Informatica®

Informatica® Cloud Data Integration
October 2023

トランスフォーメーション

Informatica Cloud Data Integration トランスフォーメーション
October 2023

発行日: 2024-01-11

目次

序文

マッピングやマップレットに含めることができるトランスフォーメーションに関する詳細については、「*トランスフォーメーション*」を参照してください。データをソースからターゲットに移動するときにどのように変換するかを学びます。

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 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 Intelligent Cloud Services Web サイト

Informatica Intelligent Cloud Services Web サイト（http://www.informatica.com/cloud）にアクセスできます。このサイトには、Informatica Cloud 統合サービスに関する情報が含まれます。

Informatica Intelligent Cloud Services コミュニティ

Informatica Intelligent Cloud Services コミュニティを使用して、技術的な問題について議論し、解決します。また、技術的なヒント、マニュアルの更新情報、FAQ（よくある質問）への答えを得ることもできます。

次の Informatica Intelligent Cloud Services コミュニティにアクセスします。

https://network.informatica.com/community/informatica-network/products/cloud-integration

開発者は、次の Cloud 開発者コミュニティで詳細情報を確認したり、ヒントを共有したりできます。

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services マーケットプレース

Informatica マーケットプレイスにアクセスすると、データ統合コネクタ、テンプレート、およびマップレットを試用したり購入したりできます。

https://marketplace.informatica.com/

データ統合のコネクタのドキュメント

データ統合のコネクタのドキュメントには、マニュアルポータルからアクセスできます。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。

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 Intelligent Cloud Services Trust Center

Informatica Intelligent Cloud Services Trust Center は、Informatica のセキュリティポリシーおよびリアルタイムでのシステムの可用性について情報を提供します。

Trust Center（https://www.informatica.com/trust-center.html）にアクセスします。

Informatica Intelligent Cloud Services Trust Center にサブスクライブして、アップグレード、メンテナンス、およびインシデントの通知を受信します。Informatica Intelligent Cloud Services Status ページには、すべての Informatica Cloud 製品の実稼働ステータスが表示されます。メンテナンスの更新はすべてこのページに送信され、停止中は最新の情報が表示されます。更新と停止の通知がされるようにするには、Informatica Intelligent Cloud Services の 1 つのコンポーネントまたはすべてのコンポーネントについて更新の受信をサブスクライブします。すべてのコンポーネントにサブスクライブするのが、更新を逃さないようにするための最良の方法です。

サブスクライブするには、Informatica Intelligent Cloud Services Status ページで **[サブスクライブして更新]** をクリックします。電子メール、SMS テキストメッセージ、Webhook、RSS フィード、またはこれらの 4 つの任意の組み合わせとして送信された通知を受信するという選択ができます。

Informatica グローバルカスタマサポート

グローバルサポートセンターには、Informatica Network または電話でお問い合わせください。

Informatica Network でオンラインサポートリソースを検索するには、Informatica Intelligent Cloud Services のヘルプメニューで **[サポートにお問い合わせください]** をクリックして、**Cloud Support** ページに移動します。**Cloud Support** ページには、システムステータス情報とコミュニティディスカッションが記載されています。追加のリソースを検索する場合や電子メールで Informatica グローバルカスタマサポートに問い合わせる場合は、Informatica Network にログインし、**[サポートが必要な場合]** をクリックしてください。

Informatica グローバルカスタマサポートの電話番号は、Informatica の Web サイト https://www.informatica.com/services-and-training/support-services/contact-us.html に掲載されています。

第 1 章

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、マッピングの一部で、データに対して実行する処理を表します。また、トランスフォーメーションでは、各トランスフォーメーションへのデータの入力方法も定義します。

各トランスフォーメーションは特定の関数を実行します。例えば、ソース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り、式トランスフォーメーションは行レベルの計算を実行します。

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

トランスフォーメーションは、アクティブまたはパッシブにすることができます。

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できます。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。

パッシブな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の数は変更されません。

ブランチ内のすべてのトランスフォーメーションがパッシブである場合、複数のブランチをパッシブなダウンストリームトランスフォーメーションに接続できます。

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ダウン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。行を連結できない可能性があります。アクティブなトランスフォーメーションは行数を変更するため、別のトランスフォーメーションからの行数と一致しない場合があります。

例えば、マッピング内の一方のブランチにはパッシブな式トランスフォーメーションがあり、もう一方のブランチにはアクティブなアグリゲータトランスフォーメーションがあるとします。アグリゲータトランスフォーメーションは、合計などの集計をグループに対して実行し、行数を削減します。式トランスフォーメーションからの行数とアグリゲータトランスフォーメーションからの行数が異なるため、ブランチを接続してもデータ統合はこれらのトランスフォーメーションからの行を結合できません。2 つのブランチを結合するには、ジョイナトランスフォーメーションを使用します。

トランスフォーメーションタイプ

トランスフォーメーションをマッピングに追加したら、トランスフォーメーションの詳細を定義できます。各トランスフォーメーションタイプには、設定可能な固有のオプションのセットがあります。

以下の表に、各トランスフォーメーションの簡単な説明を示します。

トランスフォーメーション	説明
ソース	ソースからデータを読み取ります。
ターゲット	ターゲットにデータを書き込みます。
アグリゲータ	アクティブなトランスフォーメーションで、データのグループに対して集計計算を実行します。
クレンジング	Data Quality で作成したクレンジングアセットをマッピングまたはマップレットに追加するパッシブなトランスフォーメーション。クレンジングアセットを使用して、データの形式やコンテンツを標準化します。
データマスキング	パッシブなトランスフォーメーションで、非プロダクション環境向けの現実的なテストデータとして機密データをマスクします。
データサービス	HL7 や HIPAA などの業界標準サービスやカスタマイズされたサービスなど、データサービスリポジトリからデータサービスを呼び出すアクティブなトランスフォーメーション。
重複排除	Data Quality で作成した重複排除アセットをマッピングまたはマップレットに追加するアクティブなトランスフォーメーション。重複排除アセットを使用して、データセットの重複する ID のインスタンスを検索し、必要に応じて重複を 1 つのレコードに統合します。
式	パッシブなトランスフォーメーションで、個々のデータ行に対して計算を実行します。
フィルタ	アクティブなトランスフォーメーションで、データフローからのデータをフィルタリングします。
階層ビルダ	アクティブなトランスフォーメーションで、リレーショナル入力を階層出力に変換します。
階層パーサー	階層入力をリレーショナル出力に変換するパッシブトランスフォーメーション。
階層プロセッサ	階層入力をリレーショナル出力に、またはリレーショナル入力を階層出力に、または階層出力を別のスキーマの階層出力に、あるいは階層入力を非正規化されたフラット化済みの出力に変換するアクティブなトランスフォーメーション。
入力	データをマップレットに渡すパッシブトランスフォーメーション。マップレットでは使用できますが、マッピングでは使用できません。
Java	Java で書かれたユーザーロジックコードを実行します。 アクティブまたはパッシブです。
ジョイナ	アクティブなトランスフォーメーションで、2 つのソースからのデータを結合します。
ラベラ	Data Quality で作成したラベラアセットをマッピングまたはマップレットに追加するパッシブなトランスフォーメーション。ラベラーアセットを使用して、入力フィールドの情報のタイプを識別し、各タイプのラベルをデータに割り当てます。

トランスフォーメーション	説明
ルックアップ	ルックアップオブジェクトからデータを検索します。ルックアップオブジェクトおよびルックアップ接続を定義します。また、ルックアップ条件を定義して、値を返します。 パッシブなルックアップトランスフォーメーションは 1 行を返します。アクティブなルックアップトランスフォーメーションは複数行を返します。
機械学習	機械学習モデルを実行し、予測をマッピングに返します。
マップレット	マップレットを、マッピングまたは別のマップレットに挿入します。マップレットにはトランスフォーメーションロジックが含まれます。このロジックを作成および使用すると、データを変換してからターゲットにロードできます。 マップレットのトランスフォーメーションロジックに基づいてアクティブまたはパッシブになります。
ノーマライザ	複数出現するフィールドのあるデータを処理し、複数出現するデータの各インスタンスの行を返すアクティブトランスフォーメーション。
出力	データをマップレットからダウンストリームトランスフォーメーションに渡すパッシブトランスフォーメーション。マップレットでは使用できますが、マッピングでは使用できません。
解析	Data Quality で作成した解析アセットをマッピングまたはマップレットに追加するパッシブなトランスフォーメーション。解析アセットを使用して、入力フィールドの単語または文字列を、その単語または文字列に含まれる情報のタイプに基づいて 1 つ以上の個別の出力フィールドに解析します。
Python	トランスフォーメーション機能を定義する Python コードを実行します。アクティブまたはパッシブです。
ランク	アクティブなトランスフォーメーションで、レコードを上限/下限に制限します。
ルータ	入力データへの条件の適用に使用できる、アクティブなトランスフォーメーション。
ルール仕様	Data Quality で作成したルール仕様アセットをマッピングまたはマップレットに追加するパッシブなトランスフォーメーション。ルール仕様アセットを使用して、データセットにビジネスルールのデータ要件を適用します。
シーケンスジェネレータ	値のシーケンスを生成するパッシブトランスフォーメーション。
ソーター	指定されたソート条件に従って昇順または降順にデータをソートするパッシブトランスフォーメーション。
SQL	ストアドプロシージャまたはストアド関数を呼び出すか、データベースに対してクエリを実行します。 ストアドプロシージャまたはストアド関数を呼び出す場合はパッシブです。クエリを処理する場合は、アクティブまたはパッシブにできます。
構造パーサー	フラットファイルソースの構造化されていないデータを分析して、構造化形式で書き込むパッシブトランスフォーメーション。
トランザクション制御	アクティブなトランスフォーメーションで、マッピングの実行時に行のセットをコミットまたはロールバックします。

トランスフォーメーション	説明
共有体	アクティブなトランスフォーメーションで、複数の入力グループのデータを 1 つの出力グループにマージします。
Velocity	Velocity スクリプトを実行して、データをフラット化せずに、JSON または XML 階層入力をある形式から別の形式に変換するパッシブトランスフォーメーション。
ベリファイヤ	Data Quality で作成したベリファイヤアセットをマッピングまたはマップレットに追加するパッシブなトランスフォーメーション。ベリファイヤアセットを使用して、郵便アドレスデータを検証および改善します。
Web サービス	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、データの変換または配信を行うアクティブトランスフォーメーション。

詳細モードのトランスフォーメーション

詳細モードでは、トランスフォーメーションパレットに、高度な機能を有効にするトランスフォーメーションが含まれています。マッピングを詳細モードにコピーする場合、使用可能なトランスフォーメーションを使用するためにデータフローの更新が必要になることがあります。

マッピングを詳細モードにコピーすると、詳細モードでトランスフォーメーションを使用できない場合、**[検証]** パネルに検証エラーが表示されます。エラーを解決するには、マッピングのデータロジックを更新して、詳細モードで使用できるトランスフォーメーションのみを使用します。詳細モードのトランスフォーメーションは、異なる動作をする場合があります。

次の表に、トランスフォーメーションと、それらが使用可能なマッピングを示します。

トランスフォーメーション	可用性
ソース	すべてのマッピングで使用可能
ターゲット	すべてのマッピングで使用可能
アグリゲータ	すべてのマッピングで使用可能
クレンジング	すべてのマッピングで使用可能
データマスキング	詳細モード以外で使用可能
重複排除	すべてのマッピングで使用可能
式	すべてのマッピングで使用可能
フィルタ	すべてのマッピングで使用可能
階層ビルダ	詳細モード以外で使用可能
階層パーサー	詳細モード以外で使用可能
階層プロセッサ	詳細モードで使用可能

トランスフォーメーション	可用性
入力	すべてのマッピングで使用可能
Java	すべてのマッピングで使用可能
ジョイナ	すべてのマッピングで使用可能
ラベラ	すべてのマッピングで使用可能
ルックアップ	すべてのマッピングで使用可能
機械学習	詳細モードで使用可能
マップレット	すべてのマッピングで使用可能
ノーマライザ	すべてのマッピングで使用可能
出力	すべてのマッピングで使用可能
解析	すべてのマッピングで使用可能
Python	詳細モードで使用可能
ランク	すべてのマッピングで使用可能
ルータ	すべてのマッピングで使用可能
ルール仕様	すべてのマッピングで使用可能
シーケンスジェネレータ	すべてのマッピングで使用可能
ソーター	すべてのマッピングで使用可能
SQL	詳細モード以外で使用可能
構造パーサー	すべてのマッピング
トランザクション制御	詳細モード以外で使用可能
共有体	すべてのマッピングで使用可能
Velocity	詳細モード以外で使用可能
Web サービス	詳細モード以外で使用可能

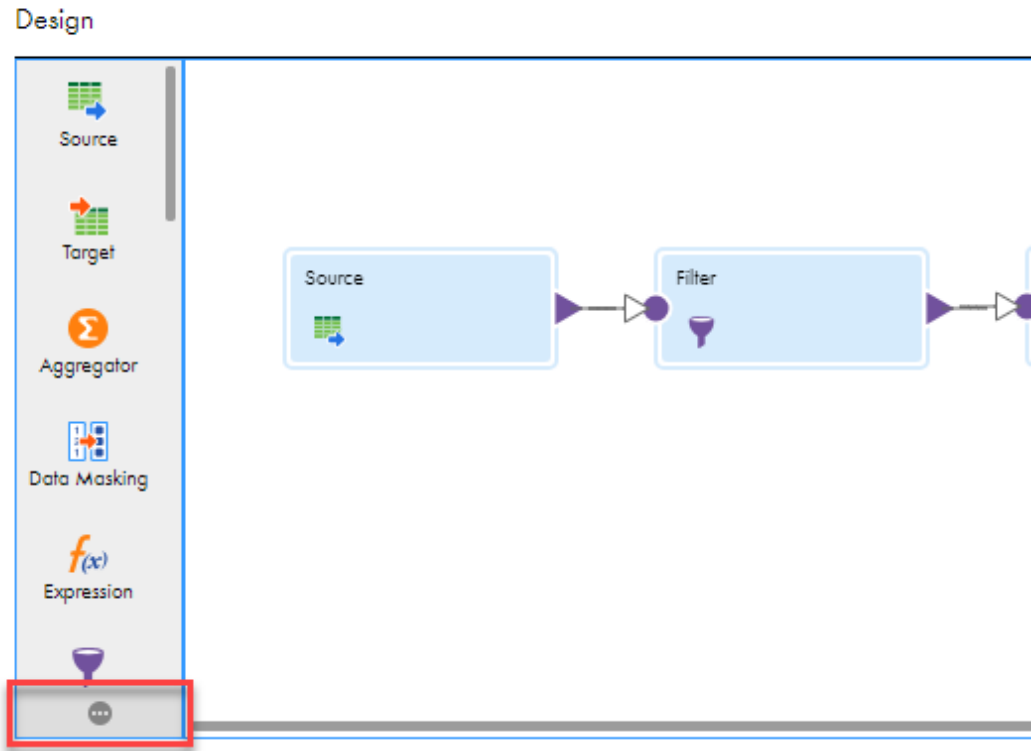
ライセンス取得済みトランスフォーメーション

マッピングで使用できるトランスフォーメーションは、組織のライセンスによって異なります。

Mapping Designer では、ライセンス取得済みトランスフォーメーションアイコンを使用して、データ統合で使用可能なすべてのトランスフォーメーション、または組織でライセンスを取得しているトランスフォーメーションのみを表示できます。デフォルトでは、トランスフォーメーションパレットにライセンス取得済みトラ

ンスフォーメーションが表示されます。すべてのトランスフォーメーションを表示するには、ライセンス取得済みトランスフォーメーションアイコンをクリックします。

ライセンス取得済みトランスフォーメーションアイコンは、次の図のようにトランスフォーメーションパレットの一番下に表示されます。



以下のオプションから選択できます。

- **ライセンス取得済みを表示**パレットには、組織で使用可能なトランスフォーメーションのみが表示されます。
- **すべて表示**パレットには、データ統合で使用可能なすべてのトランスフォーメーションが表示されます。ライセンスが取得されていないトランスフォーメーションは無効となり、マッピングを使用することはできません。

トランスフォーメーションのライセンスが有効期限切れとなった場合は、ライセンスを更新して、トランスフォーメーションを含むすべてのマッピングまたはタスクの検証、実行、インポートを行ってください。

受信フィールド

受信フィールドは、アップストリームトランスフォーメーションからトランスフォーメーションに入力されるフィールドです。

デフォルトでは、トランスフォーメーションはアップストリームトランスフォーメーションからすべての受信フィールドを継承します。ただし、このデフォルトは変更することができます。例えば、アップストリームトランスフォーメーションからのフィールドの一部が不要な場合や、アップストリームトランスフォーメーションからのフィールドの名前変更が必要な場合があります。

フィールドルールでは、アップストリームトランスフォーメーションからトランスフォーメーションへのデータの入力方法を定義します。フィールドルールを作成して、どの受信フィールドを含めるのか、除外するのか、または名前を変更するのかを必要に応じて指定できます。

フィールド名の競合は、複数のトランスフォーメーションから同じ名前のフィールドを取得する場合に発生します。アップストリームトランスフォーメーションからのフィールドが原因で発生するフィールド名の競合を解決するには、フィールド名の競合の解決を作成し、受信フィールドの名前を一括変更します。

次のリストに、フィールドが入力されて、トランスフォーメーションを通過するときに発生するイベントの順序を示します。

1. フィールド名の競合がある場合、その競合の解決ルールが実行されます。
2. フィールドルールは、フィールドがアップストリームトランスフォーメーションからトランスフォーメーションに入力されるときに実行されます。
3. トランスフォーメーションのタイプによっては、新しいフィールドがトランスフォーメーションに追加される場合があります。例えば、ルックアップトランスフォーメーションの場合、フィールドがルックアップオブジェクトからトランスフォーメーションに入力されることがあります。

フィールド名の競合

異なるトランスフォーメーションからのフィールド名が一致している場合、それらのフィールドが含まれるマッピングを検証すると、Mapping Designer によってフィールド名の競合エラーが生成されます。フィールド名の競合が発生したら、各フィールドの名前が一意になるようにする必要があります。

フィールド名の競合を解決するには、フィールドの名前を変更するフィールドルールを作成します。フィールド名の競合を解決するフィールドルールを作成する場合、アップストリームトランスフォーメーションでフィールドルールを作成します。

または、フィールド名の競合のエラーメッセージにはリンクがあるため、このリンクを使用してフィールド名の競合ルールを作成し、フィールド名の競合を解決することもできます。フィールド名の競合ルールでは、競合の原因となったフィールドだけでなく、アップストリームトランスフォーメーションからのすべてのフィールドの名前が変更されます。

フィールド名の競合ルールは、フィールドルールが有効になる前に有効になります。フィールド名の競合ルールは、アップストリームトランスフォーメーションからの受信フィールドにのみ適用できます。受信フィールドが最初にトランスフォーメーションに入力された後に発生したフィールド名の競合は、フィールド名の競合ルールでは修正できません。例えば、フィールド名の競合ルールを使用して、フィールドルールまたはアクティビティ（フィールドのルックアップなど）が原因で発生したフィールド名の競合を修正することはできません。代わりに、競合の原因となったフィールドルールまたはトランスフォーメーションを変更します。

フィールド名の競合の解決の作成

フィールド名の競合を解決するには、フィールド名の競合のエラーメッセージからアクセスする**［フィールド名の競合の解決］**ダイアログボックスを使用して、アップストリームトランスフォーメーションからのすべてのフィールドの名前を一括変更します。

1. エラーメッセージのリンクをクリックして、**［フィールド名の競合の解決］**ダイアログボックスにアクセスします。
2. 名前を一括変更するフィールドを含むアップストリームトランスフォーメーションを選択します。
3. **［一括名前変更オプション］**カラムで、プレフィックスを追加して名前を変更するのか、サフィックスを追加して名前を変更するのかを指定します。
4. フィールド名に追加するテキストを入力し、**［OK］**をクリックします。

フィールドルール

アップストリームトランスフォーメーションからの受信フィールドに基づいてフィールドルールを設定します。次に、フィールド選択条件およびフィールドの命名規則を設定します。

フィールドルールを設定する場合、次の手順を実行します。

1. 含めるまたは除外する受信フィールドを選択します。処理時間を短縮し、データセットをクリーンな状態に保つには、必要な受信フィールドのみを含めます。
2. フィールド選択条件を設定して、ルールに適用する受信フィールドを決定します。[名前付きフィールド] 選択条件を使用する場合、受信フィールドのパラメータを使用できます。
3. 必要に応じて、フィールドの名前を変更するように選択します。異なるソースからのフィールドを区別するため、またはフィールド名の競合を避けるために、受信フィールドの名前を変更できます。パターンオプションを使用する場合、パラメータを作成してフィールドの名前を一括変更できます。
4. 実行順序を確認します。複数のルールを設定する場合、マッピングタスクがルールを適用する順序を変更できます。

注: ソースを含むソーストランスフォーメーションまたはマップレットトランスフォーメーションでは、フィールドルールを設定できません。

手順 1。受信フィールドの選択

フィールドルールを設定する場合、指定した受信フィールドをルールに含めるのか、除外するのかを指定します。

[含める] / [除外する] 演算子をフィールド選択条件と組み合わせて使用し、フィールドルールの影響を受ける受信フィールドを決定します。

例えば、トランスフォーメーションですべてのバイナリフィールドを除外する場合を考えます。[除外する] 演算子を選択して、フィールド選択条件に一致する受信フィールドが現在のトランスフォーメーションに渡されないように指定します。次に、フィールド選択条件にバイナリデータ型を指定します。

手順 2.フィールド選択条件の設定

フィールドルールを設定する場合、フィールド選択条件を指定して、フィールドルールに適用する受信フィールドを決定します。

次のいずれかのフィールド選択条件を選択することができます。

すべてのフィールド

すべての受信フィールドを含めます。このオプションを **[含める]** 演算子と組み合わせて使用している場合、受信フィールドの名前を一括変更できます。

名前付きフィールド

指定した受信フィールドを含めたり、除外したりします。**[名前付きフィールド]** 選択条件を使用して、名前を変更したり、受信トランスフォーメーションに含めたり、受信トランスフォーメーションから除外したりする個々の受信フィールドを指定します。フィールド選択条件の詳細を入力する場合、すべての受信フィールドを確認し、含めるまたは除外するフィールドを選択できます。フィールドがリストに表示されていない場合、そのフィールドがソースに存在していれば追加できます。含めるまたは除外するフィールドを表すパラメータを作成することもできます。

データ型を指定

指定したデータ型の受信フィールドを含めたり、除外したりします。フィールド選択条件の詳細を入力する場合、含めるまたは除外するデータ型を選択できます。

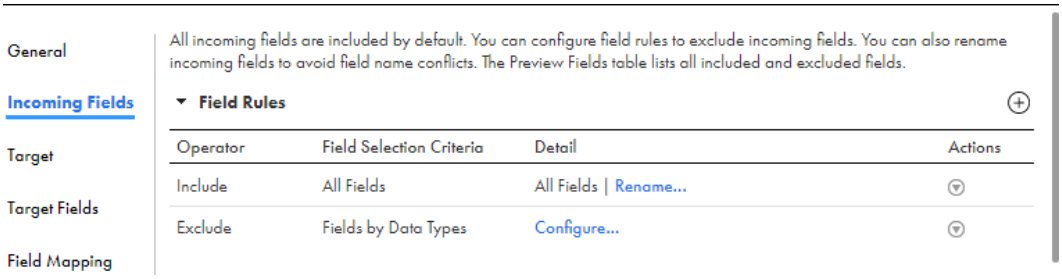
テキストまたはパターンを指定

プレフィックス、サフィックス、またはパターンで受信フィールドを含めたり、除外したりします。このオプションを使用すると、データフローで以前に名前を変更したフィールドを選択できます。フィールド選択条件の詳細の入力時に、プレフィックス、サフィックス、またはパターンの選択や、使用するルールの定義ができます。

プレフィックスオプションまたはサフィックスオプションを選択する場合、プレフィックスまたはサフィックスとして使用するテキストを入力します。例えば、「Cust」という文字列で始まるすべてのフィールドを検索するには、プレフィックスとして Cust と入力します。

パターンオプションを選択した場合は、正規表現の入力や、パターンに対するパラメータの使用ができます。表現には、Perl 互換の正規表現構文を使用する必要があります。例えば、文字列「Cust」または「Addr」で始まるすべてのフィールドを検索するには、パターン Cust*|Addr.*を入力します。フィールド名のいずれかの部分に文字列「Cust」または「CUST」を含むすべてのフィールドを検索するには、パターン*Cust.*|.*CUST.*を入力します。Perl 互換の正規表現構文の詳細については、「*関数参照*」の REG_EXTRACT 関数のヘルプを参照してください。

次の画像は、**［データ型を指定］**フィールド選択条件の選択を示しています。



次の画像は、フィールド選択条件の詳細の **［date/time］**データ型の選択を示しています。

手順 3。フィールドの名前変更

フィールド名の競合を回避するため、または複合マッピングのフィールドの入力元を明確にするためにフィールドの名前を変更します。トランスフォーメーションのフィールドルールの一部としてフィールドの名前を変更できます。フィールドルールのフィールド選択条件を指定したら、選択したフィールドの名前変更方法を指定します。

フィールドの名前は個別にまたは一括で変更できます。フィールドの名前を個別に変更する場合、受信フィールドのリストから名前を変更するフィールドを選択します。次に、選択したフィールドごとに名前を指定します。

一括で名前を変更する場合、プレフィックス、サフィックス、またはパターンを追加してすべてのフィールドの名前を変更できます。プレフィックスまたはサフィックスでフィールドの名前を変更する場合、プレフィックスまたはサフィックスとして使用するテキスト文字列を入力します。例えば、すべてのフィールドの名前をFF_<field name>として変更するように指定できます。

パターンでフィールドの名前を変更する場合、パターンを表す正規表現を入力するか、タスクのパターンを定義するパラメータを使用します。プレフィックスまたはサフィックスをすべてのフィールド名に追加する簡単な式を作成することも、特定のパターンを特定のテキストで置き換える式を作成することもできます。

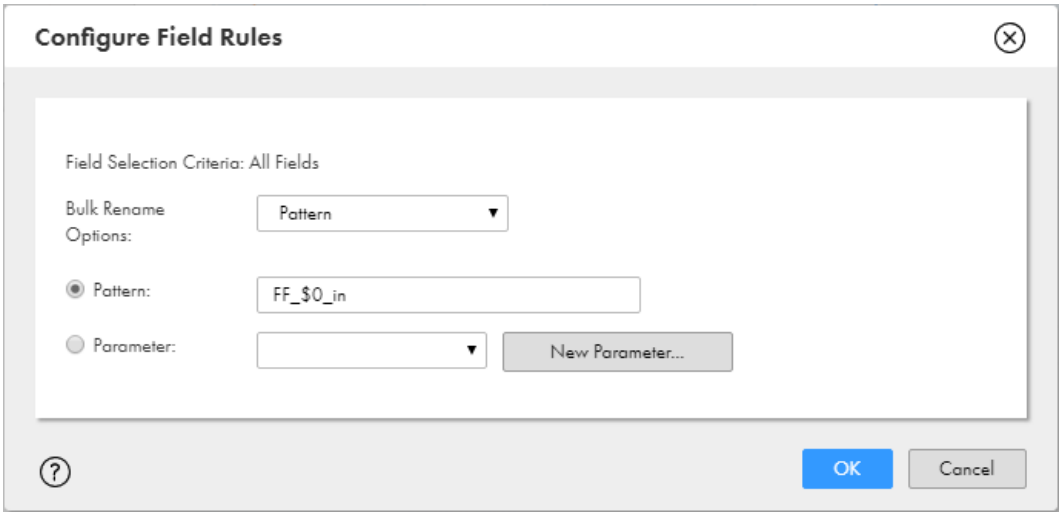
パターンをテキストで置き換えるには、次の構文で正規表現を使用します。ここでは、マッチングするパターンとそのパターンを置き換えるテキストがスラッシュで区切られています。

<マッチングするパターン>/<置換テキスト>

次の表では、フィールド名を一括変更するときの正規表現の例をいくつか示します。

目的	式
出現するすべての「Inc」を「LLC」で置き換える。	Inc/LLC
フィールド名の末尾に出現する「Inc」を「LLC」で置き換える。	Inc$/LLC
フィールド名の先頭に出現する「Loc」を「Branch」で置き換える。	^Loc/Branch
出現する「A/C」をすべて削除する。	A\/C(.*)/$1 **注:** フィールド名の 1 文字が正規表現のメタ文字の場合、その文字をバックスラッシュでエスケープしてそれがリテラルであることを示します。この例では、スラッシュがメタ文字です。
プレフィックス「FF」とサフィックス「_in」をすべてのフィールドに追加する。	FF_$0_in

次の画像は、**[フィールドルールの設定]** ダイアログボックスの［一括名前変更オプション］で［パターン］が選択され、使用するパターンが指定されている状態を示します。



フィールド名変更ルールは、フィールド名の競合などの問題が発生しないように、注意して作成します。フィールド名変更ルールによってフィールド名の競合が発生した場合、ルールを編集できます。

ヒント: アップストリームトランスフォーメーションが、名前を一括変更できないソースの場合、式トランスフォーメーションを追加してフィールドの名前を変更できます。

手順 4。ルールの実行順序の確認

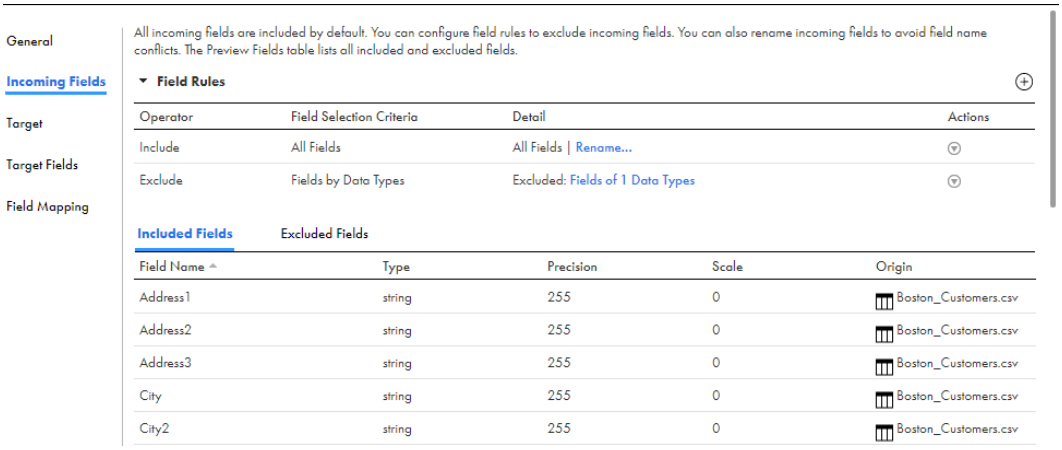
複数のフィールドルールを作成する場合、ルールが論理的順序で実行されることを確認します。

ルールの実行順序を確認するには、**[フィールドルール]** 領域でルールを参照します。マッピングタスクは、ルールの表示順序でルールを実行します。フィールドルールの順序が正しくない場合、順序を変更できます。

プレビューフィールドテーブルで、作成したルールに基づいてトランスフォーメーションの受信フィールドをプレビューすることもできます。プレビューフィールドテーブルには、組み込まれたフィールドと除外されたフィールドがすべてリストされます。例えば、バイナリフィールドを除外するフィールドルールを作成する場合、**[除外されたフィールド]** の一覧には、トランスフォーメーションから除外されるバイナリフィールドが一覧表示されます。

マッピング内のソーストランスフォーメーションで接続パラメータまたはデータオブジェクトパラメータを使用する場合、[プレビューフィールド] 領域には、トランスフォーメーションの受信フィールドは表示されません。

次の画像は、プレビューフィールドテーブルを示しています。



フィールドルールの設定の例

次の例に、トランスフォーメーションでフィールドルールを設定する方法を示します。

受信フィールド名の変更

複数の営業所から収益データを収集するマッピングを作成する必要があるとします。アップストリームトランスフォーメーションからの複数のフィールドに、ソーストランスフォーメーションのフィールドと同じ名前があることがわかっています。フィールド名の競合を避けるために、すべての受信フィールドのフィールド名を変更して、マッピング全体でソースを区別できるようにします。受信フィールドの名前を変更するために、SalesForce_というプレフィックスを付けてすべてのフィールドの名前を変更するフィールドルールを作成します。

パフォーマンスが向上するように、必要なデータのみがデータセットに含まれるようにしたいと考えています。トランザクション日に関する情報は必要ないと判断したため、データフィールドをマッピングから削除します。必須ではないフィールドを除外するルールを作成します。日付フィールドを除外するために、日時のデータ型のフィールドを除外するルールを作成します。

ルールの順序を確認します。日時フィールドを除外するルールの後に、フィールドの名前を変更するルールを実行する必要があります。名前を変更するルールの前に表示されるように、日時フィールドを除外するルールを移動します。

フィールド名のパターンの削除

括弧を使用してさまざまなパターンをグループ化し、参照を使用してパターンを置き換えることができます。正規表現では、括弧を使用して一致するパターンをグループ化します。($)参照を使用して、入力文字列内で一致するグループを選択できます。

トランスフォーメーション間でフィールドのパターンを変更する必要がある場合があります。アップストリームトランスフォーメーションのフィールドにサフィックス_out が含まれている場合は、現在のトランスフォーメーションのフィールド名からサフィックスを削除できます。_out サフィックスを削除するには、式 (.*)_out/$1 を使用します。式の(.*)部分はすべてのフィールド名文字と一致し、_out はサフィックスと一致します。また、$1 は一致した(.*)フィールド名の文字で指定された入力文字列を参照します。

次の式は、別の例を示しています。

(.*)(_)out/$1

例えば、string_out の一致した文字列では、string については$1、_については$2、out については$3 が一致します。string は一致の最初のグループであるため、string 値が$1 によって参照されます。

注: このフィールドルールを適用してフィールドの名前を変更すると、データ統合では、1 で始まるサフィックスが同じ名前の複数のフィールドに追加されます。この規則により、重複するフィールド名が削除されます。

フィールドルールの作成

Mapping Designer の **[プロパティ]** パネルの **[受信フィールド]** タブでフィールドルールを設定します。

1. **[受信フィールド]** タブの **[フィールドルール]** 領域で、ルールの実行順序に基づいてルールの行を挿入します。新しいルールの前または後に実行するルールの **[アクション]** カラムで、**[上に挿入]** または **[下に挿入]** のいずれかを選択します。・
2. ルールにフィールドを含めるのか、除外するのかを指定するには、**[演算子]** カラムで、**[含める]** または **[除外する]** のいずれかを選択します。
3. **[フィールド選択条件]** カラムで、次のいずれかの方法を選択します。
   - すべての受信フィールドの名前を一括変更するには、**[すべてのフィールド]** を選択します。
   - 指定したフィールドにルールを適用するには、**[名前付きフィールド]** を選択します。
   - 各フィールドのデータ型に基づいてルールを適用するには、**[データ型ごとのフィールド]** を選択します。
   - 特定のプレフィックス、サフィックス、またはパターンを含むすべてのフィールドにルールを適用するには、**[テキストまたはパターンを指定]** を選択します。
4. フィールド選択の詳細を指定するには、**[詳細]** カラムで、**[設定]** または **[名前の変更]** リンクをクリックします。フィールド選択条件が **[すべてのフィールド]** の場合、**[名前の変更]** リンクが表示されます。
5. **[フィールドルールの設定]** ダイアログボックスで、選択したフィールド選択基準に基づいて、ルールに適用するフィールドを選択します。または、マッピングタスクでフィールドを選択できるように、**[パラメータ]** をクリックしてパラメータを追加します。
6. フィールドの名前を変更するには、**[名前変更フィールド]** タブをクリックし、フィールドの名前を個別に変更するか、一括で変更するかを選択します。

   すべてのフィールドの名前を変更する場合、名前を一括変更する必要があります。フィールドの名前をパターンで一括変更する場合、パラメータを作成して、マッピングタスクのパターンを指定できます。
7. フィールドルールが論理的順序で実行されるようにするには、**[フィールドルール]** 領域で、ルールの表示順序を確認します。**[含まれるフィールド]** および **[除外するフィールド]** の一覧で、ルールの結果を確認します。必要に応じて、フィールドルールを適切な場所に移動します。

8. ルールを削除するには、**［アクション］**カラムで、**［削除］**を選択します。

データオブジェクトのプレビュー

ソース、ターゲット、また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、ソース、ターゲット、またはルックアップオブジェクトとして 1 つのオブジェクトを選択すると、データをプレビューできます。

このデータプレビュー機能は、マッピングデータプレビューとは異なります。マッピングデータのプレビューでは、マッピングロジックの処理の結果として変更されたデータを確認できます。マッピングデータのプレビューの詳細については、「*マッピング*」を参照してください。

データオブジェクトをプレビューするには、**［プロパティ］**パネルの**［ソース］**、**［ターゲット］**、または**［ルックアップオブジェクト］**タブを開いて、**［データのプレビュー］**をクリックします。

データをプレビューすると、データ統合では、次の情報が表示されます。

- それらのフィールドを含む、ネイティブ順での最初の 10 行を表示。
- **［ソースフィールドをアルファベット順に表示］**オプションを有効にした場合、フィールドが含まれる行をアルファベット順に表示。

ソース、ターゲット、またはルックアップオブジェクトがフラットファイルの場合は、形式オプションを設定することもできます。次の表に、フラットファイルの形式オプションを示します。

プロパティ	説明
フラットファイルタイプ	ファイルタイプ（区切りまたは固定長）。
区切り文字	区切りファイルの場合の区切り文字。
テキスト修飾子	区切りファイルの場合のテキストを修飾する文字。
エスケープ文字	区切りファイルの場合のエスケープ文字。
フィールドラベル	区切りファイルの場合、タスクがフィールドラベルを生成するか、またはソースファイルからインポートするかを決定します。ソースファイルからインポートする場合は、フィールドラベルを含む行番号を入力します。
固定長ファイル形式	固定長ファイルに使用するファイル形式。リストに同じ名前の複数の固定長ファイル形式が含まれている場合は、名前に追加されたプロジェクトとフォルダの場所を使用して、使用する適切なファイル形式を指定します。使用可能な固定長ファイル形式がない場合は、**［新規］** > **［コンポーネント］** > **［固定長ファイル形式］**を選択して作成します。

注: その他の形式オプションは、接続のタイプに基づいて使用できる場合があります。詳細については、該当するコネクタのヘルプを参照してください。

変数フィールド

変数フィールドは、計算を定義し、データを一時的に格納します。変数フィールドは、式トランスフォーメーションおよびアグリゲータトランスフォーメーションで使用できます。

変数を使用して、次の作業を実行できます。

- 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。
- 値の比較。

例えば、名と姓を連結してから名前と住所データをマージして郵送先名簿を生成する場合を想定します。これを行うには、名と姓のフィールドを連結する FullName という変数フィールドを作成します。次に、FullName 変数フィールドと住所フィールドを連結する NameAddress という式フィールドを作成します。

変数フィールドの結果はデータフローに渡されません。変数フィールドからのデータをデータフローで使用するには、変数フィールドの出力用の式フィールドを作成します。前の例の場合、連結された名と姓をデータフローに渡すために、FullName_out 式フィールドを作成します。次に、FullName 変数フィールドをそのフィールドの式として使用します。

トランスフォーメーションキャッシュ

データ統合では、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。

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。

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、データ統合は十分なメモリを割り当てることができず、タスクは失敗します。

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を得るには、データ統合がキャッシュメモリ内でトランスフォーメーションデータ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した後に、各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。

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。ソータートランスフォーメーションには 1 つのキャッシュが必要です。

以下の表に、データ統合で各キャッシュに保存される情報のタイプを示します。

トランスフォーメーション	キャッシュタイプ
アグリゲータ	インデックス。グループ化フィールドの設定に従ってグループ値を格納します。 データ。グループ化フィールドに基づいて計算結果を格納します。
ジョイナ	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	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	インデックス。グループ化フィールドの設定に従ってグループ値を格納します。 データ。グループ化フィールドに基づいて行データを格納します。
ソータ	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、データ統合で、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。

アグリゲータ、ジョイナ、ルックアップ、およびランクトランスフォーメーションの場合、データ統合では、トランスフォーメーションを実行するためのインデックスファイルとデータキャッシュファイルが作成されます。ソータートランスフォーメーションの場合、データ統合では、1 つのソーターキャッシュファイルが作成されます。デフォルトでは、データ統合は、データ統合サーバーで Secure Agent $PMCacheDir プロパティに入力されたディレクトリにキャッシュファイルを格納します。キャッシュディレクトリは、トランスフォーメーションプロパティの **[詳細]** タブで変更できます。キャッシュディレクトリを変更する場合は、指定するディレクトリが存在し、キャッシュファイルのための十分なディスクスペースがあることを確認してください。

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、固有の値を使用するようにトランスフォーメーションを設定できます。

自動キャッシュ

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。データ統合は、実行時にキャッシュメモリ要件を自動的に計算します。タスクの設定時に、データ統合が詳細セッションプロパティで割り当てることができるメモリの最大量を定義することもできます。

データ統合は、処理時間が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ます。例えば、データ統合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。

データキャッシュとインデックスキャッシュを使用するトランスフォーメーションでは、データ統合は、インデックスキャッシュよりもデータキャッシュに多くのメモリを割り当てます。また、ソータートランスフォーメーションのすべてのメモリをソーターキャッシュに割り当てます。

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。

セッションログを使用して、最適なキャッシュサイズを指定できます。セッションログに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ことで、割り当てたメモリを無駄にしないようにすることができます。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。

特定のキャッシュサイズを定義するには、トランスフォーメーションプロパティの**［詳細］**タブでキャッシュサイズの値を入力します。

キャッシュサイズの最適化

最適なマッピングパフォーマンスを得るには、データ統合がキャッシュメモリ内で完全な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を設定します。

1. トランスフォーメーションプロパティの**［詳細］**タブで、［トレースレベル］を［詳細 - 初期化］に設定します。
2. 自動キャッシュモードでタスクを実行します。
3. セッション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。

   例えば、「Joiner」というジョイナトランスフォーメーションがあるとします。セッションログには、以下のテキストが含まれます。

   ```
   CMN_1795 [2023-01-06 16:16:59.026] The index cache size that would hold [10005] input rows from the master for [Joiner], in memory, is [8437760] bytes
   CMN_1794 [2023-01-06 16:16:59.026] The data cache size that would hold [10005] input rows from the master for [Joiner], in memory, is [103891920] bytes
   ```

   ログは、インデックスキャッシュサイズとして 8,437,760 バイトが必要であり、データキャッシュとして 103,891,920 バイトが必要であることを示しています。
4. トランスフォーメーションプロパティの**［詳細］**タブで、セッションログで推奨されたキャッシュサイズの値をバイト単位で入力します。

式マクロ

式マクロは、マッピングで繰り返しや複雑な式を作成するために使用するマクロです。

式マクロを使用すれば、複数の式や定数に対して計算を実行できます。例えば、式マクロを使用することで、一連の販売範囲で複数のフィールドやラベル項目を NULL 値に置換できます。

式マクロでは、1 つまたは複数の入力フィールドでマクロのソースデータを表します。実行する計算は式で表します。出力フィールドは、計算の結果を表します。

実行時に、タスクが式を拡張してすべての入力フィールドと定数を組み込み、次に結果を出力フィールドに書き込みます。

式トランスフォーメーションおよびアグリゲータトランスフォーメーションで式マクロを作成できますが、式トランスフォーメーションでは式マクロと入出力パラメータを組み合わせることはできません。

マクロのタイプ

以下のタイプのマクロを作成できます。

垂直

垂直マクロは式を垂直に拡張します。垂直マクロは、複数の受信フィールドに同じ計算を実行するために類似する一連の式を生成します。

水平

水平マクロは式を水平に拡張します。水平マクロは拡張された式を 1 つ生成し、その式により複数のフィールドや定数が組み込まれます。

混合

混合マクロは、式を垂直および水平に拡張します。混合マクロは、複数の垂直式を生成し、それらはさらに水平にも拡張されます。

マクロ入力フィールド

マクロ入力フィールドは、式マクロで使用する入力を表すフィールドです。入力には、フィールドまたは定数を使用できます。すべての式マクロにはマクロ入力フィールドが必要です。

垂直マクロの 1 つのマクロ入力フィールドは、一連の受信フィールドを表します。

水平マクロの 1 つのマクロ入力フィールドは、一連の受信フィールドまたは一連の定数を表します。水平マクロでは、複数のマクロ入力フィールドを作成して、一連の定数を複数定義できます。

マクロ入力フィールドでは、%<macro_field_name>%の命名規則が使用されます。

例えば、式を複数のアドレスフィールドに適用することが必要な場合があります。この場合、%AddressFields%というマクロ入力フィールドを作成して、フィールドルールを定義し、使用する受信フィールドを指定します。式を設定するときには、%AddressFields%を使用して受信フィールドを表します。

垂直マクロ

マクロ式を一連の受信フィールドに適用するには、垂直マクロを使用します。

垂直マクロのマクロ入力フィールドは、受信フィールドを表します。式は、すべての受信フィールドで実行する計算を表します。またマクロ出力フィールドは、計算結果を残りのマッピングに渡す一連の出力フィールドを表します。マクロ出力フィールドでマクロ式を設定します。

マクロ出力フィールドはマクロの出力フィールドを表しますが、出力フィールドの名前はマッピングで明確に定義されません。垂直マクロの結果をマッピングに含めるには、マクロによって生成された出力フィールドを含めるように、ダウンストリームのトランスフォーメーションのフィールドルールを設定します。

垂直マクロの結果をターゲットに書き込むには、出力フィールドをターゲットトランスフォーメーションのターゲットフィールドにリンクします。

タスクが実行されると、タスクは、マクロ入力フィールドが表すフィールドごとに計算を実行する複数の式を生成します。またタスクは、マクロ出力フィールドを実際の出力フィールドで置き換え、その出力フィールドを使用して、計算結果を残りのマッピングに渡します。

注: マクロ出力フィールドでは、データの受け渡しは行われません。

例

次の垂直のマクロ式は、%Addresses%マクロ入力フィールドが表すフィールドから先頭および末尾のスペースを切り捨てます。

```
LTRIM(RTRIM(%Addresses%))
```

実行時に、タスクは、%Address%が表すフィールドからスペースを切り捨てる次の一連の式を生成します。

```
LTRIM(RTRIM(Street))
LTRIM(RTRIM(City))
LTRIM(RTRIM(State))
LTRIM(RTRIM(ZipCode))
```

垂直マクロの設定

式トランスフォーメーションの［式］タブまたはアグリゲータトランスフォーメーションの［集計］タブで垂直マクロを設定できます。

1. マクロ入力フィールドを作成して使用する受信フィールドを定義します。
2. マクロ出力フィールドを作成して出力フィールドのデータ型と命名規則を定義します。
3. マクロ出力フィールドで、マクロ式を設定します。マクロ入力フィールドをマクロ式に含めます。
4. ダウンストリームトランスフォーメーションで、マクロの結果をマッピングに含めるフィールドルールを設定します。

垂直マクロのマクロ入力フィールド

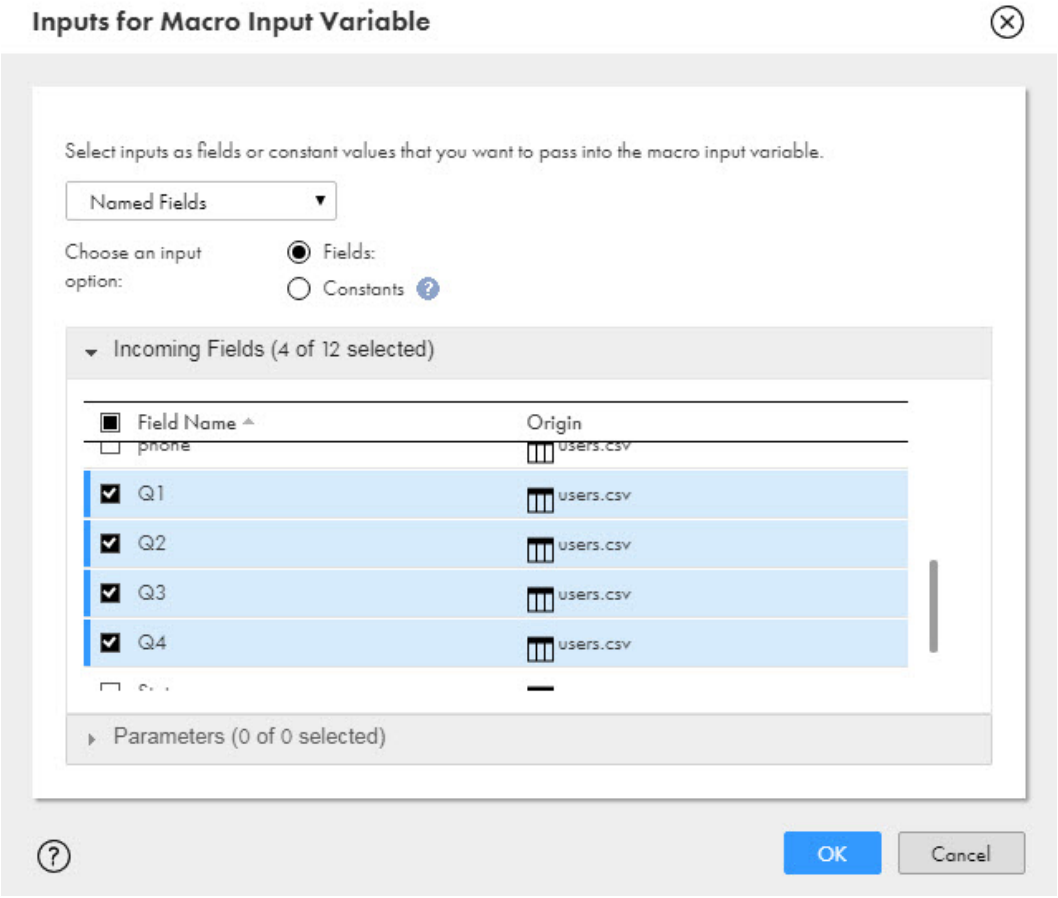
マクロ入力フィールドは、垂直マクロで使用する受信フィールドを表すために使用します。

マクロ入力フィールドを作成するとき、マクロ入力フィールドの名前を定義し、フィールドルールによって使用する受信フィールドを定義します。実行時には、マクロ入力フィールドが拡張されて選択したフィールドを表します。

