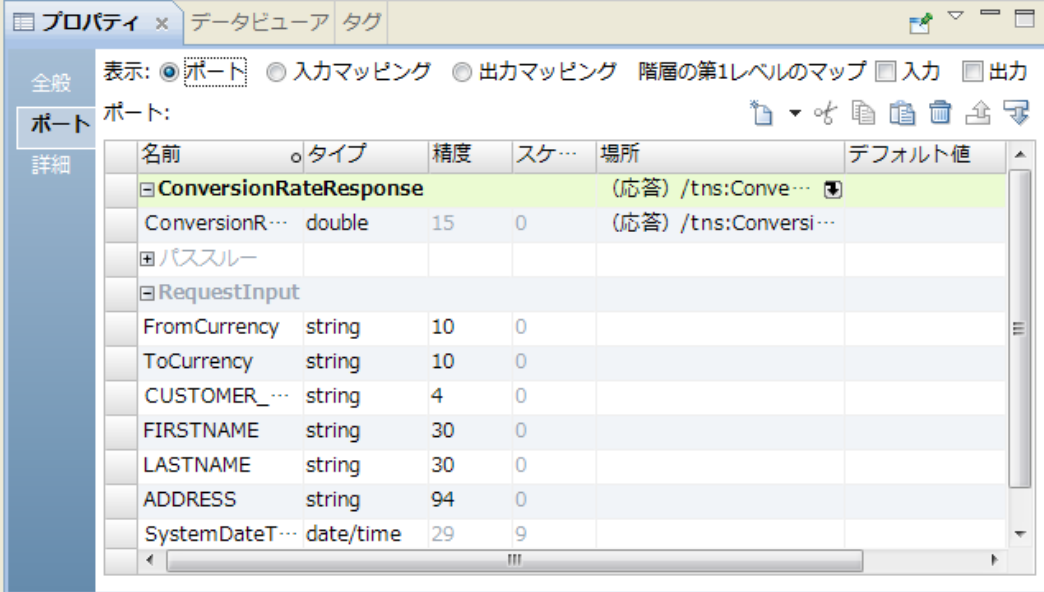
WSDL ファイルには複数の操作を定義できます。Web サービスコンシューマトランスフォーメーションを作成する場合は、実行する操作を選択します。操作入力と操作出力の階層を Web サービスコンシューマトランスフォーメーションに表示できます。これらの階層によって、SOAP リクエストメッセージと SOAP レスポンスメッセージの構造が定義されます。

1 方向入力処理を使用して WSDL をインポートすることもできます。1 方向入力処理を使用して WSDL をインポートする場合は、ダミー出力ポートを作成する必要があります。

# Web サービスコンシューマトランスフォーメーションのポート

トランスフォーメーションのポートを表示するときに、操作階層を確認する必要がない場合はポートを表示します。ポートを表示している場合は、グループの定義、ポートの定義、操作出力から出力ポートへのノードのマップを行うことができます。

以下の図に、再利用不可能な Web サービスコンシューマトランスフォーメーションのポートを示します。



名前	タイプ	精度	スケ...	場所	デフォルト値
ConversionRateResponse				(応答) /tns:Conve...	
ConversionR...	double	15	0	(応答) /tns:Conversi...	
パススルー					
RequestInput					
FromCurrency	string	10	0		
ToCurrency	string	10	0		
CUSTOMER_...	string	4	0		
FIRSTNAME	string	30	0		
LASTNAME	string	30	0		
ADDRESS	string	94	0		
SystemDateT...	date/time	29	9		

Web サービスコンシューマトランスフォーメーションには、複数の入力グループと複数の出力グループを設定できます。ポートを作成するときは、グループを作成して、そのグループにポートを追加します。操作入力階層または操作出力階層の構造に基づいて、ポートをグループ階層で定義します。キーを追加して、子グループを親グループに関連付けます。プライマリキーは、階層内の最下位のグループを除くすべてのグループに必要です。外部キーは、ルートグループを除き階層内のすべてのグループに必要です。

このトランスフォーメーションには、RequestInput という名前のルート入力グループがあります。プライマリキーをルート入力グループに追加する必要があります。プライマリキーは、string 型、bigint 型、integer 型のいずれかにする必要があります。

ルート入力グループにはパススルーポートを追加できます。パススルーポートは、データを変更せずにトランスフォーメーションを介してデータを渡します。パススルーポートは、入力データ内で 1 回しか指定できません。パススルーポートは、任意の出力グループに追加できます。出力ポートを入力ポートに関連付けます。SOAP リクエストを通じて渡した入力値は、SOAP レスポンスからの出力行に繰り返し出現します。

さらに、HTTP ヘッダー、クッキーポート、動的 URL ポート、Web サービスセキュリティ認証用ポートを、ルート入力グループに追加することができます。ルートグループ内のデータは 1 回だけ出現します。

操作出力ノードを出力ポートへマップするために、**[場所]** カラム内のフィールドをクリックし、**[場所の選択]** ダイアログボックスで階層を展開します。次に、階層からノードを選択します。

## HTTP ヘッダー入力ポート

Web サービスには追加の HTTP ヘッダーが必要な場合があります。ルート入力グループ内に入力ポートを作成して、Web サービスプロバイダに追加のヘッダー情報を渡すことができます。

HTTP ヘッダーと HTTP ポートを追加するには、追加先のルート入力グループを選択し、**[新規]** ボタンの隣の矢印をクリックします。次に、**[HTTP ヘッダー]** をクリックします。ヘッダー名とポート名を入力します。

複数の HTTP ヘッダーを作成することができます。

## その他の入力ポート

事前定義済みの入力ポートを Web サービスコンシューマトランスフォーメーションに追加できます。

以下の事前定義済み入力ポートを追加できます。

### クッキーポート

クッキー認証を使用するよう Web サービスコンシューマトランスフォーメーションを設定することができます。リモート Web サービスサーバーは、クッキーに基づいて、Web サービスコンシューマユーザーを追跡します。マッピングで Web サービスの呼び出しを繰り返し行う場合に、パフォーマンスが向上します。

クッキーポートを Web サービス要求メッセージに対して投影する場合、Web サービスプロバイダは応答メッセージでクッキー値を返します。クッキー値は、マッピング内の他のトランスフォーメーションダウンストリームに渡すことも、ファイル内に保存することもできます。クッキー値をファイルに保存する場合、そのクッキー値を Web サービスコンシューマトランスフォーメーションに対する入力として設定できます。

クッキー出力ポートは、Web サービスコンシューマトランスフォーメーションの任意の出力グループに投影できます。

### エンドポイント URL ポート

Web サービスコンシューマトランスフォーメーションは、エンドポイント URL を使用して Web サービスに接続します。エンドポイント URL は、WSDL ファイル、Web サービス接続、またはエンドポイント URL 入力ポートで定義できます。トランスフォーメーションが URL をポートで動的に受信すると、Data Integration Service は WSDL ファイルまたは Web サービス接続で定義されている URL をオーバーライドします。

Web サービスコンシューマトランスフォーメーションには、Web サービス要求ごとに URL ポート値を 1 個設定できます。エンドポイント URL ポートはルート入力グループに追加します。

### WS-Security ポート

Web サービスのセキュリティは、Web サービス接続で有効にします。Web サービスのセキュリティを有効にするときは、Web サービス接続または WS-Security 入力ポートでユーザー名とパスワードを定義する必要があります。

WS-Security ポートを追加するときは、トランスフォーメーションの入力ポートを通じてユーザー名とパスワードを渡します。トランスフォーメーションがユーザー名とパスワードをポートで動的に受信すると、Data Integration Service は Web サービス接続で定義されている値をオーバーライドします。

**注:** Web サービス接続には、HTTP と WS-Security の認証用のユーザー名とパスワードが 1 組あります。

事前定義済みの入力ポートを追加するには、**[ポート]** 領域でルート入力グループをクリックします。**[新規]** ボタンの隣の矢印をクリックし、**[その他のポート]** をクリックします。追加するポートを選択します。

# Web サービスコンシューマトランスフォーメーションの入力マッピング

トランスフォーメーションのポートを表示するときに、操作入力階層を確認するには入力マッピングを表示します。入力マッピングを表示している場合は、入力グループの定義、入力ポートの定義、操作入力ノードへの入力ポートのマッピングを行うことができます。

入力マッピングは以下の領域で構成されています。

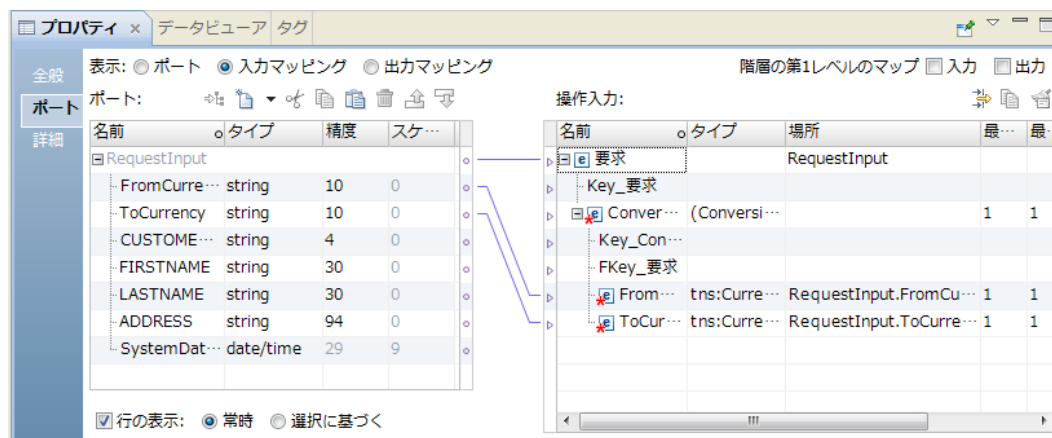
## ポート

【ポート】領域では、トランスフォーメーションの入力グループと入力ポートを作成します。

## 操作入力

【操作入力】領域には、Web サービスコンシューマトランスフォーメーションから Web サービスに伝送される SOAP リクエストメッセージ内のノードが表示されます。トランスフォーメーションの作成に使用する WSDL データオブジェクトにより、操作入力階層が定義されます。

以下の図に、再利用不可能な Web サービスコンシューマトランスフォーメーションの入力マッピングを示します。



入力ポートを作成したら、入力ポートを【ポート】領域から【操作入力】領域のノードにマップします。入力ポートを操作入力内のノードにマップすると、ポートの場所が【操作入力】領域の【場所】カラムに表示されます。

入力階層の第 1 レベルをマップするよう選択すると、Developer ツールは操作入力の第 1 レベルのノードを入力ポートにマップします。また、Developer ツールはマッピングを実行するためのポートも作成します。階層の第 1 レベルに複数回出現する親ノードが含まれ、その親ノードに 1 回または複数回出現する子ノードがある場合、Developer ツールは階層の第 1 レベルをマップしません。

1 つの文字列またはテキストの入力ポート内の XML データを、SOAP リクエストのメッセージ全体にマップすることができます。XML データを SOAP リクエストのメッセージ全体にマップする場合、その操作入力のポートをノードにマップすることはできません。

入力ポートと操作入力内のノードを結ぶ線が表示されるように設定することができます。

関連項目：

- [「Web サービス SOAP メッセージの生成の概要」 \(ページ 683\)](#)

## 入力ポートをノードにマップするためのルールとガイドライン

入力ポートを操作入力階層内のノードにマップする場合には、以下のルールを確認します。


- 階層内の 1 個のノードに対して、1 個の入力ポートをマップできます。同じポートを階層内の任意の数のキーにマップできます。
- 入力ポートとノードのデータ型には互換性が必要です。
- 1 つの入力グループ内のノードを、操作入力内の複数の階層レベルにマップできます。
- 操作入力のキーに入力ポートをマップする必要があります。キーにマップするポートのデータ型は、string、integer、または bigint であることが必要です。操作入力内の、SOAP メッセージに含める階層レベルより上位にある全レベルのキーに、データをマップします。マップするレベルと、その上位にある全レベルの外部キーを含めます。

**注:** 操作入力階層の最下位レベルだけをマップする場合は、入力ポートをキーにマップする必要はありません。

- string 型、bigint 型、または integer 型の複数の入力ポートを **【操作入力】** 領域のキーにマップして、複合キーを作成できます。複合キーの **【場所】** フィールドをクリックして、入力ポートの順序を変更したり、いずれかのポートを削除したりすることができます。

## [ビューのカスタマイズ] のオプション

**【操作入力】** 領域にキーを表示するように、操作入力階層を変更することができます。ノードの順序を定義するグループ化構造を表示することもできます。

**【ビューのカスタマイズ】** ボタン () (**【操作入力】** 領域内) をクリックします。以下のいずれかのオプションを有効にします。

### シーケンス、選択、およびすべて

要素定義が、シーケンス、選択、すべてのいずれかであるかどうかを示す線を表示します。

全グループ内のすべてのノードを SOAP メッセージ内に含める必要があります。

シーケンスグループ内のノードは、WSDL 内に定義された順序で並んでいる必要があります。

選択グループ内の少なくとも 1 つ以上のノードが、SOAP メッセージに指定されている必要があります。

### キー

**【操作入力】** 領域にキーを表示します。 **【操作入力】** 領域には、各グループ用のキーが含まれています。 **【ポート】** 領域内の入力ポートに、キーを追加できます。

## 操作入力への入力ポートのマッピング

操作入力マッピングを表示している場合は、入力グループの定義、入力ポートの定義、操作入力ノードへの入力ポートのマッピングを行うことができます。

1. Web サービスコンシューマトランスフォーメーションを開きます。
2. トランスフォーメーション入力マッピングを表示するには、以下のいずれかの方法を使用します。
  - 再利用可能なトランスフォーメーションの場合は、**【概要】** ビューをクリックします。入力マッピングの表示を選択します。

- 再利用不可能なトランスフォーメーションの場合は、**【ポート】** タブ (**【プロパティ】** ビュー内) をクリックします。入力マッピングの表示を選択します。
3. ルート入力グループのプライマリキーを定義します。
  4. 次のいずれかの方法を使用して、入力グループまたは入力ポートを **【ポート】** 領域に追加します。

オプション	説明
ノードをドラッグする	<b>【操作入力】</b> 領域のグループノードまたは子ノードを、 <b>【ポート】</b> 領域の空のカラムにドラッグします。グループノードの場合、Developer ツールによってポートのないグループが追加されます。
グループまたはポートを手動で追加する	グループを追加するには、 <b>【新規】</b> ボタンの隣の矢印をクリックし、 <b>【グループ】</b> をクリックします。ポートを追加するには、 <b>【新規】</b> ボタンの隣の矢印をクリックし、 <b>【フィールド】</b> をクリックします。
別のトランスフォーメーションからポートをドラッグする	エディタ内で、別のトランスフォーメーションから Web サービスコンシューマトランスフォーメーションへ、ポートをドラッグします。
ポートをコピーする	別のトランスフォーメーションからポートを選択し、 <b>【ポート】</b> 領域へコピーします。ポートをコピーするには、キーボードショートカットを使用するか、Developer ツールの <b>【コピー】</b> ボタンと <b>【貼り付け】</b> ボタンを使用します。
<b>【階層の第 1 レベルのマップ】</b> を選択する	<b>【階層の第 1 レベルのマップ】</b> を選択します。Developer ツールによって、操作入力の第 1 レベル内のノードが、入力ポートおよび入力グループにマップされます。また、マッピングを実行するための入力ポートと入力グループが作成されます。

5. ポートを手動で作成するか、別のトランスフォーメーションからコピーした場合は、**【操作入力】** 領域の **【場所】** カラムをクリックし、リストからポートを選択します。
6. 以下のいずれかの方法を使用して、入力ポートを複合キーとしてマップします。

オプション	説明
入力ポートをドラッグする	2 つ以上の入力ポートを選択し、操作入力階層内のキーにドラッグします。
<b>【場所の選択】</b> ダイアログボックスから入力ポートを選択する	操作入力階層でキーの <b>【場所】</b> カラムをクリックし、入力ポートを選択します。

7. 以下のいずれかの方法を使用して、ノードの場所をクリアします。

オプション	説明
<b>【クリア】</b> をクリックする	<b>【操作入力】</b> 領域でノードを 1 つ以上選択し、 <b>【クリア】</b> をクリックします。
ポートとノードを結ぶ線を削除する	操作入力領域で入力ポートとノードを結ぶ線を 1 本以上選択し、Delete キーを押します。

8. 関連付けられている WSDL データオブジェクト内に、anyType 要素、任意の要素、anyAttribute 属性、派生型の要素、または置き換えグループが含まれている場合は、**【操作入力】** 領域でオブジェクトを選択します。ノードの **【タイプ】** カラムで **【選択】** をクリックし、リストから 1 つ以上のタイプ、要素、または属性を選択します。



9. 文字列入力ポートまたはテキスト入力ポートから完全な SOAP リクエストに XML データをマップするには、ポートを右クリックし、[XML としてマップ] を選択します。

## Web サービスコンシューマトランスフォーメーションの出力マッピング

トランスフォーメーションのポートを表示するときに、操作出力階層を確認するには出力マッピングを表示します。出力マッピングを表示している場合は、出力グループの定義、出力ポートの定義、出力ポートへの操作出力ノードのマップを行うことができます。

出力マッピングは、次に挙げる領域で構成されています。

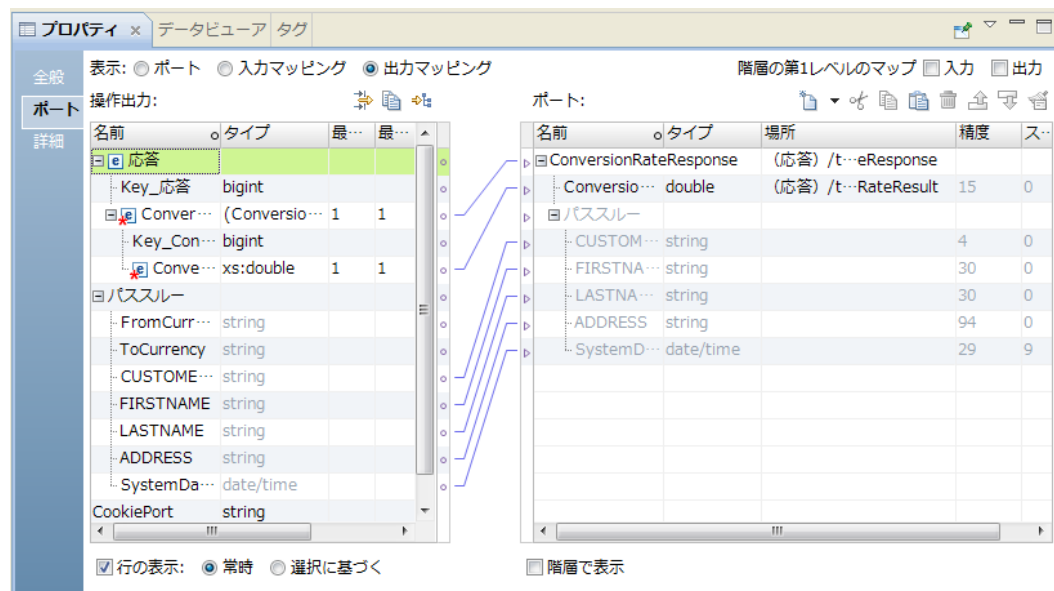
### 操作出力

【操作出力】領域には、Web サービスから Web サービスコンシューマトランスフォーメーションに返される SOAP レスponseメッセージ内のノードが表示されます。トランスフォーメーションの作成に使用する WSDL データオブジェクトにより、操作出力階層が定義されます。

### ポート

【ポート】領域では、トランスフォーメーションの出力グループと出力ポートを作成します。

以下の図に、再利用不可能な Web サービスコンシューマトランスフォーメーションの出力マッピングを示します。



出力ポートを作成したら、【操作出力】領域から【ポート】領域のポートに、ノードをマップします。操作出力から出力ポートにノードをマップすると、【ポート】領域の【場所】カラムにノードの場所が表示されます。

出力階層の第 1 レベルをマップするよう選択すると、Developer ツールは操作出力の第 1 レベルのノードを出力ポートにマップします。また、Developer ツールはマッピングを実行するためのポートも作成します。階層の第 1 レベルに複数回出現する親ノードが含まれ、その親ノードに 1 回または複数回出現する子ノードがある場合、Developer ツールは階層の第 1 レベルをマップしません。



出力ポートを階層構造で表示するように設定することもできます。子グループはそれぞれ、その親グループの下に表示されます。操作出力内のノードと出力ポートを結ぶ線が表示されるように設定することができます。

関連付けられた WSDL データオブジェクトがリポジトリから削除された場合、Developer ツールは操作ノードの場所を出力マッピングに保持します。出力マッピングを表示すると、**【ポート】** 領域の出力ポートの **【場所】** カラムに、操作ノードの場所が表示されています。別の WSDL をトランスフォーメーションに関連付けると、Developer ツールがそれぞれの場所が有効であるかどうかを調べます。場所が有効ではなくなった場合、出力マッピングの **【ポート】** 領域にある操作ノードの場所を Developer ツールがクリアします。

## 関連項目：

- [「Web サービス SOAP メッセージの解析の概要」 \(ページ 675\)](#)

## ノードを出力ポートにマップするためのルールとガイドライン

操作出力階層内のノードを出力ポートにマップする際には、以下のルールを確認します。

- 操作出力ノードおよび出力ポートのデータ型には互換性が必要です。
- グループ内の複数の出力ポートにノードをマップすることはできません。
- パススルーポートを除く各出力ポートには、有効な場所を指定する必要があります。
- 複数出現子ノードを空の出力ポートにドラッグした場合、グループを他の出力グループに関連付ける必要があります。関連付けるグループを選択すると、Developer ツールにより、グループ間に関連付けるキーが作成されます。
- 複数出現要素を、親要素が含まれているグループにドラッグする場合、含める子要素の出現回数を設定することができます。または、親グループを、トランスフォーメーション出力内の複数出現子グループで置き替えることもできます。

## SOAP メッセージを XML としてマップ

データを個別の出力ポートに返す代わりに、完全な SOAP メッセージを XML としてマップできます。

SOAP メッセージを XML としてマップすると、Data Integration Service により完全な SOAP メッセージが 1 個のポートに返されます。出力ポートは作成しないでください。

完全なメッセージをマップするには、**【操作出力】** 領域でルートグループを右クリックします。**【XML としてマップ】** を選択します。

Developer ツールにより、文字列出力ポートが作成されます。精度は 65535 バイトです。

## 【ビューのカスタマイズ】 のオプション

**【操作出力】** 領域内にクッキーポート、パススルーポート、およびキーを表示するように、操作出力階層を変更することができます。ノードの順序を定義するグループ化構造を表示することもできます。

**【ビューのカスタマイズ】** ボタン (**【操作出力】** 領域内) をクリックします。以下のいずれかのオプションを有効にします。

### シーケンス、選択、およびすべて

要素定義が、シーケンス、選択、すべてのいずれかであるかどうかを示す線を表示します。

全グループ内のすべてのノードを SOAP メッセージ内に含める必要があります。

シーケンスグループ内のノードは、WSDL 内に定義された順序で並んでいる必要があります。

選択グループ内の少なくとも 1 つ以上のノードが、SOAP メッセージに指定されている必要があります。

## キー

【操作出力】領域にキーを表示します。【操作出力】領域には、各グループのキーが含まれています。  
【ポート】領域内の出力ポートに、キーを追加できます。

## パススルーポート

【操作出力】領域には、パススルーポートが表示されます。パススルーポートとは、トランスフォーメーションを通じて、変更を加えずにデータを渡すポートのことです。操作出力内のパススルーポートを、Web サービスコンシューマトランスフォーメーションの任意の出力グループへ投影することができます。パススルーポートはデータを 1 回だけ受け取るため、SOAP メッセージ内のルートレベルにあります。

## クッキーポート

クッキーポートを表示します。クッキー認証を設定するときに、リモート Web サービスサーバーは、クッキーに基づいて、Web サービスコンシューマユーザーを追跡します。Web サービスクッキーを要求メッセージ内に投射する場合、Web サービスは応答メッセージでクッキー値を返します。操作出力内のクッキー値を、Web サービスコンシューマトランスフォーメーション任意の出力グループへ投影することができます。