マクロ入力フィールドを設定するときは、次のフィールドルールを使用します。

- すべてのフィールド
- 名前付きフィールド
- テキストまたはパターンごとのフィールド

次の画像は、Q1～Q4 フィールドを含んでいる［名前付きフィールド］フィールドルールを示しています。



垂直マクロのマクロ出力フィールド

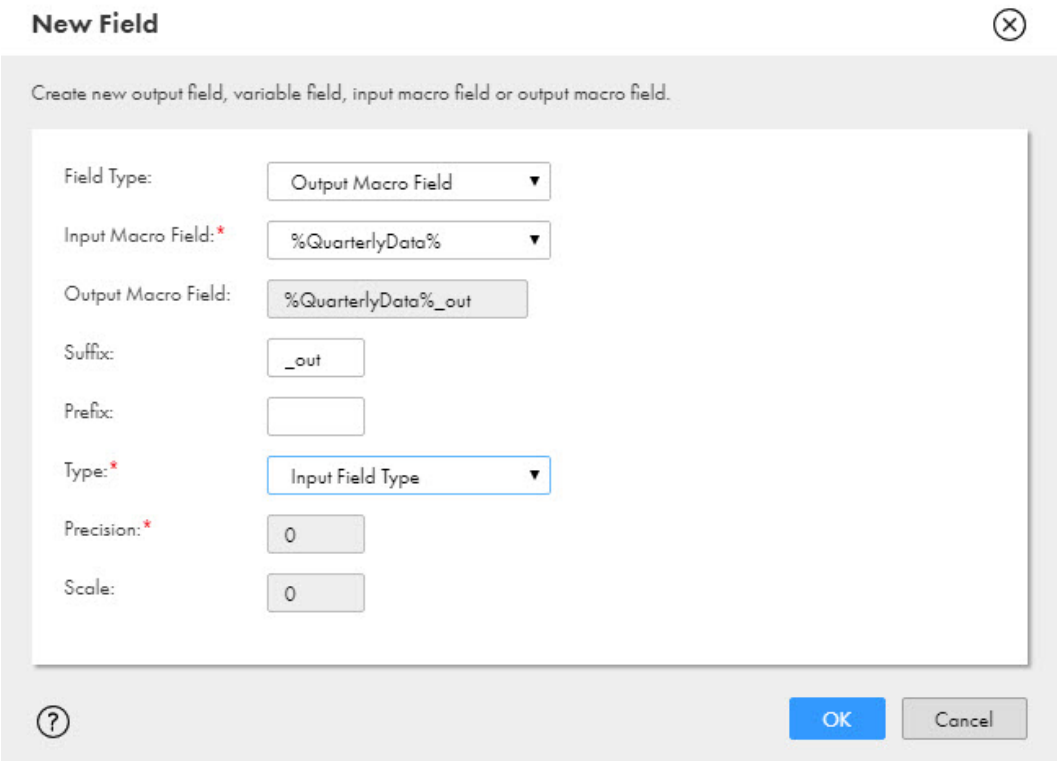
マクロ出力フィールドは、タスクが垂直マクロに対して実行時に生成する出力フィールドを表します。また、マクロ出力フィールドで使用する式も定義します。

マクロ出力フィールドを設定するときには、出力フィールドの命名規則を定義するために使用するマクロ入力フィールドを選択します。命名規則のプレフィックスやサフィックスをカスタマイズできます。マクロ出力フィールドでは、デフォルトで<macro_input_field>_out の命名規則が出力フィールドに使用されます。

出力フィールドのデータ型、精度、およびスケールを定義できます。または、受信フィールドのデータ型、精度、およびスケールを使用するようにマクロ出力フィールドを設定できます。受信フィールドのデータ型は、受信フィールドに複数のデータ型があり、式が受信データのデータ型を変更しない場合に使用します。

実行時には、タスクがマクロ出力フィールドの設定に基づいて出力フィールドを生成します。タスクは、マクロ入力フィールドが表す受信フィールドごとに出力フィールドを作成し、次に式の結果を出力フィールドに書き込みます。

例えば次の画像は、%QuarterlyData%が表す受信フィールドに基づいて出力フィールドを作成するマクロ出力フィールドを示しています。



%QuarterlyData%マクロ入力フィールドが Q1～Q4 のフィールドを表している場合、実行時にタスクによって Q1_out、Q2_out、Q3_out、Q4_out の出力フィールドが作成されます。出力フィールドは入力フィールドと同じデータ型になります。

［入力フィールドタイプ］のデータ型を選択した後に、精度やスケールを定義することはできません。

垂直マクロの出力フィールドのフィールドルール

マッピングで垂直マクロの結果を使用するには、ダウンストリームトランスフォーメーションで出力フィールドを組み込むようにフィールドルールを設定するか、ターゲットフィールドマッピングをパラメータ化します。

式マクロは、実行時まで明確に定義されないフィールドを表すため、ダウンストリームトランスフォーメーションに垂直マクロの出力フィールドを組み込むように設定する必要があります。これを実行するには、2 つの方法があります。

- ダウンストリームトランスフォーメーションで名前付きフィールドを作成します。**［受信フィールド］**タブで、名前付きフィールドルールを作成し、垂直マクロの出力フィールドごとに新しい入力フィールドを作成します。これらのフィールドは、ダウンストリームトランスフォーメーションで使用できます。
- または、ターゲットトランスフォーメーションがマクロの直接のダウンストリームである場合は、ターゲットフィールドマッピングを完全にパラメータ化します。マッピングタスクを設定すると、データ統合はターゲットにマクロ出力フィールドを作成します。受信フィールドをターゲットフィールドにマッピングします。

例

%InputDates%という名前のマクロ入力フィールドは、データを日付データ型に変換するマクロの次のソースフィールドを表します。

```
OrderDate
ShipDate
PaymentReceived
```

マクロ出力フィールドは、デフォルトの命名規則である、<マクロ入力フィールド>_out を使用します。マクロが生成する日付フィールドを使用するには、ダウンストリームトランスフォーメーションで名前付きフィールドのルールを作成します。次のフィールドを作成します。

```
OrderDate_out
ShipDate_out
PaymentReceived_out
```

作成したフィールドを含めるようにフィールドルールを設定します。

フィールドルールを作成したら、ダウンストリームトランスフォーメーションの式やフィールドマッピングでそれらのフィールドを使用できます。

垂直マクロの例

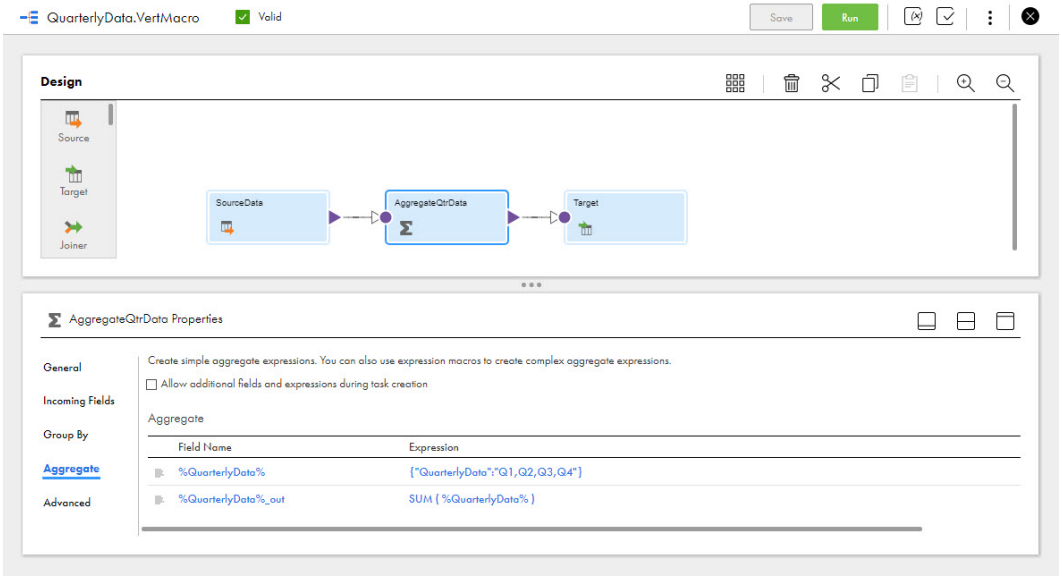
各ストアの四半期データの年間合計を求めるには、アグリゲータトランスフォーメーションで垂直の式マクロを使用できます。

アグリゲータトランスフォーメーションでは、グループ化フィールドとしてストア ID フィールドを使用します。%QuarterlyData%マクロ入力フィールドには、次の入力フィールドから売上データが読み込まれます: Q1、Q2、Q3、Q4。

%QuarterlyData%_out マクロ出力フィールドは、%QuarterlyData%マクロ入力フィールドに基づきます。四半期ごとの売上合計を求めるために、マクロ出力フィールドには次の式が含まれています: `SUM(%QuarterlyData%)`。

ターゲットトランスフォーメーションでは、フィールドルールの受信フィールドリストに、次の出力フィールドが含まれています: Q1_out、Q2_out、Q3_out、Q4_out。ターゲットフィールドマッピングでは、Qx_out フィールドがターゲットにマップされます。

次の図に、アグリゲータトランスフォーメーションでの垂直の式マクロを示します。



タスクを実行すると、式が次のように垂直方向に展開されます。

```
SUM(Q1)
SUM(Q2)
SUM(Q3)
SUM(Q4)
```

タスクは、集計を実行し、結果をターゲットに書き込むときに、ストアごとにデータをグループ化します。

水平マクロ

水平マクロは、複数の入力フィールドや定数を組み込む 1 つの複雑な式を生成するために使用します。

水平マクロでは、マクロ入力フィールドで複数の入力フィールドや定数を表すことができます。

水平マクロでは、式が、入力フィールドや定数に実行する計算を表します。式には、水平拡張関数を含める必要があります。

水平マクロ式によって 1 つの結果が生成され、トランスフォーメーション出力フィールドから結果が以降のマッピングに渡されます。水平マクロ式はトランスフォーメーション出力フィールドで設定します。

式の結果は、デフォルトのフィールドルールに従って、ダウンストリームのンスフォーメーションに渡されます。マッピングで、水平マクロの結果を組み込むためにフィールドルールを追加する必要はありません。

水平マクロの結果をターゲットに書き込むには、トランスフォーメーション出力フィールドをターゲットトランスフォーメーションのターゲットフィールドに接続します。

例

例えば、水平マクロでは、%AllFields%マクロ入力フィールドで表されたフィールドの NULL 値をチェックできます。フィールドが NULL の場合、1 が返されます。さらに、%OPR_SUM%水平拡張関数が、NULL フィールドの合計数を返します。

次の式は、マクロ内での計算を表しています。

```
%OPR_SUM[ IIF(ISNULL(%AllFields%), 1, 0) ]%
```

実行時には、%AllFields%が表すフィールドを組み込むために、アプリケーションが次のように式を水平に拡張します。

```
IIF(ISNULL (AccountID, 1,0)+IIF(ISNULL(AccountName, 1, 0)+IIF(ISNULL(ContactName, 1, 0)+IIF(ISNULL(Phone, 1, 0)+IIF(ISNULL(Email, 1, 0)...
```

水平拡張関数

水平拡張関数を使用して、式マクロに式を作成します。

水平拡張関数には、%OPR_<function_type>%の命名規則を使用します。水平拡張関数では、丸カッコではなく角かっこ（[ ]）を使用します。

［フィールド式］ダイアログボックスで、関数のリストの［水平拡張］グループに関数が表示されます。

次の水平拡張関数を使用できます。

%OPR_CONCAT%

CONCAT 関数を使用して、式マクロ内の式を拡張し、複数のフィールドを連結します。%OPR_CONCAT%は、次の式と同じ計算値を作成します。

```
FieldA || FieldB || FieldC...
```

%OPR_CONCATDELIM%

CONCAT 関数を使用して、式マクロ内の式を拡張し、複数のフィールドを連結してカンマ区切り文字を追加します。%OPR_CONCATDELIM%は、次の式と同じ計算値を作成します。

```
FieldA || ", " || FieldB || ", " || FieldC...
```

%OPR_IIF%

IIF 関数を使用して、式マクロ内の式を拡張し、IIF 文のセットを評価します。%OPR_IIF%は、次の式と同じ計算値を作成します。

```
IIF(<field> >= <constantA>, <constant1>,
  IIF(<field> >= <constantB>, <constant2>,
    IIF(<field> >= <constantC>, <constant3>, 'out of range')))
```

%OPR_SUM%

SUM 関数を使用して、式マクロ内の式を拡張し、すべてのフィールドの合計を返します。%OPR_SUM%は、次の式と同じ計算値を作成します。

```
FieldA + FieldB + FieldC...
```

水平拡張関数の詳細については、*『Function Reference』*を参照してください。

水平マクロの設定

式トランスフォーメーションの**［式］**タブまたはアグリゲータトランスフォーメーションの**［集計］**タブで、水平マクロを設定できます。

マクロ式で受信フィールドまたは定数のどちらを使用するかに基づいて水平マクロを設定します。

1. 1 つ以上のマクロ入力フィールドを作成します。
   - 受信フィールドを使用するには、マクロ入力フィールドを作成して使用する受信フィールドを定義します。
   - 定数を使用するには、使用する定数の各セットについてマクロ入力フィールドを作成します。
2. トランスフォーメーション出力フィールドを作成します。
3. トランスフォーメーション出力フィールドで、マクロ式を設定します。水平拡張関数を使用してマクロ入力フィールドを含めます。
4. マッピングに水平マクロの結果を含めるには、ダウンストリームトランスフォーメーションでデフォルトのフィールドルールを使用します。トランスフォーメーション出力フィールドを含む任意のフィールドルールを使用できます。
5. 水平マクロの結果をターゲットに書き込むには、トランスフォーメーション出力フィールドをターゲットトランスフォーメーションのターゲットフィールドに接続します。

水平マクロの受信フィールド用のマクロ入力フィールド

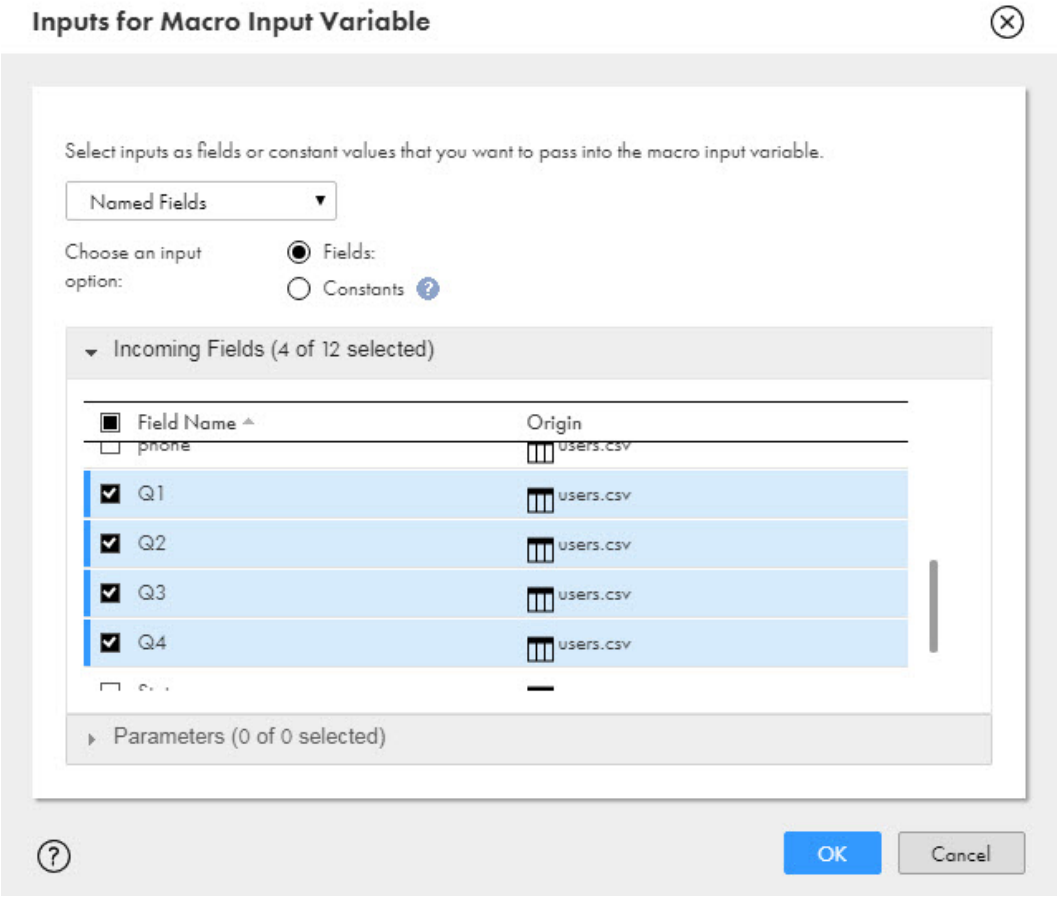
マクロ入力フィールドは、水平マクロで使用する受信フィールドを表すために使用できます。

マクロ入力フィールドを作成するとき、マクロ入力フィールドの名前を定義し、フィールドルールによって使用する受信フィールドを定義します。実行時には、マクロ入力フィールドが拡張されて選択したフィールドを表します。

マクロ入力フィールドを設定するときは、次のフィールドルールを使用します。

- すべてのフィールド
- 名前付きフィールド
- テキストまたはパターンごとのフィールド

次の画像は、Q1～Q4 フィールドを含んでいる［名前付きフィールド］フィールドルールを示しています。



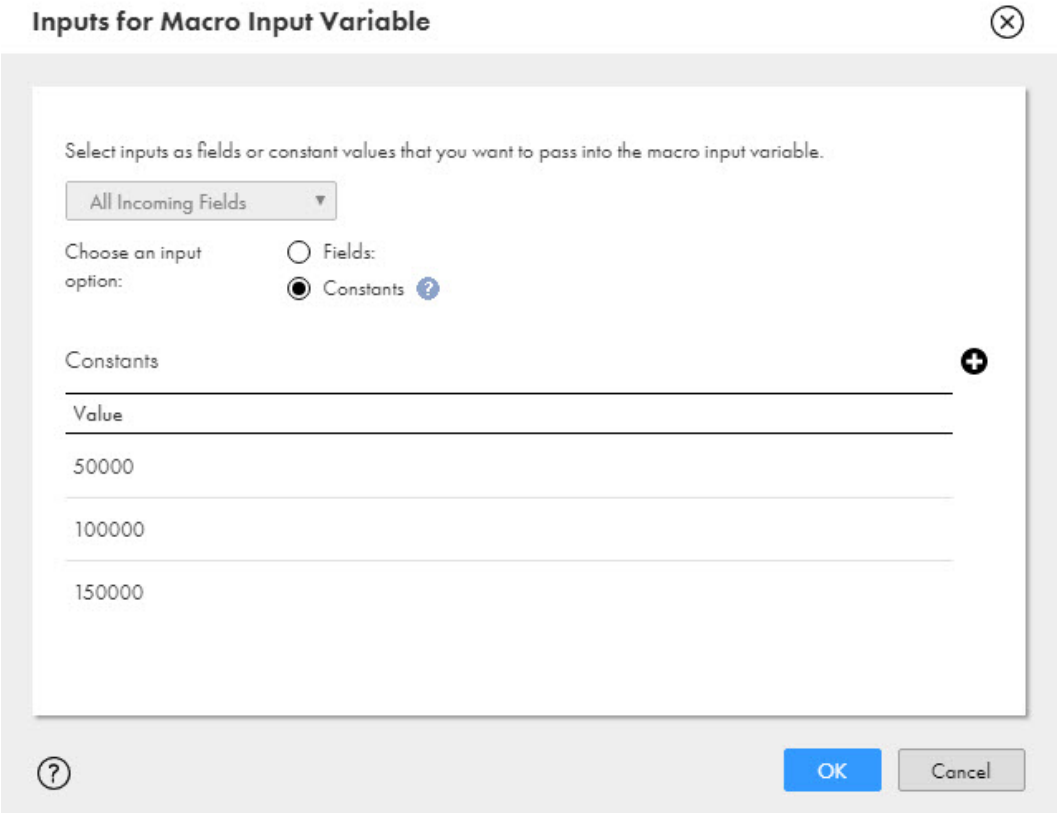
水平マクロの定数用のマクロ入力フィールド

マクロ入力フィールドは、水平マクロで使用する定数を表すために使用できます。また、複数のマクロ入力フィールドを作成して、対応する定数の複数の集合を表すこともできます。

マクロ入力フィールドを作成するとき、マクロ入力フィールドの名前を定義し、使用する定数を定義します。実行時には、マクロ入力フィールドが拡張されて定数に置き換えられ、それらの定数がリストされた順序で使用されます。

定数の複数の集合に対応する複数のマクロ入力フィールドを作成すると、タスクでは各定数の集合がリストされた順序で評価されます。

次の画像は、定数を表すマクロ入力フィールドを示しています。



実行時には、マクロ入力フィールドが拡張されて、定数が 50000、100000、150000 の順序で使用されます。

水平マクロのトランスフォーメーション出力フィールドの設定

水平マクロの式を定義したり、結果を残りのマッピングに渡したりするには、トランスフォーメーション出力フィールドを使用します。

トランスフォーメーション出力フィールドを作成するときは、フィールドの名前とデータ型を定義します。さらにマクロの式も設定します。式には、使用する水平拡張関数や任意のマクロ入力フィールドを含めます。

デフォルトのフィールドルールでは、トランスフォーメーション出力フィールドをダウンストリーム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出力フィールドを含むフィールドルールを使用して、水平マクロの結果をマッピングに渡すことができます。

水平マクロの例

給与の範囲に基づいて従業員のカテゴリを作成するには、それぞれの範囲の最小値と最大値、および対応する業務カテゴリを定義する水平マクロを作成します。

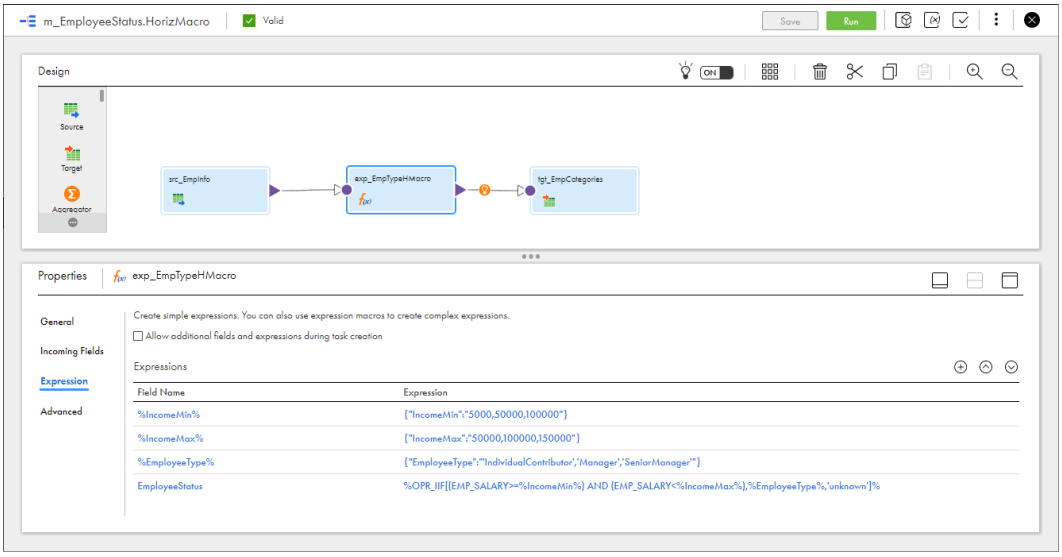
式トランスフォーメーションでは、マクロ入力フィールドで式に使用する定数を定義します。%IncomeMin%は、各給与範囲の最低値を定義し、%IncomeMax%は各給与範囲の最大値を定義します。%EmployeeType%は、各範囲に対応する業務カテゴリをリストします。

EmployeeStatus のトランスフォーメーション出力フィールドは、マッピングの結果を渡します。また、次の水平マクロ式が含まれます。

```
%OPR_IIF[ (EMP_SALARY>=%IncomeMin%) AND (EMP_SALARY<%IncomeMax%), %EmployeeType%, 'unknown' ]%
```

ターゲットトランスフォーメーションでは、デフォルトのフィールドルールによって、受信フィールドリストに EmployeeStatus トランスフォーメーション出力フィールドが組み込まれます。ターゲットフィールドのマッピングでは、EmployeeStatus がターゲットにマップされます。

次の図は、式トランスフォーメーションでの水平マクロを示しています。



水平マクロ式は、タスクを実行すると次のように拡張します。

```
IIF(Salary>=5000 AND Salary<50000), 'IndividualContributor',
  IIF (Salary>=50000 AND Salary<100000), 'Manager',
    IIF (Salary>=100000 AND Salary<150000), 'SeniorManager', 'unknown')))
```

この式は、最初の IIF 式で各マクロの入力フィールドの最初の値を使用し、それ以降もそれぞれの定数を使用しています。

混合マクロ

混合マクロは、式を垂直および水平に拡張します。混合マクロは、複数の垂直式を生成し、それらはさらに水平にも拡張されます。

混合マクロは、ビジネス要求に基づいて設定してください。垂直および水平マクロの設定ガイドラインに従って、混合マクロを作成します。

例

例えば、次の式は%OPR_IIF%水平拡張関数を使用して、%dateports%マクロ入力フィールドで表す日付フィールドの形式を「mm-dd-yyyy」の形式に変換しています。

```
%OPR_IIF[IS_DATE(%dateports%,%fromdateformat%),TO_STRING(TO_DATE(%dateports%,%fromdateformat%),'mm-dd-yyyy'),%dateports%]%
```

%fromdateformat%マクロ入力フィールドは、日付フィールドに使用される異なる日付形式の mm/dd/yy と mm/dd/yyyy を定義します。

実行時には、アプリケーションが次のように式を垂直および水平に拡張します。

```
IIF(IS_DATE(StartDate,'mm/dd/yy'),TO_STRING(TO_DATE(StartDate,'mm/dd/yy'),'mm-dd-yyyy'),
  IIF(IS_DATE(StartDate,'mm/dd/yyyy'),TO_STRING(TO_DATE(StartDate,'mm/dd/yyyy'),'mm-dd-yyyy'), StartDate))

IIF(IS_DATE(EndDate,'mm/dd/yy'),TO_STRING(To_Date(EndDate,'mm/dd/yy'),'mm-dd-yyyy'),
  IIF(IS_DATE(END _DT,'mm/dd/yyyy'),TO_STRING(TO_DATE(EndDate,'mm/dd/yyyy'),'mm-dd-yyyy'), EndDate))
```

式が垂直に拡張されて、%dateports%が表す StartDate と EndDate の式が作成されています。また、式は%fromdateformat%が表す定数を使用して受信フィールドを評価するために水平に拡張されています。

ファイルリスト

ファイルリストをフラットファイル接続のソースとして使用できます。ファイルリストは、マッピングで使用する各ソースファイルの名前とディレクトリを格納したファイルです。ファイルリストを使用すると、マッピング内の 1 つのソースオブジェクトに対して複数のソースファイルをタスクで読み取ることができます。

例えば、複数の場所からデータを収集し、これを同じマッピングを使用して処理する場合にファイルリストを使用できます。

ソーストランスフォーメーションやルックアップトランスフォーメーションなど、ソースオブジェクトを設定して、ファイルリストを読み込みます。また、各ターゲットの行にソースファイル名を書き込むこともできます。ファイルリストを使用するマッピングを実行すると、ファイルリスト内の異なるソースファイルからデータ行が読み込まれます。

ファイルリストを作成する場合には、次のルールおよびガイドラインを使用します。

- リスト内の各ファイルでは、接続で設定されているユーザー定義のコードページを使用する必要があります。
- リスト内の各ファイルは、接続で設定されたものと同じファイルプロパティを共有する必要があります。
- ファイルのパスを指定しない場合、タスクはファイルがファイルリストと同じディレクトリにあるものとみなします。
- 各パスは、Secure Agent に対してローカルなパスである必要があります。

ファイルリストを手動で作成したり、コマンドを使用してファイルリストを作成したりできます。

ファイルリストの手動作成

ファイルリストを手動で作成するには、テキストエディタでリストを作成し、テキストファイルとして保存します。このテキストファイルは Secure Agent のローカルディレクトリに格納します。

ファイルリストを作成するには、各行にファイル名を 1 つ、または各行にファイルパスとファイル名を 1 つずつ入力します。データ統合は、ファイルリストの最初のファイルからフィールド名を抽出します。

次の例に、有効な Windows ファイルリストを示します。

```
customers_canada.dat
C:\Customer_Accounts\customers_us.dat
C:\Customer_Accounts\customers_uk.dat
C:\Customer_Accounts\customers_india.dat
```

上記の例では、データ統合は、ファイルリストと同じフォルダに格納された customers_canada.dat ファイルからフィールド名を抽出しています。

ファイルリストコマンド

コマンドを使用すると、マッピングのソースファイルのリストを生成できます。有効な DOS や UNIX コマンド、バッチファイル、またはシェルスクリプトを使用できます。データ統合は、タスクの実行時にリスト内の各ファイルを読み取ります。

ソースファイルのリストが頻繁に変更される場合、または特定の条件に基づいてファイルリストを生成する場合には、ファイルリストを生成するコマンドを使用します。例えば、コマンドを使用して、ディレクトリ内のすべてのファイルまたはファイル名に基づいてファイルリストを生成できます。

コマンドを使用してファイルリストを生成する場合は、次のガイドラインに従ってください。

- バッチファイルで「/b」などのパラメータを使用する Windows コマンドを入力する必要があります。
- 各コマンド、バッチファイル、およびシェルスクリプトに、完全修飾ファイルパスを入力する必要があります。

- ファイルリストコマンドで入出力パラメータを使用することはできません。

シェルスクリプトを使用した UNIX の例

Linux マシンに保存されているパーツリストからデータを抽出する必要があります。パーツリストは、/home/dsmith/flatfile/parts ディレクトリに格納されているテキストファイルです。

次の表は、ソーストランスフォーメーションで入力したコマンドとそれに対応するシェルスクリプトの内容を示しています。

コマンド	**シェルスクリプト**（parts.sh）
/home/dsmith/flatfile/parts/parts.sh	cd /home/dsmith/flatfile/parts ls *.txt

バッチファイルを使用した Windows の例

Windows マシンに保存されている売上レコードからデータを抽出する必要があります。売上レコードファイルは C:\SalesRecords ディレクトリに保存され、命名規則 SalesRec_??-??-2017.txt を使用します。

次の表は、ソーストランスフォーメーションで入力したコマンドとそれに対応するバッチファイルの内容を示しています。

コマンド	**バッチファイル**（SalesSrc.bat）
C:\SalesRecords\SalesSrc.bat	@echo off cd C:\SalesRecords dir /b SalesRec_??-??-2017.txt

シェルスクリプトまたはバッチファイルを使用しない例

バッチファイルまたはシェルスクリプトを使用する代わりに、コマンドを使用してファイルリストを生成することもできます。例えば、次のコマンドは、source.csv という名前のファイルが含まれるファイルリストを生成します。

```
echo C:\sources\source.csv
```

コマンドサンプルファイル

コマンドを使用してファイルリストを生成する場合は、データ統合がフィールド名の抽出に使用するサンプルファイルを選択します。生成されるファイルリストにサンプルファイルが含まれていない場合、データ統合はサンプルファイルからデータを抽出しません。

生成したファイルリスト内のファイルにサンプルファイルのすべてのフィールドが含まれていない場合、データ統合は、欠落しているフィールドのレコード値を NULL に設定します。ファイルリスト内のファイルにサンプルファイルに記載されていないフィールドが含まれている場合、データ統合はこの余分なフィールドを無視します。

例えば、選択したサンプルファイルに、CustID、NameLast、および NameFirst というフィールドが含まれています。生成したファイルリスト内のあるファイルに、NameFirst フィールドが含まれていません。データ統合がファイルからデータを読み込むと、ファイル内の各レコードの名前が NULL に設定されます。

生成したファイルリスト内の別のファイルに、CustID、NameLast、NameFirst、および PhoneNo というフィールドが含まれています。サンプルファイルに PhoneNo フィールドが含まれていないため、データ統合はこのフィールドのレコードをインポートしません。電話番号をインポートする場合は、PhoneNo フィールドが含まれるサンプルファイルを選択するか、トランスフォーメーションに電話番号のフィールドを追加します。

ソーストランスフォーメーションでのファイルリストの使用

ソーストランスフォーメーションでファイルリストを使用するには、テキストファイル、バッチファイル、またはファイルリストを作成するシェルスクリプトを作成します。次に、ファイルリストが使用されるようにソーストランスフォーメーションを設定します。

1. テキストファイル、バッチファイル、またはファイルリストを作成するシェルスクリプトを作成し、Secure Agent のローカルにインストールします。
2. Mapping Designer で、ソーストランスフォーメーションを選択します。
3. **［ソース］**タブで、フラットファイル接続を選択します。
4. 作成したファイルリストを手動で使用するには、次の手順を実行します。
   a. **［ソースタイプ］**フィールドで**［ファイルリスト］**を選択します。
   b. **［オブジェクト］**フィールドで、ファイルリストを含むテキストファイルを選択します。
   c. **［フィールド］**タブで、ソーストランスフォーメーションの受信フィールドを確認します。

      データ統合は、ファイルリストの最初のファイルからソースフィールドを抽出します。ソースフィールドが正しくない場合は、フィールドを追加または削除できます。
5. コマンドで生成したファイルリストを使用するには、次の手順を実行します。
   a. **［ソースタイプ］**フィールドで、**［コマンド］**を選択します。
   b. **［サンプルオブジェクト］**フィールドで、データ統合がソースフィールドを抽出するサンプルファイルを選択します。

      いずれかのファイルを使用して、ファイルリストをサンプルファイルとして生成するか、別のファイルを選択することができます。
   c. **［コマンド］**フィールドで、使用するコマンドを入力し、`/home/dsmith/flatfile/parts/parts.sh` などのファイルリストを生成できます。
   d. **［フィールド］**タブで、ソーストランスフォーメーションの受信フィールドを確認します。

      ソースフィールドが正しくない場合は、フィールドを追加または削除したり、**［ソース］**タブをクリックして別のサンプルファイルを選択したりすることができます。
6. 必要に応じて、各ターゲット行にソースファイル名を書き込むには、**［フィールド］**タブをクリックし、**［現在処理済中のファイル名の追加］**フィールドオプションを有効にします。

   CurrentlyProcessedFileName フィールドが、フィールドテーブルに追加されます。

ルックアップトランスフォーメーションでのファイルリストの使用

ルックアップトランスフォーメーションでファイルリストを使用するには、テキストファイル、バッチファイル、またはファイルリストを作成するシェルスクリプトを作成します。次に、ファイルリストが使用され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。

1. テキストファイル、バッチファイル、またはファイルリストを作成するシェルスクリプトを作成し、Secure Agent のローカルにインストールします。
2. Mapping Designer 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。
3. **［ルックアップオブジェクト］**タブで、フラットファイル接続を選択します。
4. 作成したファイルリストを手動で使用するには、次の手順を実行します。
   a. **［ソースタイプ］**フィールドで**［ファイルリスト］**を選択します。
   b. **［ルックアップオブジェクト］**フィールドで、ファイルリストを含むテキストファイルを選択します。

c. **［戻りフィールド］**タブで、ルックアップトランスフォーメーションの受信フィールドを確認します。

データ統合は、ファイルリストの最初のファイルから戻りフィールドを抽出します。戻りフィールドが正しくない場合は、フィールドを追加または削除できます。

5. コマンドで生成したファイルリストを使用するには、次の手順を実行します。
   a. **［ソースタイプ］**フィールドで、**［コマンド］**を選択します。
   b. **［ルックアップオブジェクト］**フィールドで、データ統合が戻りフィールドを抽出するサンプルファイルを選択します。

      いずれかのファイルを使用して、ファイルリストをサンプルファイルとして生成するか、別のファイルを選択することができます。
   c. **［コマンド］**フィールドで、使用するコマンドを入力し、`/home/dsmith/flatfile/parts/parts.sh` などのファイルリストを生成できます。
   d. **［戻りフィールド］**タブで、ルックアップトランスフォーメーションの受信フィールドを確認します。

      戻りフィールドが正しくない場合は、フィールドを追加または削除したり、**［ルックアップオブジェクト］**タブをクリックして別のサンプルファイルを選択したりすることができます。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

第 2 章

ソーストランスフォーメーション

ソーストランスフォーメーションはソースからデータを抽出します。マッピングにソーストランスフォーメーションを追加する場合は、接続タイプに関連するソース接続、ソースオブジェクト、およびソースプロパティを定義します。一部の接続タイプでは、1 つのソーストランスフォーメーション内で複数のソースオブジェクトを使用できます。

ソーストランスフォーメーションを使用して、次のソースタイプからデータを読み取ることができます。

- ファイル。ソーストランスフォーメーションは、単一のソースファイルまたはファイルリストからデータを読み取ることができます。
- データベース。ソーストランスフォーメーションは、単一または複数のソーステーブルからデータを読み取ることができます。
- Web サービス。ソーストランスフォーメーションは、単一の Web サービス操作からデータを読み取ることができます。
- データ統合コネクタ。ソーストランスフォーメーションは、接続タイプに基づいて単一のソースオブジェクト、マルチグループのソースオブジェクト、または複数のソースオブジェクトからデータを読み取ることができます。

  個別のコネクタのソースに関する詳細については、オンラインヘルプの「**コネクタ**」セクションを参照してください。

1 つのマッピングで 1 つ以上のソーストランスフォーメーションを使用できます。マッピングで 2 つのソーストランスフォーメーションを使用する場合、ジョイナトランスフォーメーションを使用してデータを結合することができます。同じ構造で複数のソーストランスフォーメーションを使用する場合は、共有体トランスフォーメーションを使用してデータを単一のパイプラインにマージできます。

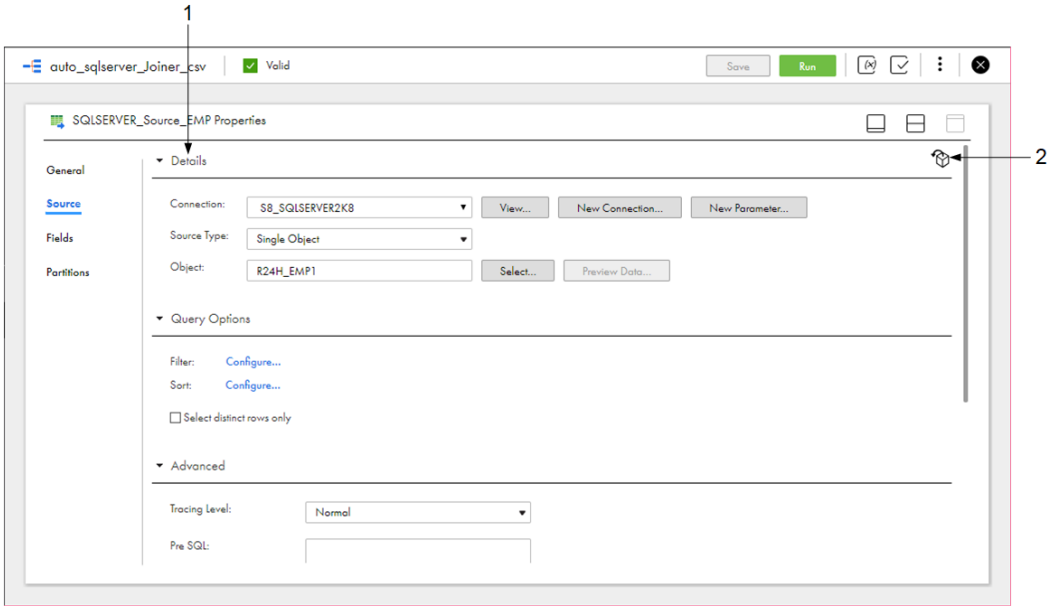
ソーストランスフォーメーションでは、表示されるソースプロパティは接続タイプによって異なります。例えば、Salesforce 接続を選択した場合、複数の関連ソースオブジェクトを使用して、Salesforce API の詳細なソースのプロパティを設定できます。対照的に、フラットファイル接続を選択する場合は、ファイルタイプを指定し、ファイル形式オプションを設定

ソースオブジェクト

［プロパティ］パネルの**［ソース］**タブで、ソーストランスフォーメーションのソースオブジェクトを選択します。

ソースオブジェクトに設定するプロパティは、接続タイプとマッピングタイプによって異なります。また、組織のライセンスによって、ソーストランスフォーメーションがマップレットの一部である場合に表示されるソースプロパティが決定されます。

次の図に、リレーショナルソースの［**ソース**］タブを示します。



1. ソース接続、ソースタイプ、ソースオブジェクトを設定するソースの詳細。
2. マッピングインベントリからソースオブジェクトを選択します。

以下の方法でソースを選択できます。

接続とソースオブジェクトを選択する。

［**詳細**］領域で、ソース接続、ソースタイプ、およびソースオブジェクトを選択します。一部のソースタイプでは、複数のソースオブジェクトを選択できます。また、新しい接続を作成することもできます。

ソースタイプは、接続タイプによって異なります。例えば、リレーショナルソースでは単一オブジェクト、複数の関連オブジェクト、または SQL クエリを選択できます。フラットファイルソースでは、単一オブジェクト、ファイルリスト、またはファイルリストコマンドを選択できます。

マッピングインベントリからソースオブジェクトを選択します。

組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されていて、［**Data Catalog**］ページでオブジェクトをマッピングに追加した場合、［**インベントリ**］パネルからソースオブジェクトを選択できます。組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されていない、またはデータカタログの検出が実行されていない場合、［**インベントリ**］パネルには何も表示されません。データカタログの検出に関する詳細については、「*マッピング*」を参照してください。

パラメータを使用する。

マッピングタスクを実行するときは、入力パラメータを使用してソース接続やソースオブジェクトを定義できます。パラメータの詳細については、「*マッピング*」を参照してください。

ファイルソース

ファイルソースには、フラットファイルと FTP/SFTP ファイルが含まれます。ファイルソースを設定する場合は、接続、ソースタイプ、および形式オプションを指定します。**［プロパティ］**パネルの**［ソース］**タブでファイルソースのプロパティを設定します。

次の表に、ファイルソースのプロパティを示します。

プロパティ	説明
接続	ソース接続の名前。
ソースタイプ	ソースタイプ。ソースタイプには、単一のオブジェクト、ファイルリスト、コマンド、またはパラメータを指定できます。
オブジェクト	ソースタイプが単一のオブジェクトの場合、このプロパティには、Customers.csv などのファイルソースを指定します。 ソースプロパティがファイルリストの場合、このプロパティには、SourceList.txt などのファイルリストを含むテキストファイルを指定します。 ソースタイプがコマンドの場合、このプロパティには、データ統合がソースフィールドをインポートするサンプルファイルを指定します。 詳細モードでは、オブジェクト名にドル記号文字$を含めることはできません。ドル記号は、パラメータ用に予約された文字です。
コマンド	ソースタイプがコマンドの場合、このプロパティには、ItemSourcesCmd.bat などのソースファイルリストを生成するコマンドを指定します。
パラメータ	ソースタイプがパラメータの場合、このプロパティには、ソースパラメータを指定します。
形式オプション	フラットファイルの形式オプション。**［形式オプション］**ダイアログボックスを開き、ファイルの形式を定義します。 区切りファイルまたは固定長ファイルのいずれかのタイプを選択できます。デフォルトは区切りファイルです。 区切りフラットファイルタイプの場合、次のファイル形式オプションを設定します。 - 区切り文字。区切り文字。カンマ、タブ文字、コロン、セミコロン、印刷不可能な制御文字、または指定したシングルバイト文字またはマルチバイト文字を使用することができます。 - 連続した区切り文字を 1 文字として扱う。1 つ以上の連続するカラム区切り文字を 1 つの文字として扱います。デフォルトでは、連続する区切り文字は NULL 値として扱われます。 - 行区切り文字。Added missing "。".改行文字。リストから改行文字を選択します。デフォルトは、改行、\012 LF（\n）です。 - テキスト修飾子。テキストを修飾する文字。 - エスケープ文字。エスケープ文字。 - フィールドラベル。タスクがフィールドラベルを生成するのか、ソースファイルからラベルをインポートするのかを指定します。 - 最初のデータ行。データの最初の行。タスクは、入力した行番号で読み取りを開始します。 タブ、スペース、または任意の印刷出力可能な特殊文字を区切り文字として使用できます。区切り文字には最大 10 文字を使用できます。区切り文字はエスケープ文字およびテキスト修飾子以外にする必要があります。 固定長フラットファイルの場合は、使用する固定長ファイル形式を選択します。リストに同じ名前の複数の固定長ファイル形式が含まれている場合は、名前に追加されたプロジェクトとフォルダの場所を使用して、使用する適切なファイル形式を指定します。固定長ファイル形式を指定していない場合は、**［新規］** > **［コンポーネント］** > **［固定長ファイル形式］**に移動して作成します。

ファイルリストとコマンドの詳細については、「ファイルリスト」 (ページ 44)を参照してください。パラメータとファイル形式の詳細については、「*マッピング*」を参照してください。

次の表に、ファイルソースの詳細プロパティを示します。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。
1000 ごとの区切り文字	1000 ごとの区切り文字。なし、カンマ、またはピリオドにすることができます。小数点記号または区切り文字と同じにすることはできません。 フィールドタイプは数値である必要があります。また、場合によってはフィールドの精度とスケールを更新する必要があります。 デフォルトは［なし］です。
小数点記号	小数点文字。カンマまたはピリオドにすることができます。1000 ごとの区切り文字、または区切り文字と同じにすることはできません。 フィールドタイプは数値である必要があります。また、場合によってはフィールドの精度とスケールを更新する必要があります。 デフォルトはピリオドです。
ソースファイルディレクトリ	フラットファイルソースの場合、ソースディレクトリの名前。 FTP ソースの場合、ソースデータのステージングに使用するローカルソースファイルディレクトリの名前とパス。 デフォルトでは、マッピングタスクはソース接続ディレクトリからソースファイルを読み取ります。 また、入力パラメータを使ってファイルディレクトリを指定することもできます。 サービスプロセス変数ディレクトリ$PMSourceFileDir を使用した場合、タスクはシステム変数への設定済みのパスにターゲットファイルを書き込みます。システム変数への設定済みのパスを見つけるには、次のディレクトリにある pmrdtm.cfg ファイルを参照してください。 `<Secure Agent installation directory>\apps\Data_Integration_Server\<Data Integration Server version>\ICS\main\bin\rdtm` また、$PMSourceFileDir 変数への設定済みのパスは、Administrator のデータ統合サーバーシステム設定の詳細にあります。
ソースファイル名	フラットファイルソースの場合、ソースファイルのファイル名、またはファイル名とパス。 また、入力パラメータを使ってファイル名を指定することもできます。 FTP ソースの場合、ソースデータのステージングに使用するローカルソースファイルの名前。
リモートファイル名	FTP ソースの場合、リモートファイルのファイル名、またはファイル名とパス。 また、入力パラメータを使ってリモートファイル名を指定することもできます。
ファイルリーダーによる文字列の NULL の切り詰め	フラットファイルソースの場合、文字列フィールド値を最初の NULL 文字から切り詰めます。 ソースファイルに NULL 文字が含まれる場合に有効にしてください。 FileRdrTreatNullCharAs カスタムプロパティを使用する場合は有効にしないでください。両方のプロパティを使用すると、データ統合におけるフラットファイルソース内の NULL 文字の取り扱い方法に関して競合する設定が作成され、タスクが失敗します。 デフォルトでは無効になっています。

データベースソース

データベースソースには、Oracle、MySQL、および Microsoft SQL Server などのリレーショナルソースが含まれています。データベースソースのソーストランスフォーメーションを設定する場合は、単一のソーステーブルまたは複数のソーステーブルを使用できます。複数のテーブルをソースとして使用する場合は、関連するテーブルを選択するか、テーブル間のリレーションシップを作成します。

データベースソースのソーストランスフォーメーションを設定するには、次のタスクを実行します。

- ソースプロパティを設定します。
- ソースに複数のテーブルが含まれている場合は、テーブルが結合されるようにソーストランスフォーメーションを設定します。関連するテーブルを結合したり、カスタムリレーションシップを指定したりできます。
- 必要に応じて、ソースデータを抽出できるようにカスタム SQL クエリを設定します。
- 必要に応じて、ソースデータをフィルタまたはソートできるようにソーストランスフォーメーションを設定します。
- テーブル名とカラム名は 74 文字を超えないようにしてください。

データベースソースのプロパティ

データベース接続、ソースタイプ、ソースオブジェクトなどのデータベースソースのプロパティを設定します。また、フィルタとソートの条件、Pre SQL と Post SQL コマンド、および出力が確定的または再現可能かどうかを指定することもできます。

次の表に、データベースソースのプロパティを示します。

プロパティ	説明
接続	ソース接続の名前。
ソースタイプ	ソースタイプ。
オブジェクト	単一ソースのソースオブジェクト。
ソースオブジェクトの追加	複数のソースのプライマリソースオブジェクト。
関連オブジェクトの追加	複数のソースの場合。選択したソースオブジェクトに関連するオブジェクトを表示します。 既存のリレーションを持つオブジェクトを選択するか、**[カスタムリレーション]** をクリックして別のオブジェクトとのカスタムリレーションを作成します。
フィルタ	レコードをフィルタリングするための条件を追加します。簡易フィルタまたは詳細フィルタを設定します。
ソート	レコードをソートするための条件を追加します。
重複しない行のみ選択	ソースから一意の行を読み取ります。SQL クエリに SELECT DISTINCT を追加します。
クエリの定義	カスタムクエリの場合。**[カスタムクエリの編集]** ダイアログボックスを表示します。有効なカスタムクエリを入力して、[OK] をクリックします。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。
Pre SQL	データをソースから読み取る前にソースに対して実行する SQL コマンド。 5000 文字までのコマンドを入力できます。
Post SQL	データをターゲットに書き込んだ後にソースに対して実行する SQL コマンド。 5000 文字までのコマンドを入力できます。
SQL クエリ	データ統合がソースからデータを読み取るために使用するデフォルトのクエリをオーバーライドする SQL クエリ。ソースデータベースでサポートされている SQL 文を入力できます。
出力が確定的かどうか	マッピングの実行間で入力データが同じ場合に、リレーショナルソースまたはトランスフォーメーション出力が変化しないケースで選択します。 このプロパティを設定するときに、パイプラインのトランスフォーメーションが常に繰り返し可能なデータを生成する場合は、Secure Agent はリカバリのためにソースデータをステージングしません。
出力は再現可能	セッションの実行間で入力データの順序が同じ場合に、リレーショナルソースまたはトランスフォーメーション出力の順序が変わらないケースで選択します。 出力が決定的で繰り返し可能な場合、Secure Agent はリカバリのためにソースデータをステージングしません。

関連オブジェクト

関連するオブジェクトを結合するようにソーストランスフォーメーションを設定できます。関連オブジェクトは、既存のリレーションまたはカスタムリレーションに基づいて結合できます。作成できるリレーションの種類は接続タイプによって異なります。

関連オブジェクトを結合するには、次のリレーションを使用します。

既存のリレーション

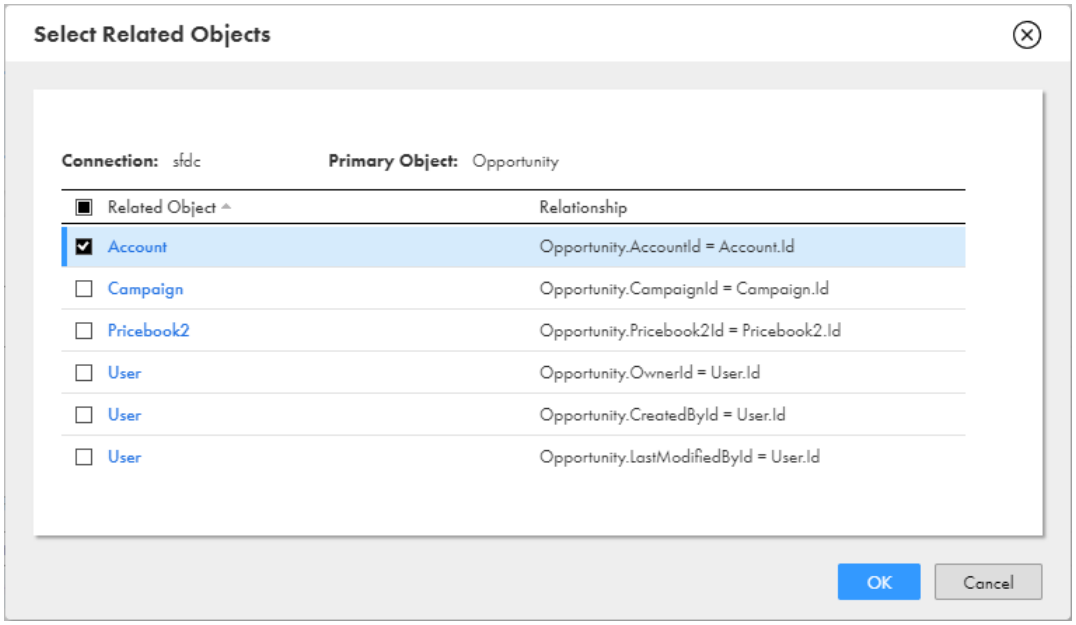
ソースシステムに定義されているリレーションを使用して関連オブジェクトを結合できます。次の接続タイプの既存のリレーションを使用すると、オブジェクトを結合できます。

- データベース
- Salesforce
- いくつかのデータ統合コネクタ

関連するオブジェクトを結合するには、プライマリオブジェクトを選択します。次に、関連するオブジェクトのリストから関連するオブジェクトを選択します。

例えば、プライマリ Salesforce ソースオブジェクトとして商談を追加した後、任意の関連オブジェクト（取引先など）を追加できます。

次の画像は、商談オブジェクトとの既存のリレーションを持つ Salesforce オブジェクトのリストを示しています。



カスタムリレーション

カスタムリレーションを作成して、同じソースシステムのオブジェクトを結合できます。カスタムリレーションを作成するには、プライマリオブジェクトを選択し、ソースシステムから別のオブジェクトを選択して、各ソースから結合条件で使用するフィールドを選択します。結合タイプと結合演算子も指定する必要があります。

以下のいずれかの結合タイプを選択できます。

内部

ノーマル結合を実行します。結合条件に一致する行が含まれます。条件に基づいて、一致しないすべての行を破棄します。

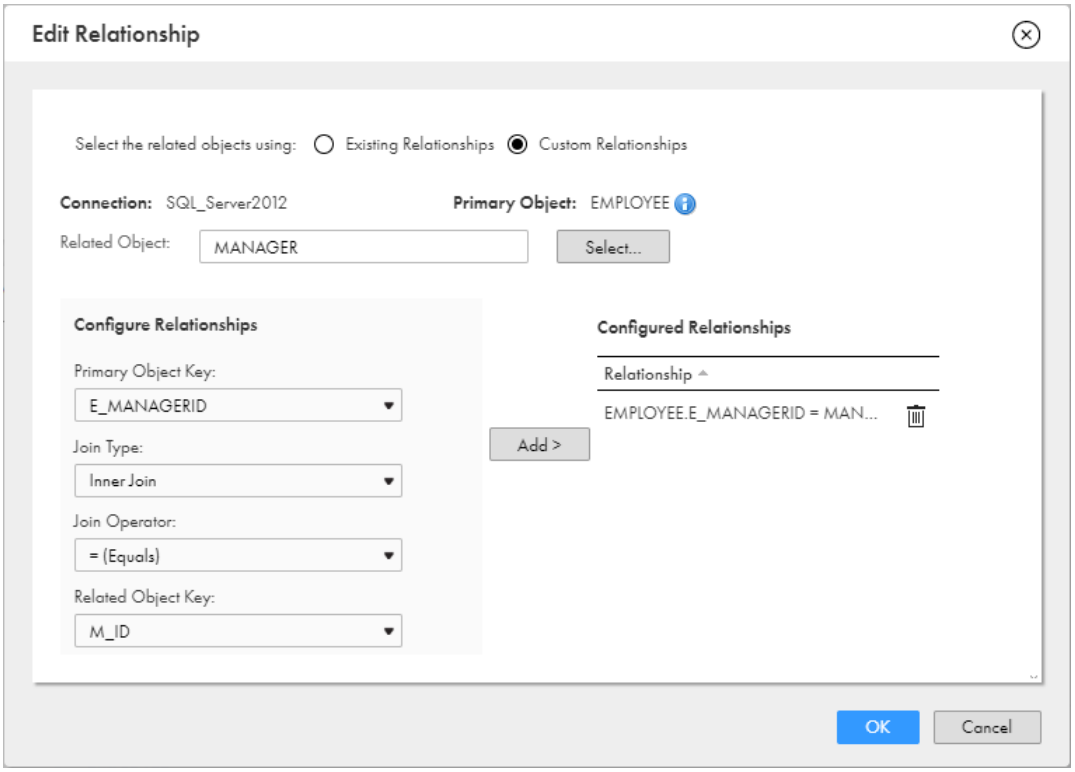
左

左外部結合を実行します。結合構文の左辺のソースのすべての行と、両方のテーブルの結合条件に一致する行が含まれます。一致しない右辺のソースの行は破棄します。

右

右外部結合を実行します。結合構文の右辺のソースのすべての行と、両方のテーブルの結合条件に一致する行が含まれます。一致しない左辺のソースの行は破棄します。

例えば、次の画像は、EMPLOYEE.E_MANAGERID フィールドと MANAGER.M_ID フィールドが一致する場合に、内部結合を使用して EMPLOYEE データベーステーブルと MANAGER データベーステーブルを結合するカスタムリレーションを示しています。



関連オブジェクトの結合

ソーストランスフォーメーションで関連オブジェクトを結合できます。ソースシステムに定義されたリレーションまたはソーストランスフォーメーションに定義されたカスタムリレーションに基づいて関連オブジェクトを結合できます。

1. **[ソース]** タブで、ソースタイプとして **[複数のオブジェクト]** を選択します。
2. **[オブジェクトおよびリレーション]** テーブルで、**[アクション]** メニューから **[ソースオブジェクトの追加]** を選択します。
3. **[ソースオブジェクトの選択]** ダイアログボックスで、ソースオブジェクトを選択します。

   [オブジェクトおよびリレーション] テーブルにソースオブジェクトが表示されます。
4. **[アクション]** メニューから **[関連オブジェクトの追加]** を選択します。
5. ソースシステムのリレーションに基づいてオブジェクトを結合するには、**[既存のリレーション]** を選択し、関連オブジェクトを選択して、**[OK]** をクリックします。
6. カスタムリレーションに基づいてオブジェクトを結合するには、**[カスタムリレーション]** を選択して次の手順を実行します。
   a. 結合で使用する **[関連オブジェクト]** を選択します。
   b. 結合で使用するプライマリオブジェクトフィールドを選択して、**[プライマリオブジェクトキー]** を設定します。
   c. 結合タイプを設定します。

      内部結合、左外部結合、または右外部結合を設定できます。

d. 結合演算子を設定します。

e. 結合で使用する関連オブジェクトフィールドを選択して、**[関連オブジェクトキー]** を設定します。

f. **[追加]** をクリックします。

リレーションシップは、**[設定済みリレーション]** テーブルに表示されます。

g. **[OK]** をクリックします。

7. 追加のソースを結合するには、プライマリソースとして動作するソースを選択して、手順 4 から 6 を繰り返します。

詳細リレーション

マッピングに含まれるソースオブジェクトが複数のソースで設定されている場合は、データベースソースの詳細リレーションを作成できます。

高度なリレーションを作成するには、**[オブジェクトおよびリレーション]** テーブルにプライマリソースオブジェクトを追加します。次に、フィールドを選択し、使用する SQL 文を記述します。ソースデータベースで有効な SQL 文を使用します。ソースからオブジェクトを追加することもできます。

カスタムリレーションを詳細リレーションに変換することもできます。これを行うには、カスタムリレーションを作成し、**[オブジェクトおよびリレーション]** テーブルの上にあるメニューから **[詳細リレーション]** を選択します。データ統合によって作成されたリレーションは編集することができます。

詳細リレーションを作成すると、ウィザードにより、定義したすべてのリレーションが編集可能な SQL 文に変換されます。

カスタムクエリ

単一オブジェクトまたは複数オブジェクトのソースオプションを使用して設定することができないデータベースソースを使用する場合、カスタムクエリを作成します。複数のテーブルの複雑な結合を実行したり、大量のソースのデータフローを入力するフィールドの数を減らしたりするために、カスタムクエリを作成できます。

カスタムクエリをソースとして使用するには、ソースタイプとして **[クエリ]** を選択し、**[クエリの定義]** をクリックします。クエリを定義する場合は、ソースデータベースに対して有効な SQL を使用します。クエリではデータベース固有の関数を使用できます。

また、カスタムクエリをルックアップソースとして使用することもできます。ルックアップトランスフォーメーションでカスタムクエリを使用する方法については、「カスタムクエリ」 (ページ 281)を参照してください。

カスタムクエリを作成する場合は、使用するソースカラムを選択するために SQL の SELECT 文を入力します。データ統合は、SQL 文を使用してソースカラムの情報を取得します。カスタムクエリを保存する前に、各カラムのデータ型、精度、スケールを編集できます。

例えば、次の SQL 文を使用して 2016 年以降のトランザクションが含まれる TRANSACTIONS テーブルに基づいてカスタムクエリを作成することもできます。

```
SELECT TRANSACTION_ID, TRANSACTION_TOTAL, TRANSACTION_TIMESTAMP from dbo.TRANSACTIONS WHERE
TRANSACTION_TIMESTAMP>'0:0:0:0 01/01/2016'
```

データ統合は、カスタムクエリのカラム名が一意であることを確認します。SQL 文で重複するカラム名が返されると、データ統合は、重複するカラム名に対して次のように数値を追加します。

```
<column_name><number>
```

保存されたマッピングでカスタムクエリを変更すると、設計時にデータ統合では、変更済みのクエリを使用してフィールドメタデータがメタデータに置き換えられます。通常、これは適切な動作です。ただし、マッピングでリレーショナルソースを使用しており、元のメタデータを保持する場合は、**[既存のフィールドメタデータを保持]** オプションを使用します。このオプションを使用すると、設計時にデータ統合では、フィールドメタ

データが更新されません。データ統合は、実行時に既存のフィールドを変更済みのクエリのフィールドにマッピングします。マッピングできないフィールドがある場合は、実行時に失敗します。

ヒント: カスタムクエリを作成する前に、ソースデータベースで使用する SQL 文をテストします。データ統合では、無効な SQL 文に特定のエラーメッセージが表示されることはありません。

ソースのフィルタリングとソート

データがデータフローに入る前に、データのフィルタリングまたはソートを行えるようにソーストランスフォーメーションを設定できます。ソースクエリオプションを使用してソースデータをフィルタリングまたはソートできます。

ソーストランスフォーメーションの［**ソース**］タブでクエリオプションを設定します。［**クエリオプション**］セクションを展開し、フィルタとソートの条件を設定します。

次のソースクエリオプションを使用することができます。

フィルタ

ソースデータをフィルタリングして、データフローに入力されるソースデータの量を制限します。以下のタイプのフィルタを作成できます。

- パラメータ化なし。ソースフィールドを選択し、フィルタに使用する演算子と値を設定します。複数のフィルタが設定されている場合、タスクは、フィルタ間に AND 演算子を使用して、リストされた順序でフィルタ式を適用します。
- 完全にパラメータ化。フィルタ式にパラメータを使用して、タスクのフィルタ式を定義します。
- 詳細。AND、OR、またはネストされた条件を使用した複合式を作成します。入力する式は、ソースからレコードを取得するために使用するクエリ内の WHERE 句になります。式では、ソースフィールド、入力および入出力パラメータ、またはシステム変数を使用できます。
  例えば、いずれかのフィールドに入力パラメータを使用し、タスクの実行時にそのパラメータを選択できます。同じパラメータを式トランスフォーメーションで再利用してフィールド式を作成したり、ターゲットトランスフォーメーションで再利用したりすることもできます。または、式の入出力パラメータを使用して、最後の実行以降に更新された行を取得できます。

  システム変数の詳細については、「*関数リファレンス*」を参照してください。パラメータの詳細については、「*マッピング*」を参照してください。

  パラメータ化されていない簡易データフィルタを詳細データフィルタに変換することはできますが、詳細データフィルタを簡易データフィルタに変換することはできません。

ソート

ソースデータをソートして、ソート済みのデータをマッピングに提供できます。例えば、ソート済みのデータを使用するアグリゲータトランスフォーメーションにソート済みのデータを提供すれば、タスクのパフォーマンスを向上できます。

データをソートする際には、1 つ以上のソースフィールドを選択します。複数のソースフィールドを選択すると、一覧表示に挙げられた順にフィールドがソートされます。

各フィールドのデータが昇順にソートされます。降順でソートする場合は、ソータートランスフォーメーションを使用できます。

ソートフィールドのパラメータを使用して、タスクのソートフィールドを定義できます。

Web サービスソース

ソーストランスフォーメーションに Workday v2 などの Web サービスコネクタを使用できます。

通常、Web サービスからのデータは階層構造です。例えば、Workday v2 ソース接続を使用する場合、データは階層構造の XML として渡されます。

ソーストランスフォーメーションの Web サービス接続を選択するときに、次の手順を実行してトランスフォーメーションを設定します。

1. Web サービス操作を選択します。
2. 要求メッセージをカスタマイズしてデータフローから不要なデータを取り除きます。
3. 階層データ構造をリレーショナル構造にマッピングします。

例えば、リレーショナルデータベースターゲットを使用したマッピングに Workday のワーカー情報を含めるとします。ソーストランスフォーメーションを作成し、Workday 接続を選択します。Get_Workers 操作を選択します。この操作は、定義された XML 構造でワーカーデータを取得します。名前と連絡先情報のみがデータに入力されるように詳細フィルタを定義します。ワーカーデータのリレーショナル構造を定義してから、フィールドをターゲットデータベースのフィールドにマッピングします。

ソースの Web サービス操作

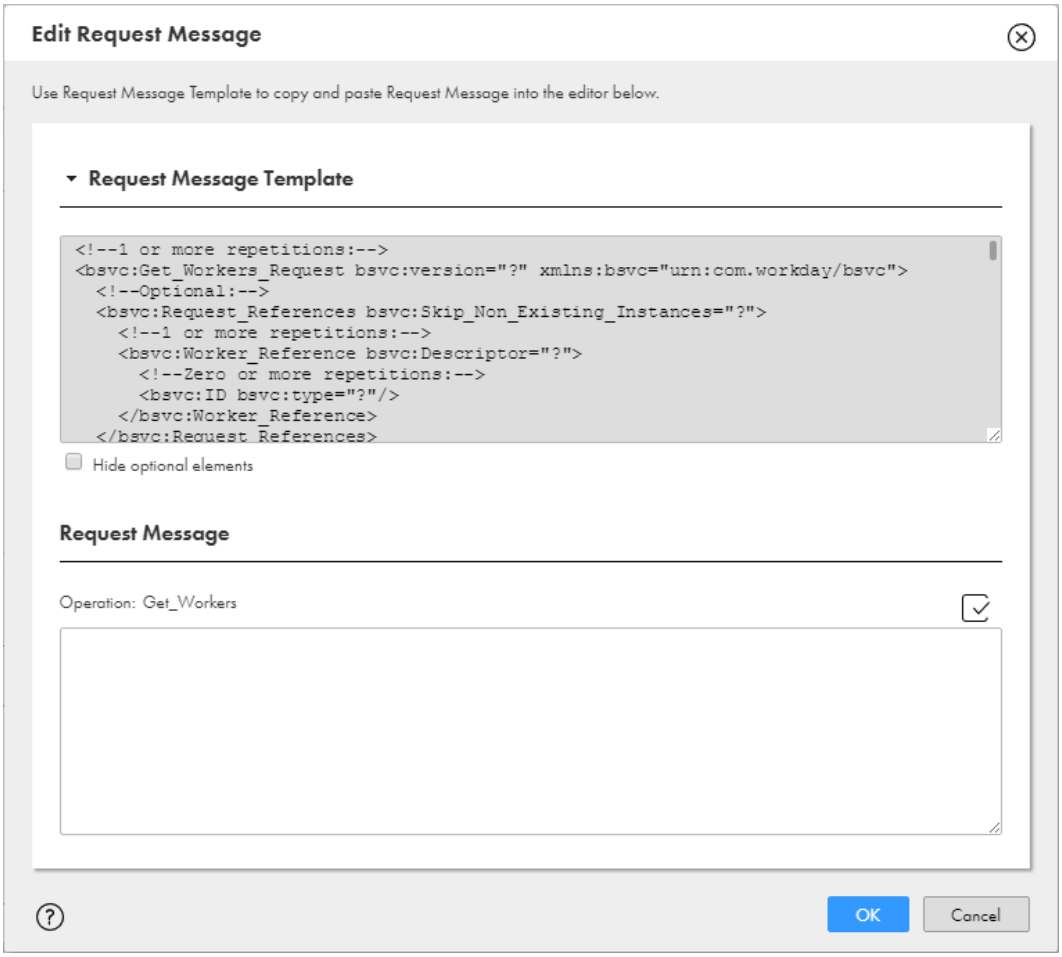
操作によって、Web サービス接続から渡すデータセットが決定され、階層データ構造が定義されます。

Web サービス接続のソースのプロパティを定義するときに、Web サービス操作を選択します。使用可能な操作は、接続によって決定されます。例えば、Workday 接続の場合、Get_Workers が使用可能な操作です。

要求メッセージ

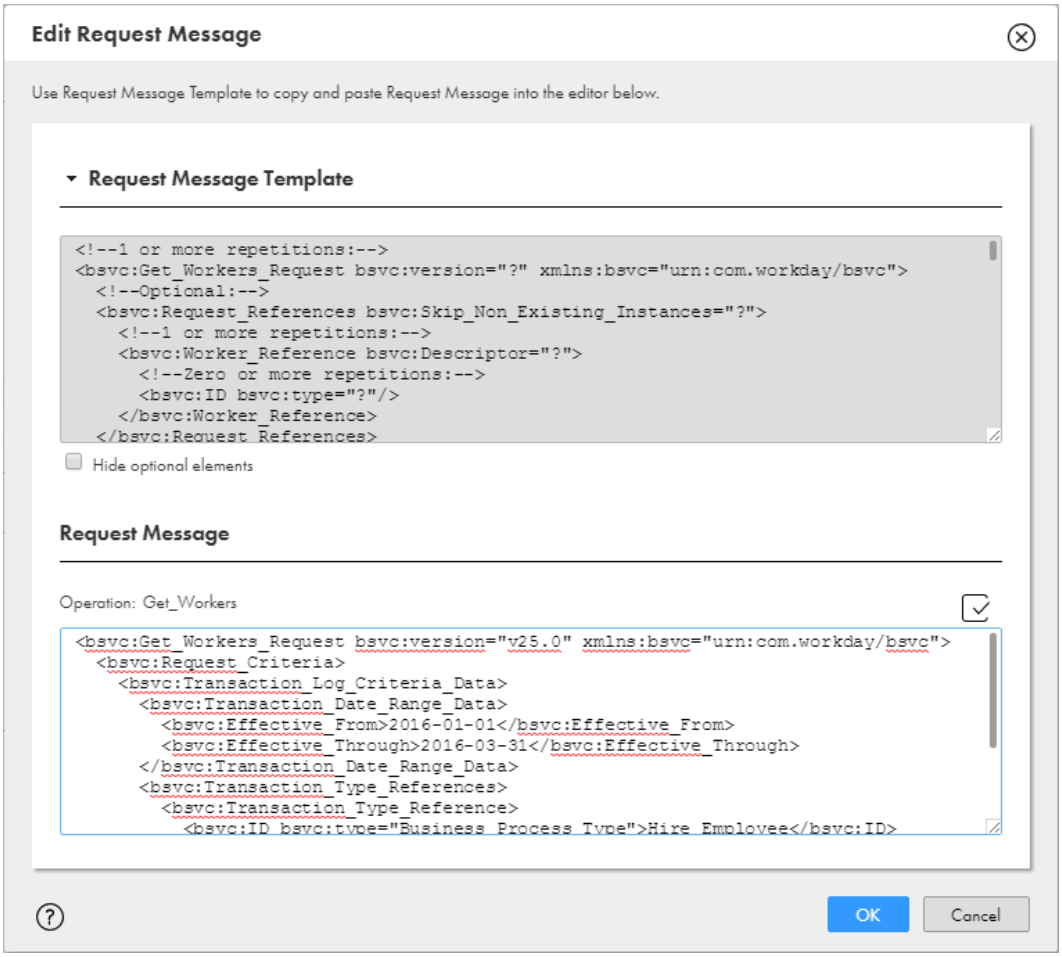
Web サービス操作には、マッピングでは使用しないデータが含まれていることがよくあります。操作に対してデフォルトの要求メッセージを使用するか、要求メッセージをカスタマイズしてデータフローを入力するデータを指定できます。

要求メッセージは XML 形式です。要求メッセージに必要な書式設定が含まれるテンプレートを使用してカスタマイズできます。要求メッセージテンプレートには、選択した操作のコンテンツが表示されます。



テンプレートをコピーして要求メッセージエディタペインに貼り付けてから、メッセージを変更します。

例えば、2016 年 1 月 1 日から 2016 年 3 月 31 日に行われたトランザクションを含める場合、次の図に示されるように要求メッセージに Effective_From と Effective_Through の値を入力します。



要求メッセージは、入出力パラメータを使用してパラメータ化できます。例えば、メッセージで特定の Effective_From と Effective_Through の日付を使用する代わりに、$$Effective_From と$ $Effective_Through のパラメータを使用できます。入出力パラメータは、要求メッセージで使用する前に［パラメータ］パネルで作成する必要があります。

入出力パラメータの詳細については、「*マッピング*」の「パラメータ」セクションを参照してください。

要求メッセージには必ず整形式の XML を使用してください。メッセージを検証して、XML が操作に必要な構造と一致することを確認できます。

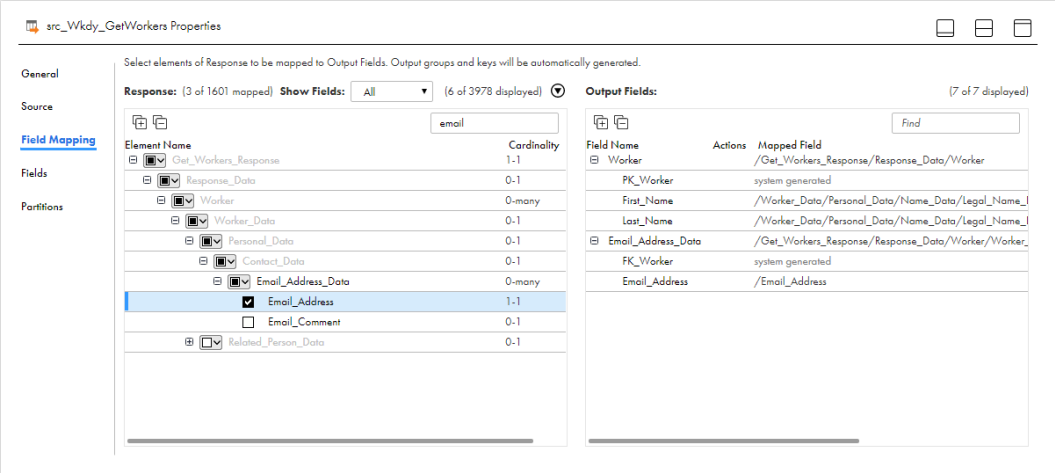
Web サービスソースのフィールドマッピング

応答フィールドを出力グループと出力フィールドのリレーショナル構造にマッピングできます。

［フィールドマッピング］タブに表示される応答には、ソースから取得するデータの階層構造が表示されます。

［応答］領域のフィールドを選択すると、生成されたプライマリキーと外部キーとともにそのフィールドがリレーショナル構造の［出力フィールド］領域に表示されます。例えば、［応答フィールド］領域で First_Name と

Last_Name を選択し、階層内の異なる親にある Email_Address を選択します。次の画像のように、［出力フィールド］領域の構造は割り当てられたプライマリキーと外部キーのあるリレーショナル構造になっています。



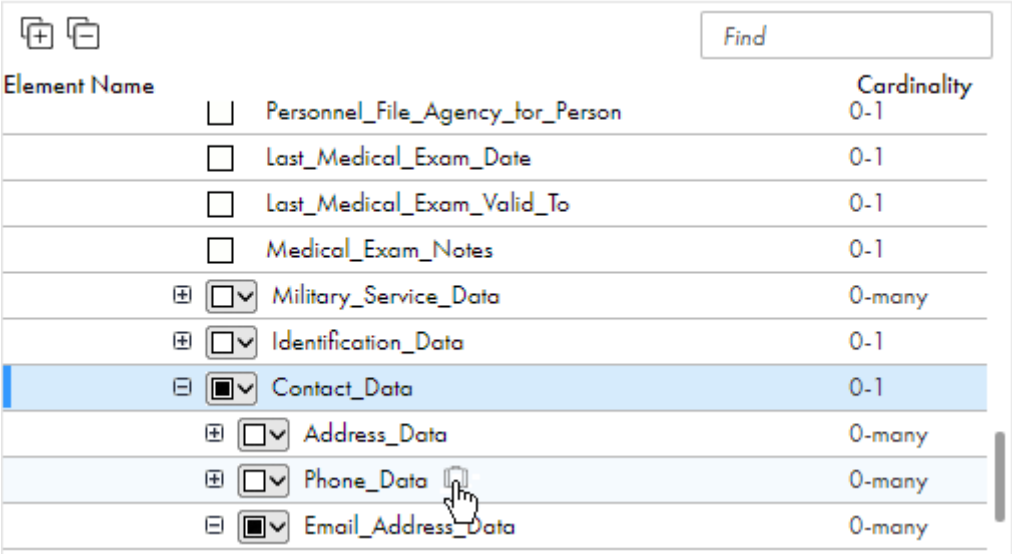
応答フィールドをリレーショナル構造にマッピングする場合、カーディナリティを考慮してください。カーディナリティにより、XML 構造の特定のポイントにおけるフィールドまたはグループの出現回数の制約が適用されます。カーディナリティが 0-多の場合、フィールドまたはグループの出現回数が 0 回以上になります。カーディナリティが 1-1 の場合、フィールドまたはグループが必須で、その出現回数は 1 回のみになります。

カーディナリティが 0-1 または 1-1 のフィールドをマッピングする場合、カーディナリティが 0-多の最初の親ノードもマッピングされます。カーディナリティが 0-多の親グループが存在しない場合、システムによって作成されます。例えば、カーディナリティが 0-1 の Email_Comment をマッピングすると、カーディナリティが 0-多の Email_Address_Data グループが自動的にマッピングされます。

パック済みフィールド

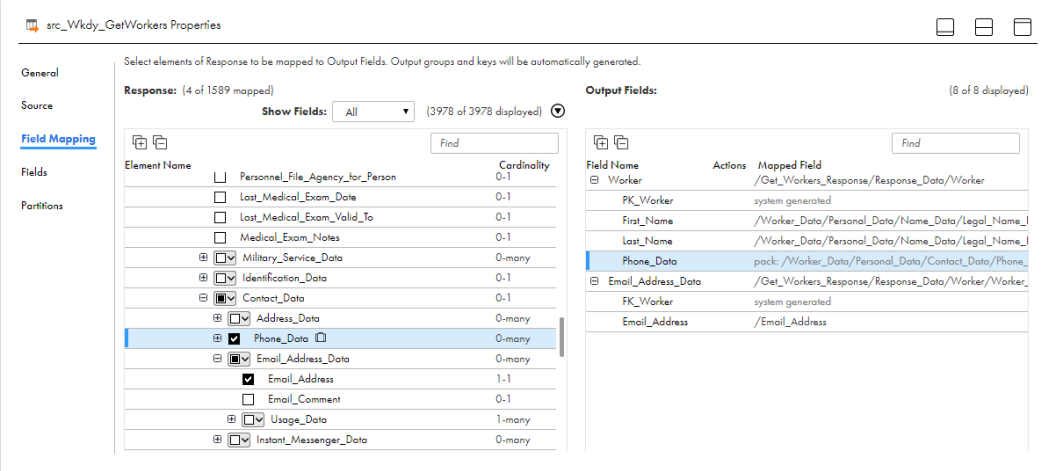
フィールドをパックして、要求メッセージの出力グループの数を削減できます。フィールドマッピングを設定するときにフィールドをパック対象としてマークできます。マッピングタスクを実行すると、タスクによって要素とその子が 1 つの XML 文字列にパックされます。

フィールドは、すでにパック対象としてマークされているソースから取得される可能性もあります。［パック］アイコンが、パック対象としてマークされた要素の横に表示されます。フィールドをパックするには、次の画像のように［パック］アイコンをクリックします。

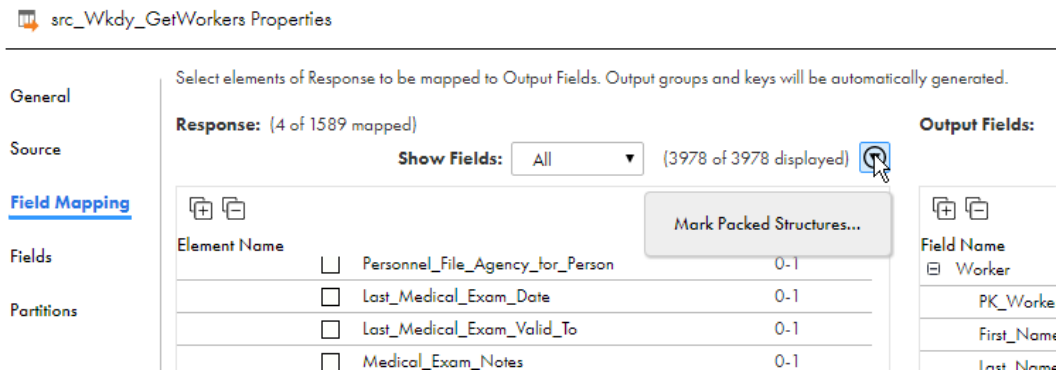


パック済みフィールドの子フィールドはパックできません。

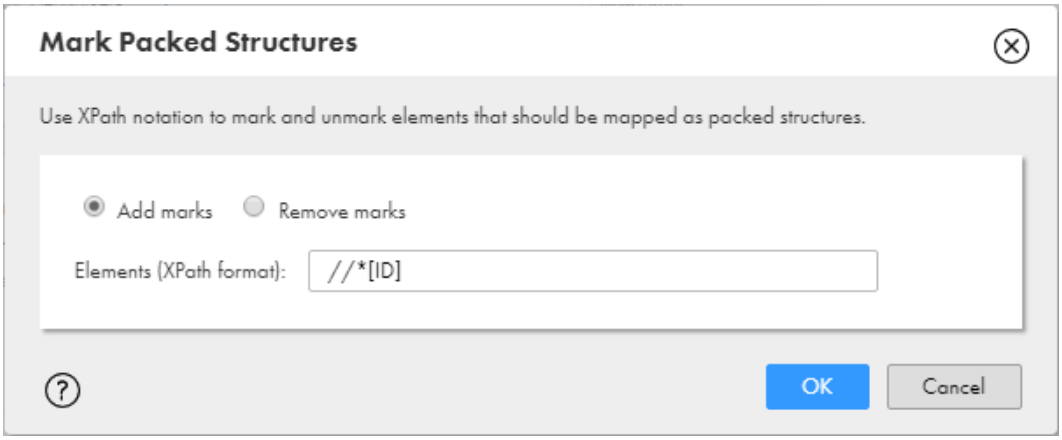
次の画像では、Phone_Data 要素がパックされることがわかります。



XPath 式を使用して、複数のフィールドにパックまたはパック解除のマークを付けることができます。次の画像のように、［応答フィールド］領域で矢印をクリックし、［パック済み構造のマーク］を選択します。



次の画像では、子としての ID を持つすべてのフィールドがパック対象としてマークされています。



パーティション

組織にパーティション化のライセンスがある場合、パーティションを使用してマッピングタスクのパフォーマンスを最適化できます。

マッピングタスクで大規模なデータセットが処理される場合、または、複雑な計算を実行するトランスフォーメーションが含まれる場合は、タスクの処理に時間がかかる場合があります。複数のパーティションを使用すると、マッピングタスクはデータを複数のパーティションに分割し、それらのパーティションを同時に処理します。これにより、パフォーマンスを最適化できます。すべてのソースタイプがパーティション化をサポートしているわけではありません。

Mapping Designer でソーストランスフォーメーションを設定する際に、パーティション化を有効にします。ソーストランスフォーメーションにパーティションを設定すると、マッピング全体でパーティション化されます。

ソースのパーティション化を有効にするには、**[パーティション化]** タブでパーティション化方法を選択します。選択できるパーティション化方法は、ソースタイプによって異なります。さまざまなタイプのソースのパーティション化の詳細については、目的のコネクタのヘルプを参照してください。

ソースタイプに応じて、次のいずれかのパーティション化方法を選択できます。

なし

マッピングタスクが、単一のパーティションですべてのデータを処理します。これがデフォルトのオプションです。

固定

マッピングタスクが、指定したパーティション数に基づいてデータの行を分散します。最大で 64 のパーティションを指定できます。

この方法は、フラットファイルソースなどの、キー範囲によるパーティション化ができないソースタイプや、キー範囲のパーティション化をサポートしないトランスフォーメーションがマッピングに含まれる場合に使用します。

マッピングで渡されるレコードの数を考慮して、マッピングに適したパーティションの数を決定してください。レコードが少ない場合、パーティション化による利点はあまりありません。

マッピングに複数のソースが含まれる場合は、ソースごとに同じ数のパーティションを指定します。

キーの範囲

マッピングタスクが、パーティションキーとして定義したフィールドに基づいてデータの行を分散します。ソース内の 1 つのフィールドをパーティションキーとして選択してから、パーティションキーの値の範囲を定義します。

この方法は、表形式のソースに使用できます（リレーショナル、Google BigQuery、および JDBC V2 のソースなど）。

キー範囲には、次のデータ型を使用できます。

- 文字列。
- すべての数値データ型。ただし、キー範囲の値には少数は使用できません。
- 日付/時刻タイプ。`MM/DD/YYYY HH24:MI:SS` の形式を使用します。

マッピングに複数のソースが含まれる場合は、ソースごとに同じ数のキー範囲を使用します。

パススルー

マッピングタスクが、パーティション間で行を再分散せずにデータを処理します。単一のパーティション内のすべての行が、そのパーティション内に留まります。パフォーマンスを向上させるために新しいパー

ティションを作成して、パーティション間のデータの分散を変更しない場合に、パススルーパーティション化を選択します。

この方法は、Amazon S3、Netezza、および Teradata などのソースに使用できます。

動的

マッピングタスクが、ソースサイズに基づいて、実行時に作成するパーティションの最適な数を決定します。

次のような状況では、マッピングをパーティション化できません。

- パラメータ化されたソースまたはソースクエリがマッピングで使用されている場合。
- マッピングに Web サービスまたは階層パーサートランスフォーメーションが含まれる場合。
- マッピングにカスタムリレーションまたは詳細リレーションを使用する複数のソースが含まれる場合。

パーティションを設定したら、Mapping Designer でマッピングを保存および実行して、パーティション設定を検証してください。

パーティション化に関するルールおよびガイドライン

マッピングをパーティション化する前に、次の規則とガイドラインに注意してください。

- 予期しない結果になることを避けるために、マッピング内のトランスフォーメーションのタイプとその順序を考慮します。マッピングタスクがパーティション化されたデータを処理する際にデータの整合性を維持できる場合は、マッピングをパーティション化できます。
- フラットファイルのパーティション化で最適なセッションパフォーマンスを得るには、大きいソースファイルを使用します。入力データの量が少ない場合は、負荷が分散されない可能性があります。
- パーティション化したマッピングにシーケンスジェネレータトランスフォーメーションが含まれる場合は、シーケンスジェネレータトランスフォーメーションにキャッシュを設定したことを確認します。キャッシュが設定されていないと、タスクが各パーティションに対して生成するシーケンス番号が連番でなくなります。
- ノーマライザトランスフォーメーションおよびシーケンスジェネレータ トランスフォーメーションで生成されるシーケンス番号は、パーティション化されたソースの場合は連番ではないことがありますが、一意です。
- パーティション化した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る場合、タスクはパーティションごとに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は、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またはアグリゲータトランスフォーメーションの前に配置する必要があります。
- キー範囲の値には、入出力パラメータを使用できません。
- マッピングに 9 個以上パーティションが含まれると、マッピングタスクのパフォーマンスが低下する場合があります。パフォーマンスを向上させるには、マッピングタスクの詳細プロパティ、［バッファブロックサイズ］および［DTM バッファサイズ］を設定します。
- Linux で、ターゲットテーブル名に Unicode 文字が含まれる場合、マルチバイトデータをサポートするためにデフォルトのロケールを UTF-8 に設定する必要があります。デフォルトのロケールを UTF-8 に設定する場合は、次の例を参照してください。

  - bash 系 UNIX シェルの場合:

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

  - csh 系 UNIX シェルの場合:

    ```
    setenv LC_ALL en_US.UTF-8
    ```

- 詳細モードのマッピングでソースにキー範囲によるパーティション化を使用する場合、次の条件がすべて当てはまるときは、マッピングが失敗します。
  - データ統合サーバーがソースを処理している。
  - 詳細クラスタがミッドストリームトランスフォーメーションを処理している。
  - 詳細クラスタが Google Cloud または Microsoft Azure 環境で実行される。

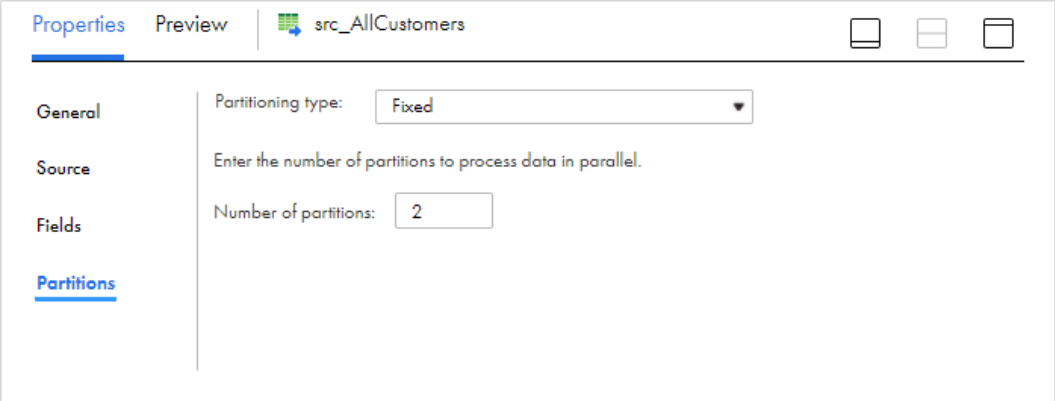
パーティション化例

次の例は、マッピングでパーティション化を設定する方法を示します。

フラットファイルソースを使用したパーティション化

1GB の大容量フラットファイルソースを使用するマッピングタスクがあるとします。パフォーマンスを最適化するために、ソーストランスフォーメーション内に 2 つのパーティションを指定します。

次の図に示すように、ソーストランスフォーメーションの **[パーティション化]** タブで、固定パーティション化を選択してパーティションの数を入力します。

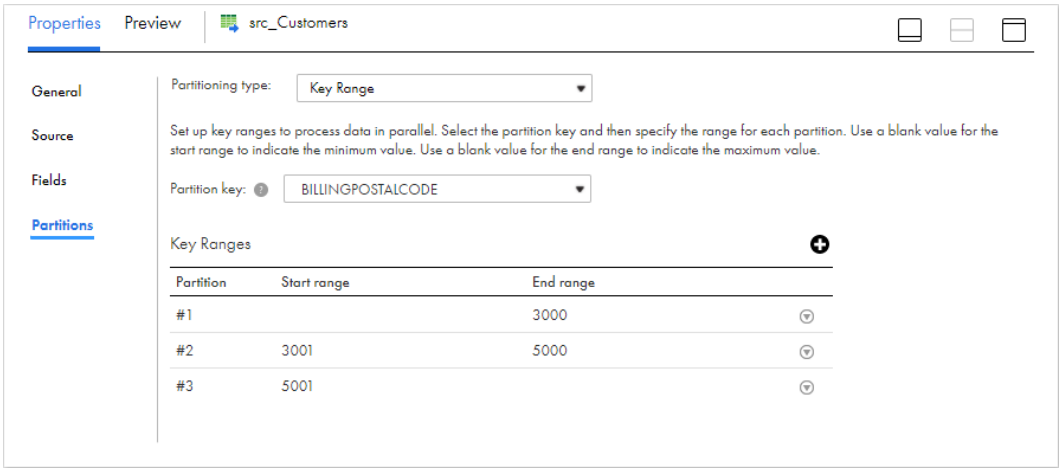


リレーショナルデータベースソースを使用したキー範囲パーティション化

顧客の名前、住所、および購入履歴がリレーショナルデータベースソースに格納されているとします。郵便番号に基づいて、次の範囲を使用してソースデータを 3 つのパーティションに分けることにしました。

- 1 つ目のパーティション: 最小値-30000
- 2 つ目のパーティション: 30001-50000
- 3 つ目のパーティション: 50001-最大値

ソーストランスフォーメーションの**［パーティション化］**タブで、キー範囲パーティション化を選択し、パーティションキーとして BILLINGPOSTALCODE フィールドを選択します。次の図に示すように、3 つのキー範囲を追加して 3 つのパーティションを作成します。



1 つ目のパーティションでは、開始の値を空白にして最小値を設定します。最後のパーティションでは、終了値を空白にして最大値を設定します。

これらの値を使用して、郵便番号が 0 から 30000 までのレコードは 1 つ目のパーティション、郵便番号が 30001 から 50000 までのレコードは 2 つ目のパーティション、郵便番号が 50001 以降のレコードは 3 つ目のパーティションで処理されます。

設定したマッピングを保存して実行し、パーティションを検証します。

詳細モードでの階層データの読み取り

詳細モードでソーストランスフォーメーションを使用して、Avro ファイル、JSON ファイル、Parquet ファイルなどの複合ファイルから階層データを読み取ることができます。詳細モードは、データを配列、マップ、または構造体として表します。

階層フィールドをパススルーフィールドとして使用して、ある複合ファイル形式のデータを別の複合ファイル形式に変換できます。例えば、階層データを Avro ソースから読み取り、そのデータを JSON ターゲットに書き込むことができます。また、階層フィールドとその子フィールドをダウンストリームトランスフォーメーションの式および条件で使用することもできます。子フィールドへのアクセスの詳細については、*関数リファレンス*の説明を参照してください。

階層フィールドは次のトランスフォーメーションに渡すことができます。

- ターゲット
- アグリゲータ
- 式
- フィルタ
- 階層プロセッサ
- ジョイナ
- ランク
- ルータ

- シーケンスジェネレータ
- ソーター

階層データを読み取るためのルールとガイドライン

階層データを読み取る場合は、次のガイドラインを考慮してください。

- 階層データを読み取るには、Amazon S3 V2 接続または Azure Data Lake Storage Gen2 接続を使用する必要があります。詳細については、該当するコネクタのヘルプを参照してください。
- XML ソースからデータを読み取るには、ソーストランスフォーメーションでインテリジェント構造モデルを使用します。インテリジェント構造モデルの詳細については、「*コンポーネント*」を参照してください。
- ソース接続またはソースオブジェクトのパラメータを使用することはできません。
- 階層フィールドに decimal データ型の子フィールドが含まれている場合、マッピングは低精度を使用して実行されます。
- トランスフォーメーションは、データの最初の行の値に基づいて精度とスケールを設定します。この最初の行は、行 0 と呼ばれることもあります。
- データの切り捨てを回避するには、データの最初の行の精度とスケールを上げます。また、最初の行に null 値が含まれていないことを確認します。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

ソースフィールド

データフローで使用するソースフィールドを設定できます。**[プロパティ]** パネルの **[フィールド]** タブでソースフィールドを設定します。

設定オプションは、接続タイプに応じて異なります。ほとんどの接続タイプでは、ソースフィールドの追加と削除、フィールドの表示方法の設定、フィールドメタデータの編集、およびソースオブジェクトからの元のフィールドの復元を行うことができます。一部の接続タイプでは、ソースオブジェクトからのみ、元のフィールドを復元できます。ソースフィールドの設定の詳細については、該当するコネクタのヘルプを参照してください。

ソースフィールドは、次の方法で設定できます。

各行にソースファイル名を追加します。

ファイルリストをソースとして使用しており、各行のソースを識別する場合は、ソースファイル名をフィールドリストに追加します。この情報をターゲットテーブルに渡すことができます。

各行にソースファイル名を追加するには、**[現在処理中のファイル名フィールドを追加する]** オプションを有効にします。**[現在処理中のファイル名フィールドを追加する]** オプションがファイルソースについて表示されます。

このオプションを有効または無効にすると、データ統合によって CurrentlyProcessedFileName フィールドが追加または削除されますが、このフィールドはソースオブジェクトと同期されません。ソースオブジェクトと同期するには、**[更新]** アイコンをクリックします。すべてのフィールドの同期、新しいフィールドのみの同期、または同期のスキップを行うことができます。

実行時に既存のフィールドを保持

マッピングの保存後にフィールドメタデータが変更された場合、データ統合では、マッピングの実行時に更新されたフィールドメタデータが使用されます。通常、これは適切な動作です。ただし、マッピングでネイティブフラットファイル接続を使用しており、設計時に使用したメタデータを保持する場合は、**[実行時に既存フィールドを保持]** オプションを有効にします。このオプションを有効にすると、データ統合のマッピングタスクでは、マッピングの作成時に使用したフィールドメタデータが使用されます。

フィールドを追加および削除します。

マッピングソースにフィールドを追加できます。リストに表示されていないソースオブジェクトからフィールドを取得するためのフィールドを追加します。フィールドを追加するには、**[フィールドの追加]** をクリックして、フィールド名、タイプ、精度、およびスケールを入力します。

また、マッピングで使用しないフィールドを削除することもできます。フィールドを削除するには、削除するフィールドを選択して、**[削除]** をクリックします。

ソート順を変更します。

ソースフィールドは、ネイティブ順、昇順、または降順に表示できます。ソート順を変更するには、**[ソート]** をクリックして、適切なソート順を選択します。

技術フィールド名またはラベルを使用します。

フィールド名は、ラベルまたは技術フィールド名で表示できます。

フィールド名の表示オプションを変更するには、**[オプション]** > **[フィールドの技術名を使用]** または **[オプション]** > **[ラベルを使用]** を選択します。

フィールドメタデータを編集します。

フィールドのメタデータを編集できます。誤って推測された情報を変更するためにメタデータを編集することがあります。メタデータを編集する場合は、名前、ネイティブタイプ、ネイティブ精度、およびネイティブスケールを変更できます（データ型に該当する場合）。一部のソースタイプでは、**[タイプ]** カラムでトランスフォーメーションデータタイプを変更することもできます。

1 つまたは複数のフィールドの名前またはメタデータを編集するには、**[オプション]** > **[メタデータの編集]** をクリックします。メタデータを編集する場合は、ラベルまたは技術フィールド名でネイティブ名を表示することもできます。ネイティブ名の表示オプションを変更するには、**[オプション]** > **[フィールドの技術名を表示]** または **[オプション]** > **[ラベルを表示]** を選択します。

フィールドのメタデータを変更する場合は、タスクの実行時にエラーが発生する可能性のある変更を行わないようにします。例えば、フィールドのネイティブ精度やネイティブスケールを増やしても、通常エラーになることはありません。一方、フィールドの精度を減らすと、データが切り捨てられる可能性があります。

ソースオブジェクトから元のフィールドを復元します。

ソースオブジェクトから元のフィールドを復元するには、**[同期]** オプションを有効にします。フィールドを同期すると、データ統合は削除されたソースフィールドを復元し、ソースに新しいフィールドを追加します。データ統合は、追加されたフィールドのうち、ソースオブジェクトで対応するフィールドがないものを削除します。

データ統合では、すべてのフィールドを同期したか新しいフィールドのみを同期したかに基づいて、既存のソースフィールドのメタデータが更新されます。すべてのフィールドを同期した場合、データ統合では、編集したフィールドメタデータがソースオブジェクトのフィールドメタデータで置き換えられます。新し

いフィールドのみを同期した場合、データローダーは、既存のソースフィールドのメタデータを保持します。データ統合は、**[名前]** フィールドに加えられた変更は元に戻しません。

複合ファイルソースのネイティブデータ型の編集

データ統合は、複合ファイルソースのネイティブデータ型を、詳細クラスタとデータ統合サーバーではそれぞれ異なる方法で処理します。

詳細クラスタでは、array、map、struct などの階層データ型には、ネイティブタイプが割り当てられます。例えば、Amazon S3 ソースの map フィールドには、ネイティブデータ型「map (string_integer)」が設定されることがあります。array、map、struct フィールドのメタデータは編集できません。

データ統合サーバーでは、データ統合は複合階層データ型を最大 4,000 文字の精度でネイティブ文字列データ型にフラット化します。ネイティブデータ型はコネクタによるものと、データ統合がソースデータを読み取るときに使用するパーサーによるものとがあります。パーサーデータ型には形式タイプのプレフィックスが付きます。例えば、Avro 形式の Amazon S3 ソースでは、パーサーによる map フィールドにはネイティブデータ型 avro_string が設定されます。コネクタフィールドとパーサーフィールドのネイティブデータ型は変更できます。

ネイティブデータ型を変更するには、ソースのメタデータを編集し、**[ネイティブタイプ]** カラムで適切なデータ型を選択します。

ネイティブデータ型を変更するときは、非パーサーデータ型をパーサーデータ型に変更できません。例えば、Amazon S3 ソースの場合、データ統合は FileName フィールドのネイティブデータ型を string に設定します。ネイティブデータ型を nstring に変更することはできますが、avro_string にはできません。同様に、パーサーデータ型を非パーサーデータ型に変更できません。