# 出力ポートへの操作出力のマッピング

操作出力マッピングを表示している場合は、出力グループの定義、出力ポートの定義、出力ポートへの操作出力ノードのマッピングを行うことができます。

1. Web サービスコンシューマトランスフォーメーションを開きます。
2. トランスフォーメーション出力マッピングを表示するには、以下のいずれかの方法を使用します。
  - 再利用可能なトランスフォーメーションの場合は、【概要】ビューをクリックします。出力マッピングの表示を選択します。
  - 再利用不可能なトランスフォーメーションの場合は、【ポート】タブ（【プロパティ】ビュー内）をクリックします。出力マッピングの表示を選択します。
3. 以下のいずれかの方法を使用して、出力グループまたは出力ポートを【ポート】領域に追加します。

オプション	説明
ノードをドラッグする	【操作出力】領域のグループノードまたは子ノードを、【ポート】領域の空のカラムにドラッグします。グループノードの場合、Developer ツールによってポートのないグループが追加されます。
グループまたはポートを手動で追加する	グループを追加するには、【新規】ボタンの隣の矢印をクリックし、【グループ】をクリックします。ポートを追加するには、【新規】ボタンの隣の矢印をクリックし、【フィールド】をクリックします。
別のトランスフォーメーションからポートをドラッグする	エディタ内で、別のトランスフォーメーションから Web サービスコンシューマトランスフォーメーションへ、ポートをドラッグします。
ポートをコピーする	別のトランスフォーメーションからポートを選択し、【ポート】領域へコピーします。ポートをコピーするには、キーボードショートカットを使用するか、Developer ツールの【コピー】ボタンと【貼り付け】ボタンを使用します。
【階層の第 1 レベルのマップ】を選択する	【階層の第 1 レベルのマップ】を選択します。Developer ツールによって、操作出力の第 1 レベル内のノードが、出力ポートおよび出力グループにマップされます。また、マッピングを実行するための出力ポートと出力グループが作成されます。

4. ポートを手動で作成するか、別のトランスフォーメーションからコピーした場合は、**【ポート】** 領域の **【場所】** カラムをクリックし、リストからノードを選択します。
5. 以下のいずれかの方法を使用して、ポートの場所をクリアします。

オプション	説明
<b>【クリア】</b> をクリックする	<b>【ポート】</b> 領域でポートを 1 つ以上選択し、 <b>【クリア】</b> をクリックします。
ノードとポートを結ぶ線を削除する	操作出力のノードと出力ポートを結ぶ線を 1 本以上選択し、Delete キーを押します。

6. 関連付けられている WSDL データオブジェクト内に、anyType 要素、任意の要素、anyAttribute 属性、派生型の要素、または置き換えグループが含まれている場合、**【操作出力】** 領域でオブジェクトを選択します。ノードの **【タイプ】** カラムで **【選択】** をクリックし、リストから 1 つ以上のタイプ、要素、または属性を選択します。
7. 完全な SOAP レスponseメッセージを XML としてマップするには、**【操作出力】** 領域でルートグループを右クリックし、**【XML としてマップ】** を選択します。

## Web サービスコンシューマトランスフォーメーションの詳細プロパティ

Web サービスコンシューマトランスフォーメーションの詳細プロパティには、追跡レベル、汎用フォールトポート、Web サービス接続、および同時 Web サービス要求のメッセージが含まれます。

**【詳細】** タブで、Web サービスコンシューマトランスフォーメーションの以下の詳細プロパティを定義できます。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

### SOAP アクション

WSDL に定義されている SOAP アクション値を、Web サービスコンシューマトランスフォーメーションの定数値でオーバーライドします。

### 汎用 SOAP フォールト処理を有効にする

WSDL で定義されていないフォールトメッセージを返します。フォールトのコードとメッセージを処理する出力ポートを GenericFault 出力グループ内に作成します。

以下の表に、SOAP 1.1 および SOAP 1.2 のフォールト出力ポートを示します。

フォールト出力ポート (SOAP 1.1)	フォールト出力ポート (SOAP 1.2)	説明
フォールトコード	コード*	フォールトの ID コードを返します。
フォールト文字列	理由*	フォールトメッセージ内のエラーの説明を返します。

フォールト出力ポート (SOAP 1.1)	フォールト出力ポート (SOAP 1.2)	説明
フォールト詳細	明細	Web サービスプロバイダから Web サービスコンシューマトランスフォーメーションへ、汎用フォールトメッセージで渡されるカスタム情報を返します。
フォールトアクタ	ロール	フォールト発生の原因となったオブジェクトに関する情報を返します。
-	ノード	フォールトが生成された SOAP ノードの URI を返します。
*コード出力ポートと理由出力ポートは階層になっています。		

**注:** コードフォールト出力ポートを展開して、SubCode フォールト出力ポートを 1 つのレベルに抽出することができます。

#### HTTP エラー処理を有効にする

Web サービスから HTTP エラーを返します。GenericFault 出力グループ内に HTTP エラー出力ポートを作成します。

#### フォールトをエラーとして扱う

フォールトメッセージをマッピングログに記録します。フォルトが発生すると、Data Integration Service によってマッピングのエラー数が 1 件加算されます。このプロパティを無効にすると、初期選択の最適化および最適化にプッシュインを利用できるようになります。デフォルトでは有効になっています。

#### 接続

Web サービスに接続する Web サービス接続オブジェクトを特定します。Developer ツールで Web サービス接続を作成します。Developer ツールまたは Administrator ツールで Web サービス接続を編集します。Web サービス接続を設定するときは、エンドポイント URL、Web サービスに必要なセキュリティのタイプ、および接続タイムアウト期間を設定します。

Web サービスコンシューマトランスフォーメーションは、エンドポイント URL を使用して Web サービスに接続します。エンドポイント URL は、WSDL ファイル、Web サービス接続、またはエンドポイント URL 入力ポートで定義できます。

以下のガイドラインを使用して、Web サービス接続をいつ設定するか判断します。

- WSDL ファイル内の URL と異なるエンドポイント URL を使用する場合、およびエンドポイント URL 入力ポートを使用しない場合に、接続を設定します。
- 接続先の Web サービスに Web サービスセキュリティ、HTTP 認証、または SSL 証明書が必要な場合に、接続を設定します。
- デフォルトの接続タイムアウト時間を変更する場合に、接続を設定します。

**注:** リポジトリ内の WSDL データオブジェクトは、Web サービス接続に関連付けることができます。関連付けられた接続は、その WSDL から作成される各 Web サービスコンシューマトランスフォーメーションのデフォルト接続になります。

#### 圧縮を有効にする

gzip 圧縮方式による SOAP リクエストのエンコードを有効にし、gzip または deflate による SOAP レスポンスのデコードを有効にします。

## XML スキーマ検証

実行時に SOAP レスポンスメッセージを検証します。【無効な XML でのエラー】または【検証なし】を選択します。

## ソート済み入力

入力データを必ずしもすべて処理しなくても、Data Integration Service が出力を生成できるようにします。入力データが操作入力階層内のキーを基準にしてソートされる場合に、ソート済み入力を有効にします。

## 最適化にプッシュイン

最適化にプッシュインを有効にします。【最適化にプッシュイン】プロパティの【開く】ボタンをクリックして、フィルタ値を受け取るフィルタポートを選択します。フィルタポートごとに、Web サービス応答にフィルタリングされたカラムを含める出力ポートを選択します。

## 副次作用あり

Web サービスが行を返す以外にいずれかの関数を実行することを示すチェックボックス。Web サービスが、行を返すことに加えて、オブジェクトを変更したり他のオブジェクトまたは関数と対話したりすると、Web サービスコンシューマトランスフォーメーションに副次作用が生じます。副次作用により、Web サービスで、データベースの変更、合計の増加、例外の発生、電子メールへの書き込み、または他の Web サービスの呼び出しが行われる場合があります。最適化にプッシュインまたは初期選択の最適化を可能にするには、【副次作用あり】プロパティを無効にします。デフォルトでは有効になっています。

## 並行処理の有効化

複数の Web サービス要求を同時に送信できるように、Web サービスに対して複数の同時接続を作成するには、Web サービスコンシューマトランスフォーメーションを有効にします。Web サービスコンシューマトランスフォーメーションを有効にして、Web サービスに対して複数の同時接続を作成するときには、合計メモリ消費量の制限と同時接続制限を設定することができます。

以下の表に、オプションを示します。

オプション	説明
並行処理の有効化	Web サービスに対して複数の同時接続を作成します。
同時接続制限	Web サービスの同時接続の数。デフォルトは 20 です。
並行処理の合計メモリ制限 (MB)	すべての同時接続の合計メモリ割り当ての制限。デフォルトは 100 MB です。

# Web サービスのエラー処理

Web サービスコンシューマトランスフォーメーションは、SOAP フォールトと HTTP エラーをマッピング内のダウストリームに渡すように設定できます。フォールトが発生したときに、そのフォールトをエラー数に加算することができます。Web サービスのエラー処理は、トランスフォーメーションの詳細プロパティで設定します。

Web サービスは、応答メッセージを返すか、またはフォールトを返します。フォールトはエラーの一種です。Web サービスでは、発生するエラーに基づいて各種のフォールトを生成できます。

Web サービスコンシューマトランスフォーメーションは、以下のタイプのフォールトを返すことができます。

## SOAP フォールト

WSDL によって定義された SOAP エラー。フォールトを Web サービス応答メッセージに含めて返す出力エラーポートを設定します。SOAP 1.1 バインディングの場合、Data Integration Service は、フォールト

のフォールトメッセージ、フォールトコード、フォールト文字列、フォールトアクタ要素を返します。  
SOAP 1.2 バインディングの場合、Data Integration Service は、フォールトのフォールトメッセージ、コード、理由、ノード、ロール要素を返します。

### 汎用 SOAP フォールト

Web サービスは実行時に汎用 SOAP フォールトを生成します。フォールト要素は、SOAP 1.1 バインディングと SOAP 1.2 バインディングで異なります。WSDL は、汎用 SOAP フォールトを定義しません。汎用 SOAP フォールトには、認証エラーや SOAP リクエストエラーなどがあります。

### HTTP エラー

トランスフォーメーション内で HTTP エラー処理を有効にすると、Developer ツールによって HTTP フォールト出力ポートが追加されます。Data Integration Service は、1 個の文字列ポート内の Web サービスから HTTP エラーを返します。HTTP エラーには、エラーコードとメッセージが含まれます。

Web サービスからの SOAP レスポンスに無効な XML データが含まれていた場合、Web サービスコンシューマトランスフォーメーションはエラーを返します。

SOAP フォールトをエラーとして扱うかどうか、設定することができます。[フォールトをエラーとして扱う]を有効にした場合に SOAP フォールトが発生すると、Data Integration Service によってマッピングのエラー数が 1 件加算されます。フォールトはメッセージログに記録されます。

## メッセージの圧縮

SOAP メッセージ圧縮を有効にした場合、Web サービスコンシューマトランスフォーメーションは Web サービス要求メッセージを圧縮し、圧縮済みの Web サービス応答メッセージを受け取ります。

Web サービスコンシューマトランスフォーメーションは、SOAP リクエストを gzip 圧縮でエンコードします。このトランスフォーメーションでは、gzip 圧縮または deflate 圧縮によってエンコードされた応答メッセージを受け付けます。

Web サービスからの応答を受信すると、Data Integration Service は SOAP メッセージ内の Content-Encoding HTTP ヘッダーを調べ、SOAP メッセージをデコードします。

デフォルトは、圧縮なしのエンコードです。Web サービスは SOAP レスポンスを圧縮しません。

以下の表に、圧縮がオンになっている場合とオフになっている場合の要求メッセージと応答メッセージのヘッダーを示します。

圧縮	ヘッダー
オン	Content-Encoding ヘッダー: gzip Accept-Encoding ヘッダー: gzip、deflate
オフ	空の Content-Encoding ヘッダー 空の Accept-Encoding ヘッダー

応答メッセージは Web サービスによってデフォルトの圧縮方式でエンコードされていることもあります。メッセージが gzip または deflate でエンコードされている場合、Web サービスコンシューマトランスフォーメーションはそのメッセージをデコードします。Web サービスが応答メッセージを予期せずエンコードしていた場合、Web サービスコンシューマトランスフォーメーションはマッピングログにメッセージを記録します。

圧縮はトランスフォーメーションの詳細プロパティで有効にします。



## 並行処理

複数の Web サービス要求を同時に送信できるように、Web サービスコンシューマトランスフォーメーションを有効にして、Web サービスに対して複数の同時接続を作成することができます。

例えば、銀行情報を問い合わせているときに、複数の行が同時に送信される並行処理が有効になるように Web サービスコンシューマトランスフォーメーションを設定することができます。入力行が 20 行ある場合は、処理を高速化するために 20 個の要求を同時に送信できます。

Web サービスコンシューマトランスフォーメーションで並行処理を有効にするときは、合計メモリ消費量の制限を設定できます。

Web サービスコンシューマトランスフォーメーションで並行処理を有効にするときは、Web サービスの同時接続の数を設定できます。

### 並行処理に関するルールとガイドライン

並行処理の使用中は以下のルールおよびガイドラインを使用します。

- 並行処理では、ソートされた入力行が Web サービスに対する複数の同時接続としてサポートされます。順序付けられた出力行はサポートされません。
- 並行処理はデータセットが 100 行を超える場合に使用します。
- Web サービスの同時接続の数が増えないようにすることをお勧めします。Web サービスの同時接続の数は、オペレーティングシステムによって使用されるソケットの数に関係します。ソケットの数が増えると費用がかかります。
- 並行処理機能の使用中は、最適なパフォーマンスのために、RAM が 100 MB 以上のマルチコアプロセッサを備えたシステムを使用します。
- 並行処理のメモリ制限は、Web サービスの起動中に並行処理のワークフローによって消費されるメモリを表します。
- Web サービスコンシューマトランスフォーメーションで並行処理を有効にするときは、メモリ消費量の制限を設定できます。メモリ消費量はサーバーの物理 RAM を超えないようにしてください。

### 並行処理のベストプラクティス

並行処理使用時の最適なパフォーマンスを実現するために、次のベストプラクティスに従ってください。

- 並行処理の合計メモリの制限と同時接続制限のデフォルト値の変更を避ける。
- 100 行未満のデータセットに対する並行処理の使用を避ける。
- 並行処理使用時のマッピングでのパススルーポートを避ける。

## フィルタの最適化

フィルタを最適化すると、マッピングを通過する行数が減り、パフォーマンスが向上します。Data Integration Service は、初期選択の最適化と最適化にプッシュインを適用することができます。

Data Integration Service でフィルタ最適化方式が適用されると、マッピング内でフィルタができるだけソースの近くに移動されます。Data Integration Service がマッピング内のトランスフォーメーションの前にフィルタを移動できない場合、フィルタロジックをトランスフォーメーションにプッシュできる可能性があります。

## Web サービスコンシューマトランスフォーメーションでの初期選択の最適化の有効化

トランスフォーメーションに副次作用がなく、フォルトをエラーとして扱うこともない場合は、Web サービスコンシューマトランスフォーメーションで初期選択の最適化を有効にします。

1. Web サービスコンシューマトランスフォーメーションの【詳細プロパティ】ビューを開きます。
2. 【フォルトをエラーとして扱う】をクリアします。
3. 【副次作用あり】をクリアします。

## Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化

Web サービスコンシューマトランスフォーメーションは、最適化にプッシュインによって、フィルタポートのフィルタ値を受け取ります。フィルタポートは、最適化にプッシュインを設定するときにフィルタポートとして特定する接続されていない入力ポートです。フィルタポートには、エンドユーザーのクエリにフィルタが含まれていない場合に Web サービスがすべての行を返すようにするデフォルト値があります。フィルタポートはパススルーポートではありません。

**注:** フィルタフィールドは、Web サービス要求のルートグループの一部である必要があります。

フィルタフィールドをパススルーポートにすることはできません。フィルタポートの設定時、ポートのデフォルト値がフィルタ条件の値に変わるため、パススルー出力ポートの値が変わります。出力パススルーポートに基づくフィルタは、予期しない結果を返します。

Web サービスコンシューマトランスフォーメーションには複数の式をプッシュすることができます。各フィルタ条件は次の形式である必要があります。

<Field> = <Constant>

フィルタ条件は AND で結合する必要があります。フィルタ条件を OR で結合することはできません。

## Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化の例

SQL データサービスは、すべての顧客の注文を返すか、ユーザーから受け取る SQL クエリに基づいて特定の顧客の注文を返します。

SQL データサービスには、次のコンポーネントから構成される論理データオブジェクトが含まれています。

### 顧客テーブル

顧客情報が含まれる Oracle データベーステーブル。

### Web サービスコンシューマトランスフォーメーション

Web サービスを呼び出して顧客の最新の注文を取得するトランスフォーメーション。Web サービスコンシューマトランスフォーメーションには、customerID と orderNum の入力ポートがあります。このトランスフォーメーションには、顧客テーブルから受け取る顧客データが含まれるパススルーポートがあります。orderNum ポートはフィルタポートで、接続されていません。OrderNum には、デフォルト値 "\*" があります。Web サービスが Web サービス要求でこの値を受け取ると、すべての注文を返します。

### 注文仮想テーブル

Web サービスから顧客データと注文データを受け取る仮想テーブル。エンドユーザーはこのテーブルを参照します。注文には、顧客カラム、orderID カラム、および顧客データと注文データが含まれます。

エンドユーザーは次の SQL クエリを SQL データサービスに渡します。



```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

クエリはマッピングを最適化するために分割されます。初期選択の最適化が使用され、フィルタロジック、customer = 23 が読み込まれた顧客テーブルに移動します。最適化にプッシュインが使用され、フィルタロジック、orderID = 56 が Web サービスコンシューマトランスフォーメーションのフィルタポートにプッシュされます。Web サービスコンシューマトランスフォーメーションでは、顧客 23 の ordersID 56 が取得されます。

## Web サービスコンシューマトランスフォーメーションによるプッシュイン最適化の有効化

トランスフォーメーションに副次作用がなく、フォルトをエラーとして扱うこともない場合は、Web サービスコンシューマトランスフォーメーションで最適化にプッシュインを有効にします。

1. Web サービスコンシューマトランスフォーメーションの **【詳細プロパティ】** ビューを開きます。
2. **【フォルトをエラーとして扱う】** をクリアします。
3. **【副次作用あり】** をクリアします。
4. **【最適化にプッシュイン】** プロパティの **【開く】** ボタンをクリックします。
5. **【最適化された入力】** ダイアログボックスで、フィルタポート名を選択します。  
複数のフィルタポートを選択できます。
6. **【出力】** カラムをクリックします。
7. フィルタポートごとに、Web サービス応答にフィルタリングされたカラムを含める出力ポートを選択します。
8. フィルタポートのデフォルト値を入力します。

**注:** Web サービスコンシューマポートは、フィルタポートではない限り、デフォルト値を設定できません。

## Web サービスコンシューマトランスフォーメーションの作成

再利用可能または再利用不可能な Web サービスコンシューマトランスフォーメーションを作成できます。再利用可能なトランスフォーメーションは、複数のマッピングで使用できます。再利用不可能なトランスフォーメーションは、単一のマッピングで使用されます。

単一の WSDL オブジェクトから SOAP 1.1 バインディングと SOAP 1.2 バインディングの Web サービスコンシューマトランスフォーメーションを作成することができます。

1. トランスフォーメーションを作成するには、次のいずれかの方法を使用します。

オプション	説明
再利用可能	Object Explorer ビューで、プロジェクトまたはフォルダーを選択します。 <b>【ファイル】</b> > <b>【新規】</b> > <b>【トランスフォーメーション】</b> をクリックします。Web サービスコンシューマトランスフォーメーションを選択し、 <b>【次へ】</b> をクリックします。
再利用不可	マッピングまたはマップレットで、トランスフォーメーションパレットからエディタに Web サービスコンシューマフォーメーションをドラッグします。

**【新しい Web サービスコンシューマトランスフォーメーション】** ダイアログボックスが表示されます。

2. [参照] をクリックし、Web サービスの要求メッセージと応答メッセージを定義する WSDL データオブジェクトを選択します。

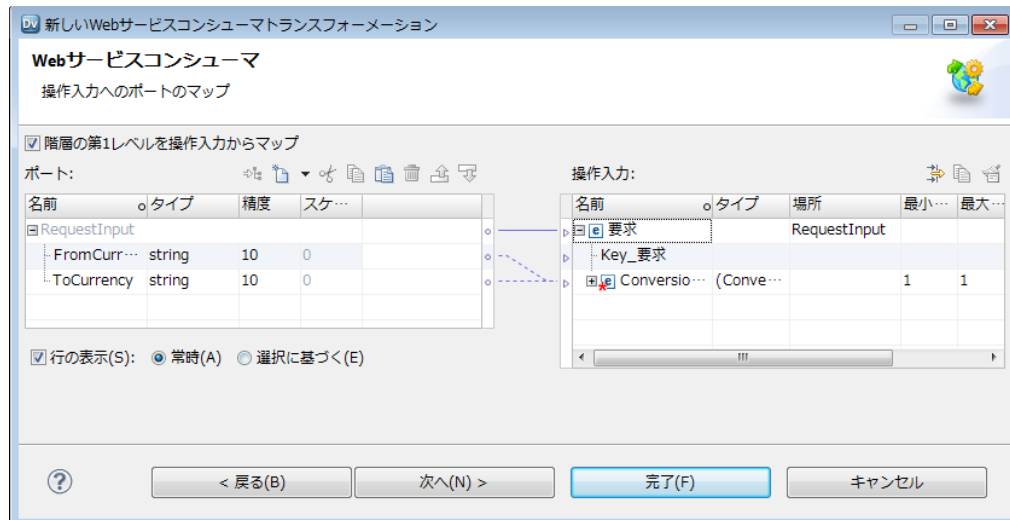
リポジトリ内に WSDL が存在しない場合、[新しい Web サービスコンシューマトランスフォーメーション] ダイアログボックスから WSDL をインポートすることができます。

3. [参照] をクリックし、WSDL から操作を選択します。

SOAP 1.1 バインディングまたは SOAP 1.2 バインディングのある操作を選択できます。

4. [次へ] をクリックします。

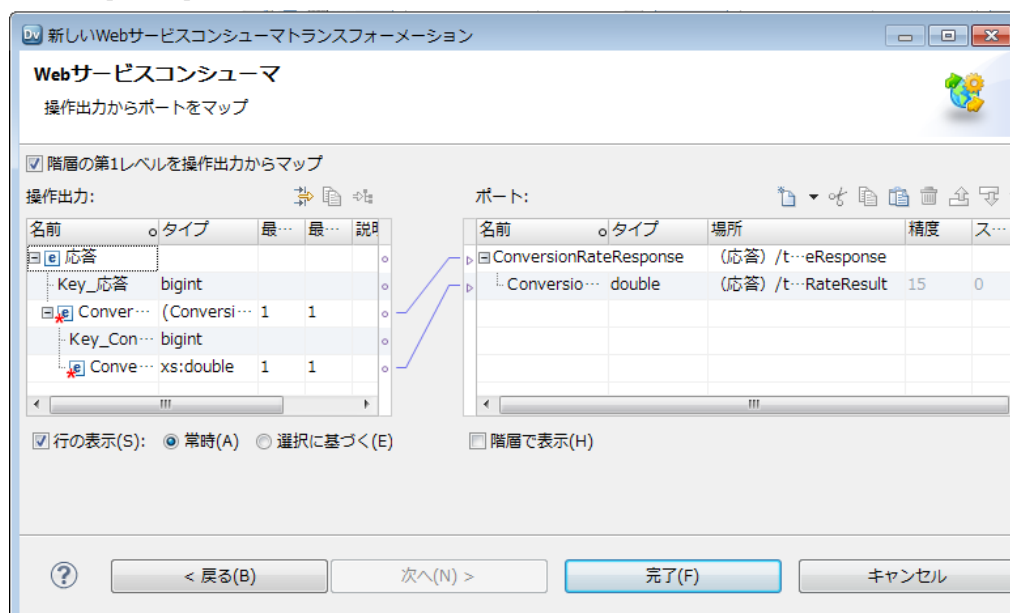
[操作入力へのポートのマッピング] 画面が表示されます。[ポート] 領域には、トランスフォーメーションの入力グループと入力ポートが表示されます。[操作入力] 領域には、要求メッセージ階層が表示されます。



5. 入力グループと入力ポートを定義し、操作入力ノードに入力ポートをマップします。

6. [次へ] をクリックします。

[操作出力からポートをマップ] 画面が表示されます。[操作出力] 領域に、応答メッセージ階層が表示されます。[ポート] 領域に、トランスフォーメーションの出力グループと出力ポートが表示されます。



7. 出力グループと出力ポートを定義し、出力ポートに操作出力ノードをマップします。
8. **【完了】** をクリックします。
9. **【詳細】** ビューをクリックし、トランスフォーメーションプロパティと Web サービス接続を設定します。

#### 関連項目：

- [「操作入力への入力ポートのマッピング」 \(ページ 658\)](#)
- [「出力ポートへの操作出力のマッピング」 \(ページ 662\)](#)
- [「Web サービスコンシューマトランスフォーメーションの詳細プロパティ」 \(ページ 663\)](#)

## Web サービスコンシューマトランスフォーメーションの例

例えば、RT100 製品ラインの注文情報を営業部門へ公開する必要があるとします。営業チームでは、注文の概要と注文の詳細を日常的に照会する必要があります。

仮想テーブル内に毎日の注文情報を公開する論理データオブジェクトを作成します。読み取りマッピングには、最新の RT100 の注文を返す Web サービスコンシューマトランスフォーメーションが含まれます。この Web サービスコンシューマトランスフォーメーションは、RT100 製品ラインに対する毎日の注文の概要と詳細をそれぞれ返す Web サービスを使用します。

### 入力ファイル

入力ファイルは、製品ライン番号を含むフラットファイルです。

この入力ファイルを定義する物理データオブジェクトを作成します。ファイルには Product\_Line というフィールドが 1 個含まれます。フィールド値は RT100 です。 **プロパティビューのランタイムビュー**で、物理データオブジェクトの場所を定義します。

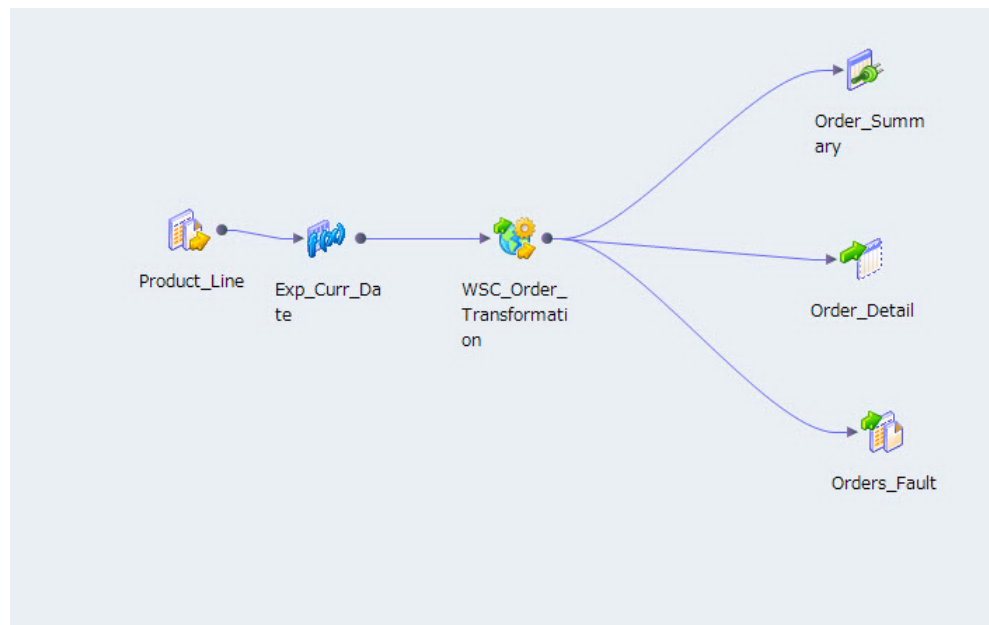
### 論理データオブジェクトモデル

所属部門のビジネスアナリストは、注文の概要と注文の詳細のテーブル構造を説明する論理データモデルを作成します。この論理データモデルには、Order\_Summary と Order\_Detail の論理データオブジェクトが含まれます。

論理データモデルを定義するスキーマが、アナリストによってモデリングツールで作成されます。このスキーマから論理データモデルをインポートし、Order\_Summary と Order\_Detail の論理データオブジェクトを作成します。

## 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マッピングは、論理データオブジェクトを通じてデータにアクセスする方法を記述したものです。



読み取りマッピングには以下のオブジェクトが含まれます。

**Product\_Line**

製品ライン番号を含む入力フラットファイル。

**Exp\_Curr\_Date トランスフォーメーション**

現在の日付と、Web サービスコンシューマトランスフォーメーションのルートレベル入力グループのプライマリーを返す式トランスフォーメーション。

**WSC\_Order トランスフォーメーション**

Web サービスを使用して注文情報を取得する Web サービスコンシューマトランスフォーメーション。トランスフォーメーションは、製品ラインと現在の日付を応答メッセージに含めて Web サービスに渡します。トランスフォーメーションは、Web サービスから注文情報を応答メッセージとして受け取ります。

**Order\_Summary テーブル**

Order\_No、Customer\_Id、Qty、Order\_Date などの注文情報を含む論理データオブジェクト。

**Order\_Detail テーブル**

Order\_No、Product\_Id、Qty、Status などの詳細な注文情報を含む論理データオブジェクト。

**Orders\_Fault**

汎用フォールトメッセージを受け取る出力フラットファイル。

## Web サービスコンシューマトランスフォーメーション

Web サービスコンシューマトランスフォーメーションは、製品ライン、日付、シーケンス番号を入力として受け取ります。このトランスフォーメーションは、Get\_Order\_Info Web サービス操作を使用して、注文情報を取得します。

Web サービスコンシューマトランスフォーメーションを作成する場合、要求と応答の Web サービスメッセージを記述する WSDL データオブジェクトを選択します。Web サービスメッセージには、XML 要素の階層グループが含まれます。要素には他の要素を含めることができます。要素によっては、複数回出現するものもあります。トランスフォーメーションを、リポジトリ内の Order\_Info WSDL オブジェクトから作成します。

トランスフォーメーションの入力ポートを設定し、ポートを操作入力階層にマップします。操作出力階層から出力ポートにノードをマップします。Web サービス接続および実行時プロパティを設定します。

### トランスフォーメーション入力マッピング

**ポート**ビューに入力マッピングを表示している場合、入力ポートを定義し、それらのポートを操作入力のノードにマップできます。

トランスフォーメーションの【**ポート**】領域には、ルートグループと Order グループがあります。ルートグループは、Request 入力グループです。Request 入力グループに、プライマリキーを表すポートを 1 個追加します。

Order グループには **Select\_Date** 入力ポートと **Select\_Product\_Line** 入力ポートがあります。

これらの入力ポートを、【**操作入力**】領域の **Order\_Date** ノードと **Product\_Line** ノードにマップします。

【**操作入力**】領域は、Web サービスコンシューマトランスフォーメーションが Web サービスに渡す要求メッセージを定義します。【**操作入力**】領域には、デフォルトでノードが表示されます。

### トランスフォーメーション出力マッピング

**ポート**ビューに出力マッピングが表示されている場合、操作出力のノードをトランスフォーメーション出力グループにマップすることで、出力ポートを定義できます。

Web サービスは、以下の階層を Web サービス応答メッセージに含めて返します。

```
Response
 Orders
 Order
 Key_Order
 Order_ID
 Order_Date
 Customer_ID
 Total_Qty
 Order_Details
 Order_Detail
 Product_ID
 Description
 Qty
 Status
```

Web サービスから複数の注文が返されます。Order は、Orders レベルの複数出現ノードです。個々の注文に対して、Web サービスは注文の詳細を複数返すことができます。Order\_Detail は、Order\_Detail レベルの複数出現ノードです。

**注:** Developer ツールにより、ユーザーインターフェースに Key\_Order ノードが追加されます。これらのキーを出力グループにマッピングすることで、グループ間のリレーションを定義できます。この例では、Order\_ID は、Order のプライマリキーであると同時に Order\_Details の外部キーでもあります。

[ポート] 領域で、以下の出力グループを作成します。

```
Order
 Order_ID
 Order_Date
 Customer_ID
 Total_Qty

Order_Detail
 Order_ID
 Product_ID
 Description
 Qty
 Status
```

Data Integration Service は、Order\_ID の値が変更されると必ず、Order グループから行を書き出します。

Data Integration Service は、Order\_ID と Product\_ID の値が変更されると必ず、Order\_Detail グループから行を書き出します。

## トランスフォーメーションの詳細プロパティ

Web サービスコンシューマトランスフォーメーションの以下の詳細プロパティを設定します。

### 汎用 SOAP フォールト処理を有効にする

SOAP フォールトメッセージを受け取る出力ポートを追加します。

### 接続

Web サービスにアクセスする Web サービス接続を選択します。

### 圧縮を有効にする

Web サービスコンシューマトランスフォーメーションにより、Web メッセージが gzip で圧縮されます。

## 第 47 章

# Web サービス SOAP メッセージの解析

この章では、以下の項目について説明します。

- [Web サービス SOAP メッセージの解析の概要, 675 ページ](#)
- [トランスフォーメーションのユーザーインターフェース, 676 ページ](#)
- [複数出現出力設定, 677 ページ](#)
- [anyType 要素の解析, 679 ページ](#)
- [派生型の解析, 680 ページ](#)
- [QName 要素の解析, 681 ページ](#)
- [代替グループの解析, 681 ページ](#)
- [SOAP メッセージ内の XML 構造の解析, 682 ページ](#)

## Web サービス SOAP メッセージの解析の概要

Data Integration Service は、Web サービストランスフォーメーションで SOAP メッセージを解析するときに行データを生成します。

Web サービス入力トランスフォーメーションおよび Web サービスコンシューマトランスフォーメーションは、SOAP メッセージの解析を行う Web サービストランスフォーメーションです。

SOAP メッセージを解析するようにトランスフォーメーションを設定するには、SOAP メッセージ階層と同様の構造の出力ポートを作成します。SOAP メッセージ階層内のノードを、作成した出力ポートにマッピングします。

正規化した出力ポートのグループ、非正規化したグループ、ピボット化したポートのグループを設定することができます。SOAP メッセージに派生型、anyType 要素、または代替グループが含まれる場合は、SOAP メッセージインスタンス内に出現する可能性のある型に基づいて別の出力グループを設定することもできます。

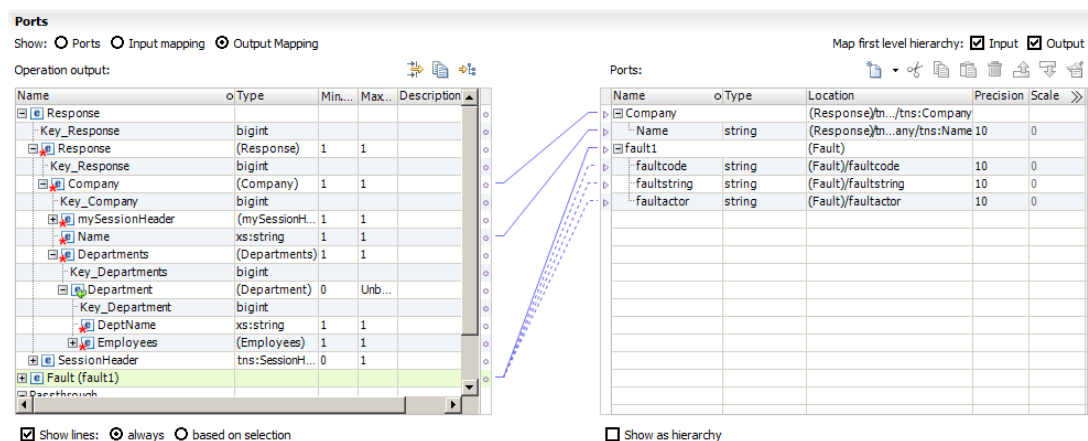
## 関連項目：

- [「Web サービスコンシューマトランスフォーメーションの出力マッピング」](#) (ページ 660)

# トランスフォーメーションのユーザーインターフェース

Web サービスコンシューマトランスフォーメーションおよび Web サービス入力トランスフォーメーションでは、ユーザーインターフェースを介して、SOAP メッセージからトランスフォーメーション出力ポートにデータをマップすることができます。

次の図に、Web サービスコンシューマトランスフォーメーションにおける SOAP 1.1 メッセージノードと出力ポート間のマッピングを示します。



## 操作領域

操作領域には SOAP メッセージ階層が含まれています。階層構造内の階層レベルは、複合ノードまたは複数出現ノードによって設定されます。Developer ツールは階層レベルにキーを割り当て、そのキーによって階層レベル間の親子リレーションが設定されます。

前の図に示した SOAP メッセージ階層には、次の階層レベルが含まれます。

### 応答または要求

応答メッセージまたは要求メッセージのルートを表すレベル。

### 会社名

最上位レベルの要求データ。

### 部門

その会社内の複数出現の部門。

### 従業員

従業員は、部門内の複合要素です。

### フォールトグループ

エラーメッセージを受信するフォールトメッセージグループ。



## ポート領域

SOAP メッセージの階層レベルから出力ポートへ、データをマップすることができます。出力ポート内の各グループは、プライマリ外部キーリレーションによって、別の出力グループに関連付けられている場合もあります。

前の図に示したトランスフォーメーションには、SOAP メッセージ内のノードグループに対応する出力ポートのグループがあります。

# 複数出現出力設定

入力トランスフォーメーションまたは Web サービスコンシューマトランスフォーメーションから複数出現のデータが返される場合、出力ポートを別の構成で設定することもできます。

正規化した出力データ、ピボット化した出力データ、または非正規化した出力データを構成することができます。

例えば、SOAP メッセージに Departments（部門）と Employees（従業員）という複合要素が含まれているとします。各部門には、複数の従業員が含まれています。Departments は Employees の親です。

SOAP メッセージには、以下のような要素の階層が含まれます。

```
Departments
 Department_ID
 Department_Name
 Employees
 Employee_ID
 Employee_Name
```

## 正規化したリレーショナル出力

正規化した出力データを作成する場合、それらのデータ値は出力グループ内で反復されません。SOAP メッセージ内の階層レベルと、出力ポートのグループとの間に、1 対 1 リレーションを作成します。

SOAP メッセージに Departments という親階層レベルと、Employees という子階層レベルが含まれている場合に、以下のようなポートのグループを作成するとします。

```
Departments
 Department_Key
 Department_ID
 Department_Name
```

```
Employees
 Department_Key
 Employee_ID
 Employee_Name
```

Department\_Key は、Employees 出力グループを Departments 出力グループに関連付ける生成キーです。

## 生成キー

出力グループを作成すると、その出力グループと別の出力グループとが、Developer ツールの生成キーを使用して関連付けられます。Developer ツールは、親グループと子グループの両方に bigint 値のキーを追加します。生成キーのキー値は、実行時に Data Integration Service によって作成されます。

## 例

SOAP 階層内に次のノードがあります。

```
Departments
 Dept_Key
 Dept_Num
 Dept_Name

 Employees
 Dept_FK
 Employee_Num
 Employee_Name
```

Departments の出力ポートグループを作成する場合、Departments ノードをポート領域の空のフィールドにマップします。Developer ツールによって、次の出力グループが作成されます。

```
Departments
 Dept_Num
 Dept_Name
```

Employees ノードをポート領域の空のフィールドにマップすると、Developer ツールによって、Employees グループを Departments グループに関連付けるよう求めるメッセージが表示されます。Employees グループを複数のグループに対して関連付けることができます。Developer ツールによって、各グループにキーが追加されます。

Developer ツールによって、以下のグループと生成キーが作成されます。

```
Departments
 Key_Departments
 Dept_Num
 Dept_Name

 Employees
 Key_Departments
 Employee_Num
 Employee_Name
```

**注:** 生成キーにノードをマップする必要はありません。キー値は実行時に Data Integration Service によって作成されます。

Developer ツールは、1 個の出力グループ内の複数のレベルに対して生成キーを作成できます。Employees グループに、以下のようなポートが含まれるとします。

```
Employees
 Key_Employees
 Key_Departments
 Key_Managers
 Employee_Num
 Employee_Name
```

Key\_Departments および Key\_Managers は、親グループをポイントする生成キーです。Key\_Employees は、Employees グループを示す生成キーです。Key\_Employees は、子グループが Employees グループに関連付けられている場合に表示されます。

## 非正規化したリレーショナル出力

リレーショナル出力を非正規化することができます。出力データを非正規化すると、親グループ内の要素の値が、個々の子要素に対して繰り返し出現するようになります。

出力データを非正規化するには、親階層レベルから出力ポートの子グループにノードをマップします。

次の例に、Employees 出力グループ内の Department\_ID および Department\_Name を示します。

```
Employees
 Department_ID
 Department_Name
```

Employee\_ID  
Employee\_Name

Department\_ID および Department\_Name は、部門内の全従業員に対して繰り返し表示されます。

Department_ID	Department_Name	Employee_ID	Employee_Name
100	経理	56500	Kathy Jones
100	経理	56501	Tom Lyons
100	経理	56509	Bob Smith

## ピボット化したリレーショナル出力

出力グループに表示される複数出現要素の数を指定することができます。

複数出現要素をピボット化するには、出力ポートの親グループに複数出現子要素をマップします。Developer ツールによって、親に含める子要素の数を指定するよう求めるメッセージが表示されます。

次の例に、親グループ部門内の 2 つの Employee\_ID インスタンスを示します。

**Departments**  
Department\_ID  
Department\_Name  
Employee\_ID1  
Employee\_ID2

## anyType 要素の解析

anyType 要素は、WSDL またはスキーマで使用されているグローバル型すべてを表します。ノードを Developer ツール内のポートにマップする場合、SOAP メッセージ内に anyType 要素の代わりに出現するデータ型を選択します。SOAP メッセージ内の anyType 要素は、複合型または xs:string に置き換える必要があります。選択したデータ型ごとに、ポートグループを作成します。

出力ポートにデータをマップする型を選択する必要があります。WSDL またはスキーマにグローバル型が含まれない場合は、Developer ツールにより anyType 要素が xs:string に置き換えられます。

操作領域内で要素型を選択するには、**[タイプ]** カラムで anyType 要素の **[選択]** をクリックします。利用可能な複合型および xs:string のリストが表示されます。

anyType 要素を派生型で置き換える場合、Data Integration Service は一度に 1 つの型に対して要素を設定します。SOAP メッセージは基本型と派生型のデータを同時に含むことはありません。

### 派生型の例

WSDL に anyType 要素が 1 つ含まれています。この要素を AddressType とその派生型である USAddressType に置き換えます。SOAP メッセージ階層に以下のグループがあります。

```
Address:AddressType (base type)
 Address: AddressType
 Street
 City

Address:USAddressType (derived type)
 Street
 City
 State
 ZipCode
```

SOAP メッセージには以下のデータが含まれています。

```
<address xsi:type="AddressType">
 <street>1002 Mission St.</street>
 <city>san jose</city>
</address>
```

```
<address xsi:type="USAddressType">
 <street>234 Fremont Blvd</street>
 <city>Fremont</city>
 <zip>94556</zip>
 <state>CA</state>
</address>
```

Data Integration Service は xsi: AddressType に対して以下の 1 行を返します。

Street	City
1002 Mission St.	San Jose

Data Integration Service は、派生型 xsi: USAddressType に対して以下の 1 行を返します。

Street	City	状態	郵便番号
234 Fremont Blvd.	Sunnyvale	CA	94556

型が xsi: USAddressType である場合、Data Integration Service は AddressType に値を設定しません。

## 派生型の解析

派生型を含む SOAP メッセージを解析することができます。SOAP メッセージからデータを受け取るポートを定義する場合、SOAP メッセージ内に出現する可能性のある型を選択します。選択した型の要素によって、作成する必要のあるポートが決まります。

例えば、WSDL に AddressType およびその派生型である USAddressType が含まれるとします。Developer ツールの操作領域で、以下のようなグループを作成できます。

```
Address
 Address: AddressType
 Street
 City

 Address:USAddressType
 Street
 City
 State
 ZipCode
```

SOAP メッセージに、以下のようなデータが含まれているとします。

```
<address>
 <street>1002 Mission St.</street>
 <city>san jose</city>
</address>
```

```
<address xsi:type="USAddressType">
 <street>234 Fremont Blvd</street>
 <city>Fremont</city>
 <zip>94556</zip>
 <state>CA</state>
</address>
```

```
<address xsi:type="USAddressType">
```

```

<street>100 Cardinal Way</street>
<city>Redwood City</city>
<zip>94536</zip>
<state>CA</state>
</address>

<address>
<street>100 El Camino Real</street>
<city>Sunnyvale</city>
</address>

```

基本型である Address に対して、Data Integration Service から以下の行が返されます。

Street	City
1002 Mission St.	San Jose
234 Fremont Blvd	Sunnyvale
100 Cardinal Way	Redwood City
100 El Camino Real	Sunnyvale

派生型である USAddress に対して、Data Integration Service から以下の行が返されます。

Street	City	状態	郵便番号
234 Fremont Blvd.	Sunnyvale	CA	94556
100 Cardinal Way	Redwood City	CA	94536

Data Integration Service は、すべての住所を基本型で返します。Data Integration Service は、米国式表記の住所データを派生型で返します。派生型には、USAddressType が基本型から継承した Street 要素と City 要素も含まれています。

## QName 要素の解析

Data Integration Service で SOAP メッセージの QName 要素を解析すると、スキーマで定義されている名前空間プレフィックスを使用するためにスキーマの名前空間に属する QName 値が更新されます。そうでない場合は、Data Integration Service により要素の値は更新されません。

例えば、スキーマには名前空間"http://user/test"に対して定義されている名前空間プレフィックス tns があるとし、SOAP メッセージには、同じ名前空間に対して定義されている名前空間プレフィックス mytns があります。Data Integration Service が QName 値 mytns:myelement を解析するとき、値は tns:myElement に変更されます。

Data Integration Service で SOAP メッセージの QName 要素を生成するとき、要素の値は更新されません。

## 代替グループの解析

代替グループは、ある要素を同じグループ内の別の要素に置き替えます。代替グループは、派生型とほぼ同じですが、各要素定義に代替グループ名が含まれている点で異なります。

代替グループ内の特定の型から要素を受け取る、出力ポートのグループを設定することができます。代替グループ内の別の型から要素を受け取る、別の出力ポートグループを設定することもできます。

# SOAP メッセージ内の XML 構造の解析

SOAP メッセージには、choice、list、union などの要素が含まれる場合があります。

Web サービストランスフォーメーションは、これらの構造を持つ SOAP メッセージを解析できますが、いくつかの制限を伴います。

## choice 要素

choice 要素の子要素は、<choice>宣言内の要素のいずれかに限定されます。

以下に、従業員または契約社員の person 要素を示します。

```
<xs:element name="person">
 <xs:complexType>
 <xs:choice>
 <xs:element name="employee" type="employee"/>
 <xs:element name="contractor" type="contractor"/>
 </xs:choice>
 </xs:complexType>
</xs:element>
```

choice 要素のマッピングは、以下の方法で行うことができます。

- 出力グループ内の個々の choice 要素に対して出力ポートを作成します。一部の要素では、出力行に NULL 値が表示されることがあります。
- 各 choice 要素に対して出力グループを作成します。上の例の場合、employee グループと contractor グループを作成します。Data Integration Service は、SOAP メッセージ内に表示される要素に基づいて行を生成します。

## list 要素

list 要素は、"Monday Tuesday Wednesday"などのように、単純型の値が複数含まれている XML 要素です。

Data Integration Service は、list を文字列値として返すことができます。SOAP メッセージにリストが含まれている場合に、リストの項目を別々の出力行にマップすることはできません。マッピング内でリスト内の要素を抽出する必要がある場合には、式フォーメーションを設定することで、リスト内の要素を個別に取り出すことができます。

## union 要素

union 要素は、複数の型の組み合わせから成る単純型です。

以下のテキストに、size\_no と size\_string という 2 つの単純型を組み合わせた、Size 要素を示します。