複合ファイルソースのネイティブデータ型の編集の詳細については、適切なコネクタのヘルプを参照してください。

トランスフォーメーションのデータ型の編集

データ統合は、ソースデータを読み取るときに、ネイティブデータ型を対応するトランスフォーメーションデータ型に変換してから、データのトランスフォームを実行します。データ統合は、ターゲットに書き込むときに、トランスフォーメーションデータ型を対応するネイティブデータ型に変換します。ソースのメタデータを編集するときは、フィールドのトランスフォーメーションデータ型を変更することができます。

ネイティブデータ型に複数のトランスフォーメーションデータ型があるようなコネクタのトランスフォーメーションデータ型を変更できます。例えば、Kafka ソースでは、ネイティブデータ型のバイナリをトランスフォーメーションデータ型のバイナリまたは文字列にマッピングできます。

トランスフォーメーションデータ型を変更するには、ソースのメタデータを編集し、**[タイプ]** カラムで適切なトランスフォーメーションデータ型を選択します。

フィールドのトランスフォーメーションデータ型を編集すると、データ統合では、ダウンストリームトランスフォーメーションでそのフィールドのデータ型が更新されます。実行時にターゲットが作成される場合は、ターゲットでそのフィールドのデータ型も更新されます。マッピングに既存のターゲットが含まれる場合は、データ型に互換性があるようにターゲットでフィールドのメタデータを編集する必要があります。

さまざまなソースタイプのトランスフォーメーションデータ型の編集の詳細については、適切なコネクタのヘルプを参照してください。

第 3 章

ターゲットトランスフォーメーション

ターゲットトランスフォーメーションを使用して、マッピングのターゲット接続およびターゲットオブジェクトを定義します。マッピングでは、1 つ以上のターゲットトランスフォーメーションを使用できます。

接続タイプに基づいて、ターゲットの詳細オプションを定義したり、新しいまたは既存のターゲットオブジェクトを使用するように指定したり、ターゲットの更新カラムを設定したりできます。表示されるターゲットオプションは、選択した接続タイプによっても異なります。例えば、Salesforce 接続を選択する場合、成功ログおよびエラーログの詳細を設定できます。

ターゲットトランスフォーメーションではファイル、データベース、およびデータ統合接続を使用できます。

ターゲットトランスフォーメーションでは、次のプロパティを使用できます。

- 全般。トランスフォーメーションの名前および説明を定義します。
- 受信フィールド。ターゲットに書き込まれるデータを定義するフィールドルールを含めます。ターゲットフィールドのプレビューができます。
- ターゲット。ターゲット接続、ターゲットオブジェクト、および詳細オプションを定義します。接続タイプに基づいて、新しいターゲットを作成したり、既存のターゲットを使用したり、更新カラムを設定したりできます。
- ターゲットフィールド。ターゲットオブジェクトのフィールドを一覧表示します。必要に応じてフィールドを追加または削除します。ターゲットフィールドのメタデータを編集することもできます。
- フィールドマッピング。アップストリームトランスフォーメーションからターゲットへのフィールドマッピングを定義します。フィールドマッピングは、既存のターゲットオブジェクトを使用する場合にのみ適用できます。

ターゲットの例

Salesforce ユーザーアカウントデータは読み取るが、ユーザー設定データは除外するマッピングでは、フラットファイルターゲットを使用できます。ソーストランスフォーメーションは、アカウントオブジェクトおよび関連するユーザーオブジェクトのデータを読み取ります。

ターゲットトランスフォーメーションは、ディレクトリ C:\UserAccountData への書き込みを行うフラットファイル接続を使用します。デフォルトの［すべてのフィールド］ルールには、すべての受信フィールドが含まれます。［名前付きフィールド］ルールを作成して、不要なユーザー設定フィールドを除外します。

［実行時に新しいターゲットを作成］を選択し、ターゲットファイルに次の名前を入力します。
SF_UserAccount_%d%m%y.csv

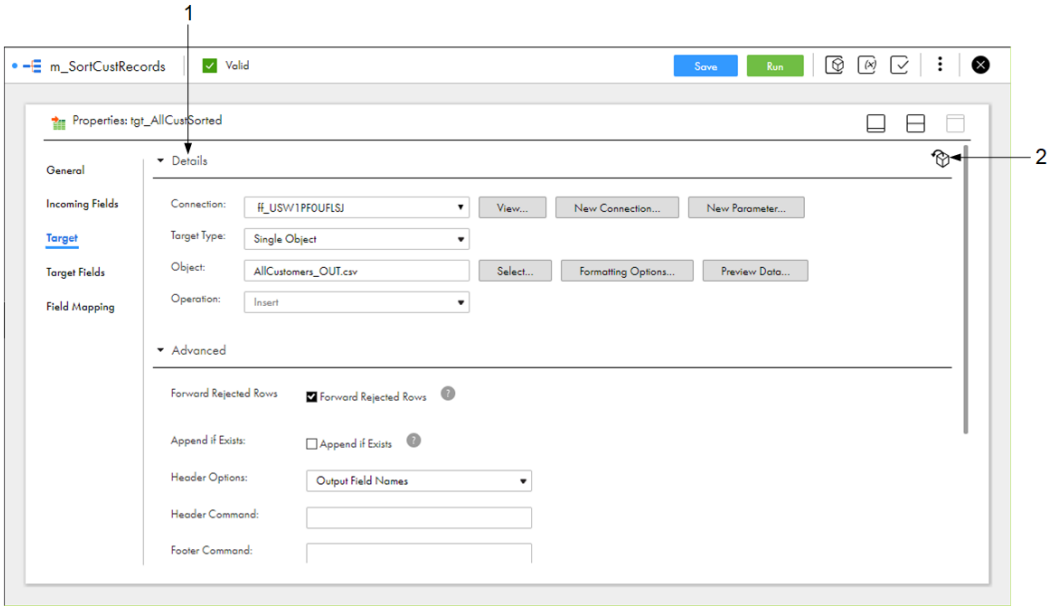
タスクが 2016 年 11 月 29 日に実行された場合、マッピングタスクは C:\UserAccountData ディレクトリに SF_UserAccount_291116.csv という名前のターゲットファイルを作成します。ターゲットファイルには、［名前付きフィールド］ルールで指定されたユーザー設定フィールドを除く、Salesforce アカウントオブジェクトおよびユーザーオブジェクトのすべてのフィールドが含まれます。

ターゲットオブジェクト

［プロパティ］パネルの［**ターゲット**］タブで、ターゲットトランスフォーメーションのターゲットオブジェクトを選択します。

ターゲットオブジェクトに設定するプロパティは、接続タイプとマッピングタイプに応じて異なります。また、組織のライセンスによって、ターゲットトランスフォーメーションがマップレットの一部である場合に表示されるターゲットプロパティが決定されます。

次の図に、フラットファイルターゲットの［**ターゲット**］タブを示します。



1. ターゲット接続、ターゲットタイプ、ターゲットオブジェクト、ターゲット操作を設定するターゲットの詳細。
2. マッピングインベントリからターゲットオブジェクトを選択します。

以下の方法でターゲットオブジェクトを選択できます。

接続とターゲットオブジェクトを選択する。

［**詳細**］領域で、ターゲット接続、ターゲットタイプ、およびターゲットオブジェクトを選択します。新しい接続を作成できます。フラットファイルターゲットおよびリレーショナルターゲットでは、実行時にターゲットオブジェクトを作成することもできます。

既存のターゲットオブジェクトを使用する場合は、ターゲットオブジェクトのリストからターゲットを選択し、それをアップストリームトランスフォーメーションにリンクします。ターゲットテーブルが変更された場合、その変更に合わせてターゲットトランスフォーメーションを更新する必要があります。物理ターゲットとターゲットトランスフォーメーションが一致しない場合、マッピングは失敗します。

フラットファイルターゲットに既存のターゲットオブジェクトを使用する場合、マッピングタスクを実行すると既存のターゲットが上書きされます。

マッピングインベントリからターゲットオブジェクトを選択します。

組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されていて、［**Data Catalog**］ページでオブジェクトをマッピングに追加した場合、［**インベントリ**］パネルからターゲットオブジェクトを選択できます。組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されていない、またはデータカタログの検出が実行されていない場合、［**インベントリ**］パネルには何も表示されません。データカタログの検出に関する詳細については、「*マッピング*」を参照してください。

パラメータを使用する。

マッピングタスクを実行するときは、入力パラメータを使用してターゲット接続やターゲットオブジェクトを定義できます。パラメータの詳細については、「*マッピング*」を参照してください。

また、ターゲット操作を定義する必要もあります。使用できるターゲット操作は、接続タイプによって異なります。

詳細クラスタでのターゲットファイルの作成

詳細クラスタで実行され、複合ファイルターゲットを含むマッピングを作成すると、ターゲットトランスフォーメーションは、［詳細］領域の［ファイル名］プロパティに入力したファイル名でフォルダを作成します。単一のファイルは作成されません。出力が一連のファイルに分割されている場合があるため、そうする必要があります。

フォルダ名には、一意の識別子が追加されることもあります。マッピングはファイル名とフォルダ内のすべてのコンテンツを認識するため、ユーザーが識別子を知っておく必要はありません。

フォルダ内のパートファイルは次のような名前になります: part-<一意の識別子>.<フォーマット>。ここで、サフィックスは csv、txt、avro、parquet、json、orc のいずれかです。

ファイルターゲット

ファイルターゲットには、フラットファイルと FTP/SFTP ファイルがあります。ファイルターゲットを設定する際には、接続、ターゲットタイプ、およびターゲットオブジェクトを指定します。

FTP/SFTP ターゲットの場合は、既存のターゲットオブジェクトを選択できます。フラットファイルターゲットの場合は、既存のターゲットオブジェクトを選択するか、実行時に新しいターゲットを作成することができます。

実行時にフラットファイルターゲットを作成する場合は、静的または動的ファイル名を指定できます。

ファイルターゲットのプロパティ

［プロパティ］パネルの**［ターゲット］**タブでファイルターゲットのプロパティを設定します。

次の表に、ファイルターゲットの詳細を示します。

プロパティ	説明
接続	ターゲット接続の名前。
ターゲットタイプ	ターゲットの種類（単一のオブジェクトまたはパラメータ）。
オブジェクト	ターゲットオブジェクトの名前。 詳細モードでは、オブジェクト名にドル記号文字$を含めることはできません。ドル記号は、パラメータ用に予約された文字です。

プロパティ	説明
形式オプション	フラットファイルの形式オプション。**［形式オプション］**ダイアログボックスを開き、ファイルの形式を定義します。 区切りファイルまたは固定長ファイルのいずれかのタイプを選択できます。デフォルトは区切りファイルです。 区切りフラットファイルタイプに書き込むには、次のファイル形式オプションを設定します。 - 区切り文字。区切り文字。カンマ、タブ文字、コロン、セミコロン、印刷不可能な制御文字、または指定したシングルバイト文字またはマルチバイト文字を使用することができます。 - テキスト修飾子。テキストを修飾する文字。 - エスケープ文字。エスケープ文字。 - フィールドラベル。マッピングタスクでフィールドラベルを生成するか、ソースファイルからラベルをインポートするかを指定します。 - 最初のデータ行。データの最初の行。タスクは、入力した行番号で読み取りを開始します。 タブ、スペース、または任意の印刷出力可能な特殊文字を区切り文字として使用できます。区切り文字には最大 10 文字を使用できます。区切り文字はエスケープ文字およびテキスト修飾子以外にする必要があります。 固定長フラットファイルタイプに書き込むには、使用する固定長ファイル形式を選択します。リストに同じ名前の複数の固定長ファイル形式が含まれている場合は、名前に追加されたプロジェクトとフォルダの場所を使用して、使用する適切なファイル形式を指定します。固定長ファイル形式がない場合、**［新規］** > **［コンポーネント］** > **［固定長ファイル形式］**をクリックして作成します。
操作	ファイルターゲットの場合、操作は常に挿入です。

次の表に、フラットファイルターゲットの詳細プロパティを示します。

プロパティ	説明
拒否された行の転送	マッピングタスクが拒否された行を拒否ファイルに転送するかどうかを示します。 行エラー処理を有効にした場合、マッピングタスクは拒否された行と削除された行を行エラーのログに書き込みます。却下ファイルは生成されません。削除された行を、行エラーログだけでなくセッションログにも書き出したい場合は、Verbose Data トレースを有効にします。 却下された行を転送しない場合、マッピングタスクは却下された行がない状態でセッションログに書き込みを行います。
1000 ごとの区切り文字	1000 ごとの区切り文字。なし、カンマ、またはピリオドにすることができます。小数点記号または区切り文字と同じにすることはできません。 フィールドタイプは数値である必要があります。また、場合によってはフィールドの精度とスケールを更新する必要があります。 デフォルトは［なし］です。
小数点記号	小数点文字。カンマまたはピリオドにすることができます。1000 ごとの区切り文字、または区切り文字と同じにすることはできません。 フィールドタイプは数値である必要があります。また、場合によってはフィールドの精度とスケールを更新する必要があります。 デフォルトはピリオドです。

プロパティ	説明
存在する場合は追加	各パーティションのターゲットファイルおよび拒否ファイルに出力データを追加します。FTP/SFTP ターゲットファイルの場合、このオプションは使用できません。 このオプションを選択しないと、マッピングタスクは出力データをターゲットファイルに書き込む前に、各ターゲットファイルを切り詰めます。ファイルが存在しない場合、マッピングタスクはファイルを作成します。
ターゲットディレクトリの作成	［出力ファイルディレクトリ］フィールドで指定されたターゲットディレクトリが存在しない場合は、ターゲットディレクトリを作成します。
ヘッダーのオプション	ファイルターゲットにヘッダ行を作成します。以下のオプションを選択することができます。 - No Header。フラットファイルターゲットにヘッダ行を作成しません。 - Output Field Names。出力フィールド名が指定されたファイルターゲットにヘッダー行を作成します。 - Use header command output。ヘッダ行を生成するには、［Header Command］フィールドでこのコマンドを使用します。例えば、ファイルターゲットのヘッダー行にデータを追加するコマンドを使用できます。 デフォルトは［No Header］です。
ヘッダーコマンド	ファイルターゲットにヘッダー行を生成するために使用するコマンド。例えば、ファイルターゲットのヘッダー行にデータを追加するコマンドを使用できます。
フッタコマンド	ファイルターゲットにフッタ行を生成するために使用するコマンド。
出力タイプ	タスクのターゲットのタイプ。ターゲットデータをファイルターゲットに書き込むには、**［ファイル］**を選択します。コマンドにデータを出力するには、**［コマンド］**を選択します。FTP/SFTP ターゲット接続に対して、**［コマンド］**を選択することはできません。
出力ファイル名	出力ファイルのファイル名、またはファイル名とパス。デフォルトでは、マッピングタスクによって、ターゲットオブジェクトの後に出力ファイルの名前が設定されます。
出力ファイルディレクトリ	フラットファイルターゲットの出力ディレクトリの名前。デフォルトでは、マッピングタスクは出力ファイルをターゲット接続ディレクトリに書き込みます。 また、入力パラメータを使用してターゲットファイルディレクトリを指定することもできます。 サービスプロセス変数ディレクトリ$PMTargetFileDir を使用した場合、タスクはシステム変数への設定済みのパスにターゲットファイルを書き込みます。システム変数の設定済みパスを見つけるには、次のディレクトリにある pmrdtm.cfg ファイルを参照してください。 `<Secure Agent installation directory>\apps\Data_Integration_Server\<Data Integration Server version>\ICS\main\bin\rdtm` また、$PMTargetFileDir 変数への設定済みのパスは、Administrator のデータ統合サーバーシステム設定の詳細にあります。

プロパティ	説明
拒否ファイルディレクトリ	拒否ファイルを書き込むディレクトリパス。デフォルトでは、マッピングタスクはすべての拒否ファイルを次のサービスプロセス変数ディレクトリに書き込みます: `$PMBadFileDir/<federated task ID>` **［拒否ファイル名］**フィールドでディレクトリとファイル名の両方を指定する場合は、このフィールドをクリアします。マッピングタスクはタスクの実行時に、このフィールドと**［拒否ファイル名］**フィールドを連結します。
拒否ファイル名	拒否ファイルのファイル名、またはファイル名とパス。デフォルトでは、マッピングタスクはターゲットオブジェクト名に従って、「<ターゲット名>.bad」のように拒否ファイルに名前を付けます。 マッピングタスクはタスクの実行時に、このフィールドと**［拒否ファイルディレクトリ］**フィールドを連結します。例えば、［拒否ファイルディレクトリ］フィールドに**「**C:\reject_file**」**を指定し、［拒否ファイル名］フィールドに**「**filename.bad**」**と入力した場合、マッピングタスクにより「C:\reject_file\filename.bad」に拒否された行が書き込まれます。

次の表では、ファイルターゲットにメッセージを書き込む場合の詳細プロパティについて説明します。

プロパティ	説明
入力タイプ	ターゲットに書き込む入力のタイプ。次のいずれかのオプションを選択します。 - バッファ - ファイル 使用可能なオプションは、アップストリームトランスフォーメーションからの受信データによって異なります。デフォルト値は［バッファ］です。
確認応答オプション	確認応答を保存するオプション。次のいずれかのオプションを選択します。 - **［セッションログに保存］**: 確認応答をセッションログに書き込みます。 - **［ファイルに保存］**: 確認応答をファイルに保存します。 - **［破棄］**: 確認応答を破棄します。
確認応答ファイルのパス	確認応答をファイルに保存する場合に、確認応答を書き込むファイルのパス。
ファイルが存在する場合の動作	確認応答ファイルがすでに存在する場合に実行する操作。次のいずれかの操作を選択します。 - **［名前の変更］**: 確認応答ファイルの名前を変更します。 - **［追加］**: ファイルに確認応答を追加します。 - **［上書き］**: 確認応答ファイルを上書きします。
確認応答が AA/CA ではない場合、ジョブは失敗します。	確認応答コードが AA または CA でない場合、ジョブが失敗します。
確認応答が AR/CR である場合はジョブを再試行します	確認応答コードが AR または CR の場合、ジョブの実行を再試行します。
メッセージ再試行の回数	メッセージの書き込みを試行する回数。
メッセージ再試行の間隔	メッセージの書き込みを再試行する時間間隔（秒単位）。

プロパティ	説明
メッセージの検証	データ統合がメッセージを検証するかどうかを示します。
拒否された行の転送	マッピングタスクが拒否された行を拒否ファイルに転送するかどうかを示します。 行エラー処理を有効にした場合、マッピングタスクは拒否された行と削除された行を行エラーのログに書き込みます。却下ファイルは生成されません。削除された行を、行エラーログだけでなくセッションログにも書き出したい場合は、Verbose Data トレースを有効にします。 却下された行を転送しない場合、マッピングタスクは却下された行がない状態でセッションログに書き込みを行います。

実行時に作成されるフラットファイルターゲット

マッピングにフラットファイルターゲットが含まれる場合は、既存のターゲットファイルを選択するか、実行時にターゲットを作成することができます。実行時にターゲットを作成すると、データ統合では、データ型、精度、およびスケールのターゲットオブジェクトメタデータがデータソースに基づいて自動的に検出されます。

ターゲットオブジェクトメタデータを編集する必要がある場合は、ソーストランスフォーメーションで編集できます。

ターゲットフィールドはアップストリームトランスフォーメーションにリンクできません。ターゲットの使用されていないフィールド数を削減する場合は、ターゲットトランスフォーメーションまたはアップストリームトランスフォーメーションのフィールドルールを設定します。

実行時にフラットファイルターゲットを作成すると、マッピングタスクは、マッピングの初回実行時にアップストリームトランスフォーメーションのフィールドに基づいて物理ターゲットを作成します。後続の実行でターゲットファイル名が変更されない場合、マッピングタスクはターゲットファイルを上書きします。マッピングの実行間でファイル名が変わった場合、マッピングタスクによって新しいターゲットが作成されます。データ統合は、デフォルトの接続ディレクトリにターゲットを作成します。

ターゲットファイルの静的または動的ファイル名を設定できます。静的ファイル名にはタイムスタンプを含めることができます。動的ファイル名では、マッピングタスクの実行時にファイル名を生成するための式が使用されます。

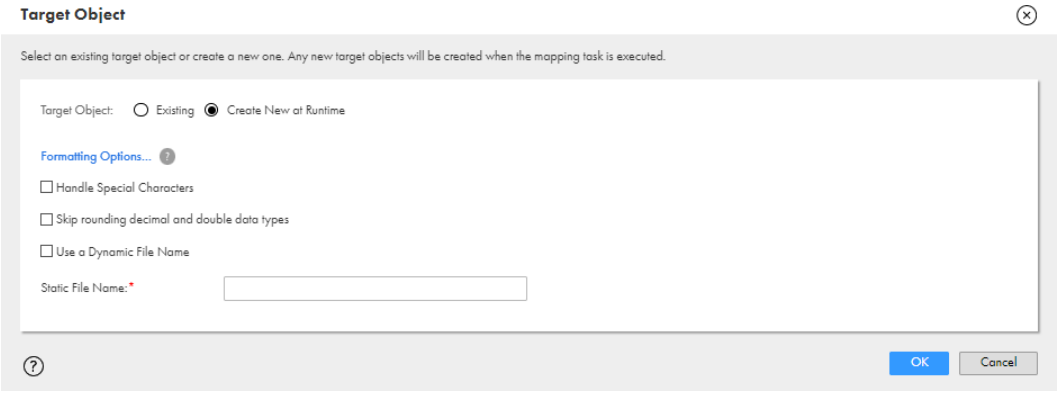
データ統合は、区切りファイルまたは固定長ターゲットファイルに書き込みを行います。デフォルトでは、データ統合は、固定長ファイル形式を使用して実行時にターゲットを作成する際、10 進型または倍精度浮動小数点数型のデータ型を使用するデータを四捨五入します。10 進型および倍精度浮動小数点数型のデータ型のデータの四捨五入をスキップすることもできます。デフォルトでは、データ統合は区切りデータの四捨五入は行いません。

静的ファイル名を持つフラットファイルターゲット

実行時にフラットファイルターゲットを作成する場合は、静的ファイル名を指定できます。ファイル名には、ファイルが作成された時間を示すタイムスタンプを含めることができます。

静的ファイル名を指定するには、**[ターゲットオブジェクト]** ダイアログボックスの **[静的ファイル名]** フィールドにファイル名を入力します。タイムスタンプを含めるには、**[特殊文字の処理]** を有効にして、ファイル名にタイムスタンプ文字を追加します。例えば、ファイル名 MyTarget_%d-%m.csv には、マッピングが実行された日と月が含まれます。

次の画像は、**[ターゲットオブジェクト]** ダイアログボックスを示しています。



タイムスタンプを含めない場合、マッピングタスクはタスクの初回実行時にターゲットファイルを作成し、後続の実行時にファイルを上書きします。

ターゲットファイル名にタイムスタンプを追加する場合、マッピングタスクはタイムスタンプの変更時にデータを新しいファイルに書き込みます。例えば、特殊文字の処理を有効にして静的ファイル名 MyTarget_%d-%m.csv を入力し、マッピングタスクを 1 月 15 日と 1 月 16 日に実行するとします。マッピングタスクはターゲットファイル MyTarget_15-01.csv と MyTarget_16-01.csv を作成します。

フラットファイルターゲットのタイムスタンプ

実行時にフラットファイルターゲットを作成する場合、タイムスタンプ情報をファイル名に追加して、ファイルがいつ作成されたのかを示すことができます。

ターゲットファイルのファイル名を指定する場合、タイムスタンプ情報をファイル名に追加するためにマッピングタスクが使用する Linux STRFTIME 関数の形式に基づいて、特殊文字を含めます。タイムスタンプは、組織のタイムゾーンに基づいています。

次の表に、使用する可能性のある一般的な STRFTIME 関数の形式を示します。

特殊文字	説明
%d	2 桁の 10 進数で表される日（01～31）。
%m	2 桁の 10 進数で表される月（01～12）。
%y	2 桁の 10 進数で表される、世紀を含まない年（00～99）。
%Y	世紀を含む年（2015 など）。
%T	24 時間表記の時刻（%H:%:M:%S に相当）。
%H	24 時間表記の時間（00～24）。
%I	12 時間表記の時間（01～12）。
%M	10 進数で表される分（00～59）。
%S	10 進数で表される秒（00～60）。
%p	AM または PM。

動的ファイル名を持つフラットファイルターゲット

実行時にフラットファイルターゲットを作成する場合は、動的ファイル名を指定できます。動的ファイル名では、ファイル名を生成するための式が使用されます。

動的ファイル名を使用すると、マッピングタスクが実行されるたびに新しいターゲットファイルを作成できます。例えば、次の式は、マッピングタスクが実行されるたびに「OrdersOut_<system_timestamp_with_second_precision>.csv」というファイルを作成します。

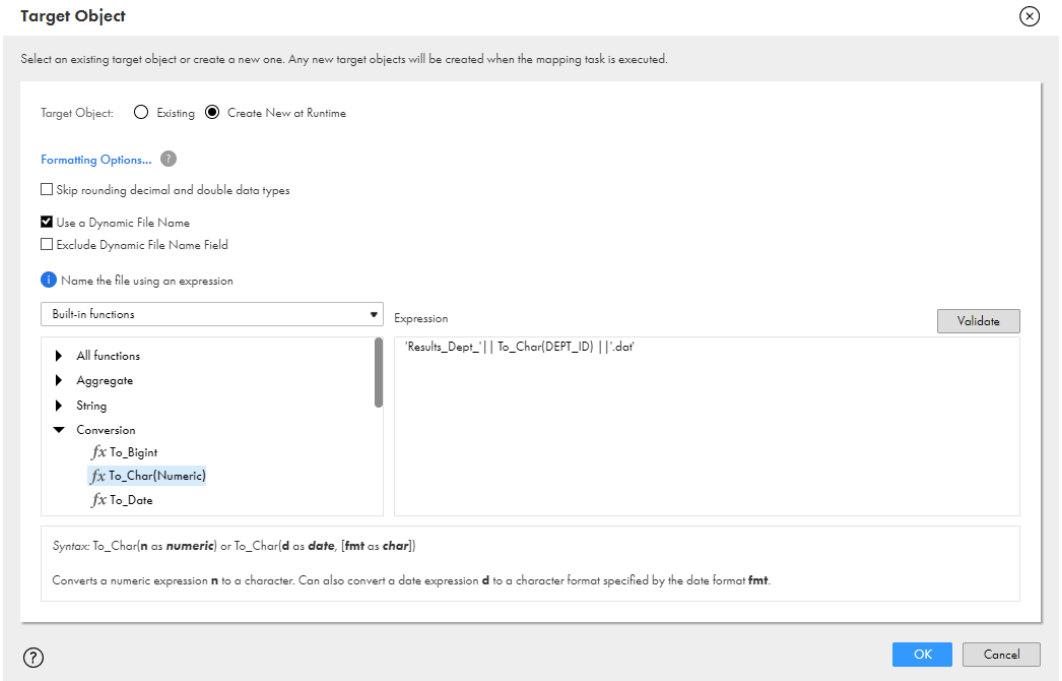
```
'OrdersOut_'||To_Char(SYSDATE, 'YYYYMMDDHH24MISS')||'.csv'
```

また、トランザクション制御トランスフォーメーションを含むマッピングで動的ファイル名を使用して、トランザクション境界が変わるたびにデータを異なるターゲットファイルに書き込むこともできます。例えば、トランザクション制御トランスフォーメーションのダウンストリームであるターゲットで次の式を使用すると、DEPT_ID フィールドが変わるたびにデータを異なるターゲットファイルにコミットできます。

```
'Results_Dept_'||To_Char(DEPT_ID)||'.dat'
```

動的ファイル名を指定するには、**[ターゲットオブジェクト]** ダイアログボックスで **[動的ファイル名を使用]** を選択し、式エディタでファイル名の式を入力します。

次の画像は、**[動的ファイル名を使用]** オプションが有効になっている **[ターゲットオブジェクト]** ダイアログボックスを示しています。



ターゲットファイル名の式には、受信フィールド名、定数、演算子、組み込み関数、およびユーザー定義関数を含めることができます。

ファイル名の式で入力フィールド名を使用する場合は、そのフィールドをターゲットから除外することを選択できます。**[動的ファイル名フィールドを除外]** オプションを有効にすると、データ統合は、式で使用される受信フィールドをターゲットに書き込みません。式に受信フィールドを 1 つだけ含めます。複数の受信フィールドを含めると、式は無効になります。

複数の受信フィールドを使用するには、ターゲットトランスフォーメーションの直前に式トランスフォーメーションを追加します。式トランスフォーメーションで、ファイル名として使用する式を保持するフィールドを設定します。ターゲットトランスフォーメーションでは、このフィールドを動的ファイル名の式として使用します。

式の作成の詳細については、*関数リファレンス*の説明を参照してください。

実行時におけるフラットファイルターゲットの作成

実行時にフラットファイルターゲットを作成するには、**［ターゲットオブジェクト］**ダイアログボックスで**［実行時に新規作成］**を選択します。次に、ターゲットファイルの静的または動的ファイル名を設定します。

1. ターゲットトランスフォーメーションの**［ターゲット］**タブで、フラットファイル接続を選択します。
2. ターゲットタイプを**［単一オブジェクト］**に設定します。
3. **［選択］**をクリックして、ターゲットオブジェクトを選択します。
4. **［ターゲットオブジェクト］**ダイアログボックスで、**［実行時に新規作成］**を選択します。
5. ターゲットファイル名を設定します。
   - タイムスタンプのない静的ファイル名を入力するには、**［静的ファイル名］**フィールドにファイル名を入力します。
   - タイムスタンプを含む静的ファイル名を入力するには、**［特殊文字の処理］**を有効にして、**［静的ファイル名］**フィールドにタイムスタンプ文字を含むファイル名を入力します。
   - 動的ファイル名を入力するには、**［動的ファイル名を使用］**を有効にして、ファイル名を作成するための式を設定します。

     ファイル名の式では、フィールド、システム変数、パラメータ、組み込み関数、およびユーザー定義関数を使用できます。
6. 必要に応じて、10 進型または倍精度浮動小数点数型のデータ型を使用する固定長フラットファイル形式を使用しており、データ統合でデータを四捨五入しないようにする場合は、**［10 進型および倍精度浮動小数点数型の四捨五入をスキップ］**を選択します。
7. **［OK］**をクリックします。

データベースのターゲット

データベースターゲットには、Oracle、MySQL、および Microsoft SQL Server などのリレーショナルソースが含まれています。

データベースターゲットのターゲットトランスフォーメーションを設定する場合は、データを単一のターゲットテーブルに書き込むことができます。既存のテーブルを選択するか、実行時にテーブルを作成することができます。

テーブル名とカラム名は 74 文字を超えないようにしてください。

データベースターゲットのプロパティ

［プロパティ］パネルの**［ターゲット］**タブでデータベースターゲットのプロパティを設定します。

次の表にデータベースターゲットプロパティを示します。

プロパティ	説明
接続	ターゲット接続の名前。 または、パラメータを定義してから、マッピングタスクで接続を指定することもできます。
ターゲットタイプ	ターゲットの種類（単一のオブジェクトまたはパラメータ）。
オブジェクト	ターゲットオブジェクトの名前。 1 つのオブジェクトを選択した場合は、データをプレビューすることもできます。
操作	ターゲット操作（挿入、更新、更新/挿入、削除、またはデータドリブンのいずれか）。
ターゲットの切り詰め	新しい行を挿入する前にターゲットオブジェクトを切り詰めます。 挿入操作およびデータドリブン操作に適用されます。
ターゲットの一括ロードの有効化	データベースの一括 API を使用して、挿入操作を実行します。 一括 API を使用して、最小限の数の API 呼び出しでデータベースに大量のデータを書き込みます。[Bulk] モードでロードすると、パフォーマンスを向上させることができますが、データベースロギングが発生しないので、リカバリを実行する機能が制限されます。 挿入操作に適用されます。
更新カラム	ターゲットデータの更新、更新/挿入、または削除を行うときに一時プライマリキーカラムとして使用するフィールド。複数の更新カラムを選択した場合、マッピングタスクは各更新カラムと AND 演算子を使用して、一致する行を特定します。 更新、更新/挿入、削除、およびデータドリブン操作に適用されます。
データ依存条件	行に挿入、更新、削除、または拒否操作のフラグを設定する式を定義できます。 例えば、次の IIF ステートメントは、ID フィールドが null の場合に、行に拒否のフラグを設定します。そうでない場合は行に更新のフラグを設定します。 `IIF (ISNULL(ID), DD_REJECT, DD_UPDATE )` データドリブン操作に適用されます。
拒否された行の転送	マッピングタスクを実行すると、拒否された行が拒否ファイルに転送されるようにします。 拒否された行を転送しない場合、マッピングタスクは拒否された行を削除してセッションログに書き込みます。 行エラー処理を有効にした場合、マッピングタスクは拒否された行と削除された行を行エラーのログに書き込みます。拒否ファイルは作成しません。削除された行を、行エラーログだけでなくセッションログにも書き出したい場合は、Verbose Data トレースを有効にします。
Pre SQL	データをソースから読み取る前にターゲットに対して実行する SQL コマンド。 5000 文字までのコマンドを入力できます。
Post SQL	データをターゲットに書き込んだ後にターゲットに対して実行する SQL コマンド。 5000 文字までのコマンドを入力できます。

プロパティ	説明
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。 UPDATE 文を入力します。または、**［設定］**をクリックしてデフォルトの UPDATE 文を生成し、デフォルトの文を変更します。 入力した UPDATE 文により、データ統合がキーカラムに基づいてターゲットを更新するために使用するデフォルトの UPDATE 文がオーバーライドされます。UPDATE 文のオーバーライドを定義すると、キー以外のカラムに基づいてターゲットテーブルを更新できます。
拒否ファイルディレクトリ	拒否ファイルを書き込むディレクトリパス。デフォルトでは、マッピングタスクはすべての拒否ファイルを次のサービスプロセス変数ディレクトリに書き込みます: `$PMBadFileDir/<federated task ID>` **［拒否ファイル名］**フィールドでディレクトリとファイル名の両方を指定する場合は、このフィールドをクリアします。マッピングタスクはタスクの実行時に、このフィールドと**［拒否ファイル名］**フィールドを連結します。
拒否ファイル名	拒否ファイルのファイル名、またはファイル名とパス。デフォルトでは、マッピングタスクはターゲットオブジェクト名に従って、「<ターゲット名>.bad」のように拒否ファイルに名前を付けます。 マッピングタスクはタスクの実行時に、このフィールドと**［拒否ファイルディレクトリ］**フィールドを連結します。例えば、［拒否ファイルディレクトリ］フィールドに**「**C:\reject_file**」**を指定し、［拒否ファイル名］フィールドに**「**filename.bad**」**と入力した場合、マッピングタスクにより「C:\reject_file\filename.bad」に拒否された行が書き込まれます。

データベースターゲットのプロパティの詳細については、目的のコネクタのヘルプを参照してください。

実行時に作成されるデータベースターゲット

マッピングにデータベースターゲットが含まれる場合は、既存のターゲットテーブルを選択するか、実行時にターゲットテーブルを作成することができます。実行時にデータベースターゲットを作成すると、データ統合では、データ型、精度、およびスケールのターゲットオブジェクトメタデータがデータソースに基づいて自動的に検出されます。

ターゲットオブジェクトメタデータを編集する必要がある場合は、ソーストランスフォーメーションで編集できます。

ターゲットフィールドはアップストリームトランスフォーメーションにリンクできません。ターゲットの使用されていないフィールド数を削減する場合は、ターゲットトランスフォーメーションまたはアップストリームトランスフォーメーションのフィールドルールを設定します。

実行時にデータベースターゲットを作成すると、マッピングタスクは、マッピングの初回実行時にアップストリームトランスフォーメーションのフィールドに基づいてデータベーステーブルを作成します。

後続の実行では、マッピングタスクは初回実行時に作成されたターゲットテーブルのデータを置き換えます。このため、最初の実行の後にマッピングを変更した場合、ターゲットフィールド数やそのメタデータへの変更が後続の実行でターゲットに反映されません。変更を確認するには、マッピングを実行する前に既存のターゲットを削除するか、ターゲットの名前を変更します。

実行時にリレーショナルターゲットを作成する場合、ターゲット操作は常に挿入です。ターゲットの切り捨てを選択できます。

注: 2020 年 9 月春リリースの前に作成されたマッピングでは、パラメータ化されたソースの Bigint データは、データ統合によって、ランタイム時に作成されたデータベースターゲット内で int データに変換されます。Bigint データを変換せずにターゲットに書き込むには、Mapping Designer でマッピングを編集し、マッピング詳細プロパティでオプションを有効にします。

2020 年 9 月春リリース以降に作成されたマッピングでは、データ統合によって Bigint データは変換されません。

実行時におけるデータベースターゲットの作成

実行時にデータベースターゲットを作成するには、**［ターゲットオブジェクト］**ダイアログボックスで**［実行時に新規作成］**を選択し、ターゲットテーブル名を入力します。

1. ターゲットトランスフォーメーションの**［ターゲット］**タブで、データベース接続を選択します。
2. ターゲットタイプを**［単一オブジェクト］**に設定します。
3. **［選択］**をクリックして、ターゲットオブジェクトを選択します。
4. **［ターゲットオブジェクト］**ダイアログボックスで、**［実行時に新規作成］**を選択します。
5. ターゲットテーブル名を入力します。
6. **［OK］**をクリックします。

リレーショナルターゲットの更新カラム

1 つ以上のフィールドをリレーショナルターゲットの更新カラムとして設定できます。更新カラムは、ターゲットテーブル内の行を一意に識別するカラムです。マッピングタスクはそれらのフィールドを使用して、ターゲット内のデータを更新、更新/挿入、または削除します。

ターゲットテーブルにプライマリキーが含まれておらず、マッピングで更新、更新/挿入、または削除の操作が使用されている場合は、更新カラムを設定します。複数の更新カラムを選択した場合、マッピングは各更新カラムと AND 演算子を使用して、一致する行を特定します。

マッピングを実行すると、フィールドマッピングを使用して、アップストリームトランスフォーメーションの行がターゲットテーブルに照合されます。マッピングタスクが 1 つの受信行を複数のターゲット行と照合した場合、一致したすべてのターゲット行に対して特定のタスク操作が実行されます。

ターゲット接続またはターゲットオブジェクトのパラメータを使用して、マッピングタスクの更新カラムを設定できます。

更新カラムの設定

更新または upsert 操作を使用してリレーショナルターゲットのデータを更新する場合、更新カラムを設定できます。

1. ［プロパティ］パネルで、**［ターゲット］**タブをクリックします。
2. リレーショナル接続を選択します。

   リレーショナルデータベース接続タイプの場合は、接続パラメータを使用することもできます。
3. 使用するターゲットタイプを選択します。
4. ターゲットオブジェクトを選択します。
5. 更新または upsert 操作を選択します。
6. 更新カラムを選択するには、**［追加］**をクリックします。

   ［更新カラム］ウィンドウにすべてのターゲットカラムが表示されます。
7. 使用するフィールドを**［ターゲットカラム］**リストから**［更新カラム］**リストに移動します。
8. **［OK］**をクリックします。

ターゲット更新のオーバーライド

デフォルトでは、データ統合により、キー値に基づいてターゲットテーブルが更新されます。しかしながら、マッピングの各ターゲットについて、デフォルトの UPDATE 文を上書きすることができます。これによって、キー以外のカラムに基づいてターゲットの更新を行うことができます。

リレーショナル接続と ODBC 接続に対してターゲット更新のオーバーライドを入力できます。詳細については、目的のコネクタのヘルプを参照してください。

ターゲットトランスフォーメーションの詳細プロパティで UPDATE 文をオーバーライドします。**[更新オーバーライド]** フィールドにターゲット UPDATE 文を入力します。または、**[設定]** をクリックしてデフォルトの UPDATE 文を生成し、その文を変更します。

ターゲットフィールドはターゲットカラム名に一致する必要があるため、更新文にはターゲットトランスフォーメーションのフィールドを指定するための「:TU」というキーワードが含まれます。この文の UPDATE 部分を変更する場合は、TU を使用してフィールドを指定する必要があります。

デフォルトの UPDATE 文をオーバーライドする場合は、データベースで有効な SQL 文を入力する必要があります。データ統合では構文は検証されません。

例

あるマッピングが各販売員の総販売実績を T_SALES テーブルに送るとします。

データ統合は、ターゲット T_SALES に対して次のデフォルト UPDATE 文を生成します。

```
UPDATE
  T_SALES
SET
  EMP_NAME = :TU.EMP_NAME,
  DATE_SHIPPED = :TU.DATE_SHIPPED,
  TOTAL_SALES = :TU.TOTAL_SALES
WHERE
  EMP_ID = :TU.EMP_ID
```

WHERE 句をオーバーライドして、Mike Smith という名前の従業員のレコードのみを更新するとします。これを行うには、WHERE 句を以下のように編集します。

```
UPDATE
  T_SALES
SET
  DATE_SHIPPED = :TU.DATE_SHIPPED,
  TOTAL_SALES = :TU.TOTAL_SALES
WHERE
  :TU.EMP_NAME = EMP_NAME AND EMP_NAME = 'MIKE SMITH'
```

ターゲット更新のオーバーライドの設定に関するガイドライン

ターゲット更新クエリを入力する場合には、以下のガイドラインに従ってください。

- ターゲット更新のオーバーライドを使う場合は、手作業ですべての予約語を引用符で囲む必要があります。
- ターゲットカラム名に次の文字のどれかが含まれている場合、デフォルトの UPDATE 文はオーバーライドできません。

  ' , ( ) < > = + - * / \ t \ n \ 0 <空白文字>
- ターゲットテーブルの個々の行について複数回更新を行った場合には、データベースには最後の更新データが入ります。マッピングに結果データの順序が定義されていない場合、同一の入力データに対して異なるマッピングを実行すると、ターゲットテーブルのデータが異なる場合があります。
- カラム参照を含まない WHERE 句は、ターゲットテーブルの行すべてを更新する場合もあれば、まったく更新しない場合もあります。これは、WHERE 句の内容とマッピングからのデータによって決まります。例え

ば、以下のクエリでは、トランスフォーメーションのいずれかの行に EMP_ID > 100 がある場合、ターゲットテーブルのすべての行について EMP_NAME を「MIKE SMITH」に設定します。

```
UPDATE T_SALES SET EMP_NAME = 'MIKE SMITH' WHERE :TU.EMP_ID > 100
```

- WHERE 句にフィールド参照が含まれていない場合には、マッピングの各行について同じ一連の行が更新されます。例えば、以下のクエリでは EMP_ID > 100 のすべての従業員が更新され、マッピングの最後の行から EMP_NAME を取得します。

  ```
  UPDATE T_SALES SET EMP_NAME = :TU.EMP_NAME WHERE EMP_ID > 100
  ```

- ターゲット操作を更新または更新/挿入に設定する場合は、詳細セッションプロパティでソース行を更新として処理するようにマッピングタスクを設定します。

ターゲット UPDATE 文の入力

ターゲット UPDATE 文は、ターゲットトランスフォーメーションの **[ターゲット]** タブで入力します。

1. ターゲットトランスフォーメーションの **[ターゲット]** タブで、詳細プロパティを開きます。
2. **[更新オーバーライド]** フィールドの横にある **[設定]** をクリックします。
3. **[更新オーバーライド SQL エディタ]** で、**[SQL の生成]** をクリックします。

   デフォルトの UPDATE 文が表示されます。
4. UPDATE 文を変更します。

   ヒント: [SQL の書式設定] をクリックして UPDATE 文を書式設定すると、読みやすくなります。

   WHERE 句を上書きしてキー以外のカラムを含めることができます。データベース予約語はすべて引用符で囲んでください。
5. **[OK]** をクリックします。

Web サービスのターゲット

ターゲットトランスフォーメーションに Workday V2 などの Web サービスコネクタを使用できます。

リレーショナル構造のフィールドを Web サービスターゲットの要求フィールドにマッピングし、階層出力を生成できます。

ターゲットトランスフォーメーションの Web サービス接続を選択するときに、次の手順を実行してトランスフォーメーションを設定します。

1. Web サービス操作を選択します。
2. 受信フィールドを Web サービスの要求構造にマッピングします。

例えば、Workday の連絡先情報を更新するとします。ターゲットトランスフォーメーションを作成し、Workday 接続を選択します。Maintain_Contact_Info 操作を選択します。受信フィールドを Workday 操作の要求にマッピングします。

ターゲットの Web サービス操作

ターゲットトランスフォーメーションでは、Web サービス操作によって Web サービスに渡すことができるデータセットが決まります。

Web サービス接続のターゲットのプロパティを定義するときに、Web サービス操作を選択します。使用可能な操作は、接続によって決定されます。例えば、Workday 接続の場合、Update_Job_Posting が使用可能な操作です。

Web サービスターゲットのフィールドマッピング

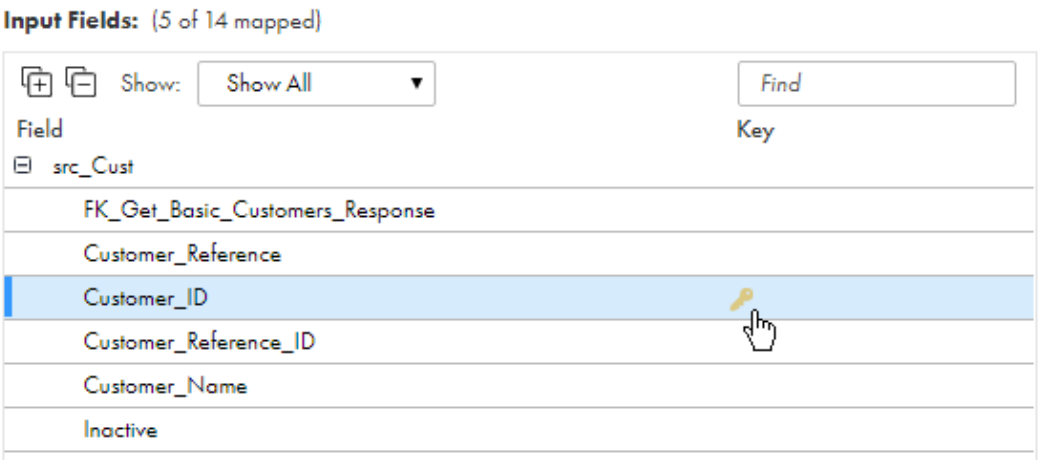
リレーショナル構造にあるフィールドを Web サービスターゲットで使用される階層構造にマッピングできます。

ターゲットトランスフォーメーションの［**フィールドマッピング**］タブで、［**ターゲットフィールド**］領域に表示されるフィールドが階層構造で表示されます。ターゲットフィールドは、ターゲットトランスフォーメーションに選択された操作の要求メッセージ構造によって決まります。

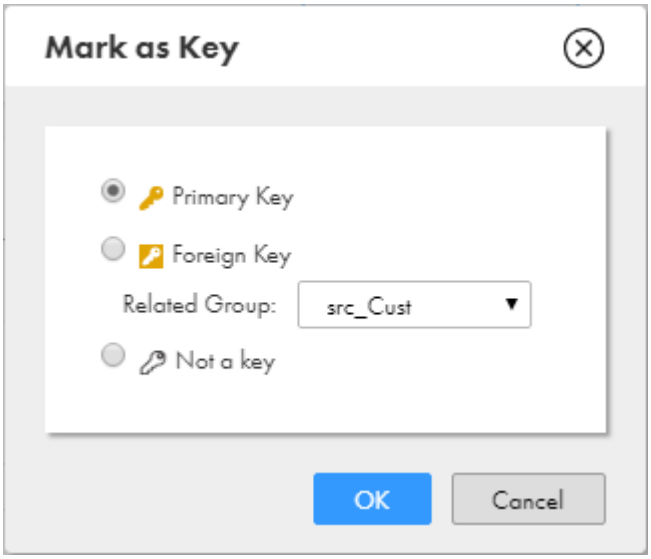
各ソースオブジェクトは、［**入力フィールド**］領域のグループとして表示されます。［**入力フィールド**］領域のフィールドを選択し、フィールドを Web サービス要求にマッピングできます。入力フィールドに複数の入力グループが含まれている場合、グループを Web サービス要求の対応するノードにマッピングします。Web サービス要求で必要なすべてのフィールドをマップする必要があります。

複数のソースオブジェクトをマッピングする場合、プライマリキーと外部キーを少なくとも 2 つのグループに割り当てる必要があります。ソースデータが、親オブジェクトのプライマリキー、および子オブジェクトの外部キーとプライマリキーでソートされていることを確認します。

キーを割り当てるには、次の画像のように、プライマリキーまたは外部キーにするフィールドの横にある［キー］アイコンをクリックします。



［**キーとしてマーク**］ダイアログボックスで、次の画像のように、プライマリキーまたは外部キーとしてフィールドを割り当て、関連グループを選択できます。



キーは Bigint データ型または String データ型にする必要があります。

パーティション

詳細モードでは、実行時に作成する一部のタイプのパーティション化されたターゲットにデータをロードするときに、パーティションキーフィールドを設定できます。一部のターゲットタイプでは、パーティションを使用して、ターゲットへのデータのロードを最適化できます。

[パーティション化] タブで、パーティションキーフィールドとパーティション化方法を設定できます。**[パーティション化]** タブは、詳細モードのターゲットに対して表示されます。

パーティションキーフィールド

実行時に作成する特定のタイプのパーティション化されたターゲットにデータをロードするときに、パーティションキーとして使用されるフィールドを設定できます。複合ファイルターゲットにデータを書き込む場合は、パーティションキーフィールドを設定する必要が生じる場合があります。

例えば、実行時に作成する Amazon S3 V2 ターゲットにデータをロードするマッピングを作成できます。ターゲットは、Avro データファイルを利用するパーティション化された Hive テーブルです。YEAR、MONTH、および DAY カラムに基づいてパーティション化されたディレクトリにデータファイルを書き込む必要があります。YEAR、MONTH、および DAY フィールドをパーティションキーとして設定します。

[パーティション化] タブの［パーティションフィールド］領域で、パーティションキーとして使用するフィールドを設定します。パーティションキーフィールドの追加、削除、および順序変更ができます。

さまざまなターゲットタイプに対するパーティションキーフィールドの設定の詳細については、目的のコネクタのヘルプを参照してください。

パーティション化方法

マッピングタスクが大規模なデータセットをロードする場合、データをロードするためにタスクに長い時間がかかる可能性があります。複数のパーティションを使用すると、マッピングタスクはデータを複数のパーティションに分割し、各パーティションのデータを同時にロードします。これにより、パフォーマンスを最適化できます。すべてのターゲットタイプがパーティション化をサポートしているわけではありません。

詳細モードのターゲットがパーティション化をサポートしている場合は、**[パーティション化]** タブの［並行処理］領域でパーティション化方法を選択できます。選択できるパーティション化方法は、ターゲットタイプによって異なります。さまざまなタイプのターゲットのパーティション化の詳細については、目的のコネクタのヘルプを参照してください。