```
<xs:element name="Size">
 <xs:simpleType>
 <xs:union memberTypes="size_no size_string" />
 </xs:simpleType>
</xs:element>
```

Size 要素を出力ポートにマップするには、その Size 要素用のポートを 1 つ作成します。作成したポートを文字列として設定します。マッピング内に別のトランスフォーメーションを設定し、データを別のデータ型に変換することができます。

## 第 48 章

# Web サービス SOAP メッセージの生成

この章では、以下の項目について説明します。

- [Web サービス SOAP メッセージの生成の概要, 683 ページ](#)
- [トランスフォーメーションのユーザーインターフェース, 684 ページ](#)
- [ポートと階層レベルのリレーション, 685 ページ](#)
- [キー, 686 ページ](#)
- [ポートのマップ, 687 ページ](#)
- [複数出現ポートのピボット化, 689 ページ](#)
- [非正規化データのマップ, 691 ページ](#)
- [派生型および要素の置き換え, 692 ページ](#)
- [SOAP メッセージ内の XML 構造の生成, 693 ページ](#)

## Web サービス SOAP メッセージの生成の概要

Data Integration Service では、SOAP メッセージを生成する際に、入力データのグループから XML データが生成されます。Web サービスコンシューマトランスフォーメーション、Web サービスの出力トランスフォーメーション、またはフォールトトランスフォーメーションを作成する場合は、SOAP メッセージ階層にマップする入力ポートを設定します。

SOAP メッセージを生成するようにトランスフォーメーションを設定するには、入力ポートのグループを作成し、各グループを SOAP メッセージ階層内のグループにマップします。SOAP メッセージの構造は、WSDL またはスキーマで定義されます。

SOAP メッセージ内のデータのグループを、非正規化入力データから設定することができます。また、複数出現入力データを、SOAP メッセージ内の複数出現ノードにピボット化することもできます。

データは、SOAP メッセージ内の派生型、anyType 要素、または置き換えグループにマップできます。トランスフォーメーションを定義するときに、SOAP メッセージに出現可能な型を選択する必要があります。選択した型によって、作成が必要となる入力ポートが決まります。

Developer ツールで SOAP メッセージ階層を表示すると、階層にキーが含まれていることがわかります。このキーは、SOAP メッセージには現れません。Data Integration Service では、このキーを使用して、SOAP メッセージ内のグループ間に親子リレーションが定義されます。キー値を設定するには、SOAP メッセージ内のキーに入力データをマップします。

## 関連項目：

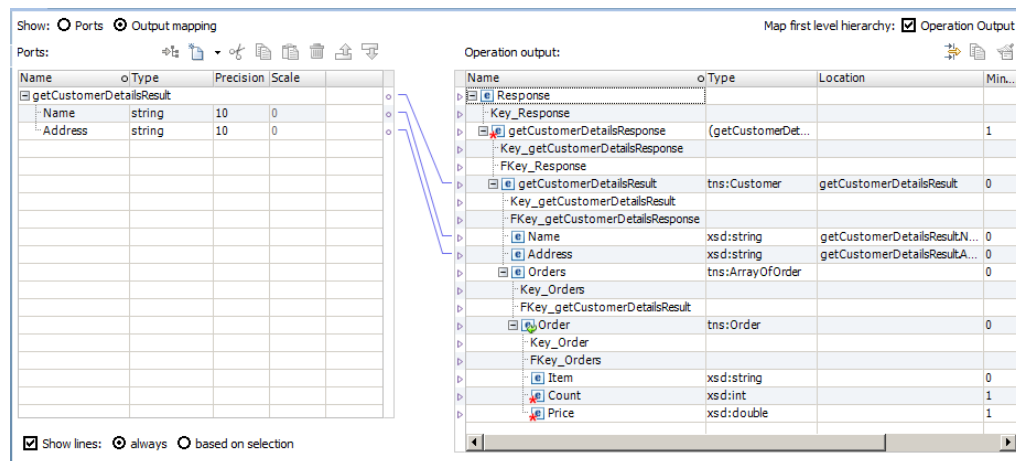
- [「Web サービスコンシューマトランスフォーメーションの入力マッピング」 \(ページ 657\)](#)

# トランスフォーメーションのユーザーインターフェース

Web サービスの出力トランスフォーメーション、フォールトトランスフォーメーション、および Web サービスコンシューマトランスフォーメーションには、SOAP メッセージを設定するために使用できるユーザーインターフェースが含まれています。

SOAP メッセージを生成するようにトランスフォーメーションを設定するには、SOAP メッセージ階層に似た構造内で入力ポートを作成します。階層の構造は、WSDL またはスキーマによって決まります。各入力ポートを SOAP メッセージ内のノードにマッピングします。

以下の図は、Web サービスの出力トランスフォーメーションにおける入力ポートと SOAP メッセージノードの間のマッピングを示しています。



## 【入力ポート】領域

入力ポートのグループを【入力ポート】領域で作成します。SOAP メッセージ階層内の各レベルについて、マッピングが必要な入力ポートを組み込みます。

Response または Request 入力グループと、データを受け取る子グループを作成する必要があります。

入力ポートグループを作成する際に、各親グループでプライマリキーを定義します。各子グループで外部キーを定義します。外部キーによって、グループが親グループに関連付けられます。

WSDL ルートレベルでデータを渡すことがなければ、Response レベルまたは WSDL ルートレベルでキーを定義する必要はありません。例えば、ルートレベルに HTTP ヘッダーが含まれている場合があります。

以下に示す顧客と注文のグループに似たポートのグループを作成するとします。

```
Response
 Response_Key

 Customer_Details_Root
 Key_Cust_Det
 FK_Response_Key

 Customer
```



```

Customer_ID
FK_Cust_Det
Name
Address

Orders
Order_Num
FK_Cust_ID

Order_Items
Order_Num
Item
Count
Price

```

## 操作領域

**【操作】** 領域には、WSDL またはスキーマで定義されている SOAP メッセージ階層内の要素が表示されます。SOAP メッセージに、WSDL またはスキーマの要素がすべて含まれているとは限りません。メッセージには、入力ポートからマップしたデータが含まれています。

複数出現ノードおよび複合ノードによって、SOAP メッセージ構造の階層レベルが定義されます。Developer ツールにより、これらのレベルにキーが追加されてレベル間の親子リレーションが作成されます。リーフレベルを除く階層内のすべてのレベルには、プライマリキーが設定されます。各子レベルには、親レベルへの外部キーが設定されます。SOAP メッセージ階層に見られるキーは、SOAP メッセージインスタンスには現れません。Data Integration Service では、SOAP メッセージ生成時にデータのレベルを関連付けるために、キーの値が必要となります。

**【場所】** カラムには、グループ名および SOAP メッセージ内の要素のデータを含んでいる入力ポートが表示されます。入力ポートをノードにマップするまで、**【場所】** カラムは空白です。

前出の図では、SOAP メッセージに顧客詳細および注文の単一インスタンスが含まれています。Orders グループには、Order という複数出現要素が含まれています。SOAP メッセージ階層では、以下のレベルがキーで関連付けられています。

```

Response
 GetCustomerDetailsResponse
 GetCustomerDetailsResult
 Orders
 Order

```

Response レベルは、レスポンスメッセージのルートを表します。Data Integration Service では、SOAP メッセージにヘッダーを添付するためにこのレベルが必要となります。

GetCustomerDetailsResponse レベルは、メッセージのルートです。

## ポートと階層レベルのリレーション

入力ポートを SOAP メッセージ階層にマップする場合は、入力グループと SOAP メッセージ階層レベルの間のリレーションを維持します。例えば、Department（部門）と Employee（従業員）の 2 つのグループがあるとしたします。

Department 入力グループは、以下の行を受け取ります。

Dept_num	名前	場所
101	HR	New York
102	Product	California

Employee 入力グループは、以下の行を受け取ります。

Dept_num	Employee
101	Alice
101	Bob
102	Carol
102	Dave

Employee グループの部門番号を、Department グループと Employee グループの間にリレーションを確立する外部キーとしてマップします。部門番号は、部門階層レベルに出現しますが、従業員レベルでは出現しません。

SOAP メッセージには、以下の XML 構造が格納されます。

```
<department>
 <dept_num>101</dept_num>
 <name>HR</name>
 <location>New York</location>

 <employee>
 <name>Alice</name>
 </employee>

 <employee>
 <name>Bob</name>
 </employee>
</department>

<department>
 <dept_num>102</dept_num>
 <name>Product</name>
 <location>California</location>

 <employee>
 <name>Carol</name>
 </employee>

 <employee>
 <name>Dave</name>
 </employee>
</department>
```

## キー

SOAP メッセージ階層にはキーが含まれています。Data Integration Service では、SOAP メッセージ内で XML 階層を構築するために、キー値が必要となります。

入力ポートデータを、SOAP メッセージ階層内のキーにマップする必要があります。データを提供する各レベルのキーに、データをマッピングします。複数出現ノードがある場合は、ノードを親に関連付ける必要があります。

SOAP メッセージ内では、キーが型のない状態で出現します。キーにマッピングするポートのデータ型は、string、integer、または bigint であることが必要です。親グループのプライマリキーと各子グループの外部キーは、データ型、精度、および位取りが同じである必要があります。生成されたキーを SOAP メッセージキーにマップできます。

ポートは、ノードおよび同じ階層レベルにあるキーにマップできます。例えば、Employee\_ID を SOAP メッセージ内のノードにマップし、さらに Employee レベルのキーにマップします。

階層内の 2 つのグループノードに親子リレーションがある場合は、以下のタスクを実行します。

- 親ノードグループのプライマリキーにポートをマップします。
- 子ノードグループの外部キーにポートをマップします。

プライマリキーを入力ポートにマップして、プライマリキーが Null である行またはプライマリキーが重複している行を削除することもできます。

同じキーに複数のポートをマップすることで、SOAP メッセージ内に複合キーを作成できます。複合キーは、データを非正規化し、かつ複数出現する値の組み合わせに対して一意のキーを維持する必要がある場合に使用します。string 値、bigint 値、または integer 値を含む複合キーを作成できます。

**注:** 操作マッピングに式トランスフォーメーションを組み込んで、キー値を生成することができます。

### 複合キーの例

以下に示すポートのグループから、一意の事業部 (Division) -部門 (Department) キーを設定します。

```
Company
 Company_Num
 Company_Name

 Division
 Company_Num
 Division_Num
 Division_Name

 Department
 Division_Num
 Dept_Num
 Dept_Name
 Location
```

Dept\_Num は事業部内では一意ですが、会社内の全事業部で一意ではありません。

事業部と部門の情報を含んだ Department グループを設定することができます。事業部番号と部門番号を複合キーの一部として設定します。

```
Department
 Division_Num + Dept_Num (key)
 Dept_Name
 Location
```

ポートをマップした順序によって、キー値が決まります。

## ポートのマップ

入力ポートを作成したら、各入力ポートを SOAP メッセージ階層にマップします。ポートの場所は、**[操作]** 領域でノードの横に表示されます。

ポートは以下のタイプのノードにマップできます。

### アトミックノード

子がなく分割不可能な単純要素または属性。

### 複数出現アトミックノード

階層内の同じ場所で複数回出現する単純要素または属性。

## 複合ノード

他の要素を含んでいる要素。

親ノードに場所がない場合は、入力グループ名が親ノードの場所として受け取られます。親ノードに場所がある場合は、階層レベル内の各ノードが同じ場所から出力場所を得る必要があります。

入力グループ名を、階層レベルの親ノードにマップできます。Developer ツールにより、階層内の親ノードの場所フィールドが更新されます。階層内でグループに属している子ノードは更新されません。入力ポートを子ノードにマップする場合は、各入力ポートが親ノードと同じ場所であることが必要です。

入力グループを階層レベルにマッピングした後に、入力グループを変更することができます。【クリア】をクリックするか、[ポート] 領域と [操作] 領域の間の線を削除することができます。線を削除するには、線のポインタをドラッグして対象の線を選択します。【削除】をクリックします。

## ポートのマップ

ポートを SOAP メッセージ内のノードにマップする場合は、ポートをマップするノードのタイプによって、Developer ツールでの結果が変わります。

以下の表に、[操作] 領域で単一ポートを各種のターゲットノードにマップした場合の結果を示します。

ターゲットノード	結果
アトミックノード	単一ポートをノードおよび場所を持たない親ノードにマップした場合、そのノードはポートの場所を受け取ります。親ノードの場所には、単一ポートの入力グループの場所が入ります。単一ポートをノードおよび場所を持つ親ノードにマップした場合は、親ノードの場所を変更し、同じレベルにある他の子ノードの場所をクリアすることができます。階層レベルの場所は、ポートのグループ名に変わります。
複数出現アトミックノードまたはそのノードのプライマリキー	単一ポートを複数出現アトミックノードにマップした場合は、Developer ツールによってアトミックノードの場所が選択したポートのグループに設定されます。
複合ノード	単一ポートを複合ノードにマップした場合は、Developer ツールによって複合ノードの場所がポートを含んでいるグループの場所に設定されます。Developer ツールから、ポートの割り当て先となる単独出現アトミックノードを指定するように求められます。 すべての単独出現アトミックノードに場所がある場合は、複合ノードをマップできません。

## グループのマッピング

入力グループを SOAP メッセージ内のノードにマッピングする場合は、ポートをマッピングするノードのタイプによって、Developer ツールでの結果が変わります。

以下の表に、**【操作】** 領域でグループをノードにマッピングした場合の結果を示します。

ターゲットノード	結果
アトミックノード	アトミックノードにグループをマッピングすることはできません。
複数出現アトミックノード	ノードとプライマリキーの場所を更新するために、入力グループ内のポートを選択するように要求されます。
複数出現複合ノード	Developer ツールにより、複合ノードの場所がグループの場所に設定されます。

## 複数のポートのマッピング

複数のポートを SOAP メッセージ内のノードにマッピングする場合は、ポートをマッピングするノードのタイプによって、Developer ツールでの結果が変わります。同じグループからマッピングを行う場合は、複数のポートを同時にマッピングできます。

以下の表に、複数ポートをノードにマッピングした場合のノードの結果を示します。

ターゲットノード	結果
単一のアトミックノード	複数のポートを単一ノードにマッピングした場合は、 <b>【操作】</b> 領域で複数の単一アトミックノードの場所を更新します。階層のレベルに十分な数の更新対象ノードがない場合は、Developer ツールによって存在するノードにのみポートがマッピングされます。
複数出現アトミックノード	複数のポートを複数出現アトミックノードにマッピングする場合は、ポートをそのノードの複数の出現箇所にピボット化します。Developer ツールにより、マッピングしたポートの数に基づいてノードのインスタンスが作成されます。投影したポートの数を示すメッセージが表示されます。
複数出現複合ノード	複数のポートを複合ノードにマッピングする場合は、更新対象となる単独出現アトミックノードを選択する必要があります。ポートをそのノードの複数の出現箇所にピボット化します。Developer ツールにより、マッピングしたポートの数に基づいてノードのインスタンスが作成されます。

## 複数出現ポートのピボット化

複数の入力ポートを、SOAP メッセージ内の複数出現ノードにマッピングできます。Developer ツールにより、入力データが SOAP メッセージ内の複数のノードにピボット化されます。

ピボット化する要素の数を変更するには、**【マッピングオプション】** ダイアログボックスで **【既存のピボット化のオーバーライド】** を選択します。

ピボット化されたポートインスタンスの 1 つを **【ポート】** 領域から削除すると、Developer ツールによってすべてのインスタンスが **【操作】** 領域から削除されます。

## ピボット化の例

入力グループに以下の行があるとします。

Num	名前	場所	emp_name1	emp_name2	emp_name3
101	HR	New York	Alice	Tom	Bob
102	製品	California	Carol	Tim	Dave

各行には、部門番号と 3 人の従業員名が含まれています。

Employee は、SOAP メッセージ階層内の複数出現ノードです。Employee のすべてのインスタンスを、入力行から SOAP メッセージ階層にマップできます。Employee のすべての出現箇所を選択します。【マップ】をクリックします。【マップオプション】ダイアログボックスで、一覧からノードを選択するように求められます。

Developer ツールにより、SOAP メッセージ階層で Employee ノードが複数の name ノードを含むように変更されます。

```
Department
 num
 name
 location
 Employee (unbounded)
 emp_name1
 emp_name2
 emp_name3
```

SOAP メッセージから以下の階層が返されます。

```
<department>
 <num>101</num>
 <name>HR</name>
 <location>New York</location>
 <employee>
 <emp_name>Alice</name>
 </employee>
 <employee>
 <emp_name>Tom</name>
 </employee>
 <employee>
 <emp_name>Bob</name>
 </employee>
</department>

<department>
 <num>102</num>
 <name>Product</name>
 <location>California</location>
 <employee>
 <emp_name>Carol</name>
 </employee>
 <employee>
 <emp_name>Tim</name>
 </employee>
 <employee>
 <emp_name>Dave</name>
 </employee>
</department>
```

# 非正規化データのマップ

非正規化データをマップし、SOAP メッセージ内の正規化ノードに渡すことができます。

非正規化データをマップする場合は、1 つの入力グループから SOAP メッセージ階層内の複数のノードにデータを渡します。SOAP メッセージ内に、以下のタイプのリレーションに似たグループリレーションを作成することができます。

## 線形ノードリレーション

ノード A はノード B の親であり、ノード B はノード C の親であり、ノード C はノード D の親です。

## 階層ノードリレーション

ノード A はノード B の親であり、ノード C の親でもあります。ノード B とノード C の間に関連はありません。

以下の表に、事業部と部門の非正規化データを含んでいる入力行を示します。

Division	Dept_Num	Dept_Name	Phone	Employee_Num	Employee_Name
01	100	経理	3580	2110	Amir
01	100	経理	3580	2113	Robert
01	101	Engineering	3582	2114	Stan
01	101	Engineering	3582	2115	Jim
02	102	Facilities	3583	2116	Jose

入力データには、一意の従業員番号および名前が含まれています。事業所データと部門データは、同じ部門および事業部に所属する従業員ごとに繰り返されます。

## 線形グループリレーション

ポートを設定する際に、Division、Department、および Employee に別々のグループを設定できます。Division は Department の親であり、Department は Employee の親です。以下の線形構造でグループを設定できます。

```
Division
 Division_Key
 Division_Num
 Division Name

 Department
 Department_Key
 Division_FKey
 Dept_Num
 Dept_Name
 Phone

 Employee
 Department_Fkey
 Employee_Num
 Employee_Name
```

SOAP メッセージには Division および Department の一意のインスタンスが含まれますが、Division\_Num と Dept\_Num は入力データ内で繰り返し出現します。Division\_Num を Division グループのプライマリキーとして定義します。Dept\_Num を Department グループのプライマリキーとして定義します。

## 階層グループリレーション

Division 親グループおよび Department と Employee の各子グループで構成されるグループ階層を作成することができます。Department と Employee の間には、プライマリキーと外部キーのリレーションがありません。

ん。Department と Employee は、Division の子です。これらのグループは、以下の構造で設定することができます。

```
Division
 Division_Key
 Division_Num
 Division_Name

 Department
 Division_FKey
 Dept_Num
 Dept_Name

 Employee
 Division_FKey
 Employee_Num
 Employee_Name
```

## 派生型および要素の置き換え

入力ポートは、SOAP メッセージ内の派生複合型、anyType 要素、および置き換えグループにマップできます。SOAP メッセージには、基本型および派生型の要素を組み込むことができます。

型リレーションでは、基本型は別の型の派生元となる型です。派生型は、基本型から要素を継承します。拡張複合型は基本型から要素を継承する派生型であり、追加の要素を含んでいます。限定複合型は、基本型からの要素の一部を制限している派生型です。

## 派生型の生成

WSDL またはスキーマに派生型が含まれている場合は、SOAP メッセージに追加する型を選択する必要があります。

例えば、WSDL で基本型 AddressType が定義されているとします。この WSDL には、AddressTypes から派生した USAddressType と UKAddressType も含まれています。

各型には以下の要素が含まれています。

- AddressType:street、city
- USAddressType (AddressType を拡張したもの):state、zipCode
- UKAddressType (AddressType を拡張したもの):postalCode、country

[操作] 領域で USAddressType を選択すると、Developer ツールによって USAddressType 要素のグループが SOAP メッセージ内に作成されます。このグループには、基本の住所から street と city が取り込まれ、USAddress の state と zipCode が取り込まれます。基本型を拡張した派生型には、必ず基本型の要素が含まれます。

選択可能なすべての派生型を SOAP メッセージ用に選択すると、Developer ツールによって以下のようなグループが SOAP 階層内に作成されます。

```
Address
 Address: Address
 Street
 City

 Address:USAddressType
 Street
 City
 State
 ZipCode
```



```
Address: UKAddressType
 Street
 City
 PostalCode
 Country
```

Address、USAddress、および UKAddress 用に入力ポートグループを定義する必要があります。

## anyType 要素および属性の生成

スキーマ要素および属性のなかには、SOAP メッセージに対して Any 型のデータを許可するものがあります。

anyType 要素は、グローバルに認識されるすべての型を表す選択肢です。ポートを SOAP メッセージの anyType 要素にマップする前に、利用可能な複合型または xs:string を選択します。WSDL またはスキーマに複合型が含まれない場合は、Developer ツールにより anyType 要素型が xs:string に置き換えられます。

操作領域内で要素型を選択するには、**【タイプ】** カラムで anyType 要素の **【選択】** をクリックします。利用可能な複合型および xs:string のリストが表示されます。

以下の要素および属性は、Any 型のデータを許可します。

### anyType 要素

要素に関連する XML ファイル内の任意のデータ型にすることができます。

### anySimpleType 要素

要素に関連する XML ファイル内の任意の simpleType にすることができます。

### ANY 内容要素

要素をスキーマで定義されている任意のグローバル要素にすることができます。

### anyAttribute 属性

要素をスキーマで定義されている任意の属性にすることができます。

## 置き換えグループの生成

置き換えグループを使用して、SOAP メッセージ内のある要素を別の要素に置き換えることができます。置き換えグループは派生型とほぼ同じように機能しますが、要素定義に置き換えグループ名が組み込まれる点異なります。

例えば、基本型 Address および派生型 USAddress と UKAddress があるとします。

```
xs:element name="Address" type="xs:string"/>
<xs:element name="USAddress" substitutionGroup="Address"/>
<xs:element name="UKAddress" substitutionGroup="Address"/>
```

SOAP メッセージ階層を設定する際に、SOAP メッセージ内で Address に置き換わる要素を選択することができます。

## SOAP メッセージ内の XML 構造の生成

WSDL またはスキーマには、choice 要素、list 要素、または union 要素が含まれている場合があります。Web サービスのトランスフォーメーションで、これらの要素を含んだ SOAP メッセージを生成することができます。

# choice 要素

choice 要素の子要素は、<choice>宣言内の要素のいずれかに限定されます。

choice 要素を含んでいる SOAP メッセージにポートをマップするには、choice 構造のすべての要素が含まれている入力グループを 1 つ作成します。例えば、品目の説明が寸法または重量であるとしてします。

item: description, choice {dimension, weight}

説明が寸法である場合、その説明は、長さ、幅、高さで構成される複合型です。

説明が重量である場合、その説明は単純な文字列型です。

この入力データの行は以下のとおりです。

description	length	width	height	weight
box	20cm	18cm	15cm	NULL
coffee	NULL	NULL	NULL	500g

SOAP メッセージには、寸法または重量の説明が入った Item グループが含まれています。

```
Item
 Description
 Dimension
 Length
 Width
 Height
 Weight
```

入力データが NULL 値であると、XML 出力の要素が欠落します。

SOAP メッセージには以下のデータが含まれています。

```
<item>
 <desc>box</desc>
 <dimension>
 <length>20cm</length>
 <width>18cm</width>
 <height>15cm</height>
 </dimension>
</item>

<item>
 <desc>coffee</desc>
 <weight>500g</weight>
</item>
```

# list 要素

list は、同じ要素または属性に複数の単純型値を入れることのできる XML 要素です。Data Integration Service では、入力データ内の list が統合された一連のデータとして表されている場合に、list を処理できません。

list 内の各項目が個別の要素である場合（ClassDates1、ClassDates2、ClassDates3 など）、Data Integration Service ではそれらの項目を list として処理できません。SOAP メッセージで list を返す必要がある場合は、式トランスフォーメーションを使用してこれらの項目を文字列へと統合できます。

以下の入力行には、曜日で構成される ClassDates という list 要素が含まれています。

CourseID	Name	ClassDates
Math 1	Beginning Algebra	Mon Wed Fri
History 1	World History	Tue Thu

Data Integration Service では、以下の XML 構造の SOAP メッセージが返されます。

```
<class>
 <courseId>Math 1</courseId>
 <name>Beginning Algebra</name>
 <classDates>Mon Wed Fri</classDates>
</class>
<class>
 <courseId>History 1</courseId>
 <name>World History</name>
 <classDates>Tue Thu</classDates>
</class>
```

## union 要素

union 要素は、複数の型の組み合わせから成る単純型です。SOAP メッセージに union 要素が含まれている場合は、文字列のデータを含んだ単一の入力ポートをマップする必要があります。

例えば、SOAP メッセージに size という要素が含まれているとします。size は、整数と文字列の union です。

```
<xs:element name="size">
 <xs:simpleType>
 <xs:union memberTypes="size_no size_string" />
 </xs:simpleType>
</xs:element>
```

入力行には、説明とサイズが指定された品目が含まれています。品目には、42 などの数値のサイズが指定されている場合があります。あるいは、large、medium、small などの文字列値のサイズが指定されている場合もあります。

以下の表に、数値のサイズと文字列のサイズが含まれている入力行を示します。

Desc	Size
shoes	42
shirt	large

品目サイズに対して 1 つのポートを作成します。ポートを文字列としてマップします。SOAP メッセージには以下の要素が格納されます。

```
<item>
 <desc>shoes</desc>
 <size>42</size>
</item>

<item>
 <desc>shirt</desc>
 <size>large</size>
</item>
```

## 第 49 章

# 加重平均トランスフォーメーション

この章では、以下の項目について説明します。

- [加重平均トランスフォーメーションの概要, 696 ページ](#)
- [加重平均トランスフォーメーションの構成, 696 ページ](#)
- [加重マッチ率の例, 697 ページ](#)
- [加重平均トランスフォーメーションの詳細プロパティ, 697 ページ](#)
- [非ネイティブ環境での加重平均トランスフォーメーション, 698 ページ](#)

## 加重平均トランスフォーメーションの概要

加重平均トランスフォーメーションはパッシブトランスフォーメーションであり、複数の一致操作からの一致スコアを読み取って、単一の一致スコアを生成します。

加重平均トランスフォーメーションに投入する各スコアに対し、数値の加重を適用することができます。加重はゼロから 1 の間の値です。各入力スコアに適用される加重を編集することで、出力スコアへの影響度を増やしたり減らしたりすることができます。重複分析で各データカラムの相対的な重要度を反映した加重を適用します。

比較トランスフォーメーションをマッピングまたはマプレットに追加する際は、加重平均トランスフォーメーションを使用します。

**注:** 加重を一致トランスフォーメーションで割り当てることもできます。一致ストラテジの設定や 1 つのトランスフォーメーションでの加重の割り当てには、一致トランスフォーメーションを使用します。一致トランスフォーメーションに一致マプレットを組み込むことができます。

## 加重平均トランスフォーメーションの構成

加重平均トランスフォーメーションを使用し、マッピングが一連の一致分析操作で生成する、全般的なマッチ率を調整できます。各入力ポートの相対加重を編集し、ソースデータセットに定義したデータ比較の優先順位

を反映します。加重平均トランスフォーメーションの各入力ポートは、比較トランスフォーメーションストラテジからのマッチ率出力を表しています。

以下の手順は、比較トランスフォーメーションを使用するマップレットまたはマッピングで再利用不可能な加重平均トランスフォーメーションを構成する処理を説明しています。

1. 一致分析マップレットまたはマッピングを開き、比較トランスフォーメーションの加重平均トランスフォーメーションダウンストリームを追加します。
2. 比較トランスフォーメーションからのスコア出力を加重平均トランスフォーメーション入力ポートに接続します。  
マップレットやマッピングにあるその他の比較トランスフォーメーションに対して、この手順を繰り返します。
3. 加重平均トランスフォーメーションで **【ポート】** タブを選択します。
4. 各入力の **【加重】** フィールドをダブルクリックし、0.001 から 1 の間で加重値を入力します。加重値は、トランスフォーメーションにおけるその他の入力と比較した場合の、入力スコアの相対的な重要性を反映したものである必要があります。
5. マップレットまたはマッピングを保存します。

## 加重マッチ率の例

一致分析マッピングを作成し、顧客データベースで重複する顧客名数を決定します。2つの比較トランスフォーメーションを追加し、データセットの **【郵便番号】** カラムおよび **【姓】** カラムのマッチ率を生成します。

多くのレコードには一致する郵便番号がありますが、一致する姓のあるレコードの数はそれよりもかなり少なくなります。これらのマッチ率の平均を算出するときは、より一意性の高い一致の重要性を強調する必要があります。

姓のマッチ率の重要性を強調するには、**【姓】** マッチ率に対してより高い加重を適用します。

例えば、姓スコア入力の **【加重】** 値を 0.8 に設定し、郵便番号スコア入力の **【加重】** 値を 0.4 に設定します。

## 加重平均トランスフォーメーションの詳細プロパティ

Data Integration Service で加重平均トランスフォーメーションのデータがどのように処理されるかを特定するためのプロパティを設定します。

ログに表示するトレースレベルを設定できます。

**【詳細】** タブでは、以下のプロパティを設定します。

### トレースレベル

このトランスフォーメーションのログに表示される情報の詳細度。Terse、Normal、Verbose Initialization、Verbose data から選択できます。デフォルトは **【Normal】** です。

# 非ネイティブ環境での加重平均トランスフォーメーション

非ネイティブ環境での加重平均トランスフォーメーション処理は、そのトランスフォーメーションを実行するエンジンに依存します。

以下の非ネイティブランタイムエンジンでのサポートを考慮します。

- Blaze エンジン。制限なくサポートされます。
- Spark エンジン。バッチマッピングで制限なしでサポートされます。ストリーミングマッピングではサポートされていません。
- Databricks Spark エンジン。サポートしません。

## 第 50 章

# ウィンドウトランスフォーメーション

ストリームデータをデータグループに集約し、データセットを処理する場合は、ウィンドウトランスフォーメーションを使用します。ウィンドウトランスフォーメーションは、パッシブトランスフォーメーションです。

バインドされていないソースから読み取る場合は、以降の処理のためにデータをバインドされたデータグループに集約することができます。バインドされた間隔をバインドされていないデータに導入するには、ウィンドウトランスフォーメーションを使用します。

ウィンドウトランスフォーメーションを設定するときは、ウィンドウのタイプと、時間ごとのデータ境界を定義します。データ境界を指定するには、ウィンドウサイズとウィンドウのスライド間隔を設定します。ウィンドウサイズは、データがデータグループとして集約される時間間隔を定義します。スライド間隔は、集約されたデータグループがさらに処理されるまでの時間間隔を定義します。ウォーターマーク遅延では、データグループに集約される遅延イベントのしきい値時間を定義します。

ウィンドウトランスフォーメーションは、Spark エンジンでのストリーミングマッピングに対してのみ実行できます。

ウィンドウトランスフォーメーションの詳細については、『*Data Engineering Streaming ユーザーガイド*』を参照してください。

## 第 51 章

# 書き込みトランスフォーメーション

この章では、以下の項目について説明します。

- [書き込みトランスフォーメーションの概要, 700 ページ](#)
- [書き込みトランスフォーメーションのプロパティ, 700 ページ](#)
- [書き込みトランスフォーメーションの作成, 706 ページ](#)

## 書き込みトランスフォーメーションの概要

書き込みトランスフォーメーションは、パッシブトランスフォーメーションです。マッピングでは、書き込みトランスフォーメーションを使用してターゲットにデータを書き込みます。書き込みトランスフォーメーションは再利用不可能です。

書き込みトランスフォーメーションは、物理データオブジェクト、論理データオブジェクト、またはパラメータから作成できます。PowerExchange アダプタソースからインポートした物理データオブジェクトから書き込みトランスフォーメーションを作成する場合は、データオブジェクトから書き込みトランスフォーメーションを作成する前に書き込み操作を指定するように求めるメッセージがマッピングエディタに表示される場合があります。

書き込みトランスフォーメーションに設定できるプロパティは、トランスフォーメーションを作成するために使用したデータオブジェクトのタイプによって異なります。

書き込みトランスフォーメーションは動的ターゲットを表すことができます。書き込みトランスフォーメーションのポート、メタデータ、およびその他のプロパティを動的に更新するように、書き込みトランスフォーメーションを設定できます。動的ターゲットの設定については、『*Informatica Developer マッピングガイド*』を参照してください。

## 書き込みトランスフォーメーションのプロパティ

書き込みトランスフォーメーションを作成したら、トランスフォーメーションのプロパティを設定できます。

トランスフォーメーションの【**プロパティ**】ビューのタブで書き込みトランスフォーメーションのプロパティを設定します。設定できるタブは、書き込みトランスフォーメーションで表しているターゲットのタイプによって異なります。



以下の表で、各プロパティタブについて説明し、各タブの対象のターゲットタイプを示します。

プロパティタブ	説明	ターゲットタイプ
全般	トランスフォーメーションのプロパティと動作を指定します。 リレーショナルデータオブジェクトおよびカスタマイズデータオブジェクトのソースでは、トランスフォーメーションの入力ポートをソースと同期します。	すべて
データオブジェクト	トランスフォーメーションのデータソースを指定します。 リレーショナルおよびカスタマイズデータオブジェクトソースでは、実行時にデータソースからデータオブジェクトのカラムを取得します。	フラットファイル リレーショナル カスタマイズデータオブジェクト
形式	フラットファイルデータソースの入力設定	フラットファイル
ポート	関連付けられたデータオブジェクト別、またはマッピングフロー別にポートの定義を設定します。	すべて
ランタイム	データ統合サービスが実行時にターゲットにデータを書き込むとき使用するプロパティ（拒否ファイルの送信先など）。 フラットファイルターゲットの場合、拒否ファイルの名前とディレクトリ。	フラットファイル リレーショナル
データオブジェクトパラメータ	データオブジェクトパラメータを表示します。マッピング用のパラメータ値を設定するか、マッピングパラメータにパラメータをバインドします。	フラットファイル カスタマイズデータオブジェクト 論理データオブジェクト
ランタイムリンク	新しいランタイムリンクの作成およびリンクプロパティの表示を行います。	すべて
詳細	トレースレベルと行順序を設定します。 リレーショナルターゲットまたは Hive ターゲットの場合は、ターゲットスキーマストラテジを設定します。	フラットファイル リレーショナル カスタマイズデータオブジェクト 論理データオブジェクト

## 全般プロパティ

書き込みトランスフォーメーションの名前および説明を設定できます。また、以下のプロパティを設定することができます。

### カラムのメタデータが変わったとき

リレーショナルターゲットおよびカスタマイズされたターゲットの場合に使用できます。次のいずれかのオプションを選択します。

- 入力ポートを同期します。Developer tool は、モデルリポジトリがデータオブジェクト用に保存しているメタデータの変更を使用して、書き込みトランスフォーメーションの入力ポートを更新します。
- 同期しない。Developer tool は、データオブジェクトのメタデータの変更内容を表示しません。

デフォルトは「入力ポートの同期」オプションです。

## 物理データオブジェクト

フラットファイルターゲットおよびカスタマイズされたターゲットの場合に使用できます。 トランスフォーメーションの作成で使用したオブジェクト。

データオブジェクト名を選択して、そのプロパティを設定できます。

## データオブジェクトのプロパティ

[データオブジェクト] タブでは、書き込みトランスフォーメーションターゲットを指定または変更して、リレーショナルデータオブジェクトおよびカスタマイズデータオブジェクトのターゲットを動的に作成できます。

以下のプロパティを設定することができます。

### 指定元

書き込みトランスフォーメーションのターゲットカラムとメタデータを指定するには、以下のいずれかのオプションを選択します。

- 値。書き込みトランスフォーメーションは、関連付けられたデータオブジェクトを使用して、ターゲットカラムとメタデータを指定します。
- パラメータ。書き込みトランスフォーメーションは、パラメータを使用して、ターゲットカラムとメタデータを指定します。

デフォルトのオプションは [値] です。

### データオブジェクト

既存のデータオブジェクトから書き込みトランスフォーメーションを作成した場合、このフィールドにはオブジェクトの名前が表示されます。書き込みトランスフォーメーションに関連付けるデータオブジェクトを変更するには、[参照] をクリックします。

### パラメータ

書き込みトランスフォーメーションに関連付けるパラメータを選択または作成します。

### 実行時に、データソースからデータオブジェクトのカラムを取得します

このオプションを有効にした場合、データ統合サービスは、メタデータとデータ定義の変更内容をターゲットテーブルから取得して、書き込みトランスフォーメーションに取り込みます。

データ統合サービスがメタデータとデータ定義の変更を取得する方法をプレビューするには、解決済みパラメータとのマッピングを表示します。

## ポートのプロパティ

[ポート] タブでは、以下のプロパティを設定することができます。

### 次によって定義されたカラム

次のいずれかのオプションを選択して、書き込みトランスフォーメーションのカラムを定義します。

- 関連付けられているデータオブジェクト。 [データオブジェクト] タブのデータオブジェクトのカラム名、メタデータ、およびその他のプロパティを使用します。
- マッピングフロー。 マッピングは、マッピングのアップストリームオブジェクトからカラム名、メタデータ、およびその他のプロパティを取得します。

デフォルトは [関連付けられているデータオブジェクト] オプションです。

### カラムリソースのプロパティ

フラットファイルターゲットおよびカスタマイズされたターゲットの場合に使用できます。 各カラムのリソースは、カラムが名前、メタデータ、およびその他のプロパティを取得する元となるデータオブジェクトです。 リソースのプロパティを変更するには、リソース名を選択します。

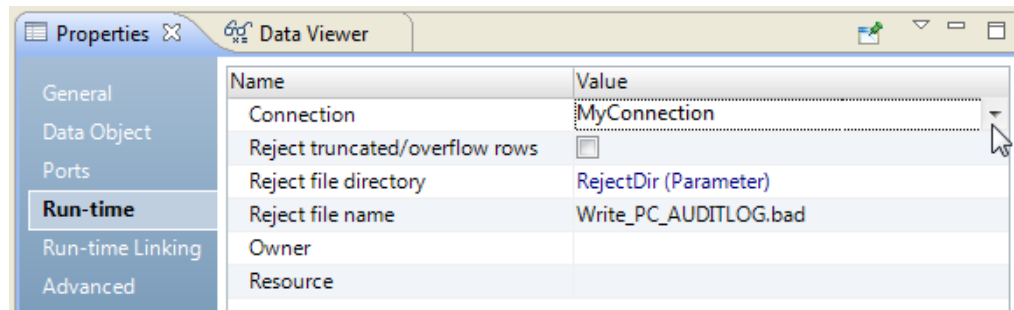
## ランタイムプロパティ

【ランタイム】タブで、次の書き込みトランザクションプロパティを設定できます。

### 接続

リレーショナルターゲットの場合に使用できます。トランスフォーメーションで使用する接続。接続を変更するには、フィールドの右側をクリックします。

次の図は、クリックするドロップダウンボタンの場所を示しています。



### 切り詰められた行またはオーバーフロー行を拒否

リレーショナルターゲットおよびカスタマイズされたターゲットの場合に使用できます。

Developer tool では、ポートからポートヘデータを渡して変換を行うことができます。変換が原因で、数値データのオーバーフローや、文字を含むカラムでの文字列の切り詰めが発生することがあります。例えば、Decimal (28, 2) ポートから Decimal (19, 2) ポートにデータを渡すと、数値データのオーバーフローが発生します。同様に、String (28) ポートから String (10) ポートにデータを渡すと、データ統合サービスは文字列を 10 文字に切り詰めます。

変換の結果オーバーフローが発生した場合、デフォルトではデータ統合サービスは、該当する行をスキップします。データ統合サービスは、拒否ファイルにデータを書き込みません。文字列の場合、データ統合サービスは文字列を切り詰めてから次のトランスフォーメーションに渡します。

このオプションを選択すると、最後のトランスフォーメーションとターゲットの間で切り詰められたデータやオーバーフローしたデータをすべてセッション拒否ファイルに含められます。データ統合サービスは、切り詰められた行やオーバーフローした行をすべて、セッション拒否ファイルまたは行エラーログに送信します。どちらに送信するかはセッションの設定によります。

### 拒否ファイルディレクトリ

拒否ファイルが存在するディレクトリ。デフォルトは RejectDir システムパラメータです。

### 拒否ファイル名

拒否ファイルのファイル名。デフォルトは<output\_file\_name>.bad です。

複数パーティションが同じフラットファイルターゲットに書き込みを行う場合、各パーティションが別々の拒否ファイルに書き込みます。拒否ファイルの名前は<output\_file\_name><partition\_number>.bad という形式になります。

## ランタイムリンクプロパティ

ランタイムリンクの作成および設定は、【ランタイムリンク】タブで行います。ランタイムリンクとは、トランスフォーメーション間のグループからグループへのリンクであり、パラメータ、リンクポリシー、またはそ

の両方を使用して実行時にリンクするポートを決定します。ランタイムリンクは、マッピングエディタで太線  
で表示されます。

次の場合に、書き込みトランスフォーメーションへのランタイムリンクを作成および設定します。

- 書き込みトランスフォーメーションのターゲットデータオブジェクトでパラメータを使用している。
- アップストリームトランスフォーメーションからのポートが実行時に変わる可能性がある。

**注:** マッピングフローに基づいてターゲットカラムを定義する場合は、書き込みトランスフォーメーションへの  
ランタイムリンクを作成しないでください。

**[ランタイムリンク]** タブでは、次のタスクを実行できます。

**ランタイムリンクを作成する。**

**[リンク]** 領域で **[新規]** ボタンをクリックして、実行時に **[新しいリンク]** ダイアログボックスでポート  
を書き込みトランスフォーメーションにリンクする、リンク元のトランスフォーメーションを選択します。

**ランタイムリンクプロパティを設定する。**

**[リンクのプロパティ]** 領域で、次のランタイムリンクプロパティを設定します。

#### パラメータ

ポート名がマッピングの実行ごとに変わる可能性があり、ポート名の値が分かっている場合に、この  
オプションを選択します。入力リンクセット型のパラメータを使用して、マッピング実行ごとに名前  
値でポートを接続します。入力リンクセットマッピングパラメータの構文は、カンマで区切られたポ  
ートのペアで構成されます（例: Afield1->Bfield2, Afield3->Bfield4 ）。

#### リンクポリシー

同じ名前のポートを自動的にリンクする場合に、このオプションを選択します。例えば、両方のマッ  
ピングオブジェクトに SALARY という名前のポートが含まれている場合、データ統合サービスはそれ  
らのポートをリンクします。ポート名に含まれるプレフィックスとサフィックスは無視できます。

データ統合サービスは、実行時に、次の順番でポート間のリンクを確立し、解決します。

- マッピングエディタで手動で作成するリンク。
- ランタイムリンク用に設定したパラメータに基づくリンク。
- ランタイムリンク用に設定したリンクポリシーに基づくリンク。

ランタイムリンクの詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

## 詳細プロパティ

データ統合サービスでの書き込みトランスフォーメーションのデータの処理方法を指定するには、詳細プロパ  
ティを設定します。

**[詳細]** タブで、以下のプロパティを設定します。

#### トレースレベル

マッピングログファイル内の詳細度を制御します。

#### ターゲットロードタイプ

ターゲットロードのタイプ。[ノーマル] または [一括] を選択します。リレーショナルリソースまたはカ  
スタマイズデータオブジェクトのターゲットロードタイプを設定できます。

[ノーマル] を選択した場合、データ統合サービスはターゲットを通常どおりにロードします。DB2、  
Sybase、Oracle、Microsoft SQL Server にロードする場合には、[一括] を選択できます。他のデータベ  
ースタイプに [一括] を指定すると、データ統合サービスがノーマルロードに戻します。バルクロードに  
よってマッピングのパフォーマンスは向上しますが、データベースロギングが発生しないので、リカバリ

機能が制限されます。バルクロードを使用して Oracle ターゲットに書き込むときに、Oracle データベースの制約を無効化することでパフォーマンスを最適化できます。

マッピングにアップデイトストラテジトランスフォーメーションが含まれる場合には、[ノーマル] モードを選択してください。[ノーマル] を選択したときに Microsoft SQL Server ターゲット名にスペースが含まれる場合は、接続オブジェクトで以下の環境 SQL を設定します。