ターゲットタイプに応じて、次のいずれかのパーティション化方法を選択できます。

なし

マッピングタスクが、すべてのデータを単一のパーティションにロードします。これがデフォルトのオプションです。

固定

マッピングタスクが、指定したパーティション数に基づいてデータの行を分散します。最大で 64 のパーティションを指定できます。

ターゲットに渡されるレコードの数を考慮して、ターゲットパーティションの適切な数を決定してください。レコードが少ない場合、パーティション化による利点はあまりありません。

パススルー

マッピングタスクが、パーティション間で行を再分散せずにデータを処理します。単一のパーティション内のすべての行が、そのパーティション内に留まります。パフォーマンスを向上させるために新しいパー

ティションを作成して、パーティション間のデータの分散を変更しない場合に、パススルーパーティション化を選択します。

動的

マッピングタスクが、実行時に作成するパーティションの最適な数を決定します。

詳細モードでの階層データの書き込み

詳細モードでターゲットトランスフォーメーションを使用して、Avro ファイル、JSON ファイル、Parquet ファイルなどの複合ファイルに階層データを書き込むことができます。

階層データを書き込む場合は、次のガイドラインを考慮してください。

- 階層データを書き込むには、Amazon S3 V2 接続または Azure Data Lake Storage Gen2 接続を使用する必要があります。詳細については、該当するコネクタのヘルプを参照してください。
- 実行時に新しいターゲットを作成する必要があります。
- ターゲット接続またはターゲットオブジェクトのパラメータは使用しないでください。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

ターゲットフィールド

データフローで使用するターゲットフィールドを設定できます。ターゲットフィールドの追加と削除、フィールドの表示方法の設定、フィールドメタデータの編集、およびターゲットオブジェクトからの元のフィールドの復元を行うことができます。

[プロパティ] パネルの **[ターゲット]** タブでターゲットフィールドを設定します。

ターゲットフィールドは、次の方法で設定できます。

フィールドを追加および削除します。

マッピングターゲットにフィールドを追加できます。フィールドを追加するには、**[フィールドの追加]** をクリックして、フィールド名、タイプ、精度、およびスケールを入力します。

また、マッピングで使用しないフィールドを削除することもできます。フィールドを削除するには、削除するフィールドを選択して、**[削除]** をクリックします。

ソート順を変更します。

ターゲットフィールドは、ネイティブ順、昇順、または降順に表示できます。ソート順を変更するには、**[ソート]** をクリックして、適切なソート順を選択します。

技術フィールド名またはラベルを使用します。

フィールド名は、ラベルまたは技術フィールド名で表示できます。

フィールド名の表示オプションを変更するには、**［オプション］** > **［フィールドの技術名を使用］**または**［オプション］** > **［ラベルを使用］**を選択します。

フィールドメタデータを編集します。

フィールドのメタデータを編集できます。誤って推測された情報を変更するためにメタデータを編集することがあります。メタデータを編集する場合は、名前、ネイティブタイプ、ネイティブ精度、およびネイティブスケールを変更できます（データ型に該当する場合）。

1 つまたは複数のフィールドの名前またはメタデータを編集するには、**［オプション］** > **［メタデータの編集］**をクリックします。メタデータを編集する場合は、ラベルまたは技術フィールド名でネイティブ名を表示することもできます。ネイティブ名の表示オプションを変更するには、**［オプション］** > **［フィールドの技術名を表示］**または**［オプション］** > **［ラベルを表示］**を選択します。

フィールドのメタデータを変更する場合は、タスクの実行時にエラーが発生する可能性のある変更を行わないようにします。例えば、フィールドのネイティブ精度やネイティブスケールを増やしても、通常エラーになることはありません。一方、フィールドの精度を減らすと、データが切り捨てられる可能性があります。

ターゲットオブジェクトから元のフィールドを復元します。

ターゲットオブジェクトから元のフィールドを復元するには、**［同期］**オプションを使用します。フィールドを同期すると、データ統合は削除されたターゲットフィールドを復元し、データ型と精度の変更を元に戻して、ターゲットに新しいフィールドを追加します。データ統合は、追加されたフィールドのうち、ターゲットオブジェクトで対応するフィールドがないものを削除します。

既存のターゲットフィールドの場合、データ統合は、編集したメタデータをターゲットオブジェクトのフィールドメタデータで置き換えます。データ統合は、**［名前］**フィールドに加えられた変更は元に戻しません。

ターゲットトランスフォーメーションのフィールドマッピング

ターゲットトランスフォーメーションのフィールドマッピングを設定し、データフローからターゲットオブジェクトへのデータの移動方法を定義します。

［フィールドマッピング］タブでフィールドマッピングを設定します。

［フィールドマッピング］タブには、受信フィールドおよびターゲットフィールドのリストが含まれています。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

ターゲットフィールドに受信フィールドをマッピングする方法。次のいずれかのオプションを選択します。

- 手動。受信フィールドをターゲットフィールドに手動でリンクします。このオプションを選択すると、自動的にマッピングされたフィールドのリンクが削除されます。フィールドを手動でマッピングするには、フィールドを［受信フィールド］リストからドラッグして、［ターゲットフィールド］リストの適切なフィールドの横に配置します。または、**［アクション］**メニューを使用して、選択されたフィールドをマッピングまたはマッピング解除したり、すべてのマッピングをクリアしたりすることができます。

- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。タスクで、すべてのフィールドマッピングを設定できます。
- 一部パラメータ化。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。
  フィールドマッピングパラメータの詳細については、「*マッピング*」を参照してください。

オプション

フィールドの表示方法や表示するフィールドを設定できます。これらを設定するには、**［オプション］**をクリックして次の表示オプションから選択します。

- フィールドラベルまたは技術フィールド名を使用してフィールドを表示する。
- マッピングされたフィールド、マッピングされていないフィールド、またはすべてのフィールドを表示する。

自動マップ

データ統合で同じ名前のフィールドを自動的にリンクして、フィールドのマッピングを手動で行う場合は、**［手動］**オプションを選択してから**［自動マップ］**メニューを開きます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

ターゲットトランスフォーメーションの設定

マッピングの結果は、既存のターゲットに保存することも、新たにターゲットを作成して保存することもできます。ターゲットを作成することを選択した場合、Secure Agent はタスクの実行時にターゲットを作成します。

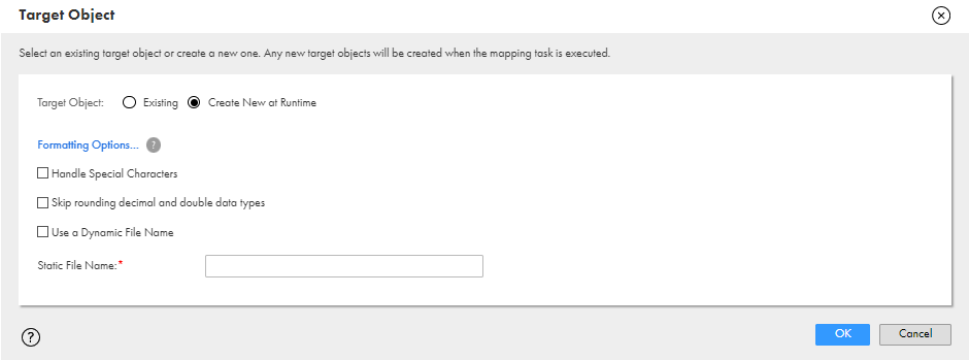
1. マッピングでターゲットトランスフォーメーションを選択します。
2. **［全般］**タブで、ターゲット名と任意の説明を入力します
3. **［受信フィールド］**タブで、ターゲットに含めるフィールドを指定するフィールドルールを設定します

   フィールドルールの詳細については、「フィールドルール」 (ページ 25)を参照してください。

4. **[ターゲット]** タブで、ターゲット接続を選択し、ターゲットの種類（例えば、**[単一オブジェクト]** や **[パラメータ]**）を指定します。

   複数のオブジェクトの選択するオプションは、NetSuite 接続でのみ使用できます。

5. ターゲットオブジェクトに入力パラメータを使用するには、既存のパラメータを選択するか、**[新しいパラメータ]** をクリックして、ターゲットオブジェクトの新しいパラメータを作成します。
6. 既存のターゲットオブジェクトを使用するには、オブジェクト名を入力するか、**[選択]** をクリックしてターゲットオブジェクトを選択します。
7. 新しいターゲットオブジェクトを作成するには、次の手順を実行します。
   a. **[選択]** をクリックします。
   b. **[ターゲットオブジェクト]** ダイアログボックスで、**[実行時に新規作成]** を選択します。



   c. フラットファイルターゲットの場合は、Accounts.csv など、拡張子を含むターゲットファイルの名前を入力します。

      ファイル名にタイムスタンプを含める場合は、**[特殊文字の処理]** を選択し、ファイル名に特殊文字を追加します（例えば、Accounts_%d%m%y%T.csv）。

      動的ファイル名を使用する場合は、**[動的ファイル名を使用]** を選択してファイル名の式を設定します。

      精度とスケールを持つ 10 進型または倍精度浮動小数点数型のデータ型を使用する固定長フラットファイル形式を使用しており、ターゲットファイルでデータを四捨五入しないようにする場合は、**[10 進型および倍精度浮動小数点数型の四捨五入をスキップ]** を選択します。

   d. リレーショナルターゲットの場合は、テーブル名を入力します。
   e. クラウドデータウェアハウスターゲットの場合は、必要に応じて、新しいターゲットオブジェクトで正確なフィールド名を使用することを選択し、オブジェクト名を入力して、テーブルの場所、テーブルタイプ、パスなどのその他のプロパティを入力します。

      さまざまな接続タイプのターゲットプロパティに関する詳細については、該当するコネクタのヘルプを参照してください。

   f. **[OK]** をクリックします。
8. フラットファイルターゲットの書式オプションを設定するには、**[形式オプション]** をクリックし、区切り文字やテキスト修飾子などの書式オプションを設定します。
9. リレーショナルターゲットの場合は、挿入操作のためにテーブルを切り捨てるかどうかなど、ターゲット操作と関連プロパティを選択します。

   実行時にリレーショナルターゲットを作成する場合、ターゲット操作は常に挿入です。

10. 必要に応じて、ターゲットの詳細プロパティを指定します。

    詳細プロパティは、接続タイプによって異なります。コネクタのプロパティの詳細については、目的のコネクタのヘルプを参照してください。

11. **［ターゲットフィールド］**タブでターゲットフィールドを設定します。

    フィールド名およびメタデータの編集、フィールドの追加、不要なフィールドの削除を行うことができます。

    実行時にターゲットを作成する場合、ターゲットフィールドを設定することはできません。

12. **［フィールドマッピング］**タブで、受信フィールドをターゲットフィールドにマッピングします。

    実行時にターゲットを作成する場合、フィールドは自動的にマップされます。

    フィールドマッピングの詳細については、「ターゲットトランスフォーメーションのフィールドマッピング」 (ページ 88)を参照してください。

第 4 章

アグリゲータ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、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ることを除いて、式トランスフォーメーションと同様に動作します。式トランスフォーメーションは、行ごとに結果を返します。

例えば、アグリゲータトランスフォーメーションを使用して、組織の各部門の従業員の平均給与を計算できます。アグリゲータトランスフォーメーションで、部門番号のグループを作成し、各グループの従業員の平均給与を計算する式を設定します。

グループ化フィールド

集計式用にデータをグループ化する方法を定義する場合は、グループ化フィールドを使用します。グループ化フィールドは、**［プロパティ］**パネルの**［Group By］**タブで設定します。

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適切な入力フィールド、入出力フィールド、出力フィールドを選択します。複数のグループ化フィールドを選択して、一意のグループの組み合わせごとに新しいグループを作成することができます。これにより、データ統合ではグループごとに定義済みの集計が実行されます。

値のグループ化を行うと、データ統合ではグループごとに 1 つの行が生成されます。値のグループ化を行わない場合、データ統合ではすべての入力行に対して 1 つの行が返されます。

アグリゲータトランスフォーメーションが詳細クラスタで実行され、入力フィールドと出力フィールドがフィールドごとにグループ化されていない場合、このトランスフォーメーションでは各グループの最後の行が集計結果とともに返されない可能性があります。

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フィールドを選択した場合、データ統合ではフィールドの順序に基づいてグループ化の順序が決定されます。グループの順序は集計結果に影響を及ぼす場合があるため、適切にグループ化が実行されるようにグループ化フィールドを並べ替えます。また、グループのフィールドを選択した後にフィールド順を変更することもできます。

例えば、TOTAL_QTY および TOTAL_PRICE という集計フィールドを作成して、それぞれの項目の合計数と合計価格をストアごとに保存するとします。フィールドごとに次の式を定義します。

- TOTAL_QTY: `SUM (QTY)`
- TOTAL_PRICE: `SUM (QTY*PRICE)`

グループ化フィールドとして STORE_ID と ITEM を定義します。

入力行には、次のようなデータが含まれます。

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

データ統合は、次の一意の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'

データ統合は、ストア ID、項目名、ストアごとの各項目の合計数、およびストアごとの各項目の合計価格を返します。

STORE_ID	ITEM	TOTAL_QTY	TOTAL_PRICE
101	'AAA'	2	4.90
101	'battery'	6	17.34
201	'battery'	5	8.35
301	'battery'	1	2.45

ソート済みデータ

ジョブのパフォーマンスを高めるために、ソート済みデータを使用するよう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。ソート済みデータを処理するためにアグリゲータトランスフォーメーションを設定するには、**[詳細]** タブで **[ソート済み入力]** を選択します。

ソート済みデータを使用するようにアグリゲータトランスフォーメーションを設定する場合は、データフローのこれより前の時点でデータをソートする必要があります。アグリゲータトランスフォーメーションでリレーショナルデータベースのデータを処理する場合は、ソースのソートキーを一意にすることも必要です。データ

が事前に正しくソートされていない場合や、ソートキーが一意でない場合は、マッピングタスクの実行時に予期しない結果を受け取ったりエラーが発生する可能性があります。

マッピングタスクは、ソート済みデータで集計計算を実行するとき、同じグループの連続する行をキャッシュします。このタスクは、別のグループのデータを読み取ると、キャッシュしたグループの集計計算を実行し、次のグループに進みます。

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 フィールドがあり、［ソート済み入力］オプションが選択されているとします。アグリゲータを通して以下のデータを渡すと、マッピングタスクは新しいグループである 201/battery を検出するとすぐに、101/battery グループの 3 つの行について集計を実行します。

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

ソート済みデータを使用しない場合は、マッピングタスクでは、すべてのデータが読み取られた後で集計計算が実行されます。

集計フィールド

集計計算を定義するには、集計フィールドを使用します。

アグリゲータトランスフォーメーションを設定するときに、集計フィールドを各計算の出力用に作成して、データフローで使用します。集計フィールドでは集計関数を使用できます。条件句と集計以外の関数も使用できます。

集計フィールドは、**［プロパティ］**パネルの**［集計］**タブで設定します。集計フィールドを設定するときは、フィールド名、データ型、精度、スケール、説明（オプション）を定義します。説明には最大 4000 文字を含めることができます。実行する計算を定義することもできます。

集計フィールドを設定するときは、トランスフォーメーション内で使用する計算に変数フィールドを使用できます。集計フィールドと変数フィールドにマクロを組み込むこともできます。

詳細モードでは、Group by フィールドが 1 つの行を返して、集計式に STDDEV 関数と VARIANCE 関数が含まれている場合、出力は NULL になります。その原因は、データ統合が Spark 3.2 を使用するためです。出力値 0 を取得するには、マッピングタスクで spark.sql.legacy.statisticalAggregate セッションプロパティを true に設定してください。

集計関数

集計フィールドの式では集計関数を使用できます。

集計関数の詳細については、*『Function Reference』*を参照して下さい。

ネストされた集計関数

ネストされた集計関数は、別の集計関数内の集計関数です。

例えば、次の式では売上が合計されて最大の数値が返されます。

```
MAX( SUM( SALES ))
```

アグリゲータトランスフォーメーションの異なる出力フィールドに、単一レベルの関数を複数含めたり、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。詳細モードでは集計関数をネストできません。

アグリゲータトランスフォーメーションの出力フィールド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フィールド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。

条件句

集計で使用され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 )
```

詳細プロパティ

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を設定できます。詳細プロパティは、セッションログメッセージのトレースレベル、トランスフォーメーションがソート済み入力を使用するかどうか、キャッシュ設定、およびトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
ソート済み入力	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。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。
キャッシュディレクトリ	データ統合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ローカルディレクトリ。 デフォルトでは、データ統合は、データ統合サーバーで Secure Agent $PMCacheDir プロパティに入力されたディレクトリを使用します。新しいディレクトリを指定する場合は、そのディレクトリが存在し、集計キャッシュのための十分なディスクスペースがあることを確認してください。

プロパティ	説明
データキャッシュサイズ	トランスフォーメーションのデータキャッシュサイズ。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。［自動］を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは［Auto］です。
インデックスキャッシュサイズ	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。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。［自動］を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは［Auto］です。
トランスフォーメーション範囲	データ統合が受信データにトランスフォーメーションロジックを適用する方法を指定します。次のいずれかのオプションを選択します。 - 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。データの行が同一トランザクション内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。 - 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、データ統合は受信トランザクションの境界を削除します。データの行がソース内のすべての行に依存している場合は、［すべての入力］を選択します。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。また、複合演算子を使用してプリミティブ子フィールドにアクセスし、その子フィールドを使用して集計計算を実行することもできます。例えば、ドット演算子を使用して構造フィールドの整数にアクセスし、その整数を引数として SUM 関数に渡すことができます。複合演算子の詳細については、*関数リファレンス*の説明を参照してください。

階層フィールドを使用する場合は、次のガイドラインを考慮してください。

- 階層フィールドを Group By フィールドとして使用することはできません。
- 集計計算の出力を配列、マップ、または構造に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例

変更データキャプチャ（CDC）システムですべての株式の最新の価格を取得する必要があるとします。

各株式の最新の価格を取得するには、マッピングを作成し、アグリゲータトランスフォーメーションとジョイナトランスフォーメーションを追加します。

- すべての株式を株式 ID に基づいてグループ化し、各株式 ID の最新の取引時間を計算するようにアグリゲータトランスフォーメーションを設定します。
- ソースデータとアグリゲータトランスフォーメーション出力を使用して、株式 ID と取引時間の自己結合を実行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ます。

ターゲットトランスフォーメーションにより、株式 ID と価格がターゲットファイルに書き込まれます。

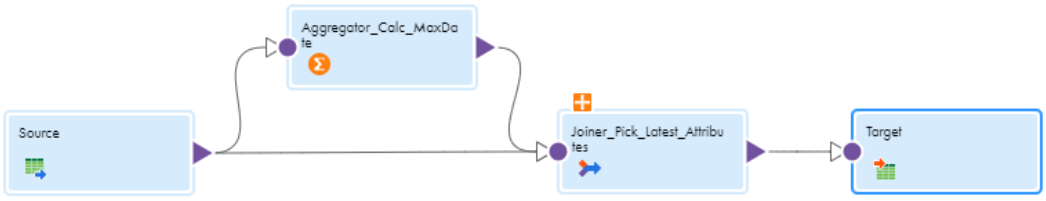
ソースデータ

以下の表に、ソースデータを示します。

STOCK_ID	PRICE	VOLUME	TRADETIME
100	1.12	125	2023.01.20.1245
105	3.45	56	2023.01.18.0821
102	2.29	25	2023.01.19.1118
101	4.56	87	2023.01.17.0901
102	2.35	38	2023.01.20.1649
100	0.99	94	2023.01.20.1355
104	1.88	67	2023.01.20.1730
103	2.11	41	2023.01.17.1211
105	3.40	90	2023.01.18.1525
104	3.40	90	2023.01.20.1634
103	3.40	90	2023.01.20.1526
105	3.40	90	2023.01.19.1745

マッピング設定

次の図は、設定するマッピングを示しています。



次の方法でトランスフォーメーションを設定します。

アグリゲータトランスフォーメーション

以下の表に、アグリゲータトランスフォーメーションで設定するためのプロパティを示します。

プロパティ	設定
受信フィールド	集計されたフィールドが一意の名前を持つように、すべての受信フィールドに「agg_」というプレフィックスを付けるフィールドルールを設定します。
グループ化	agg_stock_id フィールドに基づいてグループ化を行います。
集計	次のプロパティを持つ集計フィールドを 1 つ作成して、各株式 ID の最新の取引時間を返すようにします。 - フィールドタイプ: 出力フィールド - 名前: agg_maxtime - タイプ= string - 精度: 10 次の式を設定します: max(agg_tradetime)

ジョイナトランスフォーメーション

ジョイナトランスフォーメーションでは、次の結合条件を使用して通常の結合を設定します。

- agg_stock_id = STOCK_ID
- agg_maxtime = TRADETIME

ターゲットトランスフォーメーション

以下の表に、ターゲットフォーメーションで設定するためのプロパティを示します。

プロパティ	設定
受信フィールド	次のフィールドルールを設定します。 - すべてのフィールドを含めます。 - プレフィックス「agg_」が付いた**［テキストまたはパターンごとのフィールド］**を除外します。 含まれるフィールドのリストには、すべてのソースフィールドが表示されます: PRICE、STOCK_ID、TRADETIME、VOLUME
ターゲットフィールド	次のターゲットフィールドを追加します: STOCK_ID および PRICE
フィールドマッピング	**［正確なフィールド名］**を使用して自動マッピングを実行します。これにより、受信フィールド STOCK_ID および PRICE がターゲットフィールド STOCK_ID および PRICE にマッピングされます。

ターゲットデータ

次の表は、マッピングによりターゲットファイルに書き込まれるデータを示しています。

STOCK_ID	PRICE
100	0.99
101	4.56
102	2.35
103	3.40

STOCK_ID	PRICE
104	3.40
105	3.40

フィールドはアグリゲータトランスフォーメーションで STOCK_ID に基づいてグループ化されるため、各株式 ID は最新の価格とともに 1 つの行にリストされます。例えば、株式 105 の最近の取引価格は 3.40 ドルです。

第 5 章

クレンジングトランスフォーメーション

クレンジングトランスフォーメーションによって、Data Quality で作成したクレンジングアセットがマッピングに追加されます。クレンジングアセットは、データの形式やコンテンツを標準化するデータトランスフォーメーション操作のセットです。

クレンジングトランスフォーメーションに 1 つのクレンジングアセットを追加します。1 つ以上の入力フィールドをクレンジングアセットにマッピングできます。

クレンジングアセットは、次の操作の 1 つ以上を実行できます。

- 入力データの大文字小文字を変更する。
- 入力データの先頭および末尾のスペースを削除する。
- 入力データから値を削除する。
- 入力データ内の値を検索置換する。
- クレンジング済みのデータを 2 つ以上の入力フィールドから 1 つの新しい出力フィールドにマージします。

クレンジングアセットに複数の操作を設定できます。また、アセットに任意のタイプの操作を複数回追加できます。マッピングでは、定義した順序で入力データフィールドの操作が実行されるため、1 つのクレンジングアセットで入力フィールドデータへの複数の変更を指定できます。

クレンジングトランスフォーメーションは、別の所で設計したデータトランスフォーメーションロジックをマッピングに追加できるという点において、マップレットトランスフォーメーションと類似しています。マップレットと同様に、クレンジングアセットは再利用可能なアセットです。

クレンジングトランスフォーメーションには、クレンジングアセットに含まれるロジックは表示されず、クレンジングアセットを編集することもできません。クレンジングアセットを編集するには、クレンジングアセットを Data Quality で開きます。

クレンジングトランスフォーメーションの設定

マッピングのクレンジングトランスフォーメーションを設定する場合は、最初に、トランスフォーメーションに含めるロジックが含まれているクレンジングアセットを選択します。次に、トランスフォーメーションの 1 つ以上の受信フィールドを、クレンジングアセットによって指定されたターゲットフィールドにマッピングします。

トランスフォーメーションを設定するための手順は、クレンジングアセットによって指定された入力の数によって異なります。

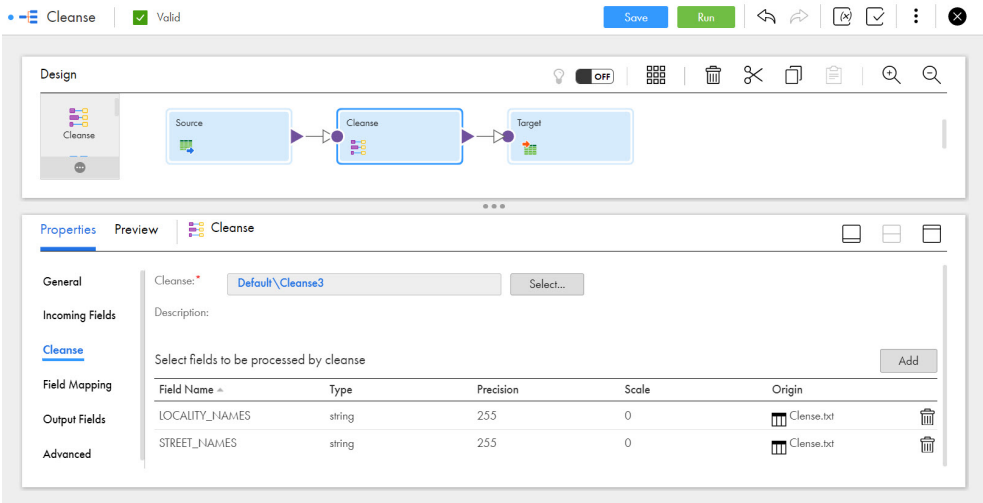
トランスフォーメーションを設定するには、以下のタスクを実行します。

1. クレンジングトランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. **[クレンジング]** タブで、トランスフォーメーションに含めるクレンジングアセットを選択します。

   単一の入力または複数の入力を含むクレンジングアセットを選択できます。

   - 単一の入力を含むクレンジングアセットを含める場合は、アップストリームオブジェクトから 1 つ以上のフィールドを入力として選択します。クレンジングアセットにフィールドをマッピングするには、**[追加]** をクリックします。

     次の図に、クレンジングアセットを選択して入力フィールドを追加する場合に使用するオプションを示します。

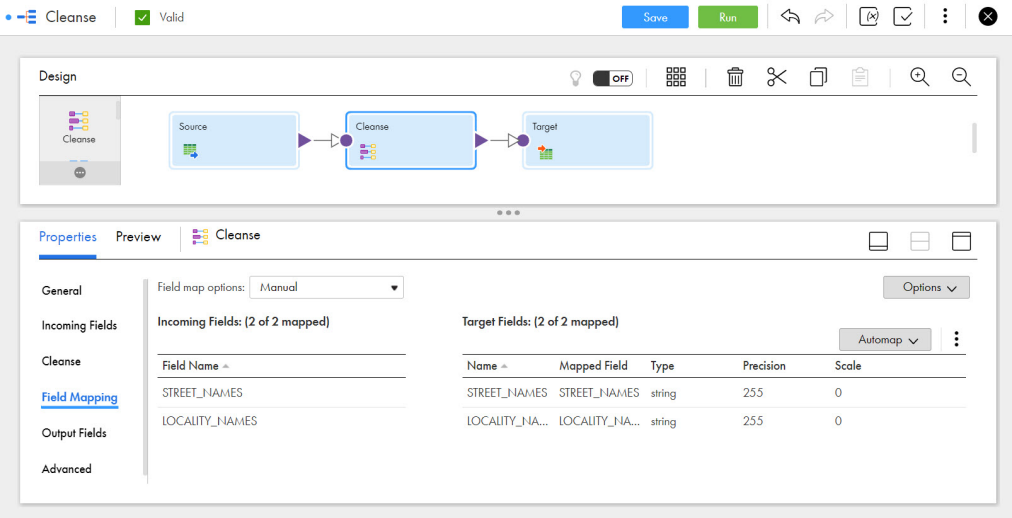


   - 複数の入力を含むクレンジングアセットを含めた場合、[クレンジング] タブに [追加] オプションは表示されません。[フィールドマッピング] タブのオプションを使用して、トランスフォーメーションの入力フィールドをアセットに接続します。
3. **[受信フィールド]** タブで、受信フィールドを検証します。

   デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。
4. **[フィールドマッピング]** タブで、トランスフォーメーションの 1 つ以上の入力フィールドをアセットに接続します。クレンジングアセットによって単一の入力が指定されている場合、データ統合ではターゲットフィールドに入力フィールドが自動的にリンクされます。クレンジングアセットによって複数の入力が指定されている場合は、手動でフィールドに入力フィールドをリンクします。

   クレンジングアセットの入力名には、トランスフォーメーションの入力フィールドの名前が反映されている場合があります。この場合、**[自動マップ]** オプションを使用してフィールドの接続を行うことができます。

次の図に、トランスフォーメーションの入力フィールドをマッピングするために使用するオプションを示します。



5. **［出力フィールド］**タブの出力フィールドのプロパティを確認します。
6. **［全般］**タブでは、オプションとして、クレンジングトランスフォーメーションの名前を変更したり、説明を追加したりすることもできます。**［詳細］**タブで、トランスフォーメーションのトレースレベルを更新することもできます。デフォルトのトレースレベルはノーマルです。

クレンジングアセットの考慮事項

クレンジングアセットは、1 つ以上のクレンジングインスタンスで構成されます。各インスタンスは、アセットによって指定された一連のクレンジング操作に対する 1 つ以上の入力フィールドを表します。

Data Quality のアセットには複数のクレンジングインスタンスを追加できるため、クレンジングトランスフォーメーションは、単一のアセットを持つさまざまな入力フィールドのセットに複数のクレンジング操作を適用することができます。

複数の入力フィールドの指定に使用されているクレンジングアセットを選択する場合は、アセットの構成を理解し、アセットによって各入力で実行されるクレンジング操作のタイプを把握しておく必要があります。クレンジングトランスフォーメーションでは、各アセット入力が発生したインスタンスは識別されません。Data Quality でアセットに複数のインスタンスを定義した場合は、各入力が属するインスタンスのレコードを作成します。アセット入力をトランスフォーメーション入力フィールドに接続するときは、レコードをガイドとして使用します。

データ品質アセットの同期

アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。

アセットのバージョンを同期するには、マッピングでトランスフォーメーションを開き、［プロパティ］パネルでトランスフォーメーショ名を選択します。例えば、クレンジングトランスフォーメーションの［プロパティ］パネルで**［クレンジング］**を選択します。同期が必要な場合は、アセットを同期するように求めるメッセージがデータ統合に表示されます。

アセットのバージョンを同期すると、データ統合には、現在のアセットのフィールド属性をマッピング内の他のアセットにプロパゲートするように求めるメッセージが表示される場合があります。現在のアセットが以前のバージョンの Data Quality で作成されている場合、データ統合にプロンプトが表示される場合があります。

フィールド属性をプロパゲートするアセットを選択します。複数のアセットを選択して、1 回の操作で属性をプロパゲートできます。

注: フィールドのプロパゲートは、現在のバージョンの Data Quality で作成したアセットに対してデフォルトで発生します。

クレンジングトランスフォーメーションのフィールドマッピング

フィールドマッピングを設定して、データを受信トランスフォーメーションフィールドからクレンジングアセットに移動する方法を定義します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをアセットの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドがあれば、そのリンクを削除します。複数の入力を含んだアセットを選択した場合、デフォルトではこのオプションは［手動］になります。
- 自動。同名のフィールドを自動的にリンクします。このオプションが指定されたフィールドを手動でリンクすることはできません。単一の入力を含んだアセットを選択した場合、デフォルトではこのオプションは［自動］になります。
- ［すべてパラメータを使用します］。パラメータを使用してフィールドマッピングを表現します。トランスフォーメーションのクレンジングアセットがパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、［すべてパラメータを使用します］オプションを選択します。
- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。［すべてパラメータを使用します］または［部分的にパラメータを使用します］をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは*フィールドマッピング*型である必要があります。

1 つのマッピング内の複数のクレンジングトランスフォーメーションに同じフィールドマッピングパラメータを使用しないでください。

オプション

［受信フィールド］リストと**［ターゲットフィールド］**リストでフィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。［手動］または［部分的にパラメータを使用します］フィールドマップオプションを選択すると、［自動マップ］オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。
- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合では Cust_Name フィールドと Customer_Name フィールドが自動的にリンクされます。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似する名前のフィールドをマッピングできます。

［自動マップ］ > **［自動マップを取り消す］**をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで**［アクション］** > **［マップ解除］**をクリックします。選択した 1 つ以上のフィールドのマッピングを解除するには、［ターゲットフィールド］コンテキストメニューで**［選択項目をマップ解除］**をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、［ターゲットフィールド］コンテキストメニューで**［マッピングのクリア］**をクリックします。

クレンジングトランスフォーメーションの出力フィールド

出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。クレンジングトランスフォーメーションは、**［フィールドマッピング］**タブでマッピングしたフィールドに基づいて出力フィールドを作成します。

［出力フィールド］タブには、各出力フィールドの名前、タイプ、精度、およびスケールが表示されます。

トランスフォーメーションにより、次の方法で出力フィールドが作成されます。

単一の出力でアセットをクレンジング

トランスフォーメーションによって、クレンジング済みのフィールドが作成されます。

出力フィールド名は、クレンジングトランスフォーメーションの名前が追加されたターゲットフィールドの名前です。例えば、Person_name というターゲットフィールドがあり、クレンジングトランスフォーメーションの名前に Cleanse_TX と入力した場合、クレンジング操作によって Person_name_Cleanse_TX という出力フィールド名が返されます。

複数の出力でアセットをクレンジング

トランスフォーメーションによって、クレンジング済みのフィールドと、アセットで自分がまたは他のユーザーが設定したマージ済み出力フィールドが追加で作成されます。

トランスフォーメーションは、各ターゲットフィールドの名前に_Cleansed のサフィックスを追加しようとします。例えば、ターゲットフィールドが FirstName である場合は、この操作によって FirstName_Cleansed という出力フィールド名が返されます。

注: 現在のバージョンの Data Quality でクレンジングアセットを作成した場合、トランスフォーメーションはサフィックス_*Cleansed* をすべての出力フィールドに適用します。古いバージョンの Data Quality にアセットを作成した場合、トランスフォーメーションが、出力に異なる命名ポリシーを適用する場合があります。詳細については「*出力フィールド名のルールとガイドライン*」を参照してください。

マージされた出力フィールド名は、アセットで設定済みのマージされたフィールドの名前となります。

クレンジングトランスフォーメーションの出力フィールドプロパティを編集することはできません。プロパティを編集するには、Data Quality でクレンジングアセットを開きます。

出力フィールド名のルールとガイドライン

Data Quality でクレンジングアセットを作成し、時間の経過とともにアセットに出力を追加できます。これは、クレンジングトランスフォーメーションが出力フィールドに適用するサフィックスのタイプに影響を与える可能性があります。

トランスフォーメーションで複数の出力を持つクレンジングアセットを使用する場合は、フィールド名がマッピング要件を満たしていることを確認してください。トランスフォーメーションが期待どおりにフィールド名を定義するようにするには、出力フィールドを再度マップする必要がある場合があります。

出力フィールド名を確認するときは、次のルールとガイドラインを考慮してください。

- 一部の古いクレンジングアセットは、単一のクレンジングインスタンスで単一の出力フィールドをサポートしていました。そのようなアセットに出力を追加し、元のインスタンスを更新しない場合、クレンジングトランスフォーメーションは新しい出力にのみ_*Cleansed* のサフィックスを適用します。トランスフォーメーションでは、元のインスタンスの出力にサフィックスを適用しません。

  クレンジングトランスフォーメーションは、アセットをトランスフォーメーションにいつ追加するかに関係なく、このポリシーを適用します。別のインスタンスに出力を追加し、元のインスタンスを変更しない場合、トランスフォーメーションは古い出力フィールドからサフィックスを削除します。
- クレンジングアセットの元のインスタンスに出力を追加すると、クレンジングトランスフォーメーションがすべての出力名に_*Cleansed* サフィックスを適用します。

  クレンジングトランスフォーメーションは、アセットの経過時間に関係なく、またアセットをトランスフォーメーションにいつ追加するかに関係なく、このポリシーを適用します。

詳細プロパティ

クレンジングトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。

第 6 章

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を使用して、機密性が高い実稼働データを、プロダクション環境以外での実際の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メーションは、カラムごとに設定されたマスキングルール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メーションは、パッシブトランスフォーメーションです。

データマスキングトランスフォーメーションでは、ポートに設定したソースのデータ型およびマスキングの種類に基づいてマスキングルールが提供されます。文字列の場合は、文字列内で置き換える文字およびマスク内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。統合サービスでは、マスキングルールで設定したロケールに基づいて文字を置き換えます。

データマスキングトランスフォーメーションを使用するには、適切なライセンスが必要です。

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。

以下のマスキング方法のいずれかを選択します。

クレジットカードマスキング

クレジットカード番号を含む文字列データ型のカラムに、クレジットカードのマスク形式を適用します。

電子メールマスキング

電子メールアドレスを含む文字列データ型のカラムに、電子メールのマスク形式を適用します。

詳細電子メールマスキング

姓、名、ドメイン名を使用した実際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。マスクできるのは、文字列データ型です。

IP **アドレスマスキング**

IP アドレスを含む文字列データ型のカラムに、IP アドレスのマスク形式を適用します。

キーマスキング

同じソースデータおよびシード値に対して確定的な結果を生成します。キーマスキングは、日時、文字列、数値データ型に適用できます。

電話マスキング

電話番号を含む文字列データ型のカラムに、電話番号のマスク形式を適用します。

ランダムマスキング

同じソースデータとマスク形式に対してランダムな結果を生成します。ランダムマスキングは、日時、文字列、数値データ型に適用できます。

SIN **マスキング**

社会保険番号を含む文字列データ型のカラムに、社会保険番号のマスク形式を適用します。

SSN **マスキング**

社会保障番号を含む文字列データ型のカラムに、社会保障番号のマスク形式を適用します。

カスタム置換マスキング

データのカラムを、カスタム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。カスタム置換マスキングは、文字列データ型を含むカラムに適用できます。

依存マスキング

別の入力カラムのディクショナリから返された値に基づいて、フィールド値をカスタムディクショナリの値に置き換えます。マスクできるのは、文字列データ型です。

置換マスキング

データのカラムを、デフォルト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。置換マスキングは、文字列データ型を含むカラムに適用できます。

URL **マスキング**

URL を含む文字列データ型のカラムに、URL のマスク形式を適用します。

マスキング方法の設定プロパティ

特定のマスキング方法をどのように動作させるかを定義するには、マスキング方法のさまざまなプロパティを設定します。データマスキングトランスフォーメーションでは、選択したマスキング方法および設定した特定の構成に基づいてデータがマスクされます。

表示される設定プロパティは、マスキング方法とデータ型によって決まります。例えば、文字列データにブラーを適用することはできません。ランダムマスキング方法を使用するときは、シード値を選択できません。

再現可能な出力

再現可能な出力は、データマスキングトランスフォーメーションから返される、一貫した値のセットです。

再現可能な出力は確定的な値を返します。例えば、名のカラムに対して再現可能な出力を設定するとします。データマスキングトランスフォーメーションでは、同じ名前がワークフローに表示されるたびに、同じマスクされた値が返されます。

再現可能なマスキングを設定できるのは、文字列データ型に、ランダムマスキング方法、置換マスキング方法、特殊マスク形式のいずれかを使用するときです。**[再現可能]** を選択してシード値を入力し、再現可能なマスキングを設定してください。

再現可能な出力をキーマスキング方法に設定することはできません。

ディクショナリの使用の最適化

[ディクショナリ出力の最適化] オプションは、マスキングのためのディクショナリ値の使用を増やし、ターゲット内の重複するディクショナリ値を減らします。

置換マスキングまたはカスタム置換マスキングを実行する場合は、ディクショナリの使用を最適化するという選択ができます。ワークフローは、選択したディクショナリの一部の値を使用してソースデータをマスクします。これらのディクショナリ値は、すべてのソースデータがターゲットでマスクされるように、複数のエントリに対して使用される場合があります。ディクショナリの使用を最適化すると、ディクショナリ値が重複して使用される可能性が低減されます。ディクショナリの出力を最適化するには、再現可能な出力のマスキングルールを設定する必要があります。

シード

シード値は、マスクされた値を生成する開始ポイントです。

データマスキングトランスフォーメーションでは、1～999 の範囲の乱数であるデフォルトのシード値が作成されます。別のシード値を入力することもできます。別のソースデータから同じマスクされたデータ値を返すには、カラムに同じシード値を適用します。例えば、4 つのテーブルに同じ Cust_ID カラムがあり、そのすべてで同じマスクを適用した値を出力するとします。このような場合は、4 つすべてのカラムを同じシード値に設定できます。

シード値をパラメータとして入力できます。シード値パラメータの名前は$$で始まる必要があります。名前にアンダースコア（_）を含めることはできますが、他の特殊文字を含めることはできません。必要なパラメータと値をパラメータファイルに追加し、実行時にパラメータファイル名を指定します。

注: シード値をパラメータとして入力する場合は、マッピングタスクでマッピングを実行する必要があります。シード値パラメータを含むマッピングを実行すると、パラメータ値を読み取ることができないため、マッピングは誤った値を使用します。

一意の置換

一意の置換マスキングを設定すると、一意のソース値ごとに一意のディクショナリ値が使用されます。

ソース値を一意のディクショナリ値でマスクするには、一意の置換マスキングを設定します。ソース値が特定のディクショナリ値でマスクされている場合、他のソース値はこのディクショナリ値ではマスクされません。

例えば、ソースデータの［名前］カラムに John という複数のエントリが含まれているとします。再現可能なマスキングを設定した場合、John というすべてのエントリは、Xyza などの同じディクショナリ値を取得します。ただし、他のソース値も同じディクショナリ値でマスクされる可能性があります。つまり、Jack というソースエントリで、ディクショナリ値 Xyza が使用されることもあります。その結果、John および Jack というすべてのエントリで同じディクショナリ値が使用されます。一意の置換マスキングを設定した場合、John というすべてのソース値で Xyza というディクショナリ値が使用されると、他のソース値では同じディクショナリ値が使用されなくなります。

一意の置換マスキングには、ストレージテーブルに対するストレージ接続が必要です。ストレージテーブルには、一意の置換マスキングに必要なソースからディクショナリへの値のマッピング情報が含まれています。

注: ソースデータにディクショナリよりも多くの一意の値が含まれている場合、すべてのソースデータをマスクするために必要な一意のディクショナリ値が足りないため、マスキングは失敗します。

マスクフォーマット

キーマスキングまたはランダムマスキングを文字列データ型に設定するときは、出力カラムの各文字を、アルファベット、数字、英数字に制限するようにマスク形式を設定します。

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は、ソース文字列の最後の部分の文字はマスクされません。

マスク形式を設定するときは、ソースフィルタ文字またはターゲットフィルタ文字を設定し、それによってマスク形式を使用します。

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。

次の表に、マスク形式文字を示します。

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。
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。

例えば、部署名の形式が次のとおりであるとします。

```
nnn-<部署名>
```

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+AAAAAAAAAAAAAAA
```

Data Masking トランスフォーメーションによって、最初の 3 文字が数字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、アルファベットに置き換えられます。

ソースフィルタ文字

キーマスキングまたはランダムマスキングを文字列データ型に設定するときは、ソースフィルタ文字を設定して、マスクする文字を選択します。

ある文字をソースフィルタ文字として設定すると、その文字は、ソースデータに出現するたびにマスクされます。ソース文字列における文字の位置や文字数にかかわらず、設定することができます。ソースフィルタ文字を設定しない場合、マスキングによってカラム内のすべてのソース文字が置き換えられます。

ソースフィルタ文字では、大文字と小文字が区別されます。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、一意のデータを返さないことがあります。

ターゲットフィルタ文字

キーマスキングまたはランダムマスキングを文字列データ型に設定するときは、ターゲットフィルタ文字を設定して、ターゲットカラムに表示する文字を制限します。

データマスキングトランスフォーメーションによって、ターゲットの文字はターゲットフィルタ文字で置き換えられます。例えば、各マスクに英文字の大文字をすべて含めるよう設定するには、次の文字を入力します。
ABCDEFGHIJKLMNOPQRSTUVWXYZ

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。

範囲

数値または日時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。

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、フィールド精度以下である必要があります。 デフォルトの範囲は、1 からフィールド精度長までです。

日付範囲

日付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。

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。下限値は、ソース値より小さい値に関する偏差です。 上限値は、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。Data Masking 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。

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。選択できるのは、年、月、日、時、分、または秒です。 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。ソース日付が 2006 年 2 月 2 日の場合、データマスキングトランスフォーメーションは 2004 年 2 月 2 日から 2008 年 2 月 2 日の日付を返します。

データマスキングトランスフォーメーションの接続

マスキング方法でカスタムディクショナリ接続またはストレージ接続を使用するには、データマスキングトランスフォーメーションに接続を追加する必要があります。

カスタムフラットファイルまたはリレーショナルディクショナリを使用できます。また、一意の置換マスキング方法には、ソースからディクショナリ値へのマッピングのためのストレージ接続が必要です。

データマスキングトランスフォーメーションの**［マスキングルール］**タブに接続を追加します。カラムにマスキング方法を設定する場合は、［マスキングルール］タブで指定したディクショナリとストレージ接続を使用できます。

［マスキングルール］タブには次のような接続フィールドが表示されます。

- リレーショナルディクショナリ接続
- フラットファイルディクショナリ接続
- ストレージ接続

2022 年 4 月リリース以前に作成されたマッピングをエクスポートした場合、マッピングのデータマスキングトランスフォーメーションにディクショナリとストレージ接続情報が含まれない場合があります。マッピングをインポートすると、フィールドは空白の状態で表示されます。2022 年 4 月リリース以降の環境にマッピングをインポートする場合にこの問題を回避するには、マッピングをエクスポートする前に、マッピングを開いて保存します。エクスポートされたマッピングには、インポート時にディクショナリとストレージ接続情報が表示されます。接続は、**［依存関係の表示］**ページの**［用途］**タブにも表示されます。

注: ［保存］ボタンを有効にするには、マッピングを変更する必要がある場合があります。

クレジットカードマスキング

クレジットカードマスキングは、組み込みのマスク形式を適用してクレジットカード番号をマスクします。特定のクレジットカード会社の形式で、マスクされた数値を作成できます。

データマスキングトランスフォーメーションでは、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は 13〜19 桁である必要があります。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。

クレジットカード番号の最初の 6 桁で、クレジットカード会社が識別されます。元のクレジットカード会社を維持するか、別のクレジットカード会社を選択して、マスキング結果に表示できます。

クレジットカードパラメータ

次の表は、クレジットカードマスキングに設定できるパラメータです。

パラメータ	説明
再現可能	タスクを複数回実行する場合、または複数のテーブルに存在するフィールドに対してマスク値を生成する場合は、同じマスク値を返します。
シード値	再現可能な出力を生成する開始番号。1 - 999 の数値を入力します。デフォルトのシード値は 190 です。シード値をパラメータとして入力できます。
発行者を維持	マスクされたクレジットカードと同じクレジットカードタイプを返す。例えば、ソースクレジットカードが Visa カードの場合、マスクされた Visa 形式のクレジットカード番号を生成する。
マスク発行者	ソースクレジットカードタイプを他のクレジットカードタイプに置き換える。［発行者を維持］を無効にした場合は、置き換える側のクレジットカードのタイプを選択する。AMEX、VISA、MASTERCARD などのクレジットカードを選択することができる。デフォルトは ANY である。

電子メールマスキング

データマスキングトランスフォーメーションでは、電子メールアドレスをマスクする場合、ランダムな文字で構成される電子メールアドレスが返されます。

例えば、Georgesmith@yahoo.com が KtrIupQAPyk@vdSKh.BICJ などとしてマスクされます。

電子メールマスク形式を使用するときは、シード値を設定する必要があります。シード値は 1〜999 のランダムな数値であり、マスクされた値の生成の開始ポイントです。別のシード値を入力することもできます。別のソースデータから同じマスクされたデータ値を返すには、カラムに同じシード値を適用します。例えば、4 つのテーブルに同じ Cust_ID カラムがあるとします。そのすべてで、同じようにマスクされた値を出力する必要があります。このような場合は、4 つすべてのカラムを同じシード値に設定してください。

詳細電子メールマスキング

姓、名、ドメイン名を使用して、現実的な電子メールアドレスを作成できます。

詳細電子メールマスキングを設定する際は、電子メールアドレス内のユーザー名とドメイン名をマスクするパラメータを設定することができます。例えば、ソーステーブルに First_Name と Last_Name というカラムがあるとします。この場合、First_Name の最初の文字と Last_Name の 7 文字を含む電子メールアドレスを設定できます。電子メールアドレスのドメイン名を定義します。マスキングタスクにより、次の構文でアドレスが作成されます。

VSingh@mycompany.com

次の表は、高度な電子メールマスキングに設定できるパラメータを示しています。

パラメータ	説明
再現可能	タスクを複数回実行する場合、または複数のテーブルに存在するフィールドに対してマスク値を生成する場合は、同じマスク値を返します。
シード値	再現可能な出力を生成する開始番号。1 - 999 の数値を入力します。デフォルトのシード値は 190 です。シード値をパラメータとして入力できます。
名	電子メール名の最初の部分として使用するカラム名。電子メールには、選択したカラムのマスク値が含まれます。
名の長さ	電子メールアドレスに含める名の文字数。
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の、ドット、ハイフン、アンダースコアなどの区切り文字。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。
姓	電子メール名で使用する、マスクされたカラムの名前。電子メールには、選択したカラムのマスク値が含まれます。
姓の長さ	電子メールアドレスに含める姓の文字数。
ドメイン名	gmail.com などのインターネットプロトコル（IP）リソースを表す文字列値。

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。11.12.23.34 は例えば 75.32.42.52 に、10.23.24.32 は例えば 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。

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、同じシード値を使用します。

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形式に影響を与えるマスキングルールを設定できます。キーマスキングでは、数値、文字列、日時のデータ型をマスクできます。

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。

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、シードを必要とするマスキングアルゴリズムが適用されます。

文字列のキーマスキングを設定すると、再現可能な出力が生成されます。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。マスク形式を定義するには、ソースフィルタ文字とターゲットフィルタ文字を定義します。ソースフィルタ文字では、マスクするソース文字を定義します。ターゲットフィルタ文字では、ソースフィルタ文字をマスクする文字を定義します。

電話番号マスキング

電話番号はランダムな数字でマスクできます。

データマスキングトランスフォーメーションでは、元の電話番号の形式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。アルファベットと特殊文字はマスクされません。

電話番号をマスクするときは、再現可能な出力を設定できます。[再現可能] を選択してシード値を入力してください。

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形式に影響を与えるマスキングルールを設定できます。

ランダムマスキングでは、日時、数値、文字列の値をマスクできます。

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。

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、フィールド精度に基づいて、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。

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、マスクフォーマットを定義します。どのソース文字をマスクするのかを定義して、それをマスクする文字を定義するには、ソースとターゲットのフィルタ文字を設定します。

社会保険番号マスキング

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次の形式になります。

```
xxx-xxx-xxx
```

SIN の開始桁

マスクされた SIN の最初の桁を定義できます。

開始桁を設定するには、[開始桁] を有効にして桁を入力します。データマスキングトランスフォーメーションは、入力した番号で始まる、マスクされた社会保険番号を作成します。