```
SET QUOTED_IDENTIFIER ON
```

### 更新オーバーライド

ターゲットのデフォルトの UPDATE 文をオーバーライドします。

### 削除

削除のフラグが設定された行をすべて削除します。

デフォルトは [有効] です。

### 挿入

挿入のフラグが設定された行をすべて挿入します。

デフォルトは [有効] です。

### ターゲットスキーマストラテジ

リレーショナルまたは Hive ターゲットテーブルのターゲットスキーマストラテジのタイプ。

次のいずれかのターゲットスキーマストラテジを選択します。

- 維持-既存のターゲットスキーマを維持。データ統合サービスは、既存のターゲットスキーマを維持します。
- 作成-実行時にテーブルを作成または置換。データ統合サービスは、実行時にターゲットテーブルを削除して、指定したターゲットテーブルに基づくテーブルで置換します。
- パラメータの割り当て。ターゲットスキーマストラテジの値を表すパラメータを割り当てておき、実行時にそのパラメータを変更できます。

### 作成または置換の DDL クエリ

定義する DDL クエリに基づき、実行時にターゲットテーブルを作成または置換します。【作成-実行時にテーブルを作成または置換】ターゲットスキーマストラテジオプションを選択したときに適用されます。

### ターゲットテーブルの切り詰め

データをロードする前にターゲットを切り詰めます。

デフォルトは [有効] です。

### ターゲットパーティションの切り詰め

データのロード前に、内部または外部のパーティション化された Hive ターゲットを切り詰めます。このオプションを選択する前に、【ターゲットテーブルの切り詰め】を選択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。

### アップデイトストラテジ

既存の行のアップデイトストラテジ。次のいずれかのストラテジを選択します。

- 更新時に更新。データ統合サービスは更新のフラグが設定された行をすべて更新します。
- 挿入時に更新。データ統合サービスは更新のフラグが設定された行をすべて挿入します。ターゲットの【挿入】オプションも選択する必要があります。
- 更新しない場合は挿入。更新のフラグが設定された行がターゲット内に存在している場合、データ統合サービスは、これらの行を更新した後、挿入のマークが付いている残りの行を挿入します。ターゲットの【挿入】オプションも選択する必要があります。

## PreSQL

データ統合サービスが、ソースを読み込む前にターゲットデータベースに対して実行する SQL コマンド。

Developer tool では、SQL は検証されません。

## PostSQL

ターゲットに書き込んだ後にターゲットデータベースに対してデータ統合サービスが実行する SQL コマンド。

Developer tool では、SQL は検証されません。

## 行順序を保持

ターゲットへの入力データの行順序を保持します。データ統合サービスが行順序を変更する可能性がある最適化を実行しないようにする場合に、このオプションを選択します。

データ統合サービスが最適化を実行すると、以前のマッピングで確立された行順序が失われる場合があります。ソート済みフラットファイルソース、ソート済みリレーショナルソース、またはソータートランスフォーメーションを使用したマッピングにおける行順序を確立できます。行順序を保持するようにターゲットを設定した場合、データ統合サービスはターゲットの最適化を実行しません。

## 制約

テーブルレベルの参照の整合性制約の SQL 文。リレーショナルターゲットのみに適用されます。

# 書き込みトランスフォーメーションの作成

書き込みトランスフォーメーションを作成する場合、トランスフォーメーションの作成元のリソースに基づいて以下のいずれかの方法を選択します。

## モデルリポジトリ内のデータオブジェクトからトランスフォーメーションを作成する。

この方法は、書き込みトランスフォーメーションのメタデータをデータオブジェクトに基づいて作成する場合に使用します。

マッピングエディタで書き込みトランスフォーメーションを作成するには、**新しい書き込みトランスフォーメーションウィザード**を使用するか、**【オブジェクトエクスプローラ】** ビューからデータオブジェクトをドラッグして、トランスフォーメーションタイプとして **【書き込み】** を選択します。

## マッピングオブジェクトのフローからトランスフォーメーションを作成する。

この方法は、アップストリームマッピングトランスフォーメーションからのメタデータのフローに基づいて書き込みトランスフォーメーションのポートを作成する場合に使用します。

## パラメータを使用してトランスフォーメーションを作成する。

この方法は、パラメータが表すオブジェクトから書き込みトランスフォーメーションがポートを継承するようにする場合に使用します。

## マッピングの別のトランスフォーメーションを使用してトランスフォーメーションを作成する。

この方法は、書き込みトランスフォーメーションがソーストランスフォーメーションと同じポートを持つようにする場合に使用します。

## データオブジェクトからの書き込みトランスフォーメーションの作成

書き込みトランスフォーメーションはマッピングエディタで作成および設定できます。

この方法は、トランスフォーメーションの特定のプロパティを設定する場合に使用することをお勧めします。例えば、トランスフォーメーションを作成して、実行時にデータソースからカラムのメタデータを取得するように設定できます。

1. マッピングエディタ内で右クリックし、**【トランスフォーメーションの追加】** を選択します。  
**【トランスフォーメーションの追加】** ダイアログボックスが開きます。
2. **【書き込み】** を選択し、**【OK】** をクリックします。  
**【新しい書き込みトランスフォーメーション】** ウィザードが開きます。
3. **【物理データオブジェクト】** または **【論理データオブジェクト】** を選択してから、**【参照】** をクリックします。  
**【データオブジェクトの選択】** ダイアログボックスが開きます。
4. データソースを選択してから、**【OK】** をクリックします。
5. 関連付けられているデータオブジェクトによって書き込みトランスフォーメーションのカラムを定義するには、**【関連付けられているデータオブジェクト】** を選択します。  
書き込みトランスフォーメーションのポートは、関連付けられているデータオブジェクトのカラムと同じです。
6. 実行時にターゲットファイルからの変更を使用してターゲットオブジェクトのカラムを動的に更新するには、**【実行時に、データソースからデータオブジェクトのカラムを取得します】** を選択します。  
データ統合サービスは、マッピングが実行されたときに、書き込みトランスフォーメーションのカラムメタデータを更新します。
7. **【完了】** をクリックします。

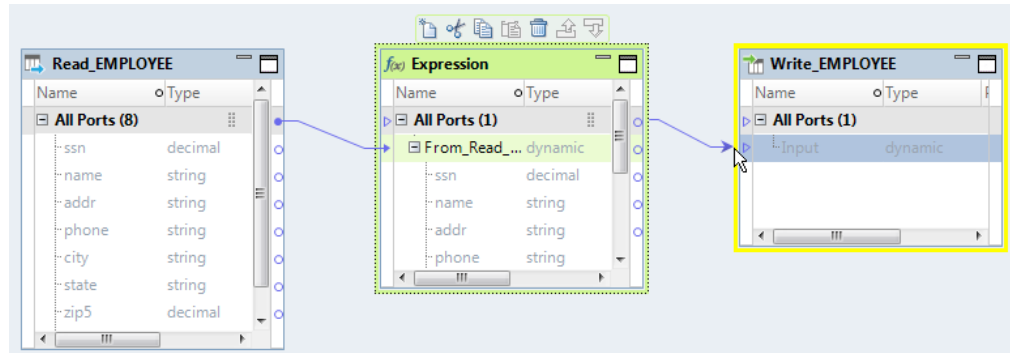
## マッピングフローからの書き込みトランスフォーメーションの作成

書き込みトランスフォーメーションはマッピングエディタで作成および設定できます。

この方法は、トランスフォーメーションの特定のプロパティを設定する場合に使用することをお勧めします。例えば、トランスフォーメーションを作成してから、マッピングフローに基づいてカラムを定義し、実行時にデータソースからカラムのメタデータを取得するように設定できます。

1. マッピングエディタ内で右クリックし、**【トランスフォーメーションの追加】** を選択します。  
**【トランスフォーメーションの追加】** ダイアログボックスが開きます。
2. **【書き込み】** を選択し、**【OK】** をクリックします。  
**【新しい書き込みトランスフォーメーション】** ウィザードが開きます。
3. **【物理データオブジェクト】** または **【論理データオブジェクト】** を選択してから、**【参照】** をクリックします。  
**【データオブジェクトの選択】** ダイアログボックスが開きます。
4. データソースを選択して、**【OK】** をクリックします。
5. **【マッピングフロー】** を選択し、**【完了】** をクリックします。
6. アップストリームの **【すべてのポート】** ポートから、書き込みトランスフォーメーションの **【入力】** ポートにポートをドラッグします。

次の図は、アップストリームポートを書き込みトランスフォーメーションの【入力】ポートに接続する方法を示しています。



書き込みトランスフォーメーションが、アップストリームマッピングオブジェクトからコラム定義を受け取ります。

7. 実行時にターゲットファイルからの変更を使用してターゲットオブジェクトのコラムを動的に更新するには、**【実行時に、データソースからデータオブジェクトのコラムを取得します】**を選択します。

データ統合サービスは、マッピングが実行されたときに、書き込みトランスフォーメーションのコラムメタデータを更新します。

## パラメータからの書き込みトランスフォーメーションの作成

パラメータとは、マッピングを実行するたびに変更できる定数値です。

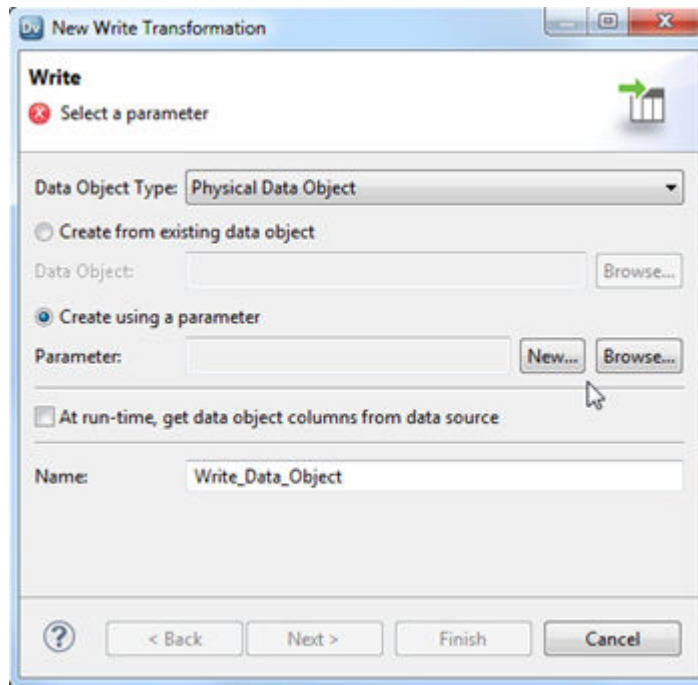
動的マッピングでは、パラメータを使用してソースおよびターゲットを変更できます。入力ルール、選択ルール、ランタイムリンク、およびトランスフォーメーションプロパティにパラメータを使用することもできます。パラメータの値を変更すると、データ統合サービスはパラメータに指定された値に従ってターゲットを作成または再作成します。

1. エディタで右クリックし、**【トランスフォーメーションの追加】**を選択します。
2. トランスフォーメーションのリストから**【書き込み】**を選択し、**【OK】**をクリックします。

**ヒント:** トランスフォーメーションの最初の文字を入力して、リストをフィルタ処理します。

**【新しい書き込みトランスフォーメーション】** ダイアログボックスが開きます。





3. **【パラメータを使用して作成】** を選択します。
4. パラメータを参照して選択するか、**【新規】** をクリックして新しいパラメータを作成してから、データオブジェクトを参照してパラメータの作成元となるオブジェクトを選択します。
5. 必要に応じて、**【実行時に、データソースからデータオブジェクトのカラムを取得します】** オプションを選択して、実行時にトランスフォーメーションカラムを更新します。
6. 必要に応じて、**【次へ】** をクリックして、トランスフォーメーションカラムを定義する方法を選択します。
7. **【完了】** をクリックします。

## 既存のトランスフォーメーションからの書き込みトランスフォーメーションの作成

マッピングに存在するトランスフォーメーションと同じポートを持つ書き込みトランスフォーメーションを作成できます。

複合ファイル、フラットファイル、またはリレーショナルリソースのターゲットを作成できます。既存のトランスフォーメーションにおけるいずれかのポートに複合ポートが含まれている場合は、リンクポリシーに基づいて、実行時に名前またはリンクポートによって複合ファイルターゲットおよびリンクポートを作成する必要があります。Developer tool では、Avro、Parquet、ORC、または JSON 複合ファイルターゲットを作成できます。

1. エディタでマッピングを開きます。
2. マッピングエディタでトランスフォーメーションを右クリックして、**【ターゲットの作成】** を選択します。  
**【ターゲットの作成】** ウィンドウが開きます。
3. 複合ファイル、フラットファイル、またはリレーショナルデータオブジェクトタイプを選択します。
4. リンクタイプを選択します。  
リンクタイプは、次のいずれかを選択できます。

### 名前でポートをリンク

書き込みトランスフォーメーションのポートはソースのポートに対応し、名前が同じです。

### マッピングフローに基づいて動的ポートをリンク

書き込みトランスフォーメーションには、マッピングフローのアップストリームオブジェクトに基づく動的ポートが含まれます。

### リンクポリシーに基づいて実行時にポートをリンク

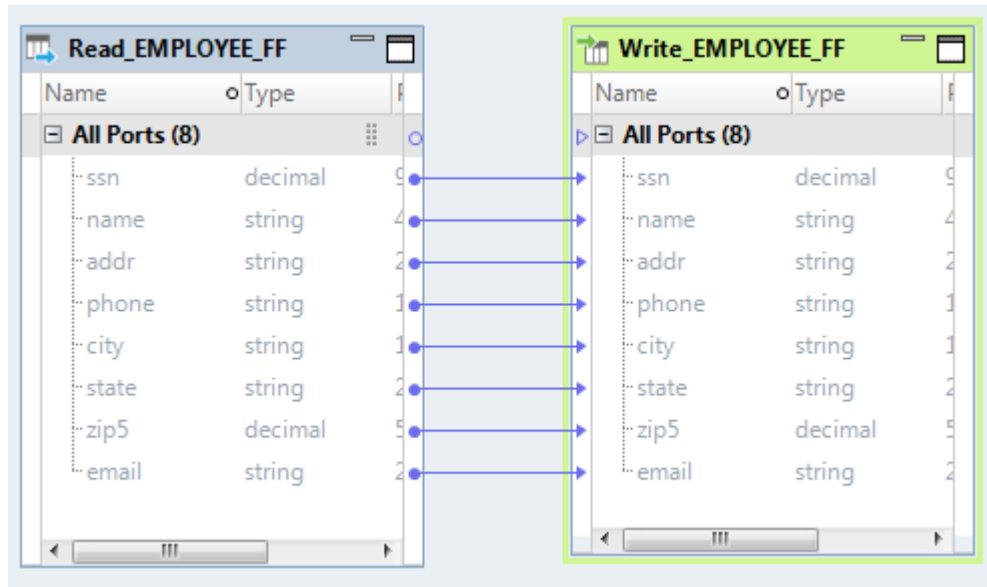
ポートは、書き込みトランスフォーメーションの [ランタイムリンク] タブに設定するリンクポリシーに基づいて、実行時にターゲットに作成されます。

動的ポートおよびランタイムリンクの設定の詳細については、『*Informatica Developer マッピングガイド*』を参照してください。

5. 新しいデータオブジェクトの名前を指定します。
6. 必要に応じて、[参照] をクリックして、データオブジェクトの場所を選択します。
7. 複合ファイルターゲットを作成することを選択した場合は、[リソース形式] ドロップダウンボックスで複合ファイル形式として Avro または Parquet を選択します。
8. [完了] をクリックします。

Developer tool は以下のタスクを実行します。

- マッピングに書き込みトランスフォーメーションを追加します。  
次の図は、読み取りトランスフォーメーションから作成された書き込みトランスフォーメーションを示しています。



- ポートをリンクします。
- 物理データオブジェクトを作成します。  
物理データオブジェクトのプロパティを構成できます。例えば、複合ファイルデータオブジェクトの HDFS 接続を指定する必要があります。

## 付録 A

# トランスフォーメーションの区切り文字

- [トランスフォーメーションの区切り文字の概要, 711 ページ](#)

## トランスフォーメーションの区切り文字の概要

トランスフォーメーションの区切り文字は、データ文字列間の区切りを示します。

次の表に、データ文字列の解析や書き込みを行う際にトランスフォーメーションで使用する区切り文字を示します。

区切り文字の名称	区切り文字記号
アットマーク	@
カンマ	,
ダッシュ	-
二重引用符	"
スラッシュ	/
ピリオド	.
シャープ	#
パイプ	
セミコロ	;
単一引用符	'
スペース	[Space キー]
タブ	[Tab キー]
下線	_

# 索引

## 数字

16 進数ソースビュー、データプロセッサ  
説明 [227](#)

## A

ABORT 関数  
使用 [52](#)  
all グループ  
REST Web サービスコンシューマトランスフォーメーションの表示  
[574](#)  
anyAttribute 属性  
Web サービスコンシューマトランスフォーメーション [657](#), [660](#)  
anyType  
ポートのマップ [693](#)  
anyType 要素  
Web サービスコンシューマトランスフォーメーション [657](#), [660](#)  
解析 [679](#)  
any 要素  
Web サービスコンシューマトランスフォーメーション [657](#), [660](#)  
API メソッド  
Java トランスフォーメーション [328](#)

## C

choice 要素  
REST Web サービスコンシューマトランスフォーメーションの表示  
[574](#)  
SOAP メッセージの解析 [682](#)  
Web サービスコンシューマトランスフォーメーションの表示 [658](#),  
[661](#)  
説明 [694](#)

## D

Data Integration Service)  
再起動モード [335](#)  
Data Transformation サービス  
モデルリポジトリへのインポート [248](#)  
複数のサービスのインポート [248](#)  
defineJExpression  
Java 式の API メソッド [345](#)  
defineJExpression メソッド  
Java トランスフォーメーション [329](#)

## E

EDataType クラス  
Java 式 [344](#)  
ERROR 関数  
使用 [52](#)

Excel  
Developer Tool へのコピー [68](#)  
トランスフォーメーションの編集 [69](#)  
トランスフォーメーションポートの設定 [68](#)  
ポートのコピー [67](#)  
ルールおよびガイドラインのコピー [69](#)  
Expression トランスフォーメーション  
Blaze エンジン [286](#)  
Databricks Spark エンジン [287](#)  
Spark エンジン [286](#)

## F

failSession メソッド  
Java トランスフォーメーション [330](#)

## G

GCID  
説明 [502](#)  
ノーマライズトランスフォーメーションの例 [513](#)  
generateRow メソッド  
Java トランスフォーメーション [330](#)  
getBytes メソッド  
Java トランスフォーメーション [347](#)  
getDouble メソッド  
Java トランスフォーメーション [347](#)  
getInRowType メソッド  
Java トランスフォーメーション [331](#)  
getInt メソッド  
Java トランスフォーメーション [348](#)  
getLong メソッド  
Java トランスフォーメーション [348](#)  
getMetadata メソッド  
Java トランスフォーメーション [332](#)  
getResultDataType メソッド  
Java トランスフォーメーション [348](#)  
getResultMetadata メソッド  
Java トランスフォーメーション [348](#)  
getStringBuffer メソッド  
Java トランスフォーメーション [349](#)  
gzip  
SOAP メッセージの圧縮 [666](#)

## H

Hive マージの使用  
オプション [646](#)  
HTTP エラー出力  
Web サービスコンシューマトランスフォーメーションの有効化 [663](#)  
HTTP 応答コード  
REST Web サービスコンシューマトランスフォーメーション [570](#)

HTTP 接続  
  REST Web サービス [575](#)  
HTTP ヘッダ  
  REST Web サービスコンシューマトランスフォーメーションへの追加 [569](#)  
HTTP ヘッダー  
  Web サービスコンシューマトランスフォーメーションへの追加 [656](#)

ID インデックスデータの永続ストレージ [453](#)

ID 照合分析  
  ID 分析定義済み [446](#)  
  インデックスデータの永続ストア [453](#)  
  インデックスデータ分析の出力プロパティ [489](#), [490](#)  
  永続インデックスデータに関するルールとガイドライン [453](#)  
  持続ステータス記述 [462](#)  
  持続ステータスコード [462](#)  
  プロセスフロー [481](#)  
  マスターデータセット [453](#)

IIF 関数  
  シーケンスジェネレータートランスフォーメーションを使用した欠落  
  キーの置き換え [589](#)  
incrementErrorCount メソッド  
  Java トランスフォーメーション [332](#)  
INOUT パラメータ  
  ストアドプロシージャの呼び出し [629](#)  
invokeJExpression  
  API メソッド [341](#)  
invokeJExpression メソッド  
  Java トランスフォーメーション [333](#)  
isNull メソッド  
  Java トランスフォーメーション [334](#)  
isResultNull メソッド  
  Java トランスフォーメーション [349](#)

## J

Java コード  
  Java トランスフォーメーション [309](#)  
  エラーの検出 [321](#)  
Java コードのコンパイル  
  [コード全体] タブ [316](#)  
Java コードスニペット  
  Java トランスフォーメーション用に作成 [311](#)  
Java コードの生成  
  Java 式 [340](#)  
Java 式  
  EDatatype クラス [344](#)  
  invokeJExpression API メソッド [341](#)  
  Java コードの生成 [340](#)  
  Java トランスフォーメーション [338](#)  
  JExpression クラス [346](#), [347](#)  
  JExprParaMetadata クラス [344](#)  
  [関数の定義] ダイアログボックスで作成 [340](#)  
  関数の設定 [340](#)  
  高度なインタフェース [343](#)  
  高度なインタフェースを使用した呼び出し [343](#)  
  高度なインタフェースの例 [346](#)  
  作成 [340](#)  
  生成 [339](#)  
  設定 [339](#)  
  単純なインタフェース [341](#)  
  単純なインタフェースの例 [342](#)  
  単純なインタフェースを使用した呼び出し [341](#)  
  呼び出し [333](#)

Java 式 (続く)  
  呼び出しに関するルールおよびガイドライン [333](#)  
  ルールおよびガイドライン [341](#), [343](#)  
Java 式の API メソッド  
  defineJExpression [345](#)  
  getBytes [347](#)  
  getDouble [347](#)  
  getInt [348](#)  
  getLong [348](#)  
  getResultDataType [348](#)  
  getResultMetadata [348](#)  
  getStringBuffer [349](#)  
  isResultNull [349](#)  
  呼び出し [349](#)  
Java トランスフォーメーション  
  API メソッド [328](#)  
  Blaze エンジン [325](#)  
  defineJExpression メソッド [329](#)  
  failSession メソッド [330](#)  
  generateRow メソッド [330](#)  
  getInRowType メソッド [331](#)  
  getMetadata メソッド [332](#)  
  incrementErrorCount メソッド [332](#)  
  invokeJExpression メソッド [333](#)  
  isNull メソッド [334](#)  
  Java コード [309](#)  
  Java コードスニペットの作成 [311](#)  
  Java プリミティブデータ型 [302](#)  
  logError メソッド [334](#)  
  logInfo メソッド [335](#)  
  NULL 値の設定 [336](#)  
  NULL 値のチェック [334](#)  
  resetNotification メソッド [335](#)  
  setNull メソッド [336](#)  
  Spark エンジン [325](#)  
  storeMetadata メソッド [337](#)  
  アクティブ [302](#)  
  [インポート] タブ [312](#), [314](#)  
  [関数] タブ [315](#)  
  [コード全体] タブ [316](#)  
  コンパイル [320](#)  
  コンパイルエラー [320](#)  
  コンパイルエラーの原因の特定 [321](#)  
  [最後] タブ [315](#)  
  作成 [319](#)  
  ステートレス [306](#)  
  設計 [305](#)  
  データ型の変換 [302](#)  
  デフォルトのポート値 [306](#)  
  トラブルシューティング [320](#)  
  トランスフォーメーション範囲 [306](#)  
  ナノ秒処理 [306](#)  
  入力行タイプの取得 [331](#)  
  [入力時] タブ [314](#)  
  パッシブ [302](#)  
  非ユーザーコードのエラー [321](#)  
  [ヘルパ] タブ [312](#), [314](#)  
  変数のリセット [335](#)  
  ポートの作成 [305](#)  
  マッピングの失敗 [330](#)  
  マッピングレベルのクラスパス [309](#)  
  メタデータの格納 [337](#)  
  メタデータの取得 [332](#)  
  ユーザーコードのエラー [321](#)  
  ログ [334](#), [335](#)  
  概要 [301](#)  
  高精度処理 [306](#)  
  出力ポート [306](#)

Java トランスフォーメーション (続く)

詳細プロパティ [306](#)

入力ポート [306](#)

非ネイティブ環境 [325](#)

Java パッケージ

インポート [312](#)

Java プリミティブデータ型

Java トランスフォーメーション [302](#)

JDK

Java トランスフォーメーション [301](#)

JExpression クラス

Java 式 [346](#), [347](#)

JExprParaMetadata クラス

Java 式 [344](#)

JRE

Java トランスフォーメーション [301](#)

## L

logError メソッド

Java トランスフォーメーション [334](#)

logInfo メソッド

Java トランスフォーメーション [335](#)

## N

NEXTVAL ポート

シーケンスジェネレータ [588](#)

NULL 値

Java トランスフォーメーションの設定 [336](#)

Java トランスフォーメーションでのチェック [334](#)

スキップ [53](#)

定数を使用した置き換え [51](#)

NULL のマッチ率

一致トランスフォーメーション [449](#)

NumRowsAffected

出力される行 [621](#)

## O

ORDER BY

上書き [394](#)

ルックアップクエリ [394](#)

## Q

Qname 要素

SOAP メッセージの解析 [681](#)

## R

Rank トランスフォーメーション

Blaze エンジン [543](#)

Databricks Spark エンジン [544](#)

resetNotification メソッド

Java トランスフォーメーション [335](#)

REST Web サービスコンシューマトランスフォーメーション

Delete メソッド [567](#)

GET メソッド [565](#)

HTTP ヘッダポート [569](#)

HTTP メソッド [565](#)

Post メソッド [566](#)

Put メソッド [566](#)

REST Web サービスコンシューマトランスフォーメーション (続く)

RequestInput ポート [568](#)

URL ポート [569](#)

XML スキーマ検証 [575](#)

インターネット媒体のタイプ [575](#)

応答コードポート [570](#)

概要 [561](#)

クッキーポート [570](#)

再利用可能 [576](#)

再利用不可 [576](#)

作成 [576](#)

出力 XML ポート [570](#)

出力ポート [568](#)

出力ポートの割付 [574](#)

出力マッピング [573](#)

出力マッピングのルール [573](#)

出力マッピングビューのカスタマイズ [574](#)

詳細プロパティ [575](#)

セキュリティ [653](#)

接続プロパティ [575](#)

設定 [563](#)

ソート済み入力 [575](#)

トランスポートレイヤセキュリティ [653](#)

トレースレベル [575](#)

入力ポート [568](#)

入力マッピング [570](#)

入力マッピングのルール [571](#)

パススルーポート [568](#)

引数ポート [569](#)

プロキシサーバーのサポート [561](#)

プロセス [563](#)

ベース URL の設定 [575](#)

ポート [568](#)

ポートへの要素のマッピング [568](#)

マッピング出力 [561](#)

マッピング入力 [561](#)

マッピング入力ポート [571](#)

メッセージの設定 [563](#)

リソースの識別 [564](#)

## S

Sequence Generator トランスフォーメーション

Blaze エンジン [597](#)

Spark エンジン [597](#)

sequence グループ

REST Web サービスコンシューマトランスフォーメーションの表示

[574](#)

setNull メソッド

Java トランスフォーメーション [336](#)

SIN 番号

再現可能なデータマスキング [217](#)

社会保険番号のマスキング [217](#)

SOAP アクション

Web サービスコンシューマトランスフォーメーションのオーバーラ

イド [663](#)

SOAP アクションのオーバーライド

Web サービスコンシューマトランスフォーメーション [663](#)

SOAP 圧縮

Web サービスコンシューマトランスフォーメーション [666](#)

SOAP 階層

入力ポートへのリレーション [685](#)

SOAP メッセージ

anyType 要素の解析 [679](#)

choice 要素の解析 [682](#)

choice 要素のまっぴんぐ [694](#)

union 要素へのポートのマッピング [695](#)

## SOAP メッセージ (続く)

- 概要 [652](#)
- keys [686](#)
- 代替グループの解析 [681](#)
- ピボット化するデータ [689](#)
- 複数出現ノードのマッピング [677](#)
- 複数の入力ポートをマッピング [689](#)
- ポートのマッピング [687](#)
- リストの要素の解析 [682](#)
- リストの要素のマッピング [694](#)

## SOAP メッセージ解析

- Qname 要素 [681](#)
- union 要素 [682](#)
- 正規化した出力 [677](#)
- 説明 [675](#)
- 派生型 [680](#)
- 非正規化した出力 [678](#)
- ピボット化した出力 [679](#)

## Sorter トランスフォーメーション

- Databricks Spark エンジン [608](#)

## SQLException ポート

- SQL トランスフォーメーション [614](#)

## SQL エラー時に続行

- SQL トランスフォーメーション [619](#), [622](#)

## SQL クエリ

- SQL トランスフォーメーション [618](#)

## SQL トランスフォーメーション

- INOUT パラメータ [629](#)
- SQLException ポート [614](#)
- SQL エラー時に続行 [622](#)
- 影響を受けた行数 [615](#)
- 概要 [610](#)
- 空として作成 [634](#)
- クエリの定義 [618](#)
- クエリ文 [627](#)
- クエリ文字列の置換 [619](#)
- 結果セット [630](#)
- 出力行数 [621](#)
- 出力行の制限 [621](#)
- 出力ポートの定義 [612](#)
- [詳細プロパティ] ビュー [615](#)
- 初期選択の最適化 [623](#)
- ストアドプロシージャ [628](#)
- ストアドプロシージャから作成 [634](#)
- ストアドプロシージャの呼び出し [629](#)
- ストアドプロシージャパラメータ [629](#)
- データベース接続の定義 [627](#)
- 入力行と出力行のカーディナリティ [619](#)
- 入力ポートの説明 [611](#)
- パススルーポート [613](#)
- パラメータのバインド [618](#)
- プッシュイン最適化 [623](#)
- ポート [611](#)
- 戻り値のポート [629](#)
- ランタイム接続 [633](#)
- 例 [624](#)
- 最適化にプッシュインのプロパティ [624](#)

## SQL 入力ポート

- SQL トランスフォーメーション [611](#)

## storeMetadata メソッド

- Java トランスフォーメーション [337](#)

# U

## Union トランスフォーメーション

- Databricks Spark エンジン [643](#)

## union 要素

- SOAP メッセージの解析 [682](#)
- 説明 [695](#)

# W

## Web サービスコンシューマトランスフォーメーション

- HTTP エラー出力の有効化 [663](#)
- HTTP ヘッダーの追加 [656](#)
- SOAP 圧縮 [666](#)
- SOAP メッセージ [652](#)
- エラー処理 [665](#)
- エンドポイント URL [656](#)
- 概要 [651](#)
- キーの表示 [658](#), [661](#)
- クッキー認証 [656](#)
- 作成 [669](#)
- 出力マッピング [660](#)
- 詳細プロパティ [663](#)
- セキュリティ [653](#)
- 操作 [653](#)
- 同時 Web サービス要求のメッセージ [663](#)
- 動的 Web サービス URL [656](#)
- 動的な WS-Security 名 [656](#)
- トランスポートレイヤセキュリティ [653](#)
- 入力マッピング [657](#)
- 汎用フォールト出力の有効化 [663](#)
- フィルタの最適化 [668](#)
- プッシュイン最適化 [668](#)
- 汎用 SOAP フォールト [665](#)
- マッピング出力ノード [660](#)
- マッピング入力ポート [657](#)
- 最適化にプッシュインの有効化 [669](#)
- 初期選択の最適化 [668](#)

## Web サービス

- anyType へのポートのマッピング [693](#)
- 代替グループ [693](#)
- 派生型 [692](#)

## Web サービス接続

- 概要 [663](#)

## Web サービストランスフォーメーション

- [場所] カラム [685](#)

## WS-Security ユーザー名

- 動的ポート [656](#)

## WSDL ファイル

- サービス要素 [652](#)
- 操作の要素 [652](#)
- バインディング要素 [652](#)
- ポート要素 [652](#)

# X

## XMap オブジェクト

- データプロセッサトランスフォーメーションでの作成 [244](#)

# あ

## アクティブトランスフォーメーション

- Java [301](#)
- 説明 [33](#)

## アクティブなトランスフォーメーション

- Java [302](#)
- Rank [537](#)

## アグリゲータトランスフォーメーション

- Blaze エンジン [143](#)

## アグリゲータトランスフォーメーション (続く)

- Databricks Spark エンジン [144](#)
- Spark エンジン [144](#)
- アップデートストラテジの組み合わせ [647](#)
- 開発 [132](#)
- 概要 [131](#)
- キャッシュ [138](#)
- キャッシュサイズ [70](#)
- グループ化ポート [135](#)
- 再利用可能なオブジェクトの作成 [141](#)
- 再利用不可能なオブジェクトの作成 [141](#)
- 集計式 [133](#)
- 詳細プロパティ [140](#)
- ソート済み入力 [138](#)
- データのソート [139](#)
- 動的マッピング [132](#)
- トラブルシューティング [142](#)
- ネストされた集計関数 [134](#)
- 非集計式 [137](#)
- ヒント [142](#)
- ポート [132](#)
- 集計関数 [134](#)
- 非ネイティブ環境 [143](#)
- 変数の使用 [47](#)

## アップデートストラテジトランスフォーメーション

- Blaze エンジン [648](#)
- Spark エンジン [649](#)
- アグリゲータの組み合わせ [647](#)
- 作成 [645](#)
- 式 [646](#)
- 動的マッピング [645](#)
- 概要 [644](#)
- 拒否された行の転送 [646](#)
- 詳細プロパティ [646](#)
- 設定手順 [645](#)
- 非ネイティブ環境 [648](#)

## アドレスバリデータトランスフォーメーション

- 非ネイティブ環境 [129](#)

## い

- 依存カラム
  - データマスキング [207](#)
- 依存関係
  - 暗黙 [65](#)
  - リンクパス [64](#)
- 依存マスキング
  - 説明 [207](#)
- 一意の出力
  - データマスキングトランスフォーメーション [206](#)
- 一意のレコードテーブル
  - 作成 [269](#)
- 一致ストラテジ
  - 重複レコードの例外トランスフォーメーション [265](#)
- 一致トランスフォーメーション
  - Blaze エンジン [467](#)
  - ID インデックステーブル [453](#)
  - ID 照合処理フロー [481](#)
  - ID 分析定義済み [446](#)
  - NULL のマッチ率 [449](#)
  - Spark エンジン [467](#)
  - クラスタスコアのオプション [449](#)
  - クラスタ分析のデータの表示 [455](#)
  - グループ化したデータ [447](#)
  - サンプル一致ストラテジ [265](#)
  - 持続ステータス記述 [462](#)
  - 持続ステータスコード [462](#)

## 一致トランスフォーメーション (続く)

- 出力形式 [487](#)
- [照合出力] ビューのオプション [472](#)
- [照合出力] ビューの出力プロパティ [490](#)
- [照合出力] ビューの照合プロパティ [489](#)
- [照合出力] ビューのプロパティ [473](#)
- 照合分析操作の設定 [466](#)
- 照合分析の概念 [445](#)
- 照合分析用マップレットの作成 [465](#)
- 使用例 [444](#)
- 単一ソースでの分析 [446](#)
- 定義済み出力ポート [461](#)
- 定義済み入力ポート [460](#)
- デュアルソースでの分析 [446](#)
- ドライバスコア [451](#)
- パフォーマンスデータの表示 [455](#)
- パフォーマンス要因 [454](#)
- フィールド分析処理フロー [468](#)
- フィールド分析定義済み [446](#)
- リンクスコア [451](#)
- 非ネイティブ環境 [467](#)

## イベント

- データプロセッサトランスフォーメーション [239](#)

## イベントタイプ

- データプロセッサトランスフォーメーション [239](#)

## イベントビュー

- データプロセッサトランスフォーメーション [240](#)

## イベントビュー、データプロセッサ

- イベントログの表示 [241](#)

## イベントログ

- 表示 [241](#)

## イベントログ、設計時

- 説明および場所 [241](#)

## インクリメント

- シーケンス間隔の設定 [591](#)

## インスタンス変数

- Java トランスフォーメーション [312](#)

## インデックスキャッシュ

- トランスフォーメーション [71](#)

## う

- ウィンドウ化
  - プロパティ [278](#)
- ウィンドウ化プロパティ
  - 順序 [278](#), [279](#)
  - パーティション [278](#), [279](#)
  - フレーム [278](#)
  - ルールおよびガイドライン [281](#)

## え

- 影響を受けた行の数
  - SQL トランスフォーメーション [615](#)
- 永続ルックアップキャッシュ
  - 概要 [426](#)
- エディタと同期
  - データプロセッサトランスフォーメーション [246](#)
- エラー
  - Java トランスフォーメーションにおけるしきい値の増加 [332](#)
  - 処理 [53](#)
- エラーイベント
  - データプロセッサトランスフォーメーション [239](#)
- エラー数
  - Java トランスフォーメーションでの増分 [332](#)



エンコード設定  
データプロセッサトランスフォーメーション [232](#)  
エンコードのガイドライン  
データプロセッサトランスフォーメーション [234](#)  
エンドポイント URL  
REST Web サービスコンシューマトランスフォーメーション [569](#)  
Web サービスコンシューマトランスフォーメーション [656](#)

## お

応答コード  
REST Web サービスコンシューマトランスフォーメーション [570](#)  
大文字小文字変換プログラムトランスフォーメーション  
概要 [165](#)  
非ネイティブ環境 [167](#)  
オプションのエラーイベント  
データプロセッサトランスフォーメーション [239](#)

## か

開始桁  
社会保険番号 [217](#)  
開始値  
シーケンスジェネレータトランスフォーメーション [590](#)  
外部キー  
シーケンスジェネレータトランスフォーメーションを使用した作成  
[588](#)  
下限しきい値  
設定 [155](#), [259](#)  
[関数] タブ  
Java トランスフォーメーション [315](#)  
関連付けトランスフォーメーション  
概要 [146](#)  
詳細プロパティ [148](#)  
トレースレベル [148](#)

## き

keys  
SOAP メッセージ階層 [686](#)  
キー  
シーケンスジェネレータトランスフォーメーションを使用した作成  
[588](#)  
キージェネレータトランスフォーメーション  
詳細プロパティ [377](#)  
トレースレベル [377](#)  
概要 [373](#)  
非ネイティブ環境 [377](#)  
キーマスキング  
数値 [203](#)  
数値のマスキング [203](#)  
説明 [203](#)  
日時の値のマスキング [204](#)  
文字列値のマスキング [203](#)  
却下されたレコード  
不良レコードの例外トランスフォーメーション [155](#)  
キャッシュ  
インデックス [71](#)  
概要 [70](#)  
キャッシュファイル [70](#)  
サイズ [72](#)  
サイズの最適化 [75](#)  
静的ルックアップキャッシュ [425](#)  
タイプ [71](#)  
データ [71](#)

キャッシュ (続く)  
データ統合サービスによるサイズの増加 [74](#)  
動的ルックアップキャッシュ [430](#)  
トランスフォーメーション [70](#)  
パーティション化 [75](#)  
ファイルのディレクトリ [72](#)  
ルックアップトランスフォーメーション [401](#), [423](#)  
キャッシュサイズ  
自動 [72](#)  
デタマスキングトランスフォーメーション [220](#)  
キャッシュされる値の数  
シーケンスジェネレータのプロパティ値 [590](#)  
キャッシュディレクトリ  
デタマスキングトランスフォーメーション [220](#)  
キャッシュファイル  
概要 [71](#)  
ディレクトリ [72](#)  
キャッシュファイルサイズ  
関連付けトランスフォーメーション [148](#)  
キージェネレータトランスフォーメーション [377](#)  
重複レコードの例外トランスフォーメーション [263](#)  
統合トランスフォーメーション [184](#)  
キャッシュファイルディレクトリ  
関連付けトランスフォーメーション [148](#)  
キージェネレータトランスフォーメーション [377](#)  
重複レコードの例外トランスフォーメーション [263](#)  
統合トランスフォーメーション [184](#)  
行  
更新のフラグ設定 [645](#)  
行順序の維持  
シーケンスジェネレータトランスフォーメーション [591](#)  
共有体トランスフォーメーション  
概要 [640](#)  
非ネイティブ環境 [643](#)  
共有ルックアップキャッシュ  
ガイドライン [427](#)  
パーティション化、ガイドライン [427](#)

## く

クエリ  
ルックアップトランスフォーメーション [394](#)  
ルックアップの上書き [395](#)  
クッキー認証  
REST Web サービスコンシューマトランスフォーメーション [570](#)  
Web サービスコンシューマトランスフォーメーション [656](#)  
クラスタ  
重複レコードの例外トランスフォーメーション [258](#)  
クラスタデータ  
出力グループ [262](#)  
クラスパス  
マッピングプロパティ [309](#)  
グループ  
ユーザー定義 [581](#)  
ルータトランスフォーメーション [581](#)  
ルータトランスフォーメーションへの追加 [584](#)  
[グループ化] タブ  
説明 [136](#)  
ポートリストパラメータ [137](#), [541](#)  
グループフィルタの条件  
ルータトランスフォーメーション [581](#)

## け

警告イベント  
データプロセッサトランスフォーメーション [239](#)

## 計算

変数の使用 [48](#)

## 結果セット

SQL トランスフォーメーション [630](#)

出力パラメータの配置 [632](#)

ストアドプロシージャのサンプル [630](#)

## 結果文字列の置換文字

データマスキングトランスフォーメーション [211](#)

## 結合条件

概要 [357](#)

## 欠落値

シーケンスジェネレータを使用した置き換え [589](#)

## 検索

Java コードのエラー [321](#)

## 検証

デフォルト値 [53](#), [56](#)

## 検証ルールオブジェクト

データプロセッサトランスフォーメーションでの作成 [245](#)

# こ

更新でなければ挿入（プロパティ）

説明 [438](#)

## 高度なインタフェース

EDataType クラス [344](#)

Java 式 [343](#)

Java 式の呼び出し [343](#)

JExpression クラス [346](#), [347](#)

JExprParaMetadata クラス [344](#)

例 [346](#)

## 考慮事項

Java トランスフォーメーション [305](#)

## [コード全体] タブ

Java コンパイルエラー [321](#)

Java トランスフォーメーション [316](#)

## コードスニペット

Java トランスフォーメーション用に作成 [311](#)

## 個別の出力

ソータトランスフォーメーション [600](#)

## コンパイル

Java トランスフォーメーション [320](#)

## コンパイルエラー

Java トランスフォーメーションのソースの識別 [321](#)

## コンポーネントビュー

データプロセッサトランスフォーメーション [227](#)

# さ

## サービス

WSDL ファイル要素 [652](#)

## サービスパラメータポート

データプロセッサトランスフォーメーション [228](#), [229](#)

## サイクル

シーケンスジェネレータトランスフォーメーションプロパティ [590](#)

## サイクル（プロパティ）

シーケンスジェネレータトランスフォーメーションプロパティ [590](#)

## [最後] タブ

Java トランスフォーメーション [315](#)

## 最大出力行数

SQL トランスフォーメーション [619](#), [621](#)

## 再利用可能

シーケンスジェネレータトランスフォーメーション [592](#)

## 再利用可能なトランスフォーメーション

説明 [57](#)

編集 [57](#)

## 再利用不可能なトランスフォーメーション

説明 [58](#)

## 作成

Java トランスフォーメーション [319](#)

シーケンスジェネレータトランスフォーメーション [595](#)

シーケンスデータオブジェクト [592](#)

## サブレコード

ノーマライザトランスフォーメーション [503](#)

## サンプルソース

データプロセッサトランスフォーメーションでの作成 [246](#)

## サンプルソースファイル

データプロセッサトランスフォーメーションでの定義 [244](#)

## サンプル入力ファイル

データプロセッサトランスフォーメーション [228](#)

# し

## シーケンスデータオブジェクト

プロパティ [592](#)

作成 [592](#)

## 式

Java トランスフォーメーション [338](#)

簡略化 [47](#)

検査 [45](#)

コメントの追加 [45](#)

トランスフォーメーション内 [43](#)

入力 [44](#)

ポートへの追加 [45](#)

## 式エディタ

式の検証 [45](#)

式のテスト [273](#)

説明 [44](#)

## 式パラメータ

結合条件の [360](#)

ルックアップ条件 [410](#)

## 式マスキング

再現可能なマスキング [201](#)

再現可能なマスキングの例 [202](#)

説明 [201](#)

ルールおよびガイドライン [202](#)

## シーケンスジェネレータトランスフォーメーション

IIF 関数を使用した欠落キーの置き換え [589](#)

NEXTVAL ポート [588](#)

値の範囲 [591](#)

開始値 [590](#)

概要 [587](#)

行順序の維持 [591](#)

サイクル [590](#), [591](#)

[増分] プロパティ [591](#)

複合キーの作成 [588](#)

プロパティ [590](#), [592](#)

ポート [587](#)

リセット [591](#)

作成 [595](#)

非ネイティブ環境 [597](#)

## 自動キャッシュサイズ

説明 [72](#)

## 自動統合

重複レコードの例外トランスフォーメーション [259](#)

## 社会保障番号

再現可能なデータマスキング [216](#)

地域コードマスキング [216](#)

## 出力グループ

ノーマライザトランスフォーメーション [513](#)

不良レコードの例外トランスフォーメーション [150](#)

## [出力の編集] ダイアログボックス

ノーマライザトランスフォーメーション [514](#)

- 出力ポート
  - Java トランスフォーメーション [305](#), [306](#)
  - エラー処理 [50](#)
  - デフォルト値 [50](#)
- 出力ポート設定
  - 説明 [282](#)
- 出力マッピング
  - REST Web サービスコンシューマトランスフォーメーション [573](#)
  - Web サービスコンシューマトランスフォーメーション [660](#)
- 手動統合
  - 重複レコードの例外トランスフォーメーション [259](#)
- ジョイナトランスフォーメーション
  - Blaze エンジン [372](#)
  - Databricks Spark エンジン [372](#)
  - Spark エンジン [372](#)
  - 同じソースからのデータの結合 [367](#)
  - 概要 [350](#)
  - 完全外部結合 [362](#)
  - キャッシュ [352](#)
  - キャッシュサイズ [70](#)
  - 式パラメータ [360](#)
  - 条件 [357](#)
  - 条件タイプ [358](#)
  - 詳細条件タイプ [358](#)
  - 詳細プロパティ [351](#)
  - 選択ルール [355](#)
  - ソースパイプラインのブロック [369](#)
  - ソート順の設定 [363](#)
  - ソート済み [370](#)
  - ソート済み入力 [363](#)
  - 単純条件タイプ [358](#)
  - 動的ポート [360](#)
  - 動的マッピング [354](#)
  - ノーマルジョイン [361](#)
  - パフォーマンス [370](#)
  - ポート [353](#)
  - ポートセレクタ [354](#)
  - ポートセレクタの使用 [359](#)
  - マスタ外部結合 [362](#)
  - マスタ行のキャッシュ [370](#)
  - 未ソート [370](#)
  - 明細外部結合 [362](#)
  - ルールおよびガイドライン [371](#)
  - 結合タイプ [361](#)
  - 非ネイティブ環境 [371](#)
- 条件
  - ジョイナトランスフォーメーション [357](#)
  - ルータトランスフォーメーション [581](#)
- 上限しきい値
  - 設定 [155](#), [259](#)
- 条件タイプ
  - ジョイナトランスフォーメーションの詳細 [358](#)
  - ジョイナトランスフォーメーションの単純 [358](#)
- 照合分析でのグループ化 [447](#)
- 詳細条件タイプ
  - ジョイナトランスフォーメーション [358](#)
- 詳細プロパティ
  - Java トランスフォーメーション [306](#)
  - REST Web サービスコンシューマトランスフォーメーション [575](#)
  - SQL トランスフォーメーション [615](#)
  - Web サービスコンシューマトランスフォーメーション [663](#)
  - 関連付けトランスフォーメーション [148](#)
  - キージェネレータトランスフォーメーション [377](#)
  - 重複レコードの例外トランスフォーメーション [263](#)
  - 統合トランスフォーメーション [184](#)
  - 不良レコードの例外トランスフォーメーション [158](#)
- 初期化
  - 変数 [49](#)

- 初期選択の最適化
  - SQL トランスフォーメーション [623](#)
  - Web サービスコンシューマトランスフォーメーション [668](#)

## す

- 数値
  - キーマスキング [203](#)
  - ランダムマスキング [200](#)
- スキーマ
  - 階層型からリレーショナルへのトランスフォーメーション [298](#)
  - データプロセッサトランスフォーメーション [231](#)
  - リレーショナルから階層型へのトランスフォーメーション [559](#)
- スクリプト
  - データプロセッサトランスフォーメーションでの作成 [244](#)
- スクリプトヘルプビュー
  - データプロセッサトランスフォーメーション [227](#)
- スコープ
  - ポートセレクタ [275](#), [408](#)
  - マスタおよび明細 [355](#)
- スタートアップコンポーネント
  - データプロセッサトランスフォーメーション [230](#)
- ステートレス
  - Java トランスフォーメーション [306](#)
- ストアドプロシージャ
  - SQL トランスフォーメーション [628](#)
  - 結果セットの例 [630](#)
  - 入力ポートと出力ポート [629](#)
  - パラメータ [629](#)
  - 変数に書き込み [48](#)
  - 戻り値 [630](#)
  - 戻り値のポート [629](#)
  - 例 [632](#)
- ストリーマ
  - 説明 [226](#)
- ストレージのコミット間隔
  - データマスキングトランスフォーメーション [220](#)
- ストレージの暗号化
  - データマスキングトランスフォーメーション [220](#)
- ストレージの暗号化キー
  - データマスキングトランスフォーメーション [220](#)
- すべてのグループ
  - Web サービスコンシューマトランスフォーメーションの表示 [658](#), [661](#)

## せ

- 生成キー
  - Web サービス出力グループ [677](#)
- 静的コード
  - Java トランスフォーメーション [312](#)
- 静的変数
  - Java トランスフォーメーション [312](#)
- 静的ルックアップキャッシュ
  - 概要 [425](#)
- 制約
  - ソースの最適化 [552](#)
- 設計
  - Java トランスフォーメーション [305](#)
- 設計時イベントログ
  - 説明および場所 [241](#)
- 接続
  - REST Web サービス [575](#)
  - Web サービス [663](#)
- 接続されたルックアップ
  - 概要 [393](#)

接続されたルックアップ (続く)

説明 [391](#)

接続されていないトランスフォーメーション  
ルックアップトランスフォーメーション [393](#)

接続されていないルックアップ

概要 [393](#)

説明 [391](#)

〔設定〕 ビュー

重複レコードの例外トランスフォーメーション [259](#)

不良レコードの例外トランスフォーメーション [155](#)

不良レコードの例外トランスフォーメーションの例 [161](#)

設定ビュー

データプロセッサトランスフォーメーション [232](#)

選択条件

ポートセレクト [275, 408](#)

選択ルール

動的マッピング [275, 409](#)

ジョイナトランスフォーメーション [355](#)

ポートセレクト [275, 408](#)

## そ

操作

WSDL ファイル要素 [652](#)

〔操作出力〕 領域

Web サービスコンシューマトランスフォーメーションのカスタマイズ [661](#)

〔操作入力〕 領域

Web サービスコンシューマトランスフォーメーションのカスタマイズ [658](#)