再現可能な社会保険番号

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フィールドを設定すると、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。

社会保険番号ごとに一意の値が返されます。

社会保障番号（SSN）マスキング

社会保障番号マスキングでは、組み込みのマスク形式が適用されて社会保障番号が変更されます。

データマスキングトランスフォーメーションでは、社会保障庁による最新の High Group List に基づいて、無効な社会保障番号が生成されます。High Group List には、社会保障局が発行した有効な番号が記載されています。Data Masking トランスフォーメーションでは、以下の場所の High Group List にアクセスします。

<インストールディレクトリ>\Informatica Cloud Secure Agent\highgroup.txt

データマスキングトランスフォーメーションでは、High Group List にない社会保障番号が生成されます。社会保障庁では、High Group List を毎月更新しています。High Group List の最新バージョンは、次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。番号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、High Group List 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。

カスタム置換マスキング

カスタム置換マスキングを使用して、作成したフラットファイルまたはリレーショナルディクショナリの実際のテストデータにプロダクションデータを置き換えます。

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。例えば、男性と女性の名が含まれるディクショナリを作成できます。このディクショナリを使用して、男性と女性の両方の名が含まれるカラムに対する置換マスキングを実行します。

カスタム置換マスキングを設定して、ターゲットカラムを一意のソースカラム値ごとに一意のマスク値に置き換えることができます。一意のマスキングには、ソースからディクショナリ値へのマッピングのためのストレージ接続が必要です。

マスキングルールの割り当てでディクショナリまたはストレージ接続を使用する前に、ディクショナリとストレージ接続をトランスフォーメーションに追加する必要があります。**[マスキングルール]** タブでトランスフォーメーションに接続を追加します。フラットファイルディクショナリの場合は、**[設定]** | **[接続]** ビューからフラットファイルディクショナリへの接続を作成し、その接続をトランスフォーメーションに追加します。

カスタム置換マスキングを設定する場合は、ディクショナリタイプとディクショナリ接続を選択します。次に、使用するカラムをディクショナリから選択します。英語以外の文字をサポートするには、フラットファイル接続から別のコードページを使用できます。

マスキングタスクが機能するためには、フラットファイル接続のコードページと Secure Agent システムのコードページに互換性がある必要があります。

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シード値を設定する必要があります。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。

注: マッピングを実行する前に、ディクショナリファイルが次の場所にあることを確認します: <Secure Agent インストールディレクトリ>\apps\Data_Integration_Server\data

カスタム置換パラメータ

次の表では、カスタム置換マスキングに設定するパラメータについて説明します。

パラメータ	説明
フラットファイルディクショナリ リレーショナルディクショナリ	使用するカスタムディクショナリのタイプを選択します。トランスフォーメーションには、必要なディクショナリ接続が含まれている必要があります。 フラットファイルを選択した場合は、ディクショナリファイルを指すディレクトリとのフラットファイル接続を作成する必要があります。 ランタイム環境ですべての Secure Agent がフラットファイルディクショナリを使用できるようにするには、ファイルが次の場所にあることを確認します。 <Secure Agent インストールディレクトリ>\apps\Data_Integration_Server\data
ディクショナリ	選択するカスタムディクショナリ。 選択した内容に応じて、リストには、リレーショナルまたはフラットファイルディクショナリが含まれます。
ディクショナリカラム	カスタムディクショナリからの出力カラム。フラットファイルディクショナリの場合、フラットファイルにカラムヘッダーが含まれていると、ディクショナリカラムを選択できます。
ソート順	リレーショナルディクショナリに使用できます。エントリのソートの基準となるディクショナリカラム。ディクショナリのエントリの順序が変更された場合でも確定的な結果を生成するには、ソートカラムを指定します。例えば、リレーショナルディクショナリを移動してエントリの順序が変わった場合は、シリアル番号カラムでソートを行って、一貫したデータのマスクを実行します。 **注**: 選択したカラムには、一意の値が含まれている必要があります。重複する値を含む可能性のあるカラムを使用してデータの並べ替えを行わないでください。
ルックアップ入力カラム	オプション。ディクショナリでルックアップ操作を実行するソース入力カラム。
ルックアップディクショナリカラム	ルックアップ入力カラムの値を入力する場合に必要です。入力ポートと比較するディクショナリカラム。ソースは、ルックアップ入力とルックアップディクショナリの値が一致するディクショナリ行の値に置き換えられます。
ルックアップエラー定数	オプション。ディクショナリからのルックアップ条件に一致するものがない場合に設定できる定数値。デフォルトは空の文字列です。
再現可能	タスクを複数回実行する場合、または複数のテーブルに存在するフィールドに対してマスク値を生成する場合は、同じマスク値を返します。
シード値	再現可能な出力を生成する開始番号。1 - 999 の数値を入力します。デフォルトのシード値は 190 です。シード値をパラメータとして入力できます。

パラメータ	説明
ディクショナリの使用の最適化	ディクショナリからのマスクされた値の使用を増やします。**[再現可能]** オプションを選択した場合に使用できます。一意の置換を有効にした場合、このプロパティは使用できません。
一意	再現可能な置換に使用できます。ターゲットカラムを、一意のソースカラム値ごとに一意のディクショナリ値で置き換えます。ディクショナリファイルよりもソースに一意の値が多い場合、データマスキング操作は失敗します。デフォルトは一意でない置換です。

依存マスキング

依存マスキングは、データのカラムを、別のカラムのデータをマスキングするために使用するカスタムディクショナリの値に置き換えます。依存マスキングを使用するには、少なくとも 1 つの他のソースカラムをカスタム置換ルールでマスキングする必要があります。

例えば、カスタム置換ルールを使用してソースデータの名前カラムをマスキングします。Personal_Information ディクショナリの［名前］カラムの値で値をマスキングするようにルールを設定します。

別のカラムで依存マスキングを設定して、同じディクショナリ内の対応するカラムの値でソースをマスキングすることができます。例えば、［年齢］カラムに依存マスキングを適用します。［名前］カラムを依存カラムとして選択します。次に、Personal_Information ディクショナリから対応するカラムを依存出力カラムとして選択できます。ディクショナリから［年齢］カラムを選択すると、マスキングルールは名前の値に対応する年齢の値を使用します。

依存マスキングパラメータ

ソースカラムに依存マスキングを適用するには、少なくとも 1 つのカラムをカスタム置換ルールでマスキングする必要があります。

以下の表は、依存マスキングに設定することができるパラメータを示します。

プロパティ	説明
依存カラム	ソースカラムに関連付けるカスタム置換マスキング用に設定された入力カラム。リストからカラムを選択します。置換マスキングを使用して設定したカラムがリストに表示されます。
依存出力カラム	ソースデータカラムをマスキングするために使用するディクショナリカラム。依存カラムをマスキングするために使用されるディクショナリのカラムを一覧表示します。ディクショナリカラムのリストから必要なカラムを選択します。

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ときは、データに基づいて置換マスキングのタイプを選択します。

次の表に、使用可能な置換マスキングのタイプを示します。

名前	説明
名前の置換	名前のディクショナリファイルのデータでソースデータを置き換えます。
女性の名前の置換	女性の名前のディクショナリファイルのデータでソースデータを置き換えます。
男性の名前の置換	男性の名前のディクショナリファイルのデータでソースデータを置き換えます。
姓の置換	姓のディクショナリファイルのデータでソースデータを置き換えます。
役職の置換	役職のディクショナリファイルのデータでソースデータを置き換えます。
米国郵便番号の置換	米国郵便番号のディクショナリファイルのデータでソースデータを置き換えます。
番地の置換	番地のディクショナリファイルのデータでソースデータを置き換えます。
都市の置換	米国の都市名のディクショナリファイルのデータでソースデータを置き換えます。
州の置換	米国の州名のディクショナリファイルのデータでソースデータを置き換えます。
国の置換	国名のディクショナリファイルのデータでソースデータを置き換えます。

データマスキングトランスフォーメーションは、ディクショナリファイルでルックアップを実行し、ソースデータをディクショナリのデータに置き換えます。Secure Agent をダウンロードするときに、ディクショナリファイルをダウンロードしてください。ディクショナリファイルは次の場所に保存されます。

```
<Secure Agent installation directory>\apps\Data_Integration_Server\data
```

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シード値を設定する必要があります。

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。

URL アドレスマスキング

Data Masking 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。

マスクルールパラメータ

パラメータは、マッピング内の値のプレースホルダです。マッピングでデータマスキングトランスフォーメーションオブジェクトを設定するときに、マスクルールパラメータを使用できます。

ソース接続情報がない場合は、マスクルールパラメータを使用します。マッピングタスクを実行するときに、マスキング方法をソースフィールドに割り当てる場合にも、マスクルールパラメータを使用できます。例えば、マッピング内のソーストランスフォーメーションオブジェクトでパラメータが使用されている場合は、マッピングを作成するときにマスキング方法を割り当てることはできません。ソース接続を使用してマッピングを作成してから、追加フィールドをマスク対象のソースに追加できます。このマッピングを作成するときに、マスクルールパラメータを使用します。その後、複数のタスクを作成し、同じマッピングに基づいてさまざまなデータをマスクできます。

パラメータの詳細については、「*マッピング*」を参照してください。

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を作成して、ソースデータをマスクします。トランスフォーメーションオブジェクトにマスキング方法を割り当てるか、マッピングの実行時に、マスクルールパラメータを入力してマスキング方法を割り当てることができます。

Mapping Designer 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作成します。マッピングを作成するときには、設定するソーストランスフォーメーションとターゲットトランスフォーメーションが事前にキャンバス上に存在しています。マスクするソースデータを表すように、ソーストランスフォーメーションを設定します。マスクされたデータを格納するターゲット接続を表すように、ターゲットトランスフォーメーションを設定します。

1. トランスフォーメーションパレットからマッピングキャンバスにデータマスキングトランスフォーメーションをドラッグします。
2. データマスキングトランスフォーメーションオブジェクトをデータフローに接続します。
3. Mapping Designer で、データマスキングトランスフォーメーションオブジェクトを選択します。

   ［プロパティ］タブにプロパティが表示されます。
4. **［全般］**タブで、トランスフォーメーションオブジェクトの名前と、説明（省略可能）を入力します。
5. **［受信フィールド］**タブで、ターゲットにコピーするデータを定義するフィールドルールを設定します。

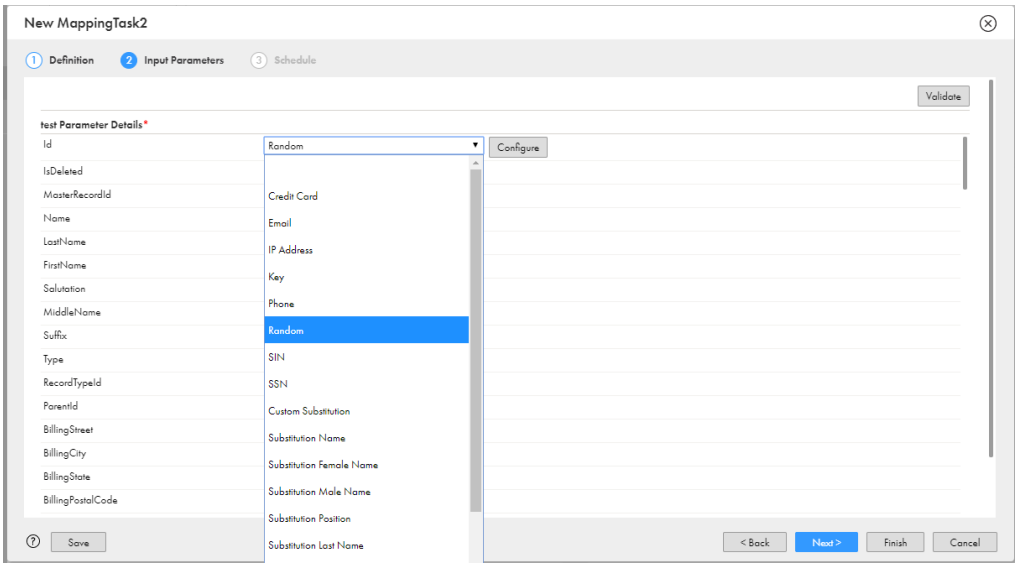
   デフォルトでは、トランスフォーメーションにすべてのフィールドが含まれます。
6. **［マスキングルール］**タブでは、以下のプロパティを設定することができます。
   - パラメータ。実行時にマスキング方法を割り当てるには、別のマスクルールパラメータを追加または作成します。
   - リレーショナルディクショナリ接続。カスタム置換マスキングにリレーショナルディクショナリを使用するには、リレーショナル接続のリストからディクショナリ接続を選択します。
   - フラットファイルディクショナリ接続。カスタム置換マスキングにフラットファイルディクショナリを使用するには、フラットファイル接続のリストからディクショナリ接続を選択します。
   - ストレージ接続。一意の置換マスキングを設定するには、接続のリストからストレージ接続を選択します。
   - 追加。マッピング内のマスキング方法を設定するには、**［追加］**をクリックして、マスクするフィールドを選択します。**［設定］**をクリックし、必要なマスキング方法を選択して設定します。マッピングにマスキング方法を割り当てた場合は、実行時にマスキング方法を編集することはできません。

7. ターゲットトランスフォーメーションを選択し、受信フィールドをターゲット内のフィールドにマップします。

   デフォルトでは、データマスキングトランスフォーメーションによって、ソーストランスフォーメーションオブジェクトから含めるフィールドにプレフィックス"out_"が追加されます。フィールド名が一致しないため、自動フィールドマッピングを使用してターゲットトランスフォーメーションオブジェクトのフィールドをマッピングすることはできません。自動フィールドマッピングを使用する場合は、ターゲットトランスフォーメーションの受信フィールド名を変更します。パターン名変更のオプションを選択し、パターンに"out_/"を入力してプレフィックスを削除します。

8. マッピングを保存して実行します。

   マッピングでマスクルールパラメータを使用した場合は、マッピングタスクの実行時にマスキング方法を割り当てて設定します。
   次の画像は、マスクルールパラメータを使用するマッピングタスクの 2 番目の手順を示しています。



注: マッピングタスクのカラムにマスキング方法を割り当ててから、マッピングのカラムの割り当てを変更した場合、マッピングタスクの設定が優先されます。マッピングタスクでカラムのルール割り当ての選択を解除した場合、マッピングタスクはカラムに割り当てられたマスキング方法をマッピングで使用します。

マスクされた一貫性のある出力

ソースデータをマスクするためにさまざまなツールを使用する場合があります。

次のツールを使用すると、同じソースデータから同じようにマスクされた出力を生成できます。

- Informatica Intelligent Cloud Services
  Informatica Intelligent Cloud Services で次のツールを使用できます。
  - Informatica Intelligent Cloud Services 上のマスキングタスク
  - Informatica Intelligent Cloud Services 上でデータマスキングトランスフォーメーションを使用したマッピングが含まれるマッピングタスク
  - Informatica Intelligent Cloud Services 上の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

- Test Data Management（オンプレミス）
- データマスキングトランスフォーメーションを含む PowerCenter マッピング

ルールおよびガイドライン

マッピング、タスク、またはワークフローを実行してマスクされた一貫性のある出力を生成する前に、次のルールおよびガイドラインを確認してください。

- 置換マスキングルールを使用して、マスクされた一貫性のある出力を生成します。
- マスキングルールでは、同じディクショナリを使用する必要があります。
- 繰り返し可能なオプションは［オン］に設定する必要があります。
- 同じシード値を使用します。

置換マスキングルールはディクショナリの値を使用して、マスクされた出力を作成します。Informatica Intelligent Cloud Services とオンプレミスの Test Data Management のデフォルトのディクショナリは同じです。同じ置換ルールを使用すると、ワークフローは同じディクショナリを使用してソースデータを置き換えます。そのため、同じシード値を使用すると、ディクショナリが同じであれば、すべての行に同じ代替値が使用されます。

Informatica Intelligent Cloud Services では、<Secure Agent のインストールディレクトリ>\apps\Data_Integration_Server\data にディクショナリファイルがあります。

オンプレミスの Test Data Management では、<Informatica のインストールディレクトリ>\server\infa_shared\LkpFiles にディクショナリファイルがあります。

繰り返し可能なオプションを［オン］に設定して、タスクまたはワークフローが同じソース値に対してディクショナリ値を繰り返すようにする必要があります。

例

次の例を検討します。

ソースデータには、ターゲットデータの氏名がマスクされるようにマスクする必要がある名カラムと姓カラムが含まれます。

次の方法を使用すると、マスクされた出力を生成できます。

- Informatica Intelligent Cloud Services でマスキングタスクを実行します。
- Informatica Intelligent Cloud Services でデータマスキングトランスフォーメーションが含まれるマッピングタスクを実行します。
- Informatica Intelligent Cloud Services でデータマスキングトランスフォーメーションが含まれるマッピングを実行します。
- Test Data Management でデータマスキングプランを実行します。
- データマスキングトランスフォーメーションが含まれる PowerCenter マッピングを実行します。

マスクされた出力を生成するには、次の上位タスクを実行します。

1. 名カラムをマスクするには、名の置換マスキングルールを使用します。繰り返し可能なオプションを［オン］に設定します。シード値を入力します。
2. 姓カラムをマスクするには、姓の置換マスキングルールを使用します。繰り返し可能なオプションを［オン］に設定します。シード値を入力します。
3. セットアップで利用可能なデフォルトのディクショナリを使用します。ディクショナリを変更しないでください。

Informatica Intelligent Cloud Services、Test Data Management のワークフロー、または出力を生成する PowerCenter マッピングで、マスキングタスク、マッピング、またはマッピングタスクを実行する場合は、同じソースデータに対してマスクされた同じ出力を生成します。

データマスキングトランスフォーメーション 例

非プロダクション環境でテストする場合は、現実的なデータが必要です。プロダクションデータには機密データのカラムが含まれます。このデータを使用すると、機密データが危害を受けるリスクを負う可能性があります。データマスキングトランスフォーメーションを使用して、機密データをマスクしてから、データをテスト環境で使用してください。

プロダクションデータに、テーブル Personnel_Information および次のデータが含まれているとします。

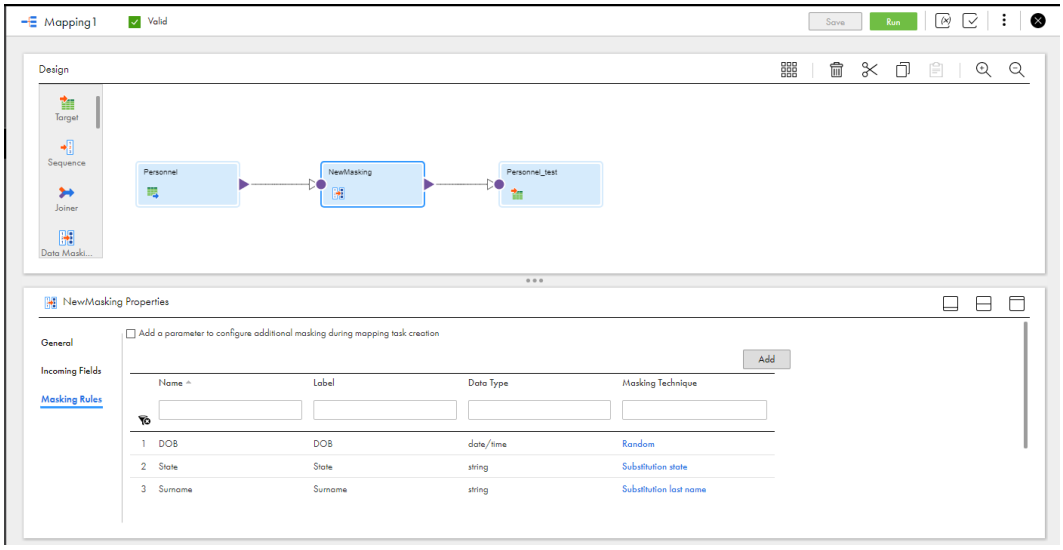
First Name	Surname	DOB	Address	State
Uma	Hilel	03/12/1985	24, Atkins Avenue	CA
John	Sen	07/15/1963	49, Wheeler Road	MN
Keiko	Burns	11/24/1989	13, Draker Drive	PA
Isadora	Buckley	08/16/1990	20, Fountain Center	CA

Mapping Designer では、テーブル Personnel_Information のソーストランスフォーメーションとして *Personnel* を追加します。ターゲットトランスフォーメーション Personnel_test を追加してください。

データマスキング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、データフローに接続します。

Surname、DOB、State の各カラムをマスクして、機密データを確実にマスクする必要があります。姓の置換のマスキング方法を使用すると、Surname カラムをマスクできます。このマスキング方法では、カラム内のデータが姓のディクショナリファイルのデータで置き換わります。DOB カラムのマスクには、ランダム日付のマスキング方法を使用できます。State カラムをマスクするには、都道府県の置換マスキング方法を使用します。

このマスキング方法では、カラム内のデータが米国の州名のディクショナリファイルのデータで置き換わります。



マッピングの実行が正常に完了すると、マスクされたデータを出力テーブル Personnel_test で確認できます。

First Name	Surname	DOB	Address	State
Uma	Acothley	05/12/1985	24, Atkins Avenue	Michigan
John	Mcgovern	04/15/1963	49, Wheeler Road	Oklahoma
Keiko	Garsia	03/24/1990	13, Draker Drive	Montana
Isadora	Sonnier	01/16/1991	20, Fountain Center	South Carolina

テーブル Personnel_test のマスクされたデータをテスト環境で使用できるようになりました。

第 7 章

データサービストランスフォーメーション

データサービストランスフォーメーションを使用して、データサービスリポジトリからデータサービスを呼び出します。これには、HL7 や HIPAA などの業界標準サービスやカスタマイズされたサービスが含まれます。例えば、トランスフォーメーションを使用してデータサービスを呼び出し、ヘルスケアパートナーと交換する HIPAA メッセージを処理できます。

デフォルトでは、データサービスリポジトリには、業界標準を処理するための、追加設定不要なデータサービスが含まれています。データサービスリポジトリでカスタマイズされたサービスを作成し、データサービストランスフォーメーションを使用してそれらを呼び出すこともできます。データサービスとデータサービスリポジトリの詳細については、Administrator のヘルプにある「*データサービスリポジトリ*」を参照してください。

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、トランスフォーメーションが呼び出すデータサービスを定義します。以下のいずれかのオプションを選択することができます。

- 特定の標準および特定の使用タイプの単一のメッセージタイプを処理するには、単一のデータサービスを使用します。HIPAA メッセージの場合、パートナーにメッセージと検証メッセージを送信するデータサービスを選択することもできます。
- さまざまな標準および使用タイプからの複数のメッセージタイプのメッセージを処理するには、動的サービス名を使用します。動的サービス名を使用した場合は、データサービス名をパラメータとしてトランスフォーメーションに渡します。

データサービストランスフォーメーションは、選択した内容に基づいて、メッセージデータをファイルまたはバッファとしてダウンストリーム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で選択したステータストレースレベルとデータサービスの使用タイプに基づいて、トランスフォーメーションによって追加の出力フィールドが作成される場合があります。マッピングにさらにターゲットトランスフォーメーションを追加し、データサービストランスフォーメーションを設定して、さまざまな出力タイプをさまざまなターゲットに送信することができます。例えば、メッセージデータを 1 つのターゲットに送信し、ステータストレースレポートを別のターゲットに送信します。

データサービストランスフォーメーションを使用するには、適切なライセンスが必要です。

動的サービス名

データサービストランスフォーメーションで動的サービス名を使用して、さまざまな標準および使用タイプのメッセージタイプのメッセージを処理します。動的サービス名を使用した場合は、データサービス名をパラメータとしてトランスフォーメーションに渡します。

データサービスの名前には、業界標準、標準のバージョン、メッセージタイプ、およびサービスの使用タイプが次の構文で示されます。

```
<industry standard>_<version>_<message type>_<usage type>
```

使用タイプは以下のいずれかになります。

- パーサー。パートナーからメッセージを受信するには、パーサーサービスを使用します。
- シリアライザ。パートナーにメッセージを送信するには、シリアライザサービスを使用します。
- 制限されたシリアライザ。パートナーに検証メッセージなどのメッセージを送信するには、制限されたシリアライザサービスを使用します。HIPAA メッセージに適用されます。

例えば、HIPAA メッセージ 270 バージョン 5010A1 を受信するために使用するサービスの名前は、HIPAA_5010A1_270_Parser です。

動的サービス名を使用するには、マッピングを設定するときに次のアクションを実行する必要があります。

- データサービス名をパラメータとしてデータサービストランスフォーメーションに渡すようにマッピングを設定します。例えば、式トランスフォーメーションをマッピングに追加し、メッセージからデータサービス名を抽出してその名前をデータサービストランスフォーメーションに渡すように式トランスフォーメーションを設定します。
- データサービスのプロパティを設定するときは、動的サービス名を使用します。
- データサービストランスフォーメーションのフィールドマッピングで、svc_name フィールドを前のトランスフォーメーションの Service_Name フィールドにマッピングします。

ステータストレースメッセージ

ステータスメッセージを出力フィールドに書き込むように、データサービストランスフォーメーションのステータストレースレベルを設定できます。Mapping Designer は、データサービストランスフォーメーションで UDT_Status_Code および UDT_Status_Message 出力フィールドを作成します。

説明のみを書き込むことを選択した場合、トランスフォーメーションはステータスコードと次のいずれかのステータスメッセージを返します。

ステータスコード	ステータスメッセージ
1	成功
2	Warning
3	失敗
4	Error
5	致命的なエラー

完全なステータスを書き込むことを選択した場合、トランスフォーメーションはステータスコードと詳細な XML ステータスメッセージを返します。

データサービスのプロパティ

データサービスのプロパティを設定して、入力および出力タイプ、トランスフォーメーションによって呼び出されるデータサービス、およびステータストレースレベルを［**データサービス**］タブで定義します。

以下のプロパティを設定します。

プロパティ	説明
入力タイプ	データサービス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信する入力のタイプ。次のいずれかのオプションを選択します。 - バッファ。データサービストランスフォーメーションは、inputBuffer 入力フィールドでソースデータを受信します。 - ファイル。データサービストランスフォーメーションは、inputBuffer 入力フィールドでソースファイルパスを受信します。 デフォルト値は Buffer です。
出力タイプ	トランスフォーメーションがダウンストリームトランスフォーメーションに渡す出力のタイプ。次のいずれかのオプションを選択します。 - バッファ。データサービストランスフォーメーションは、データを outputBuffer 出力フィールドに渡します。 - ファイル。データサービストランスフォーメーションは、出力を outputBuffer 出力フィールドに output.xml ファイルとして書き込みます。 デフォルト値は Buffer です。
サービス名	データサービスリポジトリに存在するデータサービスからデータサービスを選択します。特定の標準および使用タイプの単一のメッセージタイプを処理するマッピングに対しては、このオプションを選択します。 **ヒント**: データサービスに基づいて入力フィールドと出力フィールドに自動的に入力するには、**［フィールドにデータを取り込む］**をクリックします。 動的サービス名を使用しない場合に適用されます。
動的サービス名	サービス名をパラメータとしてダウンストリームトランスフォーメーションに渡すには、このオプションを選択します。さまざまな標準および使用タイプからの複数のメッセージタイプのメッセージを処理するマッピングに対しては、このオプションを選択します。
ステータストレースレベル	データサービストランスフォーメーションが出力フィールドに書き込むジョブステータスの詳細レベル。次のいずれかのオプションを選択します。 - なし。トランスフォーメーションは、ステータスを出力フィールドに書き込みません。ジョブの情報は、セッションログで確認できます。 - 説明のみ。トランスフォーメーションは、ステータスコードと簡単な説明を UDT_Status_Code 出力フィールドに書き込みます。 - 完全なステータス（XML）。トランスフォーメーションは、ステータスコードを UDT_Status_Code 出力フィールドに書き込み、詳細な XML ステータスメッセージを UDT_Status_Mesage 出力フィールドに書き込みます。 デフォルトは［なし］です。

データサービストランスフォーメーションの入力フィールド

[プロパティ] パネルの **[入力フィールド]** タブで入力フィールドを表示および追加します。**[入力フィールド]** タブには、各入力フィールドの名前、タイプ、精度、およびスケールが表示されます。

デフォルトでは、データサービストランスフォーメーションは inputBuffer フィールドを作成します。設定するトランスフォーメーションプロパティとデータサービスタイプに応じて、データサービストランスフォーメーションによって追加の入力フィールドが作成される場合があります。

追加のファイル、ファイル名、またはパラメータをデータサービスに渡す必要がある場合は、追加の入力フィールドを追加できます。各フィールドのフィールド名、タイプ、および精度を設定します。

次の表に、データサービス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プロパティに基づいて作成するデフォルトの入力フィールドを示します。

フィールド名	説明
inputBuffer	入力タイプがバッファである場合、ソースデータを受信します。 入力タイプがファイルである場合、ソースファイル名およびソースファイルパスを受信します。
OutputFilename	出力タイプがファイルである場合、出力ファイル名を受信します。
Service_Name	動的サービス名のプロパティを有効にすると、データサービス名を受信します。

次の表に、データサービストランスフォーメーションに作成できる追加の入力フィールドを示します。

フィールドタイプ	入出力	説明
サービスパラメータ	Input	データサービスの入力パラメータを受信します。
追加の出力(ファイル)	出力	出力ファイル名を受け取ります。
追加の入力(バッファ)	Input	データサービストランスフォーメーションに渡す入力データを受信します。

データサービストランスフォーメーションの出力フィールド

出力フィールドは、**[プロパティ]** パネルの **[出力フィールド]** タブに表示されます。**[出力フィールド]** タブには、各出力フィールドの名前、タイプ、精度、およびスケールが表示されます。

データサービストランスフォーメーションは、処理するメッセージに outputBuffer 出力フィールドを作成します。トランスフォーメーションで選択したステータストレースレベルとデータサービスの使用タイプに基づいて、トランスフォーメーションによって追加の出力フィールドが作成される場合があります。例えば、

HIPAA で制限されたシリアル化サービスの場合、トランスフォーメーションによって、エラーレポートと検証レポートの出力フィールドが作成されます。

カスタムメッセージを使用する場合は、カスタムメッセージ要素の追加の出力フィールドを設定します。パラメータを出力に渡すには、入力/出力フィールドタイプでパススルーフィールドを作成し、それらを出力フィールドにマッピングします。

ターゲットトランスフォーメーションをマッピングに追加すると、データサービストランスフォーメーションによってすべての出力フィールドがターゲットトランスフォーメーションにマッピングされます。異なるターゲットで出力を受信するには、マッピングにさらにターゲットトランスフォーメーションを追加し、ターゲットに送信しない出力フィールドを除外するためのフィールドルールを設定します。例えば、メッセージと検証レポートを 2 つの異なるターゲットで受信するには、2 つのターゲットトランスフォーメーションをマッピングに追加します。メッセージを受信するターゲットから検証レポートの出力フィールドを除外し、検証レポートを受信するターゲットから outputBuffer 出力フィールドを除外します。

データサービストランスフォーメーションフィールドのマッピング

[プロパティ] パネルの **[フィールドマッピング]** タブで、データサービストランスフォーメーションのフィールドマッピングを表示および設定します。

デフォルトでは、データサービストランスフォーメーションは、アップストリームのトランスフォーメーションから受信した入力を、前のトランスフォーメーションの inputBuffer 受信フィールドにマッピングします。動的サービス名を使用している場合は、svc_name 受信フィールドを前のトランスフォーメーションの Service_Name フィールドにマッピングします。

第 8 章

重複排除トランスフォーメーション

重複排除トランスフォーメーションによって、Data Quality で作成した重複排除アセットがマッピングに追加されます。

重複排除トランスフォーメーションを使用して、データセットの重複レベルを分析し、必要に応じて重複レコードのセットを 1 つの優先レコードに統合します。重複排除トランスフォーメーションは、レコードの *ID* 情報を分析します。ID は、人または組織を特定するレコード内のデータ値のグループです。

重複排除および統合は、次のタイプのデータプロジェクトで役立つ操作です。

- 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。
- 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。
- 重複する ID 情報を格納するレコードを特定または排除する必要があるプロジェクト。

重複排除および統合操作

重複排除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、トランスフォーメーションによって各入力レコードの ID データが分析されます。トランスフォーメーションは、入力レコード間の類似度を表す一連のパーセンテージスコアを生成します。複数のレコードが指定されたしきい値を超えるスコアで相互に一致する場合、レコードは重複と見なされます。

トランスフォーメーションに追加する重複排除アセットは、重複レコードが満たす必要があるしきい値スコアなど、重複排除操作の比較条件を指定します。

統合は、重複排除アセットがトランスフォーメーションに対して指定できるオプションのプロセスです。統合時に、トランスフォーメーションは重複排除プロセスで特定された一致レコードのセットを評価します。トランスフォーメーションは、各セットでレコードの優先バージョンを選択または作成します。

重複排除および統合プロセスは、Data Quality ユーザーが重複排除アセットで設定します。アセットで定義された条件の詳細については、Data Quality ユーザーにお問い合わせください。

重複排除および統合のルールとガイドライン

マッピングに重複排除トランスフォーメーションを追加する場合は、以下のルールとガイドラインを考慮してください。

ID **分析のためのフィールドのマッピング**

トランスフォーメーションに追加する重複排除アセットは、個人名や組織名などの ID のタイプを指定します。アセットは、ID タイプを重複排除操作の*目的*と見なします。アセットにおける ID のタイプは、トランスフォーメーションが入力フィールドで検索すると想定される情報のタイプを定義します。

トランスフォーメーションの適切な入力フィールドを、トランスフォーメーションが示すターゲットフィールドにマッピングする必要があります。必要に応じて、オプションの入力フィールドを他のフィールドにマッピングすることもできます。

スコアとしきい値

重複排除トランスフォーメーションは、入力データでペアになる可能性のある各レコードのスコアを計算します。トランスフォーメーションは、一致する重複レコードの各セット内のレコードのスコアを返します。同じセットに属さないレコードのスコアは返しません。

トランスフォーメーションは、一致セット内のレコード間のリレーションをリンクスコアおよびドライバスコアとして表します。

SequenceId **フィールドと** GroupKey **フィールド**

[フィールドマッピング] タブでは、トランスフォーメーションによって、アセットで指定されたフィールドに GroupKey フィールドと SequenceId フィールドが追加されます。GroupKey フィールドは必須です。SequenceId フィールドは詳細モードでは必須です。

グループキーは、トランスフォーメーションで入力レコードをサブセットにソートし、各サブセットに対して個別の重複分析を実行できるようにするデータ値です。適切なグループキーを選択すると、マッピング結果の品質を低下させずに、マッピングの実行にかかる時間を短縮できます。グループの詳細については、「重複分析におけるグループ」 (ページ 131)を参照してください。

シーケンス ID 値は、トランスフォーメーションで入力レコードを読み取る順序を決定します。SequenceId フィールドへデータを提供できるフィールドが入力レコードに含まれていない場合、トランスフォーメーションではレコードが入力データセットに出現する順序で読み取られます。

メタデータフィールド

[出力フィールド] タブでは、トランスフォーメーションにより、一致レコードのペアのスコア値を表示するフィールドが追加されます。このフィールドは、各レコードが属する一致レコードのセットも識別します。重複排除アセットで統合プロセスが指定されている場合、メタデータフィールドは各レコードセットの優先レコードを指定します。トランスフォーメーションは、優先レコードを*存続*レコードと見なします。

このフィールドを使用してマッピング結果を把握します。

ID ポピュレーションデータ

重複排除プロセスでは、国固有の参照データファイルを使用して入力データの ID 情報を評価します。参照データファイルは*ポピュレーションファイル*と呼ばれます。重複排除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、トランスフォーメーションによって、アセットで指定された国のポピュレーションファイルと入力データが比較されます。

Secure Agent をインストールすると、Data Quality によってポピュレーションファイルが Secure Agent ホストマシンにダウンロードされます。ファイルをダウンロードするためにアクションを実行する必要はありません。

ポピュレーションファイルのプロパティの詳細については、Data Quality オンラインヘルプで重複排除のガイドを参照してください。

重複分析におけるグループ

重複分析マッピングでは、重複排除トランスフォーメーションが実行する必要のあるデータ比較の回数が多数に及ぶため、時間がかかることがあります。比較の回数は、選択したフィールドのデータ値の数に関係しています。

次の表は、マッピングが 1 つのフィールドに対して実行する計算の回数を示しています。

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かかる時間を短縮するには、入力レコードを **[グループ]** に割り当てるように重複排除トランスフォーメーションを構成します。

グループとは、指定したフィールド上の、同一の値を含む一連のレコードです。グループ化されたデータに対して重複分析を実行すると、重複排除トランスフォーメーションは各グループ内のレコードのみを分析し、各グループからの結果を単一の出力データセットに結合します。データをグループ化するフィールドは、**GroupKey** フィールドです。適切なグループキーを選択すると、マッピング分析の精度を大きく損なうことなく、重複排除トランスフォーメーションで実行する必要がある比較の総数を減らすことができます。重複排除トランスフォーメーションで GroupKey フィールドを選択します。

次の表は、10 個のグループに分類するマッピングが 1 つのフィールドに対して実行する計算の回数を示しています。

データ値の数	グループの数	グループサイズ	比較総数（全グループ）
10,000	10	1,000	500 万
10 万	10	10,000	5 億
100 万	10	10 万	500 億

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。

- GroupKey フィールドには、さまざまな同一の値（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。
- 重複分析に関連する情報を含むグループキーを選択しないでください。例えば、インデックスキーフィールドを GroupKey フィールドとして選択しないでください。グループ作成の目的は、重複する性質があって、それが分析の目的には関係しない値に従ってデータを整理することです。
- グループキーを選択するときは、入力データに対して有効なサイズのグループをトランスフォーメーションで作成できるかどうかを考慮してください。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レコード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。
- データにグループキーに適したフィールドが含まれていない場合は、必要なグループサイズにレコードを分類するためにトランスフォーメーションが使用できるデータカラムを作成してください。例えば、100 万件のレコードを含むデータセットの場合、1 から 50 までの一連の値を繰り返すカラムを作成するとします。各グループのレコードはデータセット内に均等に分散され、グループ化されたデータに対して重複分析を続行できるようになります。

- レコードがグループに分類されないようにする場合は、すべてのレコードに同じ値が含まれる GroupKey フィールドを指定します。適切なフィールドがない場合は、フィールドを作成します。例えば、すべての値が *Group1* であるデータカラムを作成し、そのカラムを［GroupKey］フィールドとして選択します。マッピングを実行すると、重複排除トランスフォーメーションによって GroupKey フィールドの値ごとにレコードが分類され、すべてのレコードが同じグループに割り当てられます。
- 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。

例: グループキーカラムの選択

銀行で、重複した銀行口座所有者の検索を行おうとしているとします。銀行の顧客データセットには顧客の名前と住所のカラムが含まれており、銀行は重複排除アセットの目的として**［連絡先］**を選択しました。銀行は、入力レコードをグループに分類し、各グループに対して重複分析を実行することを決定しました。銀行は、グループを作成する重複排除トランスフォーメーション内のカラムを選択する必要があるとします。

次の表に、データセットの一部を示します。

顧客 ID	**姓**	**名**	**住所 1**	City	**状態**	**郵便番号**	**国**
90999990	Armstrong	Al	6121 SUNSET BLVD.	LOS ANGELES	CA	90028	USA
90999907	Baldwin	Lynn	1600 EL CAMINO REAL, SUITE 1500	MENLO PARK	CA	94025	USA
90999917	Baldwyn	Linn	1600 EL CAMINO REAL, #1500	MENLO PK	CA	94025	USA
90999859	Belleperche	Carmen	9255 SUNSET BLVD.	LOS ANGELES	CA	90069	USA
90999876	Clark	Wick	777 S. FIGUEROA	LOS ANGELES	CA	90071	USA
90999859	Bachtin	Guy	30 S. WACKER	CHICAGO	IL	60606	USA
90999868	Dicintio	David	181 WEST MADISON ST	CHICAGO	IL	60602	USA
90999869	Ash	Pascal	335 WEST 16TH STREET	NEW YORK	NY	10011	USA
90999996	Bachtin	David	1633 BROADWAY	NEW YORK	NY	10022	USA
90999994	Carpenter	Brad	30 BROAD ST	NEW YORK	NY	42304	USA
90999820	Dedmond	David	ONE FINANCIAL SQUARE	NEW YORK	NY	10008	USA
90999902	Backwell	Chris	901 SE OAK, WILLAMETTE PLZ	PORTLAND	OR	97214	USA
90999897	Askerup	Nancy	400 MARKET STREET	HOUSTON	TX	77027	USA
90999904	Choy	Shelley	1177 WEST LOOP SOUTH	HOUSTON	TX	77027	USA
90999886	Cote	Lian	530 E. SWEDESFORD RD.	HOUSTON	TX	77027	USA
90999999	Croteau	Paul	3829-55 GASKINS ROAD	HOUSTON	TX	77027	USA

このシナリオでは、レコードを並べ替えるために最適なカラムとして［州］カラムを使用します。トランスフォーメーションで、GroupKey フィールドとして**［州］**カラムを選択します。

GroupKey フィールドとして［州］カラムを選択すると、重複排除操作により、各州に 1 つずつ、合計 5 つのグループが作成されます。銀行の顧客情報の中で、異なる州で同じ連絡先情報を持つ顧客がいる可能性は非常に低いと言えます。さらに、データには顧客 ID カラムが含まれており、これによって重複排除プロセスの信頼性が高まります。

顧客 ID カラムはプライマリキーフィールドであるため、グループ作成の候補としては適していません。GroupKey フィールドとしてカラムを選択すると、重複排除操作により、すべての一意の ID に対してグループが作成され、これによりすべてのレコードに対してグループが作成されます。

同様に、［国］カラムについてもすべての行に同じ値が含まれているため、グループ作成の候補としては適していません。GroupKey フィールドとして［国］カラムを選択すると、重複排除操作によりすべてのレコードが同じグループに追加されます。銀行の顧客情報の中には、同じ名前の個別の顧客が全国に 2 人以上いる可能性があるため、そうした顧客のエントリが重複しないようにしたいと思います。

重複排除トランスフォーメーションの設定

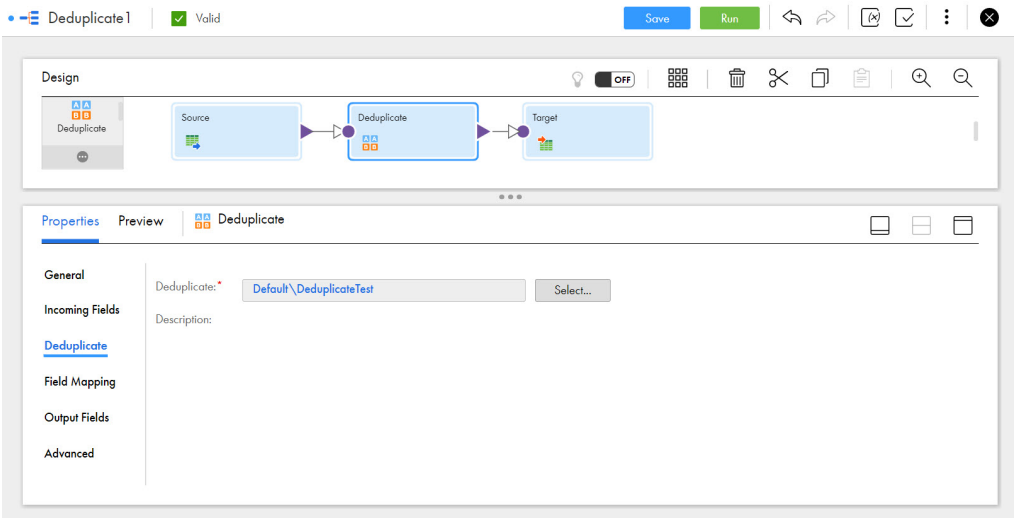
マッピングの重複排除トランスフォーメーションを設定する場合は、最初に、トランスフォーメーションに含めるロジックが含まれている重複排除アセットを選択します。次に、適切な受信フィールドをアセットにマッピングします。

重複排除アセットは、トランスフォーメーションで検索される ID のタイプを指定します。ID タイプにより、トランスフォーメーションに接続する必要がある入力フィールドのタイプが決まります。重複排除トランスフォーメーションで想定されるフィールドの詳細については、アセットを設定した Data Quality ユーザーにお問い合わせください。

トランスフォーメーションを設定するには、以下のタスクを実行します。

1. 重複排除トランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. **［重複排除］**タブで、トランスフォーメーションに含める重複排除アセットを選択します。

   次の図は、重複排除アセットを選択するために使用するオプションを示しています。



3. **［受信フィールド］**タブで、アップストリームオブジェクトからトランスフォーメーションに入力されるフィールドを確認します。

デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。

4. **［フィールドマッピング］**タブで、1 つ以上の受信フィールドを重複排除アセットのフィールドに接続します。

   マッピングするフィールドには、アセットで指定されたタイプの ID が記述されたデータが含まれている必要があります。また、重複分析中にグループを作成するためにトランスフォーメーションで使用できる GroupKey フィールドをマッピングする必要があります。

   タブの**［受信フィールド］**セクションにはアップストリームオブジェクトのフィールドがリストされ、**［ターゲットフィールド］**セクションにはアセットで指定されたフィールドがリストされます。

   受信フィールド名は、トランスフォーメーションのターゲットフィールドの名前を反映している可能性があります。この場合、**［自動マップ］**オプションを使用してフィールドの接続を行うことができます。

5. **［出力フィールド］**タブの出力フィールドのプロパティを確認します。

   出力フィールドには、重複排除および統合プロセスの結果に関する重要な情報を示すいくつかのメタデータフィールドが含まれます。メタデータフィールドの詳細については、「重複排除トランスフォーメーションのメタデータフィールド」 (ページ 136)を参照してください。

6. **［全般］**タブでは、オプションとして、重複排除トランスフォーメーションの名前を変更したり、説明を追加したりすることもできます。**［詳細］**タブで、トランスフォーメーションのトレースレベルを更新することもできます。デフォルトのトレースレベルはノーマルです。

注: アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。データ品質アセットの同期の詳細については、「「データ品質アセットの同期」 (ページ 102)」を参照してください。

重複排除トランスフォーメーションのフィールドマッピング

データがアップストリームトランスフォーメーションから重複排除トランスフォーメーションに移行して重複排除アセットの入力に接続する方法を定義するように、フィールドマッピングを設定します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

注: 受信フィールドを重複排除アセットの 1 つのフィールドにマッピングします。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドがあれば、そのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。
  トランスフォーメーションの重複排除アセットがパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、［すべてパラメータを使用します］オプションを選択します。

- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。［すべてパラメータを使用します］または［部分的にパラメータを使用します］をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは*フィールドマッピング*型である必要があります。

1 つのマッピング内の複数の重複排除トランスフォーメーションに同じフィールドマッピングパラメータを使用しないでください。

オプション

［受信フィールド］リストと**［ターゲットフィールド］**リストでフィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。［手動］または［部分的にパラメータを使用します］フィールドマップオプションを選択すると、［自動マップ］オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。
- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合では Cust_Name フィールドと Customer_Name フィールドが自動的にリンクされます。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似する名前のフィールドをマッピングできます。

［自動マップ］ > **［自動マップを取り消す］**をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで**［アクション］** > **［マップ解除］**をクリックします。選択した 1 つ以上のフィールドのマッピングを解除するには、［ターゲットフィールド］コンテキストメニューで**［選択項目をマップ解除］**をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、［ターゲットフィールド］コンテキストメニューで**［マッピングのクリア］**をクリックします。

重複排除トランスフォーメーションのメタデータフィールド

重複排除トランスフォーメーションには、重複排除および統合プロセスのメタデータを示す一連の定義済みのフィールドが含まれています。トランスフォーメーションはデフォルトでフィールドを作成し、マッピングの実行時にフィールドにデータを取り込みます。

［フィールドマッピング］タブのメタデータフィールド

［フィールドマッピング］タブの**［ターゲットフィールド］**リストには、次のメタデータフィールドがあります。

GroupKey

重複分析のためにトランスフォーメーションで入力レコードをグループにソートするために使用するデータ値が含まれます。

SequenceId

トランスフォーメーションに入力される各レコードの一意の識別子が含まれます。

トランスフォーメーションは、シーケンス ID 値を使用して Out_DriverId および Out_LinkId データ内のレコードを識別します。SequenceId フィールドをマッピングしていない場合、トランスフォーメーションは OutRowId フィールドの値をレコードの一意の識別子として使用します。

［出力フィールド］タブのメタデータフィールド

［出力フィールド］タブには、次のメタデータフィールドがあります。

Out_ClusterId

各レコードが属するクラスタの識別子が含まれます。

注: 重複排除プロセスにおけるクラスタとは、データ値が重複しきい値を超える程度に相互に一致しているレコードのセットです。同一セット内のレコードは、同じ ID を識別する可能性が高くなります。一意のレコードはそのレコード自体と完全に一致するため、1 つのセットに 1 つのレコードのみが含まれる場合があります。

Out_ClusterSize

現在のレコードが属するセット内のレコード数が含まれます。セット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。

Out_DriverId

各一致レコードセット内のドライバレコードの識別子が含まれます。ドライバレコードは、SequenceId 入力フィールドに対してセット内で値が最小のレコードです。トランスフォーメーションで SequenceId フィールドが使用されない場合、ドライバレコードは、一致セット内で Out_RowId 値が最小のレコードです。

Out_DriverScore

一致レコードセット内の現在のレコードとドライバレコードとの間の類似度を表すスコアが含まれます。

Out_IsSurvivor

統合プロセスで指定された優先レコードの識別子が含まれます。

Out_LinkId

現在のレコードと一致し、それを一致レコードセットにリンクしたレコードの識別子が含まれます。

Out_LinkScore

2 つのレコード間のスコアが含まれます。このスコアにより、レコードが一致レコードセットに追加されます。Out_LinkId フィールドは、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。

Out_RowId

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が含まれます。

一意の識別子のフィールドを SequenceId フィールドにマッピングしていない場合、トランスフォーメーションは Out_RowId 値を使用してレコードを識別します。

メタデータフィールドの選択

メタデータフィールドを使用すると、重複レコード間のリレーションに関する重要な情報が得られます。例えば、メタデータには、2 つのレコード間の類似度を数値で表す Out_LinkScore フィールドがあります。Out_LinkScore フィールドを選択する場合は、Out_LinkId フィールドも選択します。Out_LinkId フィールドは、Out_LinkScore 値が示すレコードのペア内の他方のレコードを特定します。

Out_DriverId 値は、一致レコードセット内のすべてのレコードのベンチマークを示します。Out_DriverId 値は、現在のレコードと、セット内でシーケンス ID または行 ID の値が最小のレコードとの間のスコアです。ID が最小のレコードは、重複排除プロセスでセットに最初に追加されたレコードでもあります。

リンクスコアとドライバスコア

重複排除プロセスは、重複排除トランスフォーメーションの出力にリンクスコアとドライバスコアのデータを追加します。これらのスコアを使用すると、重複レコード間のリレーションへの理解を深めることができます。