操作領域

Web サービストランスフォーメーション [685](#)

挿入でなければ更新 (プロパティ)

説明 [437](#)

ソースの最適化

制約 [552](#)

ソータ

可変長 [604](#)

ソータトランスフォーメーション

キャッシュ [603](#)

〔ソート〕 タブ [600](#)

動的マッピング [599](#)

ソータートランスフォーメーション

Blaze エンジン [606](#)

Spark エンジン [607](#)

概要 [598](#)

キャッシュサイズ [70](#)

非ネイティブ環境 [606](#)

ソートキー

設定 [600](#)

〔ソート〕 タブ

ソータトランスフォーメーション [600](#)

ソートプロパティ

重複レコードの例外トランスフォーメーション [263](#)

統合トランスフォーメーション [184](#)

ソース行のフィルタリング

ルックアップトランスフォーメーション [397](#)

ソース文字列の文字

データマスキングトランスフォーメーション [211](#)

## た

ターゲット

リレーショナル [644](#)

代替グループ

SOAP メッセージの解析 [681](#)

代替グループ (続く)

Web サービス [693](#)

Web サービスコンシューマトランスフォーメーション [657, 660](#)

単一ソースでの照合分析 [446](#)

単純条件タイプ

ジョイナトランスフォーメーション [358](#)

単純なインタフェース

Java 式 [341](#)

Java トランスフォーメーション API メソッド [341](#)

例 [342](#)

## ち

置換マスキング

説明 [204](#)

マスキングプロパティ [205](#)

致命的エラーイベント

データプロセッサトランスフォーメーション [239](#)

重複レコードテーブル

生成 [260](#)

重複レコードの例外トランスフォーメーション

概要 [257](#)

クラスタ [258](#)

クラスタ出力の例 [268](#)

構成設定の例 [266](#)

サンプル出力 [267](#)

自動統合 [259](#)

出力グループ [262](#)

詳細プロパティ [263](#)

設定 [269](#)

〔設定〕 ビュー [259](#)

重複レコードテーブルの生成 [260](#)

トレースレベル [263](#)

標準出力グループ [262](#)

ポート [260](#)

マッピングの例 [264](#)

例 [264](#)

レコードのソート [263](#)

## つ

通知イベント

データプロセッサトランスフォーメーション [239](#)

## て

ディクショナリ

再現可能な式マスキング [201](#)

置換データマスキング [204](#)

ディクショナリ情報

データマスキングトランスフォーメーション [206](#)

ディビジョントランスフォーメーション

詳細プロパティ [256](#)

品質の問題のサンプルストラテジ [153](#)

概要 [250](#)

非ネイティブ環境 [256](#)

定数

NULL 値との置き換え [51](#)

データ

一時保存 [47](#)

データオブジェクト

重複したパラメータ化 [405](#)

パラメータ化 [550](#)

データ処理トランスフォーメーション

スタートアップコンポーネント [230](#)

データプロセッサ 16 進数ソースビュー  
説明 [227](#)  
データプロセッサイベントビュー  
イベントログの表示 [241](#)  
データプロセッサトランスフォーメーション [240](#)  
データプロセッサトランスフォーメーション  
XMap 設定 [237](#)  
XML 設定 [237](#)  
エンコード設定 [232](#)  
サービスとしてエクスポート [247](#)  
サービスパラメータポート [229](#)  
作成 [242](#)  
出力ポート [230](#)  
設定 [232](#)  
説明 [226](#), [227](#)  
データビューアでテスト [246](#)  
入力ポート [228](#)  
ビュー [227](#)  
ポート [228](#)  
ユーザーログ [242](#)  
出力制御設定 [234](#)  
処理設定 [236](#)  
非ネイティブ環境 [249](#)  
データマスキングトランスフォーメーション  
IP アドレスのマスキング [215](#)  
URL のマスキング [216](#)  
依存データマスキング [207](#)  
格納テーブル [201](#), [205](#)  
キャッシュサイズ [220](#)  
キャッシュディレクトリ [220](#)  
クレジットカードのマスキング [214](#)  
再現可能な SIN 番号 [217](#)  
再現可能な SSN [216](#)  
再現可能な式マスキング [201](#)  
式マスキング [201](#)  
式マスキングのガイドライン [202](#)  
社会保障番号のマスキング [216](#), [218](#)  
ストレージのコミット間隔 [220](#)  
説明 [198](#)  
ソース文字列の文字 [211](#)  
置換マスキング [204](#)  
置換マスキングのディクショナリ [204](#)  
置換マスキングプロパティ [205](#), [206](#)  
ディクショナリ名の式マスキング [201](#)  
デフォルト値ファイル [217](#)  
電話番号のマスキング [216](#)  
特殊マスク形式 [213](#)  
範囲 [212](#)  
日付値のマスキング [212](#)  
ブラー [212](#)  
マスキング方法 [199](#)  
マスク形式 [210](#)  
ランタイムプロパティ [220](#)  
ランダムマスキング [199](#)  
一意の出力 [220](#)  
共有ストレージテーブル [220](#)  
社会保険番号のマスキング [217](#)  
電子メールアドレスのマスキング [214](#)  
非ネイティブ環境 [223](#)  
データ型  
Java トランスフォーメーション [302](#)  
データキャッシュ  
トランスフォーメーション [71](#)  
デフォルトグループ  
ルータトランスフォーメーション [581](#)  
デフォルト値  
概要 [50](#)  
検証 [53](#), [56](#)

デフォルト値 (続く)  
出力ポート [50](#), [51](#)  
データマスキング [217](#)  
入力 [53](#), [56](#)  
入力ポート [50](#), [51](#)  
パススルーポート [50](#), [51](#)  
ユーザー定義 [51](#)  
ルール [55](#)  
デュアルソースでの照合分析 [446](#)

## と

統合トランスフォーメーション  
概要 [182](#)  
詳細プロパティ [184](#)  
トレースレベル [184](#)  
レコードのソート [184](#)  
非ネイティブ環境 [197](#)  
同時 Web サービス要求のメッセージ  
Web サービスコンシューマトランスフォーメーションの有効化 [663](#)  
動的 URL  
Web サービスコンシューマトランスフォーメーション [656](#)  
動的式  
概要 [281](#)  
作成 [284](#)  
出力ポート設定 [282](#)  
例 [281](#)  
動的ポート  
結合条件 [360](#)  
ルックアップ条件 [402](#)  
動的マッピング  
ポートセクタ [275](#), [409](#)  
アグリゲータトランスフォーメーション [132](#)  
アップデートストラテジトランスフォーメーション [645](#)  
ジョイナトランスフォーメーション [354](#)  
ソータトランスフォーメーション [599](#)  
フィルタトランスフォーメーション [289](#)  
ランクトランスフォーメーション [538](#)  
ルータトランスフォーメーション [580](#)  
ルックアップ条件 [407](#)  
ルックアップトランスフォーメーション [402](#)  
選択ルール [275](#), [409](#)  
動的ルックアップキャッシュ  
使用例 [431](#)  
説明 [430](#)  
フラットファイルソースの使用 [430](#)  
ルックアップ SQL オーバーライド [436](#)  
ドライバスコア  
一致トランスフォーメーション [451](#)  
トラブルシューティング  
Java トランスフォーメーション [320](#)  
トランスフォーマ  
説明 [226](#)  
トランスフォーメーション  
再利用不可 [58](#)  
トランスフォーメーション  
Excel での編集 [69](#)  
Excel でのポートの設定 [68](#)  
Hadoop 環境 [37](#)  
Java [301](#)  
アクティブ [33](#)  
エラー処理 [53](#)  
開発 [41](#)  
概要 [32](#)  
キャッシュ [70](#)  
キャッシュサイズ [72](#), [74](#)  
キャッシュサイズの最適化 [75](#)

## トランスフォーメーション (続く)

キャッシュのパーティション化 [75](#)

キャッシュファイル [71](#)

再利用可能 [57](#), [58](#)

再利用可能なトランスフォーメーションの編集 [57](#)

再利用不可 [58](#)

作成 [58](#)

式 [43](#)

式の検証 [45](#)

シーケンスジェネレータ [587](#)

接続された [33](#)

接続されていない [33](#)

複数グループ [42](#)

メタデータのコピー [68](#)

## トランスフォーメーションポート

概要 [60](#)

## トランスフォーメーション範囲

Java トランスフォーメーション [306](#)

## トランスフォーメーションのブロック

説明 [42](#)

## トランスポートレイヤセキュリティ

REST Web サービスコンシューマトランスフォーメーション [653](#)

Web サービスコンシューマトランスフォーメーション [653](#)

## トレースレベル

関連付けトランスフォーメーション [148](#)

キージェネレータトランスフォーメーション [377](#)

重複レコードの例外トランスフォーメーション [263](#)

統合トランスフォーメーション [184](#)

# な

## ナノ秒処理

Java トランスフォーメーション [306](#)

# に

## 日時の値

データマスキング [204](#)

## 入力階層

ノーマライザトランスフォーメーション [504](#)

## 入力行

行タイプの取得 [331](#)

## 入力ポート

Java トランスフォーメーション [305](#), [306](#)

デフォルト値 [50](#)

## [入力ポート] 領域

SOAP メッセージの生成 [684](#)

## 入力マッピング

REST Web サービスコンシューマトランスフォーメーション [570](#)

Web サービスコンシューマトランスフォーメーション [657](#)

## [入力時] タブ

Java トランスフォーメーション [314](#)

# の

## ノーマライザトランスフォーメーション

GCID [502](#)

概要 [501](#)

キーの生成 [515](#)

作成 [516](#)

サブレコード [503](#)

出力グループ [512](#), [513](#)

出力グループの編集 [514](#)

使用例 [517](#)

定義の例 [518](#)

## ノーマライザトランスフォーメーション (続く)

入力階層 [504](#)

入力グループおよび出力グループの例 [519](#)

入力ポート [505](#)

フィールドのマージ [506](#), [507](#)

複数出現フィールド [502](#)

複数出現レコード [503](#)

他のトランスフォーメーションからポートをドラッグする [517](#)

マッピング出力の例 [520](#)

マッピングの例 [518](#)

詳細プロパティ [515](#)

非ネイティブ環境 [521](#)

## ノーマライザビュー

説明 [504](#)

# は

## パーサー

説明 [226](#)

## パーサートランスフォーメーション

非ネイティブ環境 [534](#)

## パーサートランスフォーメーション

概要 [524](#)

## バインディング

WSDL ファイル要素 [652](#)

## [場所] カラム

Web サービストランスフォーメーション [685](#)

## バスルーポート

SQL トランスフォーメーション [613](#)

デフォルト値 [50](#)

式トランスフォーメーション [272](#)

## 派生型

SOAP メッセージの解析 [680](#)

Web サービス [692](#)

## 派生型要素

Web サービスコンシューマトランスフォーメーション [657](#), [660](#)

## パッシブトランスフォーメーション

Java [301](#), [302](#)

概要 [33](#)

シーケンスジェネレータ [587](#)

## バッファ出力ポート

データプロセッサトランスフォーメーション [230](#)

## バッファ入力ポート

データプロセッサトランスフォーメーション [228](#)

## パフォーマンス

変数を使用して改善する [47](#)

## パラメータのバインド

SQL トランスフォーメーション [618](#)

## パラメータ

フィルタ条件 [290](#)

ルータトランスフォーメーション [583](#)

## [パラメータ] オプション

ジョイナトランスフォーメーション [360](#)

## 範囲

数値のマスキング [212](#)

## 汎用フォールト出力

Web サービスコンシューマトランスフォーメーションの有効化 [663](#)

# ひ

## 非正規化した出力

SOAP メッセージ解析 [678](#)

## 非正規化した入力

Web サービスポート [691](#)

## 日付の値

ランダムデータマスキング [200](#)

ピボット化した出力  
  SOAP メッセージ解析 [679](#)

ピボット化したデータ  
  SOAP メッセージ [689](#)

非ユーザーコードのエラー  
  Java トランスフォーメーション [321](#)

ヒューマンタスク  
  不良レコードの例外 [153](#)

標準出力グループ  
  重複レコードの例外トランスフォーメーション [262](#)

標準出力ポート  
  説明 [155](#)

品質の問題  
  ポートの割り当て先 [158](#)

品質の問題ポート  
  空白の対 NULL [150](#)  
  説明 [154](#)

**ふ**

ファイル出力ポート  
  データプロセッサトランスフォーメーション [230](#)

ファイル入力ポート  
  データプロセッサトランスフォーメーション [228](#)

フィールド一致分析  
  フィールド分析定義済み [446](#)  
  プロセスフロー [468](#)

フィルタトランスフォーメーション  
  Blaze エンジン [293](#)  
  NULL 値を含む行 [292](#)  
  概要 [288](#)  
  詳細プロパティ [292](#)  
  動的マッピング [289](#)  
  パフォーマンスのヒント [292](#)  
  フィルタ条件 [290](#)  
  フィルタ条件パラメータ [290](#)  
  非ネイティブ環境 [292](#)

フィルタポート  
  Web サービスコンシューマトランスフォーメーション [668](#)

フォールトをエラーとして扱う  
  Web サービスコンシューマトランスフォーメーションの有効化 [663](#)

複合キー  
  REST Web サービスコンシューマトランスフォーメーション [571](#)  
  Web サービスコンシューマトランスフォーメーション [657](#)  
  シーケンスジェネレータトランスフォーメーションを使用した作成 [588](#)

副次作用  
  SQL トランスフォーメーション [623](#)  
  Web サービスコンシューマトランスフォーメーション [668](#)

複数グループ  
  トランスフォーメーション [42](#)

複数出現フィールド  
  ノーマライズトランスフォーメーション [502](#)

ブッシュイン最適化  
  SQL トランスフォーメーション [623](#)  
  SQL トランスフォーメーションでの有効化 [624](#)  
  Web サービスコンシューマトランスフォーメーション [668](#)

ブラー  
  数値 [213](#)  
  日付の値 [213](#)

プライマリキー  
  シーケンスジェネレータトランスフォーメーションを使用した作成 [588](#)

不良レコードの例外トランスフォーメーション  
  概要 [149](#)  
  空白の品質の問題ポート [150](#)  
  出力グループ [150](#)

不良レコードの例外トランスフォーメーション (続く)

出力ポート [155](#)  
出力例 [162](#)  
詳細プロパティ [158](#)  
正常レコードの出力例 [164](#)  
設定 [158](#)  
[設定] ビュー [155](#)  
入力ポート [154](#)  
品質の問題 [153](#)  
不良レコードの出力例 [162](#)  
プロセスフロー [151](#)  
ポートグループ [154](#)  
マッピング [151](#)  
マプレットの例 [159](#)  
問題の出力例 [163](#)  
例 [159](#)

不良レコードテーブルの生成  
  不良レコードの例外トランスフォーメーション [157](#)

プロパティ  
  読み取りトランスフォーメーション  
    プロパティ [546](#)  
  書き込みトランスフォーメーション  
    プロパティ [700](#)

**へ**

変数ポート  
  概要 [47](#)  
  式トランスフォーメーション [272](#)

**ほ**

ポート  
  Excel からのコピー [67](#)  
  Java トランスフォーメーション [305](#)  
  SOAP メッセージへのマップ [687](#)  
  位置によるリンク [63](#)  
  作成 [60](#)  
  シーケンスジェネレータトランスフォーメーション [587](#)  
  自動的にリンク [62](#)  
  手動でリンク [62](#)  
  設定 [61](#)  
  重複レコードの例外トランスフォーメーション [260](#)  
  デフォルト値の概要 [50](#)  
  トランスフォーメーションでプロパゲートされる属性 [65](#)  
  名前によるリンク [62](#)  
  非正規化した Web サービストランスフォーメーション [691](#)  
  評価順 [48](#)  
  不良レコードの例外トランスフォーメーション [154](#)  
  変数ポート [47](#)  
  リンク [61](#)  
  リンクのルールおよびガイドライン [63](#)  
  ルータトランスフォーメーション [584](#)

ポートセレクタ  
  選択ルール [275](#), [408](#), [409](#)  
  作成 [276](#), [356](#), [411](#)  
  ジョイナトランスフォーメーションで [359](#)  
  ジョイナトランスフォーメーション [354](#), [355](#)  
  選択ルール [275](#), [408](#), [409](#)  
  動的式で [281](#)

ポート属性  
  プロパゲート [64](#)

ポートの割り当て  
  品質の問題 [158](#)

[ポート] ビュー  
  SQL トランスフォーメーション出力 [612](#)



ポートリストパラメータ  
  [グループ化] タブ [137](#), [541](#)  
ポート値  
  Java トランスフォーメーション [306](#)  
汎用 SOAP フォールト  
  Web サービスコンシューマトランスフォーメーション [665](#)

## ま

マージトランスフォーメーション  
  概要 [522](#)  
  非ネイティブ環境 [523](#)  
マスキング方法  
  データマスキング [199](#)  
マスキングルール  
  結果文字列の置換文字 [211](#)  
  ソース文字列の文字 [211](#)  
  特殊マスク形式 [213](#)  
  範囲 [212](#)  
  ブラー [212](#)  
  マスク形式 [210](#)  
マスク形式  
  特殊マスク形式 [213](#)  
  文字列値のマスキング [210](#)  
マップパー  
  説明 [226](#)  
マッピング  
  行に対する更新のフラグ設定 [645](#)  
  ルータトランスフォーメーションの使用 [585](#)  
マッピングの失敗  
  Java トランスフォーメーション [330](#)  
マッピングパラメータ  
  ルックアップ SQL オーバーライドで [395](#)  
マッピング変数  
  ルックアップ SQL オーバーライドで [395](#)  
マップレット  
  リファレンス [231](#)

## め

メソッド  
  Java トランスフォーメーション API [328](#)

## も

文字列  
  リンク付け [538](#)  
文字列値  
  カスタムデータマスキング [199](#)  
  キーデータマスキング [203](#)  
文字列の置換  
  SQL トランスフォーメーション [619](#)  
問題テーブル  
  生成 [157](#)  
[問題の割り当て] ビュー  
  不良レコードの例外トランスフォーメーション [157](#)

## ゆ

ユーザーコードのエラー  
  Java トランスフォーメーション [321](#)  
ユーザー定義グループ  
  ルータトランスフォーメーション [581](#)

ユーザーログ  
  データプロセッサトランスフォーメーション [242](#)  
ユーザー定義メソッド  
  Java トランスフォーメーション [312](#)

## よ

要素  
  共用体 [695](#)  
呼び出し  
  Java 式の API メソッド [349](#)  
読み取りデータオブジェクト  
  パラメータ化 [551](#)  
予約語  
  ルックアップクエリ [396](#)

## ら

ライブラリ  
  データプロセッサトランスフォーメーションでの作成 [245](#)  
ラベラトランスフォーメーション  
  概要 [378](#)  
  非ネイティブ環境 [389](#)  
リンク付け  
  データグループ [540](#)  
  文字列値 [538](#)  
リンクトランスフォーメーション  
  RANKINDEX ポート [539](#)  
  Spark エンジン [543](#)  
  オプション [538](#)  
  概要 [537](#)  
  キャッシュ [541](#)  
  キャッシュサイズ [70](#)  
  グループの定義 [540](#)  
  詳細プロパティ [542](#)  
  動的マッピング [538](#)  
  ポート [538](#)  
  リンクポート [540](#)  
  非ネイティブ環境 [543](#)  
  変数の使用 [47](#)  
リンクポート  
  リンクトランスフォーメーション [540](#)  
ランタイムイベントログ  
  データプロセッサトランスフォーメーション [241](#)  
[ランタイム] ビュー  
  SQL トランスフォーメーション [627](#)  
ランダムマスキング  
  数値 [199](#)  
  日付値のマスキング [200](#)  
  文字列値のマスキング [199](#)

## り

リストの要素  
  SOAP メッセージの解析 [682](#)  
  説明 [694](#)  
リセット  
  シーケンスジェネレータトランスフォーメーション [591](#)  
リファレンスビュー  
  データプロセッサトランスフォーメーション [231](#)  
[リファレンス]ビュー  
  階層型からリレーショナルへのトランスフォーメーション [298](#)  
  リレーショナルから階層型へのトランスフォーメーション [559](#)  
リボジトリビュー  
  データプロセッサトランスフォーメーション [227](#)



リレーショナルから階層型へのトランスフォーメーション  
開発 [559](#)  
作成 [559](#)  
ポート [558](#)  
ポートの作成 [559](#)  
例 [556](#)  
説明 [555](#)  
リレーショナルデータオブジェクト  
動的読み取りトランスフォーメーションの作成 [553](#)  
リンク  
1 対 1 [62](#)  
1 対多 [62](#)  
リンクスコア  
一致トランスフォーメーション [451](#)

## る

ルータートランスフォーメーション  
非ネイティブ環境 [586](#)  
ルータートランスフォーメーション  
概要 [579](#)  
グループ [581](#)  
グループフィルタの条件 [581](#)  
詳細プロパティ [585](#)  
動的ポート [583](#)  
動的マッピング [580](#)  
パラメータ [583](#)  
ポート [584](#)  
マッピングでの接続 [585](#)  
例 [581](#)  
ルール  
デフォルト値 [55](#)  
ルールおよびガイドライン  
ウィンドウ化プロパティ [281](#)  
ルックアップ  
キャッシュを使用しない [425](#)  
ルックアップ SQL オーバーライド  
動的キャッシュ [436](#)  
ルックアップキャッシュ  
永続 [426](#)  
概要 [401](#), [423](#)  
静的 [425](#)  
定義 [423](#)  
動的 [430](#)  
ルックアップクエリ  
ORDER BY [394](#)  
上書き [395](#), [397](#)  
上書きのガイドライン [396](#)  
説明 [394](#)  
デフォルトクエリ [394](#)  
予約語 [396](#)  
ルックアップ条件  
生成されたポートの使用 [407](#)  
説明 [399](#)  
データマスキングトランスフォーメーション [206](#)  
パラメータで指定 [410](#)  
ルックアップソース  
パラメータ化 [405](#)  
ルックアップソースフィルタ  
ルックアップの制限 [397](#)  
ルックアップトランスフォーメーション  
Blaze エンジン [420](#)  
Databricks Spark エンジン [422](#)  
Spark エンジン [420](#)  
永続キャッシュ [426](#)  
開発 [394](#)  
概要 [391](#)  
キャッシュ [401](#), [423](#)  
キャッシュサイズ [70](#)  
キャッシュサイズ, 削減 [395](#)  
キャッシュを使用しないルックアップ [425](#)  
クエリのプロパティ [401](#)  
再利用可能なオブジェクトの作成 [415](#)  
再利用不可能なオブジェクトの作成 [416](#)  
詳細プロパティ [414](#)  
接続 [391](#), [393](#)  
接続されていないルックアップの作成 [417](#)  
デフォルトクエリ [394](#)  
動的キャッシュ [430](#)  
動的ポート [402](#)  
動的マッピング [402](#)  
ポート名の競合 [404](#)  
マッピングパラメータとマッピング変数 [395](#)  
未接続 [391](#), [393](#)  
[ランタイム] プロパティ [413](#)  
リレーショナルパラメータ化データオブジェクト [405](#)  
ルックアップクエリのオーバーライド [395](#)  
ルックアップ条件 [398](#)  
ルックアップ条件のルールおよびガイドライン [400](#)  
ルックアップの上書き [397](#)  
非ネイティブ環境 [420](#)  
ルックアップの上書き  
ガイドライン [396](#)

## れ

例  
動的式 [281](#)  
パーティションキーおよびオーダーキー [279](#)  
例外トランスフォーメーション  
[問題の割り当て] ビュー [157](#)  
レコード  
ノーマライザトランスフォーメーション [503](#)

## ろ

ローカル変数  
概要 [47](#)  
ログ  
Java トランスフォーメーション [334](#), [335](#)  
定義 [240](#)  
ログ、設計時イベント  
説明および場所 [241](#)