リンクスコアとは、同一一致セットのメンバとして識別する 2 つのレコード間のスコアです。トランスフォーメーションは、特定のレコードと、しきい値を超えるスコアでそのレコードと最初に一致するレコードとの間にリンクを作成します。

LinkId フィールドは、リンクスコアが適用されるレコードを識別します。リンクスコアとリンク ID の値は、レコードのペアが入力データ内で最も一致することを示すものではありません。リンクスコアとリンク ID の値は、一致レコードセットの構成を特定するためのものです。

ドライバスコアとは、一致レコードセットに最初に追加されたレコードと、同一セット内の別のレコードとの間のスコアです。トランスフォーメーションは、シーケンス ID または行 ID の値を使用してセット内の最初のレコードを識別します。ドライバスコアは、セット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。

注: 重複分析では、入力レコードの単一のスコアセットが生成されます。ドライバスコアとリンクスコアは、レコード間の異なるリレーションを表すもので、異なるタイプの重複分析を示すものではありません。ドライバスコアとリンクスコアの割り当ては、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。特定のレコードのペアのドライバスコアがしきい値を下回る可能性があります。

リンクスコアとドライバスコアの例

重複排除トランスフォーメーションで名字データのカラムを分析します。重複排除アセットで、重複レコードのしきい値を 0.825 と定義します。

次の表に、トランスフォーメーションから返される可能性のある結果を示します。

名字	シーケンス ID	ClusterId	ClusterSize	DriverId	DriverScore	LinkId	LinkScore
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

結果から、名字データに関する次の情報が得られます。

- SMITT と SMITS は、他のどのレコードにも一致しません（スコアがしきい値を満足しない）。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 の ClusterSize 値が 1 であることから、それぞれが各セット内で唯一のレコードであることがわかります。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 つだけの一致レコードセットを検索します。
- SMITH と SMITH は、リンクスコアが 1 です。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。このトランスフォーメーションにより、レコードが単一の一致レコードセットに追加されます。ClusterId 値は、これらのレコードが同じセットに属することを示しています。
- SMYTH と SMYTHE は、リンクスコアが 0.83333 です。スコアが重複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、レコードが単一の一致レコードセットに追加されます。

重複排除トランスフォーメーションの出力フィールド

重複排除トランスフォーメーションは、［フィールドマッピング］タブの出力フィールドを表示するほか、トランスフォーメーションで作成した一連のメタデータフィールドも表示します。これらのフィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。

タブには、出力フィールドの名前、タイプ、精度、およびスケールが表示されます。

重複排除トランスフォーメーションの出力フィールドプロパティを編集することはできません。プロパティを編集するには、Data Quality で重複排除アセットを開きます。

トランスフォーメーションで作成されるメタデータフィールドの詳細については、「重複排除トランスフォーメーションのメタデータフィールド」 (ページ 136)を参照してください。

詳細プロパティ

重複排除トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。

第 9 章

式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は 1 行内で値を計算します。式トランスフォーメーションを使用して、非集計計算を実行します。

例えば、賞与の割合の調整や名と姓の連結などの処理を式トランスフォーメーションを使用して実行できます。

式トランスフォーメーションを設定する際には、データフローで使用する各計算の出力について式フィールドを作成します。トランスフォーメーション内で使用する計算の変数フィールドを作成します。

式フィールド

式フィールドは、受信フィールドに対して実行する計算を定義したもので、結果の出力フィールドとして機能します。受信フィールドに対して計算を実行するために必要な数だけ式フィールドを使用できます。

式フィールドを設定するときは、フィールドのプロパティと実行する計算を定義します。

以下のタイプの式フィールドを作成できます。

- 出力
- 変数
- 入力マクロ
- 出力マクロ

出力フィールド

出力フィールドを作成して、式マクロを必要としない計算を定義します。

次の表では、出力フィールドを作成するときに定義できるプロパティについて説明します。

プロパティ	説明
名前	フィールドの名前。
タイプ	フィールドのデータ型。
精度	数値の全桁数。例えば、数値 123.45 の精度は 5 です。 精度は 1 以上でなければなりません。 decimal フィールドでは 38 を超える精度は指定できません。

プロパティ	説明
スケール	数値の小数点の右側の桁数。例えば、数値 123.45 のスケールは 2 です。 スケールは 0 以上でなければなりません。 数値のスケールは、その精度より小さい値にする必要があります。 すべてのデータ型で編集できるわけではありません。
デフォルト値	デフォルト値は、トランスフォーメーションで出力エラーが発生した場合に実行する動作をマッピングタスクに指定します。デフォルト値は、詳細モードのマッピングでは使用できません。次のいずれかの値を入力できます。 - ERROR('トランスフォーメーションエラー')トランスフォーメーションエラーが発生すると、マッピングタスクは行をスキップし、エラーをセッションログまたは行エラーログに書き込みます。 - 定数または定数式。マッピングタスクは、エラーを定数または定数式に置き換えます。ログには何も書き込まれません。 - ABORT。トランスフォーメーションは強制終了し、マッピングタスクはセッションログにメッセージを書き込みます。 デフォルトは ERROR('transformation error')です。 データ統合では、マッピングを保存または検証するときに、出力フィールドのデフォルト値が検証されます。無効な値を入力すると、Mapping Designer ではマッピングが無効としてマークされます。
説明	フィールドの説明（省略可能）。説明には最大 4000 文字を含めることができます。

変数フィールド

式トランスフォーメーション内で使用する計算を定義するための変数フィールドを作成します。

次の表では、変数フィールドを作成するときに定義できるプロパティについて説明します。

プロパティ	説明
名前	フィールドの名前。
タイプ	フィールドのデータ型。
精度	数値の全桁数。例えば、数値 123.45 の精度は 5 です。 精度は 1 以上でなければなりません。 decimal フィールドでは 38 を超える精度は指定できません。
スケール	数値の小数点の右側の桁数。例えば、数値 123.45 のスケールは 2 です。 スケールは 0 以上でなければなりません。 数値のスケールは、その精度より小さい値にする必要があります。 すべてのデータ型で編集できるわけではありません。
説明	フィールドの説明（省略可能）。説明には最大 4000 文字を含めることができます。

入力マクロフィールド

式マクロで使用する入力を表す入力マクロフィールドを作成します。マクロのタイプに応じて、入力はフィールドまたは定数になります。

次の表では、入力マクロフィールドを作成するときに定義できるプロパティについて説明します。

プロパティ	説明
名前	入力マクロフィールドの名前。
説明	フィールドの説明（省略可能）。説明には最大 4000 文字を含めることができます。

出力マクロフィールド

出力マクロフィールドを作成して、すべての受信フィールドまたは定数に対して実行する計算を定義します。マクロ入力フィールドは、受信フィールドまたは定数を表します。

次の表では、出力マクロフィールドを作成するときに定義できるプロパティについて説明します。

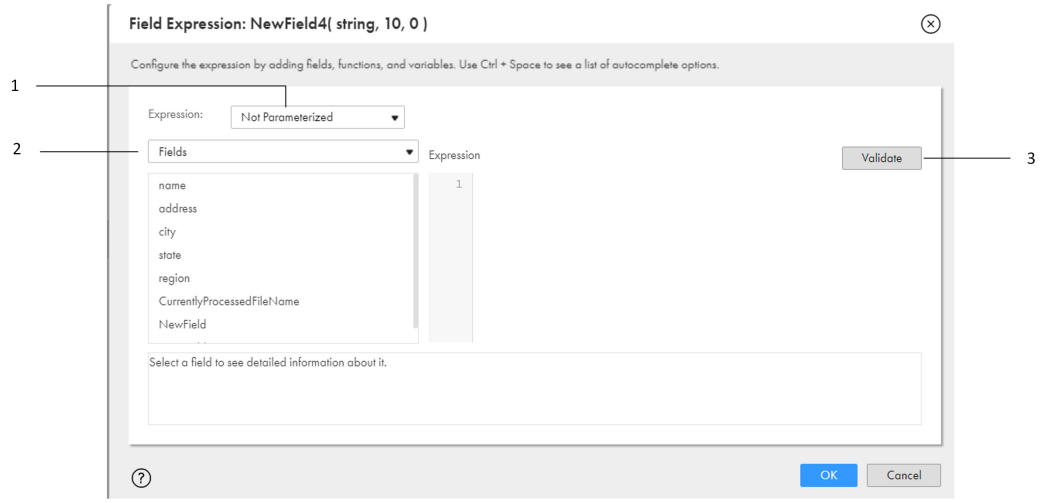
プロパティ	説明
入力マクロフィールド	式マクロの入力を表す入力マクロフィールドの名前。
出力マクロフィールド	次の形式の出力マクロフィールドの名前: <prefix>%<input mactiro field name>%<suffix> 出力マクロフィールド名には、プレフィックス、サフィックス、またはその両方が含まれている必要があります。
サフィックス	出力マクロフィールド名のサフィックス。 デフォルトは_out です。
プレフィックス	出力マクロフィールド名のプレフィックス。 デフォルトは［なし］です。
タイプ	フィールドのデータ型。
精度	数値の全桁数。例えば、数値 123.45 の精度は 5 です。 精度は 1 以上でなければなりません。 decimal フィールドでは 38 を超える精度は指定できません。
スケール	数値の小数点の右側の桁数。例えば、数値 123.45 のスケールは 2 です。 スケールは 0 以上でなければなりません。 数値のスケールは、その精度より小さい値にする必要があります。 すべてのデータ型で編集できるわけではありません。
説明	フィールドの説明（省略可能）。説明には最大 4000 文字を含めることができます。

式エディタ

式エディタを使用して、式フィールドを設定します。式には、定数、変数、組み込み関数、およびユーザー定義関数を含めることができます。また、関数内に関数をネストすることで複雑な式を作成できます。

式を設定するには、式エディタに式を入力します。

次の図に、式エディタおよび実行できるアクションを示します。



1. 式をパラメータ化する。
2. フィールド、システム変数、パラメータ、組み込み関数、およびユーザー定義関数を切り替える。
3. 式を検証する。

使用するオブジェクトの横にある **[追加]** をクリックすることで、ソースフィールド、関数、および変数を式に追加できます。式を手動で入力することも可能です。

または、**Ctrl+スペース**キーを押すと、推奨される引数と関数のリストがインラインで表示されます。データ統合は、関数の引数とキーストロークのタイプに基づいた推奨事項を提供します。インラインの推奨事項は、階層ソースデータには使用できません。

式を検証するには、**[検証]** をクリックします。データ統合で、式が検証されます。

式のトランスフォーメーション言語コンポーネント

トランスフォーメーション言語には、簡単な式や複雑な式を作成するための、次の構成要素があります。

- フィールド。ソースフィールドの名前を使用して、フィールドの値を参照します。
- リテラル。数値リテラルまたは文字列リテラルを使用して、特定の値を参照します。
- 関数。SQL に似た関数を使用して、タスク内のデータを変更します。
- 演算子。トランスフォーメーション演算子を使用して、算術演算の実行、データの結合や比較を行う式を作成します。
- 定数。事前定義の定数（TRUE など）を使用して、一定に保たれる値を参照します。

式の構文

ORDERS などのフィールドや、10 などの数値定数のみを含む単純な式を作成できます。また、複雑な式として、関数の中に別の関数をネストしたり、トランスフォーメーション言語演算子を使って異なる複数のフィールドを結合したりすることもできます。

注: トランスフォーメーション言語は標準 SQL に基づいていますが、2 つの言語には異なる点もあります。

文字列リテラルと数値リテラル

式には数値リテラルまたは文字列リテラルを含むことができます。

文字列リテラルを一重引用符で囲みます。例:

```
'Alice Davis'
```

文字列リテラルでは大文字と小文字が区別されます。一重引用符を除くすべての文字を使用できます。例えば、次のような文字列は使用できません。

```
'Joan's car'
```

一重引用符を含む文字列を返すには、CHR 関数を使用します。

```
'Joan' || CHR(39) || 's car'
```

数値リテラルでは一重引用符を使用しないでください。含めたい数値をそのまま入力します。例:

```
.05
```

または

```
$$Sales_Tax
```

式へのコメント追加

次のコメント指定子を使用して、式にコメントを挿入できます。

- 2 本のダッシュ:

  ```
  -- These are comments
  ```

- 2 本のスラッシュ:

  ```
  // These are comments
  ```

データ統合タスクは、コメント指定子の前にある行のすべてのテキストを無視します。例えば、2 つの文字列を連結する場合、式の途中に次のようなコメントの付いた式を挿入します。

```
 -- This expression concatenates first and last names for customers:
FIRST_NAME -- First names from the CUST table
|| // Concat symbol
LAST_NAME // Last names from the CUST table
// Joe Smith Aug 18 1998
```

データ統合タスクはコメントを無視し、式を次のように評価します。

```
FIRST_NAME || LAST_NAME
```

新しい行にコメントを続けることはできません。

```
-- This expression concatenates first and last names for customers:
FIRST_NAME -- First names from the CUST table
|| // Concat symbol
LAST_NAME // Last names from the CUST table
Joe Smith Aug 18 1998
```

この場合、最終行が有効な式でないため、データ統合タスクは式を評価しません。

予約語

定数、演算子、システム変数などの一部のキーワードは、特定の関数に対する予約語となっています。予約語には以下のものがあります。

- :EXT
- :INFA
- :LKP
- :MCR
- :SD
- :SEQ
- :SP
- :TD
- AND
- DD_DELETE
- DD_INSERT
- DD_REJECT
- DD_UPDATE
- FALSE
- NOT
- NULL
- OR
- PROC_RESULT
- SPOUTPUT
- TRUE
- WORKFLOWSTARTTIME

次の予約語は Informatica Intelligent Cloud Services で使用されます。

- ABORTED
- DISABLED
- FAILED
- NOTSTARTED
- STARTED

- STOPPED
- SUCCEEDED

注: 予約語をフィールドの名前に使用することはできません。予約語は式の中であらかじめ定義された意味を持ちます。

ウィンドウ関数

詳細モードでは、ウィンドウ関数を使用してステートフルな計算を簡潔に表現できます。ウィンドウ関数は、大規模なデータセットから小さいサブセットを取得して、処理と分析を行います。

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。

ウィンドウ関数を使用して、次のタスクを実行します。

- アップストリームまたはダウンストリームの行からデータを取得します。
- 行のグループに基づいて累積合計を計算します。
- 行のグループに基づいて累積平均を計算します。

ウィンドウ関数を定義する前に、**[ウィンドウ]** タブで次のウィンドウプロパティを設定します。

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。

選択するフィールドによって、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。

ウィンドウ関数を含む式をパラメータ化することはできません。式がパラメータ化されている場合、マッピングタスクでウィンドウ関数を指定することはできません。

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。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。

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。

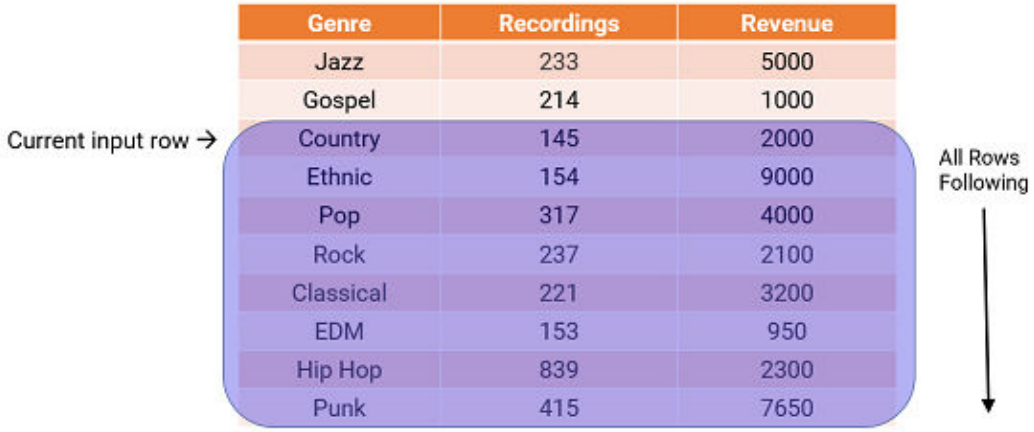
次の図は、開始オフセットが-1 で終了オフセットが 1 のフレームを示しています。

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。前図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。

［前のすべての行］および**［後のすべての行］**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［前のすべての行］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［後のすべての行］のフレームを示しています。



フレームオフセットがパーティション外部である場合、集計関数はこのフレームを無視します。フレームのオフセットがパーティションまたはテーブルの内部でない場合、集計関数はパーティション内の行のみを処理します。集計関数により、スキップされた行がログファイルに一覧表示されます。この関数は、スキップされた行に対して、デフォルト値の ERROR('トランスフォーメーションエラー')、NULL、または事前定義された定数のいずれかの応答を返します。

例えば、テーブルを販売者 ID 別、オーダーを数量別のパーティションに分けるとします。開始オフセットを-3 に設定し、終了オフセットを 4 に設定します。

次の図に、現在の入力行のパーティションとフレームを示します。

このフレームには、合計 8 行が含まれますが、計算はパーティション内に留まります。このフレームを使用して AVG 関数を定義した場合、この関数はパーティション内の数量の平均を計算し、18.75 を返します。

フレームを定義する場合、以下のルールとガイドラインを考慮します。

- LEAD および LAG では、関数の引数で指定するフレームが使用され、**[ウィンドウ]** タブで設定するフレームは無視されます。
- 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。

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

すべての行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設定します。

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。

また、データを昇順または降順に並べ替える必要もあります。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ランダムに並べられます。

パーティションキーとオーダーキーのウィンドウプロパティを定義するときは、次のルールとガイドラインを考慮してください。

- 階層フィールド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。
- 一意のフィールドをパーティションキーおよびオーダーキーとして定義します。

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

各パーティション内で MAX 関数を実行して、売り上げのよいコーヒーの 2 つはエスプレッソとアメリカーノであり、売り上げのよいお茶の 2 つは白茶と紅茶であると判断できます。

例: ウィンドウを使用した有効期限の計算

あなたは自分の担当顧客 2 社の財務プランに関する情報を持つ銀行家です。各プランに、開始日が関連付けられています。

各顧客のために、次のプランの有効化日付に基づいて、現在のプランの有効期限を知りたいとします。前のプランは新しいプランの開始時に終了するため、前のプランの終了日は、次のプランの開始日マイナス 1 日です。

次の表に、顧客コード、関連プランコード、各プランの開始日を示します。

CustomerCode	PlanCode	StartDate
C1	00001	2014-10-01
C2	00002	2014-10-01
C2	00002	2014-11-01
C1	00004	2014-10-25
C1	00001	2014-09-01
C1	00003	2014-10-10

有効期限を計算するには、次のタスクを実行します。

1. パーティションキーおよびオーダーキーを定義します。
   次のウィンドウプロパティを設定して、データを顧客コードでパーティション化し、データを開始日の昇順に並べます。

プロパティ	値	説明
フレーム	指定されていません。	LEAD 関数は、オフセット引数に基づいて行にアクセスし、フレームを無視します。
パーティションキー	CustomerCode。	行を顧客コード別にグループ化し、計算が個々の顧客に基づくようにします。
オーダーキー	StartDate 昇順。	開始日の昇順で、データを時系列に配置します。

   次の表に、顧客コードでグループ化された開始日順のデータを示します。

CustomerCode	PlanCode	StartDate
C1	00001	2014-09-01
C1	00002	2014-10-01
C1	00003	2014-10-10

CustomerCode	PlanCode	StartDate
C1	00004	2014-10-25
C2	00001	2014-10-01
C2	00002	2014-11-01

2. ウィンドウ関数の定義
   入力ごとに次の行にアクセスする LEAD 関数を定義します。
   次の計算を実行する式フィールドを設定します。
   ```
   LEAD ( StartDate, 1, '01-Jan-2100' )
   ```
   LEAD 関数の詳細については、*『関数リファレンス』*を参照してください。
3. ADD_TO_DATE 関数の定義
   アクセスした日付から 1 日を除算する ADD_TO_DATE 関数を使用します。
   次の計算を実行する式フィールドを設定します。
   ```
   ADD_TO_DATE ( LEAD ( StartDate, 1, '01-Jan-2100' ), 'DD', -1, )
   ```
   次のプランの開始日から 1 日を除算することで、現在のプランの終了日がわかります。
   次の表に、各プランの終了日を示します。

CustomerCode	PlanCode	StartDate	EndDate
C1	00001	2014-09-01	2014-09-30
C1	00002	2014-10-01	2014-10-09
C1	00003	2014-10-10	2014-10-24
C1	00004	2014-10-25	2099-12-31*
C2	00001	2014-10-01	2014-10-31
C2	00002	2014-11-01	2099-12-31*

**LEAD 関数は、これらのプランがまだ終了していないために、デフォルト値を返しました。これらの行はパーティション外であるため、ADD_TO_DATE 関数は 01-Jan-2100 から 1 日を除算し、2099-12-31 を返しました。*

例: ウィンドウを使用した GPS ping のフラグ付け

あなたの組織はトリップ ID、イベント ID、およびタイムスタンプを含む車両からの GPS ping を受信します。各 ping 間の時間差を計算し、前の行との時間差が 60 秒未満である場合、行にスキップとのフラグを付けようとしています。

イベントを時系列で順序付け、イベントをトリップ別にパーティション化します。前の行のイベント時刻にアクセスするウィンドウ関数を定義し、ADD_TO_DATE 関数を使用して、2 つのイベント間の時間差を計算します。

ウィンドウプロパティ

[ウィンドウ] タブで、次のウィンドウプロパティを定義します。

プロパティ	値	説明
フレーム	指定されていません	ウィンドウ関数は、オフセット引数に基づいて行にアクセスし、フレームを無視します。
パーティションキー	trip_id。	行をトリップ ID 別にグループ化し、計算が同じトリップからのイベントに基づくようにします。
オーダーキー	_event_id 昇順。	イベント ID の昇順で、データを時系列に配置します。

ウィンドウ関数

前の行からイベント時刻を取得する次の LAG 関数を定義します。

```
LAG ( _event_time, 1, NULL )
```

LAG 関数の詳細については、『*関数リファレンス*』を参照してください。

2 つの日付間の時間を計算する次の DATE_DIFF 関数を定義します。

```
DATE_DIFF ( _event_time, LAG ( _event_time, 1, NULL ), 'ss' )
```

DATE_DIFF が 60 秒未満の場合、または_event_time が NULL の場合、行にスキップとのフラグを付けます。

```
IIF ( DATE_DIFF < 60 or ISNULL ( _event_time ), 'Skip', 'Valid' )
```

出力

トランスフォーメーションは、次の出力を生成します。

Trip ID	Event ID	Event Time	Time Difference	Flag
101	1	2017-05-03 12:00:00	NULL*	Skip
101	2	2017-05-03 12:00:34	34	Skip
101	3	2017-05-03 12:02:00	86	Valid
101	4	2017-05-03 12:02:23	23	Skip
102	1	2017-05-03 12:00:00	NULL*	Skip
102	2	2017-05-03 12:01:56	116	Valid
102	3	2017-05-03 12:02:00	4	Skip
102	4	2017-05-03 13:00:00	3480	Valid
103	1	2017-05-03 12:00:00	NULL*	Skip
103	2	2017-05-03 12:00:12	12	Skip
103	3	2017-05-03 12:01:12	60	Valid

*これらの行の前の行は、パーティションの境界外であるため、LAG 関数は NULL 値を生成します。

例: ウィンドウでの集計関数の実行

各従業員の給与と対応する部門の平均給与を比較するとします。

次の表に、部門名、従業員 ID 番号、従業員の給与の一覧を示します。

Department	Employee	Salary
Development	11	5200
Development	7	4200
Development	9	4500
Development	8	6000
Development	10	5200
Personnel	5	3500
Personnel	2	3900
Sales	3	4800
Sales	1	5000
Sales	4	4800

すべての従業員を計算に含めるための、バインドなしフレームを設定し、各従業員の給与とその従業員の部門内の平均給与との間の差異を計算する集計関数を定義します。

ウィンドウプロパティ

[ウィンドウ] タブで、次のウィンドウプロパティを定義します。

プロパティ	値	説明
開始 offset	前のすべての行	現在の入力行の前に表示される行数を記述します。
終了オフセット	後のすべての行	現在の入力行の後に表示される行数を記述します。
オーダーキー	昇順の給与。	データを給与の昇順に配置します。
パーティションキー	部門	行を部門別にグループ化します。

[前のすべての行] および [後のすべての行] を選択した場合は、すべてのパーティション行が含まれます。例えば、現在の行が 3 行目だとします。3 行目は「Development」パーティション内にあるため、このフレームには、「Development」パーティション内の 3 行目前後のすべての行に加えて、3 行目が含まれます。

ウィンドウ関数

式トランスフォーメーションでウィンドウプロパティを設定するため、集計関数はウィンドウ関数として使用できます。

各従業員の給与とその従業員の部門内の平均給与との間の差異を計算する次の集計関数を定義します。

```
Salary - AVG ( Salary ) = Salary_Diff
```

アウトプット

トランスフォーメーションは、次の給与の差異を生成します。

Department	Employee	Salary	Salary_Diff
Development	11	5200	-820
Development	7	4200	-520
Development	9	4500	180
Development	8	6000	180
Development	10	5200	980
Personnel	5	3500	200
Personnel	2	3900	200
Sales	3	4800	-66
Sales	1	5000	-66
Sales	4	4800	134

どの従業員が同一部門の平均給与よりも少なく、または多く稼いでいるかを特定できます。この情報に基づき、その他のトランスフォーメーションを追加し、データをより詳しく調べることができます。例えば、ランクトランスフォーメーションを追加し、各従業員の同一部門内での数値的なランクを生成できます。

詳細プロパティ

式トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルやトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。また、階層フィールドを式で使用したり、階層フィールドを複合演算子とともに使用して、式のプリミティブ子フィールドにアクセスすることもできます。複合演算子の詳細については、*関数リファレンス*の説明を参照してください。

階層フィールドを式で使用する場合は、次のガイドラインを考慮してください。

- 式のパラメータは使用しないでください。
- 式の出力を配列、マップ、または構造にすることはできません。

第 10 章

フィルタトランスフォーメーション

フィルタトランスフォーメーションは、指定されたフィルタ条件に基づいてデータフローのデータをフィルタリングします。ジョブのパフォーマンスを向上させるには、フィルタトランスフォーメーションをマッピングソースの近隣に配置して、データフローから不要なデータを削除してください。

フィルタ条件は TRUE または FALSE を返す式です。ある行に対してフィルタ条件が TRUE を返すと、フィルタトランスフォーメーションはその行を残りのデータフローに渡します。フィルタ条件が FALSE を返すと、フィルタトランスフォーメーションはその行を破棄します。

データは 1 つ以上の条件に基づいてフィルタリングできます。例えば、ある日付範囲内のデータを使用する場合は、指定された日付範囲の前および後のデータを削除する条件を作成します。

フィルタトランスフォーメーションには単一のトランスフォーメーションをリンクします。複数のトランスフォーメーションをフィルタトランスフォーメーションにマージすることはできません。

フィルタ条件

フィルタ条件は、TRUE または FALSE を返す式です。

1 つ以上の簡易フィルタ条件を作成できます。簡易フィルタ条件は、フィールド名、演算子、および値から成ります。例えば、Sales > 0 は、すべての売上高が 0 より大きい行を保持します。詳細モードでは、フィルタ条件が数値結果として評価される必要があります。

フィルタ条件では大文字と小文字が区別されます。簡易フィルタでは次の演算子を使用できます。

- =（等しい）
- <（より小さい）
- >（より大きい）
- <=（以下）
- >=（以上）
- ! =（等しくない）

複数の簡易フィルタ条件を定義すると、マッピングタスクによって、指定した順番で条件が評価されます。マッピング設定タスクは、AND 論理演算子を使用して各フィルタ条件を評価し、条件を結合します。そして、すべてのフィルタ条件を満たす行を返します。

詳細フィルタ条件を使用すると複合フィルタ条件を定義できます。詳細フィルタ条件を設定する際には、AND または OR 論理演算子を使用して複数の条件を組み込むことができます。条件の評価結果を表す定数を使用できます。0 は FALSE と等価、ゼロ以外の任意の値は TRUE と等価です。

フィルタ条件のタイプを簡易から詳細に変更すると、Mapping Designer によって、設定済みの簡易フィルタ条件が詳細フィルタ条件に取り込まれます。簡易フィルタ条件は利用しても削除してもかまいません。パラメータは変換の対象になりません。

NULL 値を含む行をフィルタリングするには、ISNULL 関数を使用してフィールド値をテストします。スペースを含む行をフィルタリングするには IS_SPACES を使用します。

例えば、First_Name フィールドに NULL 値を含む行を除外するには、IIF(ISNULL(First_Name),FALSE,TRUE) という条件を使用します。この条件は、First_Name フィールドが NULL の場合に FALSE を返すという意味です。マッピングタスクは行を破棄します。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。

詳細プロパティ

フィルタトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルやトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。また、階層フィールドを詳細フィルタ条件で使用したり、階層フィールドを複合演算子とともに使用して、フィルタ条件のプリミティブ子フィールドにアクセスすることもできます。複合演算子の詳細については、*関数リファレンス*の説明を参照してください。

階層フィールドをフィルタ条件で使用する場合は、次のガイドラインを考慮してください。

- 階層フィールドを簡易フィルタ条件で使用することはできません。

- フィルタ条件のパラメータは使用しないでください。

第 11 章

階層ビルダートランスフォーメーション

階層ビルダトランスフォーメーションは、リレーショナル入力を階層出力に変換します。

このトランスフォーメーションは、アップストリームトランスフォーメーションからのリレーショナル入力を処理し、ダウンストリームトランスフォーメーションに次の出力タイプのいずれかを渡します。

- JSON
- XML
- Avro
- Parquet
- ORC

階層ビルダトランスフォーメーションは、次のオプションに基づいて階層出力を生成します。

トランスフォーメーションに関連付ける階層スキーマまたはインテリジェント構造モデル。

スキーマをトランスフォーメーションに関連付けるには、既存のスキーマを選択するか、新しいスキーマを作成します。インテリジェント構造モデルを関連付けるには、指定したサンプルファイルから新しいモデルを生成します。

入力を JSON または XML に変換するには、階層スキーマをトランスフォーメーションに関連付けます。スキーマによって、想定される出力データの階層を定義します。既存の階層スキーマを使用するか、設計時に新しいスキーマを作成することができます。XML サンプルファイルまたは XML スキーマファイルから XML 出力用のスキーマを作成できます。JSON 出力のスキーマを作成するには、JSON サンプルファイルを使用します。JSON スキーマファイルを使用しないでください。

入力を Avro、Parquet、または ORC に変換するには、インテリジェント構造モデルを作成します。設計時に、サンプルファイルからの想定される出力データの階層をモデルによって定義します。

選択した出力形式。

文字列またはバイナリを選択できます。

例えば、トランスフォーメーションが大量のデータを処理し、出力フィールドのサイズが 100 MB を超えるような場合に、データをフラットファイルに書き込むように選択することができます。

データのマッピング。

スキーマ要素にリンクされるリレーショナル要素を定義して階層出力を提供するように、フィールドマッピングを設定します。

階層ビルダトランスフォーメーションの使用

マッピングで階層ビルダ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. 必要に応じて、階層スキーマを作成します。
2. 階層ビルダトランスフォーメーションをマッピングに追加します。
3. 階層スキーマまたはインテリジェント構造モデルを階層ビルダトランスフォーメーションに関連付けます。トランスフォーメーションを設定するときに、既存の階層スキーマをトランスフォーメーションに関連付けるか、階層スキーマまたはインテリジェント構造モデルを作成します。
4. 必要に応じて、追加の出力設定を設定します。
5. フィールドマッピングを設定して、スキーマまたはモデル要素にリンクするリレーショナル要素を選択します。
6. 必要に応じて、詳細なプロパティを設定します。

マップレットで階層ビルダトランスフォーメーションを使用する場合は、実行時エラーを防ぐために、組み合わせたマプレットトランスフォーメーション名が 80 文字を超えないようにしてください。詳細については、「マップレットトランスフォーメーション名」 (ページ 316)を参照してください。

サンプルまたはスキーマ

階層スキーマを作成する場合、基となる階層スキーマとして JSON サンプルファイルまたは XSD スキーマをインポートします。

可能であればスキーマを使用することをお勧めします。スキーマに再帰的な要素を含めることはできません。

サンプルを使用する場合、サンプルはトランスフォーメーションで処理する代表的なデータであり、かつ包括的である必要があります。サンプルには、トランスフォーメーションで処理する可能性のあるすべてのフィールドを含めます。フィールドの値とタイプの組み合わせについても、可能性のあるものをすべて含めます。フィールドの長さは典型的なものにする必要があります。

階層スキーマ

階層スキーマは、データ統合にインポートするスキーマファイルまたはサンプルファイルに基づいています。サンプルファイルをインポートすると、データ統合によってサンプルファイルの構造に基づいてスキーマが生成されます。スキーマでは、入力データの想定される階層を定義します。

階層スキーマは 2 つの方法で作成できます。選択したトランスフォーメーションに関連付けられるスタンドアロン階層スキーマを作成できます。または、特定のトランスフォーメーション内から階層スキーマを作成できます。

スタンドアロン階層スキーマを作成する場合、スキーマのベースとして JSON サンプルファイルまたは XSD ファイルをインポートします。

スタンドアロン階層スキーマとして作成したのか、特定のトランスフォーメーションの一部として作成したのかに関係なく、階層スキーマは任意のトランスフォーメーションに関連付けることができます。

階層スキーマは、作成、編集、削除できます。階層スキーマを編集し、ルートやスキーマ定義を変更できます。ただし、トランスフォーメーションで使用した階層スキーマは、編集または削除できません。階層スキーマを削除する前に、トランスフォーメーションで使用していないことを確認してください。

階層スキーマのルールとガイドライン

階層ビルダトランスフォーメーションに関連付ける階層スキーマを作成するときは、次のルールとガイドラインを考慮してください。

- スキーマの要素数は 10,000 未満である必要があります。
- スキーマには、最大 10,000 個のフィールドを含めることができます。
- スキーマには、最大 20,000 個のグループを含めることができます。
- スキーマには、最大 20 レベルのデータ階層を含めることができます。
- スキーマに 5,000 を超えるポートと 10 を超えるグループが含まれる場合、処理時間に影響することがあります。

階層スキーマの作成

データ統合で階層スキーマを作成します。

1. **［新規］** > **［コンポーネント］** > **［階層スキーマ］** をクリックします。
2. **［新規階層スキーマ］** ページで、名前と説明を入力します。階層スキーマの名前を指定する必要があります。
3. プロジェクトの場所を参照して選択します。
4. スキーマまたはサンプルファイルを選択するには、**［アップロード］** をクリックします。**［ファイルの選択］** をクリックし、XSD ファイルを参照するか、サンプルの JSON ファイルを選択して、**［OK］** をクリックします。

   JSON サンプルファイルを追加すると、データ統合によってサンプルからスキーマが生成されます。
5. 使用可能なルート要素が複数ある XSD ファイルを選択した場合は、ドロップダウンメニューから 1 つのルートを選択します。
6. 別のスキーマを参照するスキーマを選択した場合は、参照先のスキーマもアップロードする必要があります。参照先のスキーマをアップロードするには、**［アップロード］** をクリックし、目的のスキーマファイルを参照して **［OK］** をクリックします。
7. 階層スキーマを保存するには、**［OK］** をクリックします。

インテリジェント構造モデル

入力を Avro、Parquet、または ORC に変換する場合は、階層ビルダトランスフォーメーションを設定するときにインテリジェント構造モデルを作成します。

モデルの作成時に、想定される出力データの階層を表すサンプルファイルを選択します。データ統合ではファイルに基づいてモデルが作成されます。

出力設定

階層出力のスキーマや形式など、出力データのプロパティを設定します。

次の表に、出力プロパティを示します。

プロパティ	説明
スキーマ	階層出力のスキーマ。次のいずれかのタスクを実行してスキーマを設定します。 - 既存の階層スキーマを選択します。 - XSD ファイルまたは JSON サンプルファイルから階層スキーマを作成します。 - Avro、Parquet、または ORC サンプルファイルに基づいてインテリジェント構造モデルを作成します。
精度	出力のバッファサイズ。
NULL 値	JSON ベースのスキーマをトランスフォーメーション関連付けるときに、トランスフォーメーションが NULL 値を保持するか省略するかどうかを決定します。
出力形式	出力データの形式（文字列またはバイナリ）。
ファイルに書き込み	トランスフォーメーションがデータをフラットファイルに書き込むかどうかを決定します。このオプションを有効にした場合、**[ファイルパス]** フィールドにファイルパスを入力します。パスをパラメータ化することはできません。 **ヒント**: トランスフォーメーションが大量のデータを処理し、出力フィールドのサイズが 100 MB を超えるような場合は、データをフラットファイルに書き込みます。
パススルーフィールドを有効にする	マッピングされていないフィールドをダウンストリームトランスフォーメーションに渡すかどうかを決定します。 階層ビルダトランスフォーメーションに複数のアップストリームトランスフォーメーションが接続されている場合、トランスフォーメーションは最初に接続されたアップストリームトランスフォーメーションからフィールドを渡します。ダウンストリームトランスフォーメーションにはすべてのアップストリームトランスフォーメーションのフィールドが表示される場合がありますが、マッピングの実行時に余分なフィールドは渡されません。

階層スキーマの選択

階層ビルダトランスフォーメーションの既存の階層スキーマを選択できます。

1. 階層ビルダトランスフォーメーションの **[プロパティ]** パネルで、**[出力設定]** タブをクリックします。
2. **[スキーマ]** フィールドの横の **[選択]** をクリックします。

   [スキーマの選択] ダイアログボックスが表示されます。
3. 一覧から階層スキーマを選択します。
4. 階層スキーマを検索するには、検索条件を選択し、検索フィールドに検索語句を入力して、**[検索]** をクリックします。

   階層スキーマは名前または説明で検索できます。階層スキーマのソートには、名前、説明、または最終変更日を使用できます。
5. 階層ビルダトランスフォーメーションに含める階層スキーマを選択し、**[OK]** をクリックします。

   選択した階層スキーマが **[プロパティ]** パネルに表示されます。

階層スキーマの作成

XSD ファイルまたはサンプル JSON ファイルに基づいて階層スキーマを作成します。

1. 階層ビルダトランスフォーメーションの［**プロパティ**］パネルで、［**出力設定**］タブをクリックします。
2. スキーマを作成するには、［**新規**］ > ［**新しいスキーマを作成**］をクリックします。

   ［**新規階層スキーマ**］ページが表示されます。
3. スキーマの名前を入力します。オプションとして、スキーマの説明を入力します。
4. プロジェクトの場所を参照して選択します。
5. スキーマまたはサンプルファイルを選択するには、［**アップロード**］をクリックします。［**ファイルの選択**］をクリックし、XSD ファイルを参照するか、サンプルの JSON ファイルを選択して、［**OK**］をクリックします。

   JSON サンプルファイルを追加すると、データ統合によってサンプルからスキーマが生成されます。
6. 使用可能なルート要素が複数ある XSD ファイルを選択した場合は、ドロップダウンメニューから 1 つのルートを選択します。
7. 別のスキーマを参照するスキーマを選択した場合は、参照先のスキーマもアップロードする必要があります。参照先のスキーマをアップロードするには、［**アップロード**］をクリックし、目的のスキーマファイルを参照して［**OK**］をクリックします。
8. 階層スキーマを保存するには、［**OK**］をクリックします。

インテリジェント構造モデルの作成

Avro、Parquet、または ORC サンプルファイルに基づいてインテリジェント構造モデルを作成します。

1. 階層ビルダトランスフォーメーションの［**プロパティ**］パネルで、［**出力設定**］タブをクリックします。
2. スキーマを作成するには、［**新規**］ > ［**サンプルファイルから自動生成**］をクリックします。

   ［**サンプルファイルからのインテリジェント構造モデルの作成**］ページが表示されます。
3. モデルの名前を入力します。必要に応じて、このモデルの説明を入力します。
4. プロジェクトの場所を参照して選択します。
5. サンプルファイルを参照して選択し、［**作成**］をクリックします。

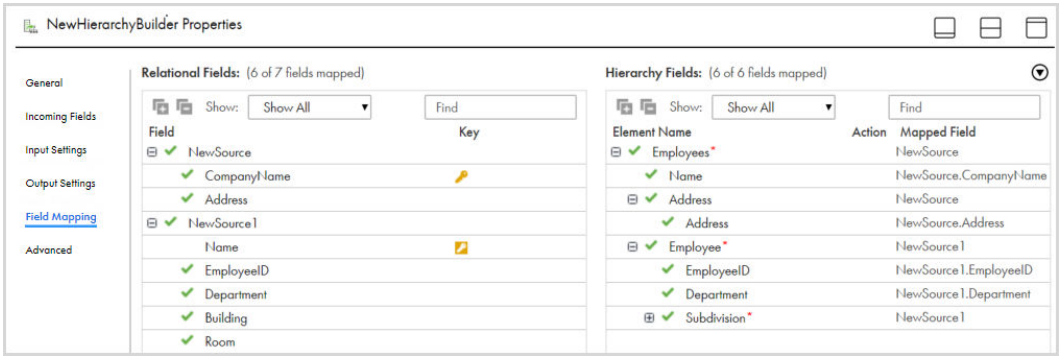
   データ統合でモデルが作成され、そのモデルがトランスフォーメーションに関連付けられます。

フィールドマッピング

階層ビルダートランスフォーメーションのフィールドマッピングを設定して、階層出力を提供するためにどのリレーショナル要素がスキーマ要素にリンクするかを定義します。［**プロパティ**］パネルの［**フィールドマッピング**］タブでフィールドマッピングを設定します。グループが複数ある場合は、フィールドマッピングエディ

タを使用してプライマリキーと外部キーを定義します。また、フィールドマッピングエディタは、リレーショナルフィールドをスキーマ要素にリンクする場合にも使用します。

次の画像は、フィールドマッピングエディタを示しています。



フィールドマッピングエディタの左側には、リレーショナルフィールドが表示されます。エディタの右側にはスキーマ要素が表示されます。この例のトランスフォーメーションには、2 つのリレーショナルグループがあるため、プライマリキーと外部キーを指定する必要があります。この例では、**[NSId]** フィールドは、**[NewSource]** グループのプライマリキーであり、**[NSLeadId]** フィールドは、**[NewSource1]** グループにリンクする外部キーです。**[NS1Id]** フィールドは、**[NewSource1]** グループのプライマリキーです。

プライマリキーには、**[キー]** カラムにプライマリキーのアイコンが表示されます。外部キーには外部キーアイコンが表示されます。

リレーショナルフィールドをスキーマ要素にリンクするには、リレーショナル要素をスキーマ要素にドラッグします。**[マッピングされたフィールド]** カラムに、スキーマ要素がマッピングされているリレーショナルフィールドが表示されます。

入力のリレーショナルフィールドが構成するグループが 1 つのだけの場合、データは非正規化された入力として扱われ、プライマリキーと外部キーを定義する必要はありません。

プライマリキーまたは外部キーの選択

プライマリキーまたは外部キーとして指定するリレーショナルフィールドを選択します。

1. 階層ビルダートランスフォーメーションの **[プロパティ]** パネルで、**[フィールドマッピング]** タブをクリックします。

   フィールドマッピングエディタに、トランスフォーメーション入力要素と出力フィールドが表示されます。エディタの左側には、リレーショナルフィールドが表示されます。エディタの右側にはスキーマ要素が表示されます。

2. リレーショナルフィールドを検索するには、検索フィールドに検索語句を入力して、**[検索]** をクリックします。
3. プライマリキーまたは外部キーとしてリレーショナルフィールドを指定するには、フィールドの **[キー]** カラムをクリックします。

   キーセレクタウィンドウが表示されます。

4. キーがプライマリキーか外部キーかを選択します。
5. キーが外部キーの場合、フィールドを外部キーとして関連付けるグループを選択します。
6. プライマリキーまたは外部キーを削除するには、フィールドの **[キー]** カラムをクリックし、**[キーではありません]** を選択します。
7. すべてのキーをクリアするには、**[階層フィールド]** パネルの上部にあるアクションメニューをクリックして、**[キーのクリア]** を選択します。

マッピングするフィールドの選択

リレーショナルフィールドをスキーマ要素にリンクして、リレーショナル入力フィールドを階層出力要素に変換する方法を定義します。

1. 階層ビルダートランスフォーメーションの［**プロパティ**］パネルで、［**フィールドマッピング**］タブをクリックします。

   フィールドマッピングエディタに、トランスフォーメーション入力要素と出力フィールドが表示されます。エディタの右側にはスキーマ要素が表示されます。エディタの左側には、リレーショナル入力フィールドが表示されます。
2. リレーショナルフィールドを検索するには、検索フィールドに検索語句を入力して、［**検索**］をクリックします。
3. 子スキーマ要素を表示するには、フィールドを展開します。
4. 要素を検索するには、検索フィールドに検索語句を入力して、［**検索**］をクリックします。
5. リレーショナル入力フィールドをスキーマ要素にリンクするには、リレーショナルフィールドをスキーマ要素にドラッグします。
6. リレーショナル入力フィールドをスキーマ要素に自動的にリンクするには、スキーマ要素を選択し、［**階層フィールド**］パネルの上部にあるアクションメニューをクリックして、［**選択項目をマップ**］を選択します。
7. スキーマ要素へのリレーショナル入力フィールドを自動的にリンク解除するには、次のオプションのいずれかを選択します。
   - スキーマ要素を選択します。［**アクション**］カラムの削除アイコンをクリックします。
   - スキーマ要素を選択します。［**階層フィールド**］パネルの上部にあるアクションメニューをクリックし、［**選択項目をマップ解除**］を選択します。
8. すべてのリンクをクリアするには、［**階層フィールド**］パネルの上部にあるアクションメニューをクリックして、［**マッピングのクリア**］を選択します。

詳細プロパティ

階層ビルダトランスフォーメーションの詳細プロパティを設定できます。詳細プロパティによって、セッションログメッセージのトレースレベルとトランスフォーメーション範囲を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
トランスフォーメーション範囲	データ統合が入力データにトランスフォーメーションロジックを適用する方法を指定します。次のいずれかのオプションを選択します。 - 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。データの行が同一トランザクション内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。 - 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、データ統合は入力トランザクションの境界を削除します。データの行がソース内のすべての行に依存している場合は、[すべての入力] を選択します。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

階層ビルダトランスフォーメーションの例

リレーショナルデータを階層データに変換し、そのデータをターゲットファイルに階層形式で書き込むとします。

スキーマファイルを使用して出力データの階層を定義する階層スキーマを設定する必要があります。

次の例は、使用するスキーマ階層を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.itemfield.com"
targetNamespace="http://www.itemfield.com" elementFormDefault="qualified">
 <xs:element name="Employees">
  <xs:complexType>
   <xs:sequence>
     <xs:element name="Name" type="xs:string" minOccurs="0"/>
     <xs:element name="Address" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element name="Employee" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="EmployeeID" type="xs:string" minOccurs="0"/>
```

```
        <xs:element name="Department" type="xs:string" minOccurs="0"/>
        <xs:element name="Subdivision" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="Building" type="xs:string" minOccurs="0"/>
           <xs:element name="Room" type="xs:string" minOccurs="0"
           maxOccurs="unbounded"/>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

次の例は、使用する 1 つ目の入力ファイルデータを示しています。

```
CompanyName,Address
First National Bank,874 Louis Road
Jackson Industry,13 Sydney Drive
```

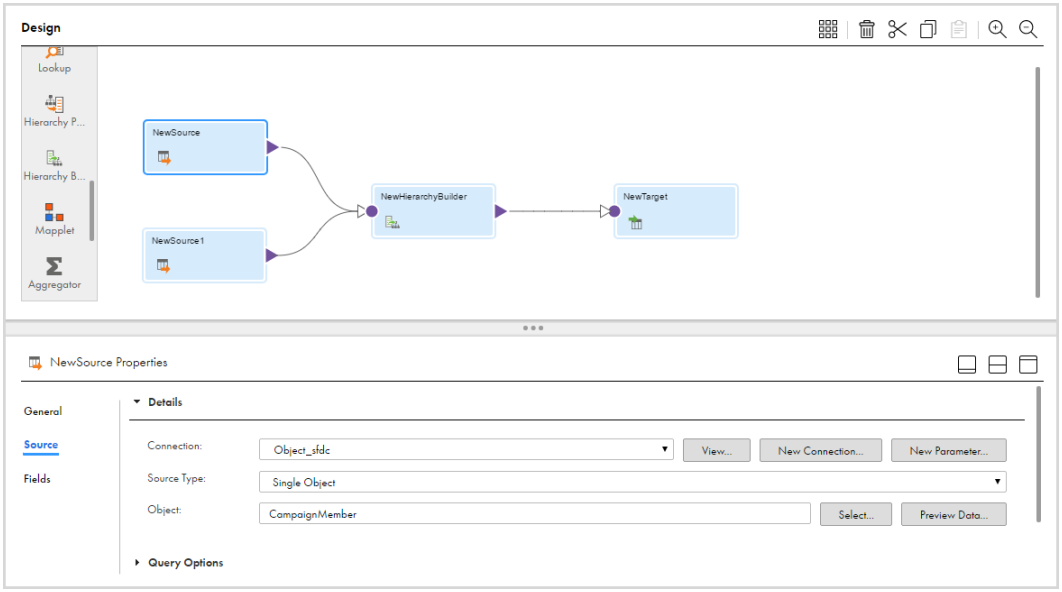
次の例は、使用する 2 つ目の入力ファイルデータを示しています。

```
Name,EmployeeID,Department,Building,Room
First National Bank,122,Credit,6,1532
First National Bank,261,Credit,6,2251
First National Bank,431,Credit,6,5312
Jackson Industry,3875,Manufacture,C,673
Jackson Industry,2837,Manufacture,B,211
```

データ統合で、使用するスキーマ階層を持つ階層スキーマを作成します。

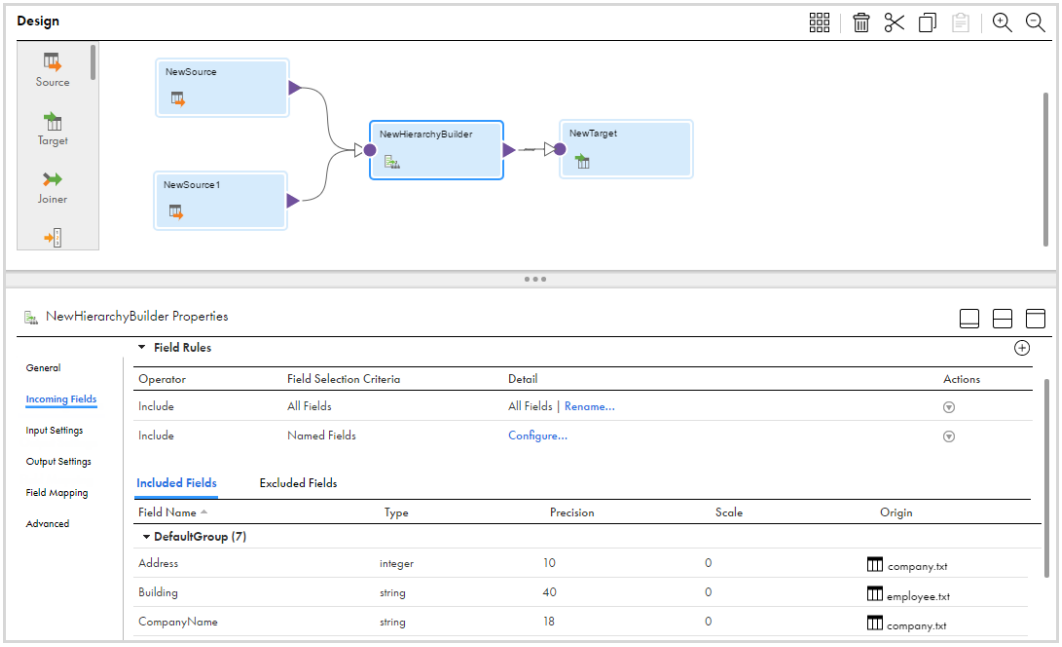
入力データを解析するには、マッピングで階層ビルダートランスフォーメーションを使用して階層入力のデータを変換します。

Mapping Designer で、2 つのソースオブジェクトを追加します。これらのオブジェクトは、解析するデータファイルへのパスが格納されたフラットファイルです。次の図に、ソーストランスフォーメーションの 1 つを示します。



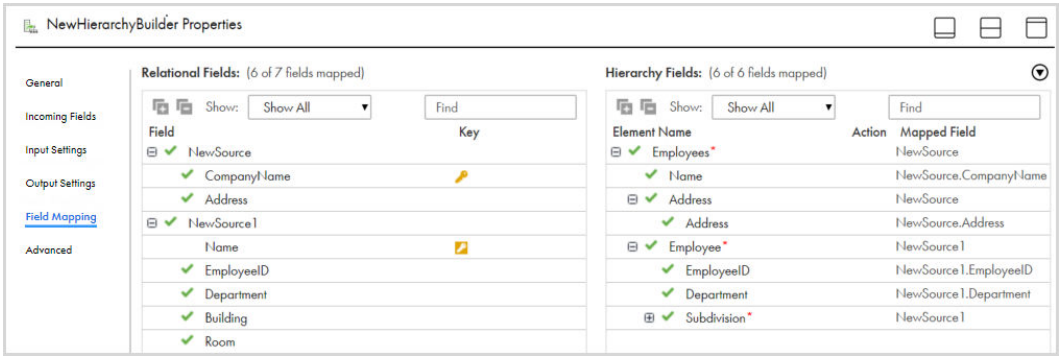
階層ビルダートランスフォーメーションを追加し、NewHierarchyBuilder という名前を使用します。作成した階層スキーマを使用するようにこのトランスフォーメーションを設定します。

NewHierarchyBuilder トランスフォーメーションにソースオブジェクトを接続します。受信データをトランスフォーメーションの各フィールドにマッピングするには、NewHierarchyBuilder トランスフォーメーションを選択します。**[受信フィールド]** タブで、フィールド名の競合がないことを確認します。次の図に、入力フィールド選択を示します。



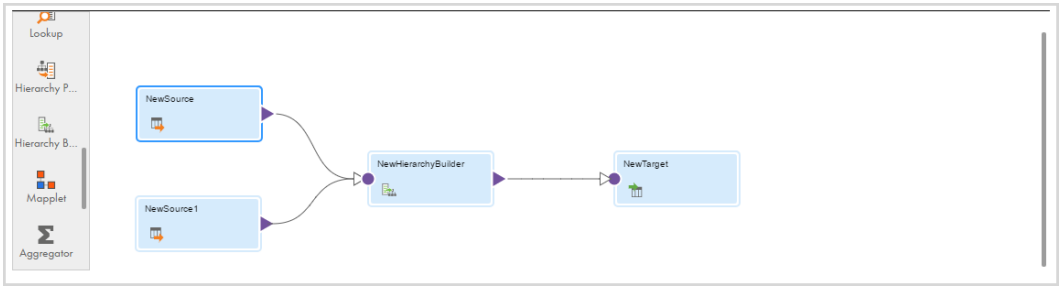
リレーショナルフィールドを階層出力にマッピングするには、**[フィールドマッピング]** タブでプライマリキーおよび外部キーを選択します。次に、階層出力のスキーマ要素にリンクするリレーショナルフィールドを選択します。

次の図に、フィールドマッピング選択を示します。



フィールドのファイルターゲットオブジェクトを追加します。

次の図に、マッピングを示します。



マッピングを実行して、階層形式のデータをターゲットトランスフォーメーションに書き込みます。

次の例は、階層出力を示しています。

```
<Employees>
<Name>First National Bank</Name>
<Address>874 Louis Road</Address>
<Employee>
<EmployeeID>122</EmployeeID>
<Department>Credit</Department>
<Subdivision>
<Building>6</Building>
<Room>1532</Room>
</Subdivision>
</Employee>
<Employee>
<EmployeeID>261</EmployeeID>
<Department>Credit</Department>
<Subdivision>
<Building>6</Building>
<Room>2251</Room>
</Subdivision>
</Employee>
<Employee>
<EmployeeID>431</EmployeeID>
<Department>Credit</Department>
<Subdivision>
<Building>6</Building>
<Room>5312</Room>
</Subdivision>
</Employee>
</Employees>
<Employees>
<Name>Jackson Industry</Name>
<Address>13 Sydney Drive</Address>
<Employee>
<EmployeeID>3875</EmployeeID>
<Department>Manufacture</Department>
<Subdivision>
<Building>C</Building>
<Room>673</Room>
</Subdivision>
</Employee>
<Employee>
<EmployeeID>2837</EmployeeID>
<Department>Manufacture</Department>
<Subdivision>
<Building>B</Building>
<Room>211</Room>
</Subdivision>
</Employee>
</Employees>
```

第 12 章

階層パーサートランスフォーメーション

階層パーサートランスフォーメーションは、階層入力をリレーショナル出力に変換します。このトランスフォーメーションはアップストリームトランスフォーメーションからの XML または JSON 入力を処理し、ダウンストリームトランスフォーメーションにリレーショナル出力を提供します。

サンプルファイルまたはスキーマファイルの出力データの想定される階層を定義する階層スキーマを設定できます。階層パーサー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に関連付けられた階層スキーマに基づいて階層入力を変換します。既存の階層スキーマを使用するか、階層スキーマを設定できます。

注: JSON ベースのスキーマオブジェクトを作成するには、JSON スキーマではなく JSON サンプルを使用します。

複雑な階層構造を解析する場合は、階層ファイル入力をより包括的に処理するために構造パーサートランスフォーメーションを使用することを検討してください。詳細については、第 32 章,「構造パーサートランスフォーメーション」 (ページ 402)を参照してください。

階層パーサートランスフォーメーションを使用するには、適切なライセンスが必要です。

階層パーサートランスフォーメーションの使用

マッピングで階層パーサー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. 階層スキーマを作成します。
2. マッピングに階層パーサートランスフォーメーションを追加します。
3. 階層スキーマを階層パーサートランスフォーメーションに関連付けます。
4. フィールドマッピングを設定し、リレーショナル出力を提供するスキーマ要素を選択します。

階層パーサーのルールとガイドライン

階層パーサートランスフォーメーションを使用する場合は、次のルールとガイドラインを考慮してください。

ソースファイルのサイズ

階層パーサートランスフォーメーションは、最大 1.5 GB までのサイズのソースファイルを処理できます。1.5 GB を超えるファイルを処理するには、インテリジェント構造モデルを使用します。インテリジェント構造モデルの詳細については、「*コンポーネント*」を参照してください。

ブールデータ型の処理

階層パーサートランスフォーメーションは、ブールデータ型の入力に対して常に 0 を返します。

XSD ファイルに基づく階層スキーマ

XSD ファイルに基づく階層スキーマを階層パーサートランスフォーメーションに関連付ける場合、次の XSD コンポーネントタイプでは階層パーサートランスフォーメーションを使用できないことに注意してください。

- xsd:any
- xsd:type
- mixed
- xsi:type
- デフォルト値
- 固定値
- タイプなしの場合: `<xs:element name="A" maxOccurs="unbounded"/>`

10,000 を超える要素と属性、再帰的な要素、または複雑な深い階層を含む大規模または複雑な XSD ファイルを解析するには、構造パーサートランスフォーメーションを使用します。詳細については、第 32 章, 「構造パーサートランスフォーメーション」 (ページ 402)を参照してください。

マップレットトランスフォーメーションでの階層パーサートランスフォーメーションの使用

マップレットで階層パーサートランスフォーメーションを使用する場合は、実行時エラーを防ぐために、結合されたマップレットトランスフォーメーション名が 80 文字を超えないようにしてください。詳細については、「マップレットトランスフォーメーション名」 (ページ 316)を参照してください。

サンプルまたはスキーマ

階層スキーマを作成する場合、基となる階層スキーマとして JSON サンプルファイルまたは XSD スキーマをインポートします。

可能であればスキーマを使用することをお勧めします。スキーマに再帰的な要素を含めることはできません。

サンプルを使用する場合、サンプルはトランスフォーメーションで処理する代表的なデータであり、かつ包括的である必要があります。サンプルには、トランスフォーメーションで処理する可能性のあるすべてのフィールドを含めます。フィールドの値とタイプの組み合わせについても、可能性のあるものをすべて含めます。フィールドの長さは典型的なものにする必要があります。

階層スキーマ

階層スキーマは、データ統合にインポートするスキーマファイルまたはサンプルファイルに基づいています。サンプルファイルをインポートすると、データ統合によってサンプルファイルの構造に基づいてスキーマが生成されます。スキーマでは、入力データの想定される階層を定義します。

階層スキーマは 2 つの方法で作成できます。選択したトランスフォーメーションに関連付けられるスタンドアロン階層スキーマを作成できます。または、特定のトランスフォーメーション内から階層スキーマを作成できます。

スタンドアロン階層スキーマを作成する場合、スキーマのベースとして JSON サンプルファイルまたは XSD ファイルをインポートします。

スタンドアロン階層スキーマとして作成したのか、特定のトランスフォーメーションの一部として作成したのかに関係なく、階層スキーマは任意のトランスフォーメーションに関連付けることができます。

階層スキーマは、作成、編集、削除できます。階層スキーマを編集し、ルートやスキーマ定義を変更できます。ただし、トランスフォーメーションで使用した階層スキーマは、編集または削除できません。階層スキーマを削除する前に、トランスフォーメーションで使用していないことを確認してください。

階層スキーマのルールとガイドライン

階層パーサートランスフォーメーションに関連付ける階層スキーマを作成するときは、次のルールとガイドラインを考慮してください。

- スキーマの要素数は 10,000 未満である必要があります。
- スキーマには、最大 10,000 個のフィールドを含めることができます。
- スキーマには、最大 500 個のグループを含めることができます。
- スキーマには、最大 20 レベルのデータ階層を含めることができます。
- スキーマに 5,000 を超えるポートと 100 を超えるグループが含まれる場合、処理時間に影響することがあります。

階層スキーマの作成

データ統合で階層スキーマを作成します。

1. **［新規］** > **［コンポーネント］** > **［階層スキーマ］** をクリックします。
2. **［新規階層スキーマ］** ページで、名前と説明を入力します。階層スキーマの名前を指定する必要があります。
3. プロジェクトの場所を参照して選択します。
4. スキーマまたはサンプルファイルを選択するには、**［アップロード］** をクリックします。**［ファイルの選択］** をクリックし、XSD ファイルを参照するか、サンプルの JSON ファイルを選択して、**［OK］** をクリックします。

   JSON サンプルファイルを追加すると、データ統合によってサンプルからスキーマが生成されます。
5. 使用可能なルート要素が複数ある XSD ファイルを選択した場合は、ドロップダウンメニューから 1 つのルートを選択します。
6. 別のスキーマを参照するスキーマを選択した場合は、参照先のスキーマもアップロードする必要があります。参照先のスキーマをアップロードするには、**［アップロード］** をクリックし、目的のスキーマファイルを参照して **［OK］** をクリックします。
7. 階層スキーマを保存するには、**［OK］** をクリックします。

入力設定

階層パーサー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に関連付けられた階層スキーマに基づいて階層入力をリレーショナルデータに変換します。

次の表に、入力のプロパティを示します。

プロパティ	説明
入力タイプ	ソーストランスフォーメーションから階層パーサートランスフォーメーションにデータを渡す場合は、**［バッファ］**入力タイプを選択します。 ソーストランスフォーメーションから代わりに参照でデータを渡す場合は、**［ファイル］**入力タイプを選択します。ソーストランスフォーメーションによって入力ファイルへのパスが渡される場合は、この入力タイプが適用されます。
スキーマ	既存の階層スキーマを選択してトランスフォーメーションに追加するか、XSD ファイルまたは JSON サンプルファイルから階層スキーマを作成します。
パススルーフィールドの有効化	マッピングされていないフィールドをダウンストリームトランスフォーメーションに渡すかどうかを決定します。 階層パーサートランスフォーメーションに複数のアップストリームトランスフォーメーションが接続されている場合、トランスフォーメーションに接続された最初にアップストリームトランスフォーメーションからフィールドが渡されます。ダウンストリームトランスフォーメーションにはすべてのアップストリームトランスフォーメーションのフィールドが表示される場合がありますが、マッピングの実行時に余分なフィールドは渡されません。

階層スキーマの選択

階層パーサートランスフォーメーションの階層スキーマを選択します。

1. 階層パーサートランスフォーメーションの**［プロパティ］**パネルで、**［入力設定］**タブをクリックします。
2. **［選択］**をクリックします。

   ［スキーマの選択］ダイアログボックスが表示されます。
3. 一覧から階層スキーマを選択します。
4. 階層スキーマを検索するには、検索条件を選択し、検索フィールドに検索語句を入力して、**［検索］**をクリックします。

   階層スキーマは名前または説明で検索できます。階層スキーマのソートには、名前、説明、または最終変更日を使用できます。
5. 階層パーサートランスフォーメーションに含める階層スキーマを選択し、**［OK］**をクリックします。

   選択した階層スキーマが**［プロパティ］**パネルに表示されます。

サンプルからの階層スキーマの作成

1. 階層パーサートランスフォーメーションの**［プロパティ］**パネルで、**［入力設定］**タブをクリックします。
2. スキーマを作成するには、**［新規］**をクリックします。

   ［新規階層スキーマ］ページが表示されます。
3. **［新規階層スキーマ］**ページで、名前と説明を入力します。階層スキーマの名前を指定する必要があります。

4. プロジェクトの場所を参照して選択します。
5. スキーマまたはサンプルファイルを選択するには、**[アップロード]** をクリックします。**[ファイルの選択]** をクリックし、XSD ファイルを参照するか、サンプルの JSON ファイルを選択して、**[OK]** をクリックします。

   JSON サンプルファイルを追加すると、データ統合によってサンプルからスキーマが生成されます。
6. 使用可能なルート要素が複数ある XSD ファイルを選択した場合は、ドロップダウンメニューから 1 つのルートを選択します。
7. 別のスキーマを参照するスキーマを選択した場合は、参照先のスキーマもアップロードする必要があります。参照先のスキーマをアップロードするには、**[アップロード]** をクリックし、目的のスキーマファイルを参照して **[OK]** をクリックします。
8. 階層スキーマを保存するには、**[OK]** をクリックします。

入力フィールド選択

階層パーサートランスフォーメーションに入力フィールド選択を設定して、階層パーサーのトランスフォーメーションの入力フィールドにマッピングするアップストリームトランスフォーメーションフィールドを定義します。

[プロパティ] パネルの **[入力フィールド選択]** タブを設定します。

次の入力フィールド選択オプションを設定できます。

自動マップ

フィールドを一致する名前とリンクします。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。
- スマートマップ。データ統合で似ている名前のフィールドを照合します。例えば、受信フィールドが Employee_Name、階層スキーマ入力フィールドが Emp_Name の場合、データ統合で Employee_Name フィールドと Emp_Name フィールドを自動的にリンクします。

[正確なフィールド名] と [スマートマップ] を同じフィールドマッピングで使用できます。例えば、[正確なフィールド名] を使用して同じ名前のフィールドを照合してから、[スマートマップ] を使用して類似した名前のフィールドをマッピングできます。

[オートマップ] > **[オートマップを元に戻す]** をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**[アクション]** > **[マップ解除]** をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、[正確なフィールド名] を使用すると、データ統合はマッピングされたフィールドを強調表示します。[スマートマップ] を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

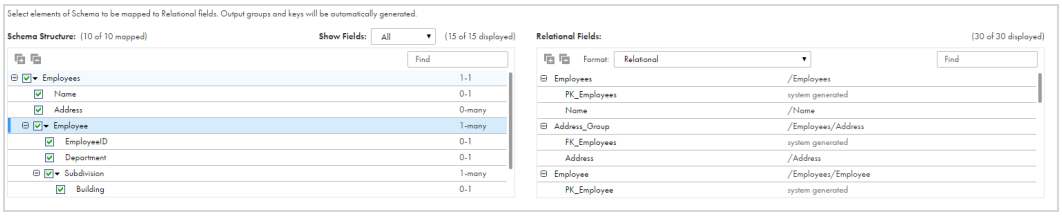
オプション

[受信フィールド] リストに表示されるフィールドを制御します。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示します。入力フィールドリストにフィールド名を表示する方法を決定します。技術フィールド名またはラベルを使用します。

フィールドマッピング

階層パーサートランスフォーメーションでフィールドマッピングを設定し、リレーショナル出力を提供するスキーマ要素を定義します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。フィールドマッピングエディタを使用して、スキーマ要素を選択または除外します。スキーマ要素を除外すると、対応するリレーショナルフィールドがエディタにより削除されます。

次の画像は、フィールドマッピングエディタを示しています。



フィールドマッピングエディタの左側には、スキーマ要素が表示されます。フィールドマッピングエディタの右側には、リレーショナルフィールドが表示されます。トランスフォーメーションでは、複数回出現する各入力要素の個別の出力グループが作成されます。また、トランスフォーメーションではプライマリキーと外部キーが作成されてグループに割り当てられます。プライマリキーは、フィールド名のプレフィックス PK_で表されます。外部キーは、プレフィックス FK_で表されます。

各リレーショナルフィールドの［リレーショナルフィールド］パネルには、リレーショナルフィールドのマッピング元となった階層要素の XPath 式が表示されます。

スキーマ要素を含めるように選択した場合、エディタの右側には対応するリレーショナルフィールドが表示されます。スキーマ要素を除外すると、対応するリレーショナルフィールドがエディタにより削除されます。

ある要素のネストされたすべての子要素を含めるまたは除外するように選択できます。または、階層レベルの 1 つ下にある直下の子要素を含めるまたは除外するように選択することもできます。

リレーショナル出力を非正規化することを選択できます。リレーショナル出力を非正規化する場合は、マッピング対象として選択した階層要素が、すべての 1 つのグループのリレーショナルフィールドにマッピングされます。

リレーショナルフィールドとループを削除できます。プライマリキーが他のグループの外部キーとして機能するグループを削除すると、エディタにより、その他のグループが更新または削除されます。削除したグループが別の繰り返し要素内でネストされる繰り返し要素を反映していた場合、上位レベルの繰り返し要素のプライマリキーが他のグループの外部キーになります。削除するグループが最上位レベルの繰り返し要素を反映している場合、エディタですべてのグループが削除されます。

変換する要素の選択

リレーショナル出力フィールドに変換するスキーマ要素を選択します。

1. 階層パーサートランスフォーメーションの**［プロパティ］**パネルで、**［フィールドマッピング］**タブをクリックします。

   フィールドマッピングエディタに、トランスフォーメーション入力要素と出力フィールドが表示されます。エディタの左にはスキーマ要素が表示されます。エディタの右にはリレーショナル出力フィールドが表示されます。
2. 子要素を表示するには、要素を展開します。
3. 要素を検索するには、検索フィールドに検索語句を入力して、**［検索］**をクリックします。

4. リレーショナル出力を非正規化するには、**［フォーマット］**フィールドで**［非正規化］**を選択して、リレーショナル出力に含める要素を選択します。デフォルトでは、リレーショナル出力は非正規化されず、**［形式］**フィールドの設定は**［リレーショナル］**になります。
5. リレーショナル出力に子要素のない要素を含めるには、要素を選択します。
6. リレーショナル出力から子要素のない要素を除外するには、要素をクリアします。
7. 子要素がある要素の場合、次のオプションから選択できます。
   - リレーショナル出力にすべての子要素を含めるには、要素を右クリックして**［すべての子孫をマッピングする］**を選択します。
   - リレーショナル出力に直下の子要素を含めるには、要素を右クリックして**［直下の子をマッピングする］**を選択します。
   - リレーショナル出力からすべての子要素を除外するには、要素を右クリックして**［すべての子孫のマッピングを解除する］**を選択します。
   - リレーショナル出力から直下の子要素を除外するには、要素を右クリックして**［直下の子のマッピングを解除する］**を選択します。
8. リレーショナルフィールドを検索するには、検索フィールドに検索語句を入力して、**［検索］**をクリックします。
9. 出力からリレーショナル出力フィールドを除外するには、フィールドをクリックします。

出力フィールド

階層パーサートランスフォーメーションで使用するスキーマ要素を選択すると、［プロパティ］パネルの**［出力フィールド］**タブにスキーマ出力フィールドが表示されます。

Mapping Designer には、各出力グループの各出力フィールドについて、名前、タイプ、精度、スケール、基点が表示されます。

出力フィールドの精度を編集するには、その精度をクリックして、必要な精度を入力します。

プライマリキーフィールドと外部キーフィールド以外のトランスフォーメーション出力フィールドを編集できます。

出力グループの選択

トランスフォーメーションをダウンストリームトランスフォーメーションにリンクする場合、ダウンストリームトランスフォーメーションにマッピングする出力グループを 1 つ選択する必要があります。

1. トランスフォーメーションをダウンストリームトランスフォーメーションにリンクします。

   ［出力グループの選択］ダイアログボックスが表示されます。
2. 出力グループを検索するには、検索フィールドに検索語句を入力して、**［検索］**をクリックします。
3. 出力グループを選択します。
4. **［OK］**をクリックします。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

階層パーサートランスフォーメーションの例

階層データをリレーショナルデータに変換し、そのデータをターゲットファイルにリレーショナル形式で書き込むとします。

スキーマファイルを使用して入力データの階層を定義する階層スキーマを設定する必要があります。

次の例は、使用するスキーマ階層を示しています。

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.itemfield.com"
targetNamespace="http://www.itemfield.com" elementFormDefault="qualified">
<xs:element name="root">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Emp_Details" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="employee">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Employeeid" type="xs:short"/>
          <xs:element name="Name">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="Firstname" type="xs:string"/>
             <xs:element name="Lastname" type="xs:string"/>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         <xs:element name="Dependents" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element name="address" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Addressid" type="xs:byte"/>
         <xs:element name="Employeeid" type="xs:short"/>
         <xs:element name="Line1" type="xs:string"/>
         <xs:element name="Line2" type="xs:string"/>
         <xs:element name="City" type="xs:string"/>
         <xs:element name="State" type="xs:string"/>
         <xs:element name="Zipcode" type="xs:int"/>
         <xs:element name="Fromdate" type="xs:date"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="EmpAttribute" type="xs:string"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

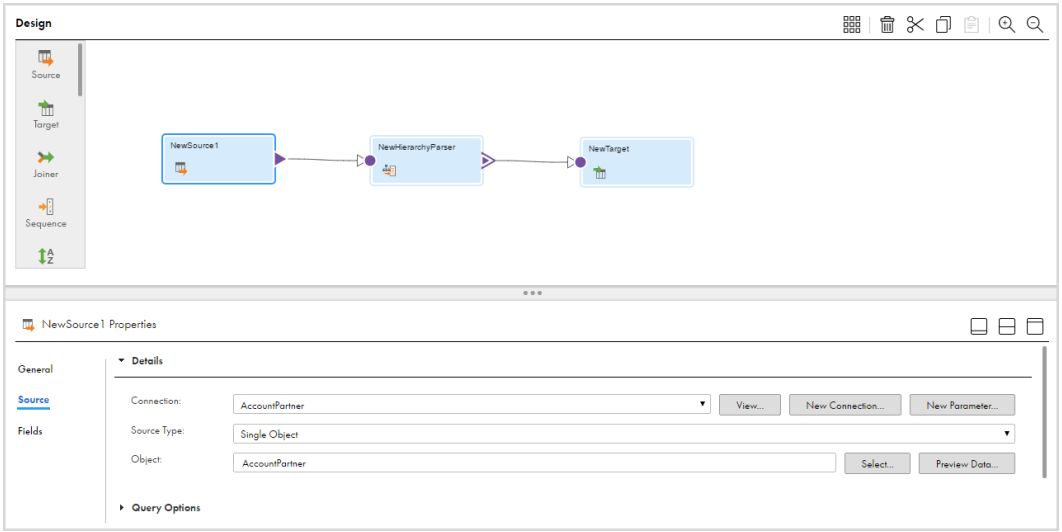
次の例は、使用する入力ファイルデータを示しています。

```
<itm:root xmlns:itm="http://www.itemfield.com">
  <!--Zero or more repetitions:-->
  <itm:Emp_Details EmpAttribute="string">
    <itm:employee>
      <itm:Employeeid>1</itm:Employeeid>
      <itm:Name>
        <itm:Firstname>Gina</itm:Firstname>
        <itm:Lastname>Aniston</itm:Lastname>
      </itm:Name>
      <!--1 or more repetitions:-->
      <itm:Dependents>anyType</itm:Dependents>
    </itm:employee>
    <!--Optional:-->
    <itm:address>
      <itm:Addressid>2</itm:Addressid>
      <itm:Employeeid>1</itm:Employeeid>
      <itm:Line1>Long Tech Park</itm:Line1>
      <itm:Line2>Industrial Zone</itm:Line2>
      <itm:City>Wichita</itm:City>
      <itm:State>KA</itm:State>
      <itm:Zipcode>773301</itm:Zipcode>
      <itm:Fromdate>2011-09-29</itm:Fromdate>
    </itm:address>
  </itm:Emp_Details>
</itm:root>
```

データ統合で、使用するスキーマ階層を持つ階層スキーマを作成します。

入力データを解析するには、マッピングで階層パーサートランスフォーメーションを使用して階層入力のデータを変換します。Mapping Designer で、解析するデータへのパスが含まれているフラットファイルをソースオブジェクトとして追加します。

次の図に、選択したソーストランスフォーメーションを示します。

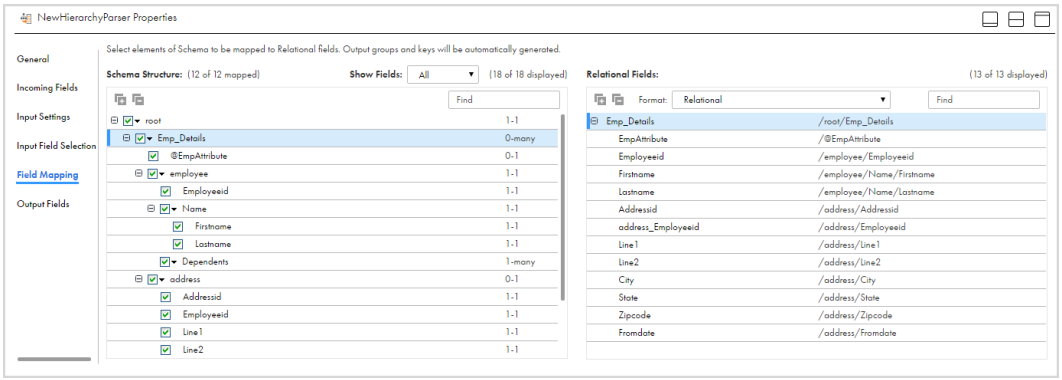


階層パーサートランスフォーメーションを追加し、NewHierarchyParser という名前を使用します。作成した階層スキーマを使用するようにこのトランスフォーメーションを設定します。

ソースオブジェクトを NewHierarchyParser トランスフォーメーションに接続します。受信データをトランスフォーメーションの各フィールドにマッピングするには、NewHierarchyParser トランスフォーメーションを選択します。**[入力フィールド選択]** タブで、選択した受信フィールドをソーストランスフォーメーションから NewHierarchyParser 階層スキーマ入力フィールドにマッピングします。

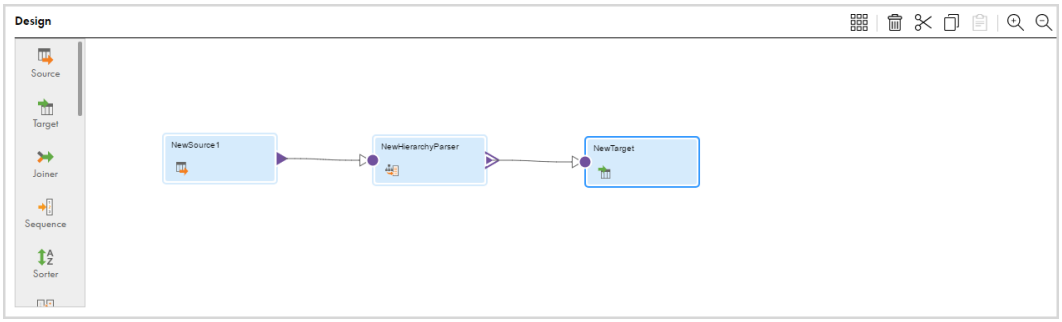
データをリレーショナルフィールドにマッピングするには、**[フィールドマッピング]** タブで、出力へのリレーショナルフィールドとして反映されるスキーマ要素を選択します。

次の図に、フィールドマッピング選択を示します。



フィールドのファイルターゲットオブジェクトを追加します。

次の図に、マッピングを示します。



マッピングを実行して、リレーショナル形式のデータをターゲットトランスフォーメーションに書き込みます。

次の例は、リレーショナル出力を示しています。

EmpAttribute	Employeeid	Firstname	Lastname	Addressid	Employeeid1	Line1	Line2
string	1	Gina	Aniston	2	1	Long Tech Park	Industrial Zone

City	State	Zipcode	Fromdate
Wichita	KA	773301	9/29/2008 12:00:00 AM

第 13 章

階層プロセッサトランスフォーメーション

詳細モードでは、階層プロセッサトランスフォーメーションを使用して、複雑なデータソースからのデータを処理できます。トランスフォーメーションによって、階層入力またはリレーショナル入力を読み取り、それをリレーショナル、階層、またはフラット化済みの非正規化出力に変換できます。

階層プロセッサトランスフォーメーションは、構造または配列を表す階層フィールドの処理を行うアクティブなトランスフォーメーションです。

階層プロセッサトランスフォーメーションの概要

ソースデータの形式と目的の出力形式に応じて、いくつかの選択肢の中から要件を満たす処理ストラテジを選択できます。

階層プロセッサトランスフォーメーションは、次のモードで動作します。

- 階層からリレーショナル。1 つの階層入力グループを複数の出力グループに変換します。これには、区切りフラットファイルまたはリレーショナルファイルを含めることができます。
- リレーショナルから階層。最大 5 つのリレーショナル入力グループを 1 つの階層出力グループに変換します。
- 階層から階層。1 つ以上の階層入力グループを異なるスキーマを持つ 1 つの階層出力グループに変換します。
- 階層からフラット化済み。1 つの階層入力グループを 1 つのフラット化済みの非正規化出力グループに変換します。

デフォルトの出力形式はリレーショナルです。

階層プロセッサトランスフォーメーションとの間でやり取りするデータには、Microsoft Azure Data Lake Store V2 接続または Amazon S3 V2 接続が必要です。

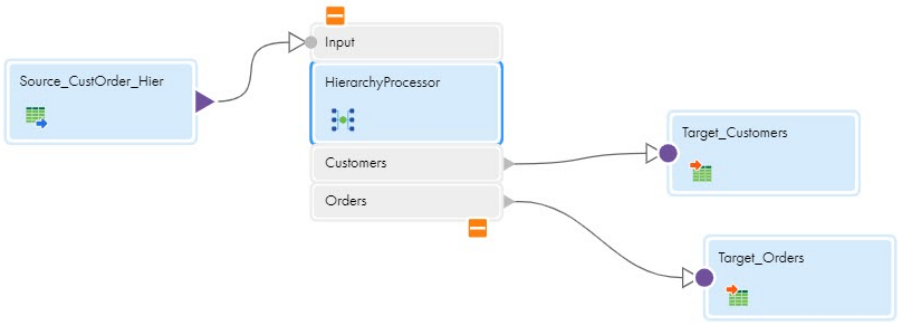
階層プロセッサトランスフォーメーションの詳細については、YouTube で次のビデオをご覧ください:

- Creating a Hierarchical to Relational Mapping Using the Hierarchy Processor Transformation
- Creating a Relational to Hierarchical Mapping Using the Hierarchy Processor Transformation
- Modifying the Schema of Hierarchical Data in an Elastic Mapping

階層型からリレーショナルへのデータ処理

階層データをリレーショナル出力に変換するマッピングでは、1 つの階層入力グループを処理し、そのデータを複数のリレーショナル出力グループに書き込むことができます。出力データは、正規化されたリレーショナルデータとして、または区切りフラットファイルに書き込むことができます。

次の図に、マッピングの例を示します。



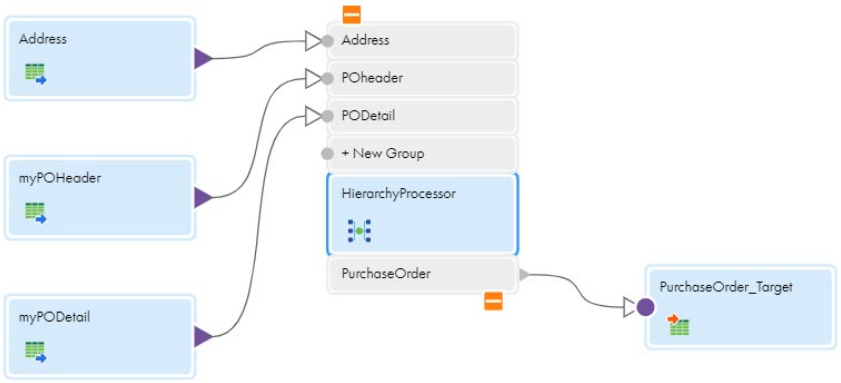
このマッピングのデータソースは、顧客および注文のデータが含まれる複合ファイルです。データは、顧客データのファイルおよび注文データのファイルという 2 つのリレーショナルファイルに流れます。

詳細については、「階層プロセッサトランスフォーメーションを使用したリレーショナル出力の定義」 (ページ 184)を参照してください。

リレーショナルから階層型へのデータ処理

リレーショナルデータを階層出力に変換するマッピングでは、最大 5 つのリレーショナル入力グループを使用して、1 つの階層出力グループに書き込むことができます。このトランスフォーメーションにより、構造と配列を作成できます。また、データソースの結合、データフィールドによるグループ化と並べ替え、特定の情報のフィルタリング、入力データおよび出力データの集計を行うこともできます。

次の図に、マッピングの例を示します。



このマッピングでは、ソース入力に 3 つのリレーショナルファイル（顧客住所データ、発注書、発注書の詳細）が含まれています。3 つのソースファイルからのデータを組み合わせた 1 つの複合ファイルにデータが流れ込みます。

詳細については、「階層プロセッサトランスフォーメーションを使用した階層出力の定義」 (ページ 195)を参照してください。

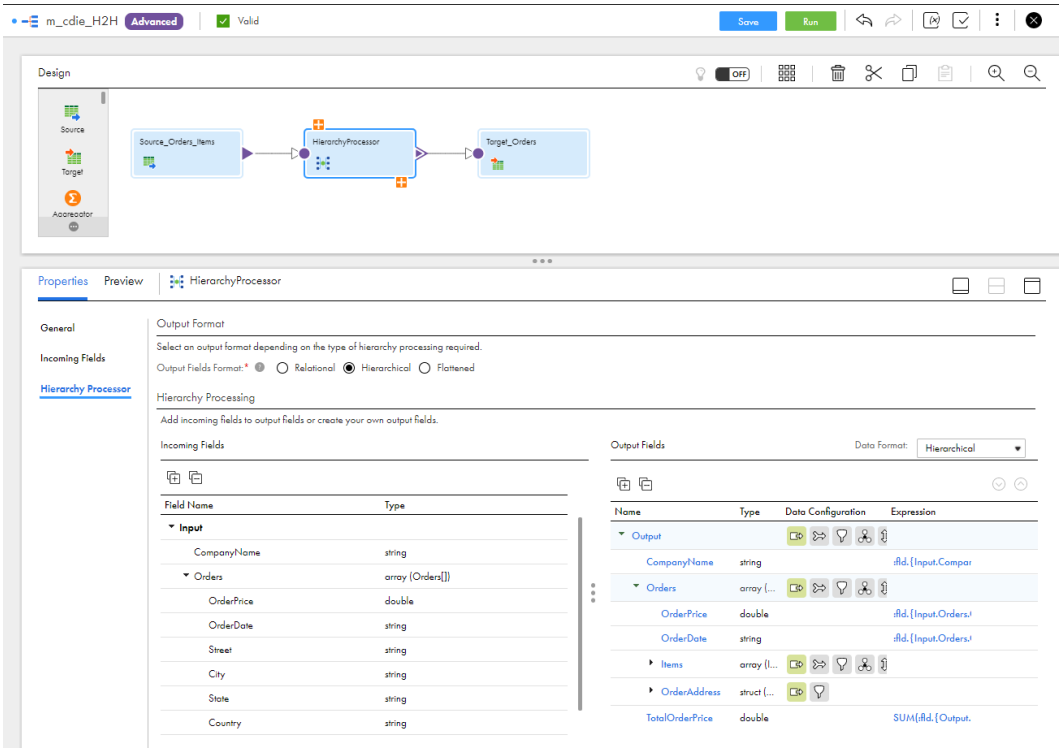
階層型から階層型へのデータ処理

階層データを階層データに変換するマッピングでは、1 つ以上の階層入力グループから読み取りを行って、1 つの階層出力グループに書き込むことができます。

特定のスキーマから別のスキーマに階層入力を変換できます。プリミティブフィールド、構造、および配列からデータを読み取り、データを別の構造に変換できます。

また、変換中のデータを変換することもできます。データソースの結合、グループ化フィールドとソート順フィールドの設定、特定の情報のフィルタリング、受信データと出力データの集計を行うことができます。

次の図に、階層プロセッサトランスフォーメーションを使用して階層データを異なる構造の階層データに変換するマッピングの例を示します。



このマッピングでは、データソースは注文と項目のデータを含む JSON ファイルです。データは、注文情報を含んだ別の JSON ファイルに渡されます。[階層プロセッサ] トランスフォーメーションが選択されており、**[階層プロセッサ]** タブに受信データと出力データの構造が表示されています。

詳細については、「階層プロセッサトランスフォーメーションを使用した階層出力の定義」 (ページ 195)を参照してください。

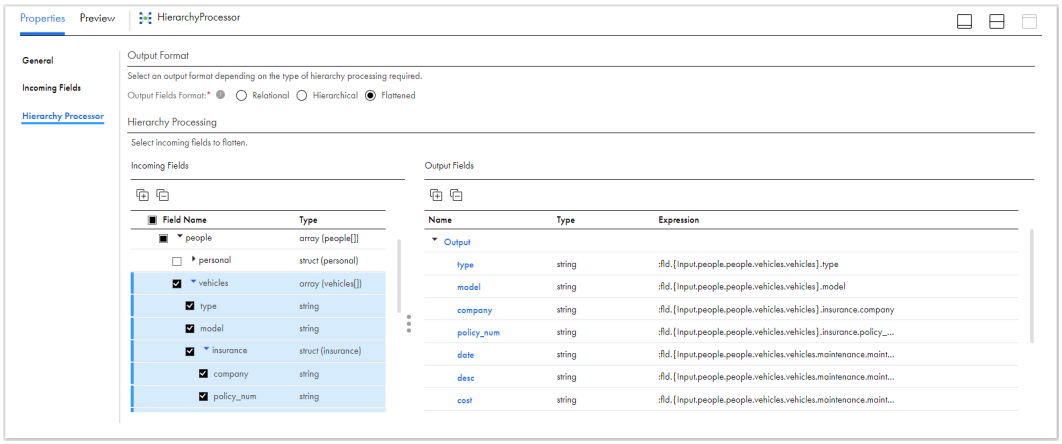
階層からフラット化済みのデータ処理

階層プロセッサトランスフォーメーションには、出力データに対する [フラット化済み] オプションが含まれています。[フラット化済み] 出力形式を使用して、階層入力を非正規化出力に変換します。

階層データをフラット化済みデータに変換するマッピングでは、1 つの階層入力グループから読み取りを行って、1 つのフラット化済み出力グループに書き込むことができます。プリミティブフィールド、構造体、配列からデータを読み取り、完全に非正規化した出力ファイルをすばやく作成できます。受信フィールドの一部のみをフラット化および非正規化することもできます。

兄弟配列を含むデータソースの場合、複雑な結合を必要とせずに、出力データを簡単に非正規化できます。必要な受信フィールドの横にあるチェックボックスをオンにします。階層プロセッサトランスフォーメーションは、フィールドを出力に追加し、式を自動的に作成します。

次の図に、階層プロセッサトランスフォーメーションを使用して階層データをフラット化したデータに変換するマッピングを示します。



このマッピングでは、データソースは personal と vehicle のデータを含む JSON ファイルです。データは、vehicle 情報を含んだフラット化済みファイルに渡されます。[階層プロセッサ] トランスフォーメーションが選択されており、**[階層プロセッサ]** タブに受信データと出力データの構造が表示されています。

詳細については、「階層プロセッサトランスフォーメーションを使用したフラット化済みの出力の定義」 (ページ 227)を参照してください。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

フィールドの制限

エラスティックマッピングは、最大 7,000,000 までの入力フィールドと出力フィールドをサポートします。すべてのマッピングにはこの制限が適用されます。ただし、階層プロセッサトランスフォーメーションを含んだマッピングには複合データが含まれるため、この制限に近づく可能性が高くなります。

階層プロセッサトランスフォーメーションに含まれる結合、子フィールド、ネストされたフィールド、およびフラット化された配列が多いほど、マッピングがフィールド制限を超える可能性は高まります。

マッピングが制限を超過した場合、マッピングコンパイルログに次のメッセージが表示されます。

```
[LDTM_0502] The mapping [<mapping name>] failed because the number of fields in the compiled mapping exceeds the threshold: [7,000,000]. Number of fields: [<actual number>]. Create multiple mappings to process the data incrementally.
```

エラーを解決するには、複数のマッピングを作成して複合データを段階的に処理するか、マッピングのサイズを小さくします。

リレーショナル出力の処理

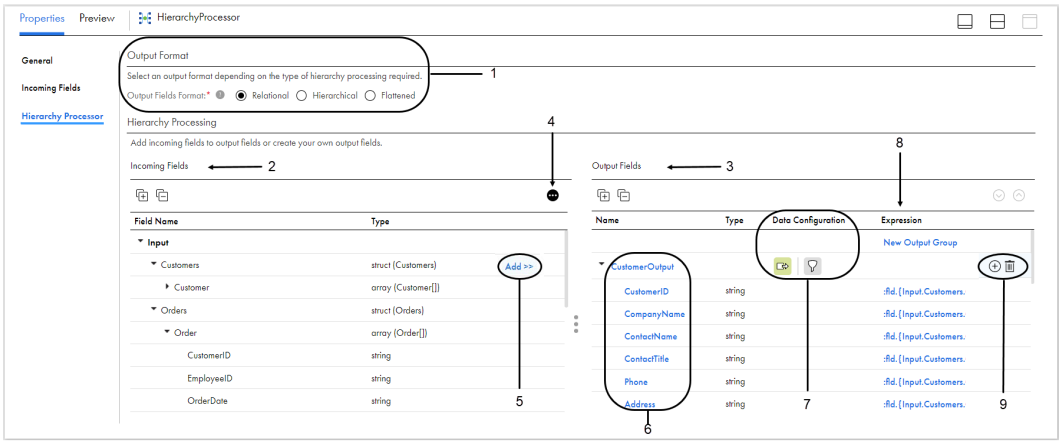
階層プロセッサトランスフォーメーションにより、1 つの階層入力グループを複数の出力グループに変換できます。これらの出力グループには、区切りフラットファイルまたはリレーショナルファイルを含めることができます。

例えば、顧客情報と注文情報を含む顧客注文ファイルがあるとします。このファイルは、階層的な JSON 形式です。リレーショナル顧客テーブルを作成して、マスターデータベースの情報を更新したいと考えています。また、どの注文が増加しているかを確認するために、個別の区切り注文ファイルも必要です。

階層プロセッサトランスフォーメーションを使用したリレーショナル出力の定義

階層プロセッサトランスフォーメーションを定義するには、**[階層プロセッサ]** タブを使用して受信フィールドを出力フィールドにマッピングし、出力データ構造を設定して、必要に応じてリレーショナル出力用のキーを生成します。

次の図に、階層入力とリレーショナル出力を持つ **[階層プロセッサ]** タブを示します。



1. 出力形式。受信階層データを 1 つ以上のリレーショナル出力グループに変換するには、**[リレーショナル]** を選択します。
2. 入力グループ、受信フィールド。これらのフィールドを使用して出力フィールドにマッピングします。
3. 出力グループ、出力フィールド。これらのフィールドを使用して複合出力ファイルを作成します。
4. キーの生成。必要に応じて、入力グループにキーを生成し、出力グループ間のリレーションを定義します。
5. 出力グループへの受信フィールドの追加。フィールドを出力グループに追加するために使用します。
6. 出力フィールド名。フィールド名をクリックして、フィールド名またはデータ型を変更します。
7. データ設定アイコン。出力グループとフィールドを設定するために使用します。
8. 式。式をクリックして、出力フィールドの式を表示またはカスタマイズします。
9. 出力フィールドの追加または削除。出力フィールドを作成または削除する場合に使用します。

ヒント: ［受信フィールド］パネルまたは［出力フィールド］パネルのサイズを変更して、情報を見やすくすることができます。

階層プロセッサトランスフォーメーションの設定

リレーショナル出力を生成するように階層プロセッサトランスフォーメーションを設定できます。

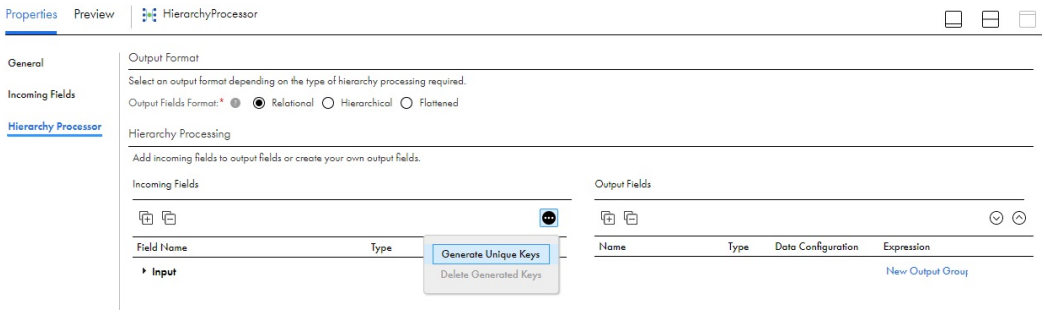
1. **[リレーショナル]** 出力形式を選択します。
2. 受信フィールドを出力に追加するか、手動でフィールドを出力に追加して、トランスフォーメーション出力を設定します。

3. 必要に応じて、入力グループにキーを生成し、出力グループ間のリレーションを定義します。
4. 出力グループとフィールドを設定します。
   a. データソースを設定します。
   b. 必要に応じて、出力フィールド式をカスタマイズします。
   c. 必要に応じて、トランスフォーメーションにフィルタを追加します。

一意キーの生成

階層データをリレーショナル出力に変換するマッピングでは、必要に応じて階層入力グループの一意キーを生成できます。

次の図に、**[階層プロセッサ]** タブで一意のキーを生成するオプションを示します。



一意キーを生成するときは、入力グループのプライマリキーと、入力グループ内の各配列要素のキーを生成します。各キーは、グローバル一意キーと、追加フィールドごとに増加する値を組み合わせたものです。生成されたキーは、他の受信フィールドと同じように出力グループにマッピングできます。

入力親要素から子データセットの出力グループにキーフィールドをマッピングすると、出力データまたは出力グループにプライマリキーと外部キーのリレーションが生じます。このリレーションは、生成されたキーのマッピング方法に基づいて、出力側で生成されます。

リレーショナル出力グループへの受信フィールドの追加

出力には、受信フィールドを個別に追加することや入力グループ全体を追加することができます。受信フィールドは、出力グループ、配列、または構造フィールドに追加できます。必要に応じて、出力フィールドの名前を変更したり削除したりすることができます。

出力フィールドを追加するには、追加する受信フィールドまたは入力グループにカーソルを合わせ、**[追加]** ボタンをクリックします。

受信フィールドのデータ型と出力タイプに基づいて、次の方法でフィールドを追加できます。

受信フィールドの追加

選択した受信フィールドを、選択した出力グループまたはフィールドに追加します。

プリミティブデータ型の受信フィールドを出力に追加するには、[受信フィールドの追加] を選択します。

入力グループの追加

グループ内のすべての受信フィールドを、選択した出力グループまたはフィールドに追加します。

プリミティブデータ型の受信フィールドを出力に追加するには、[入力グループの追加] を選択します。

単一オカレンスの子の追加

構造の配下にネストされた単一オカレンスの子フィールドを含んだ、フィールドの下にあるすべての単一オカレンスの子を出力に追加します。配列の下にネストされた単一オカレンスの子は追加されません。

［単一オカレンスの子の追加］オプションは、受信構造または配列フィールドを選択した場合に選択することができ、出力はリレーショナルとなります。詳細については、「単一オカレンスの子の追加」 (ページ 186)を参照してください。

すべての子孫の追加

フィールドの下にあるすべての子（すべての配列と構造を含む）を出力に追加します。受信フィールドに配列が含まれている場合、データ統合では、配列ごとに個別の出力グループが作成されます。

［すべての子孫の追加］オプションは、受信構造または配列フィールドを選択した場合に選択することができ、出力はリレーショナルとなります。詳細については、「すべての子孫の追加」 (ページ 187)を参照してください。

単一オカレンスの子の追加

受信構造および配列フィールドを出力に追加し、出力形式がリレーショナルである場合は、単一オカレンスの子を追加できます。［単一オカレンスの子の追加］オプションを使用すると、構造の配下にネストされた単一オカレンスの子フィールドを含んだ、フィールドの下にあるすべての単一オカレンスの子が出力グループに追加されます。

［単一オカレンスの子の追加］を選択した場合、受信フィールドには子オブジェクトが必要です。

選択したフィールドに配列が含まれている場合、この配列には複数の要素が含まれる可能性があるため、配列とその子は追加されません。

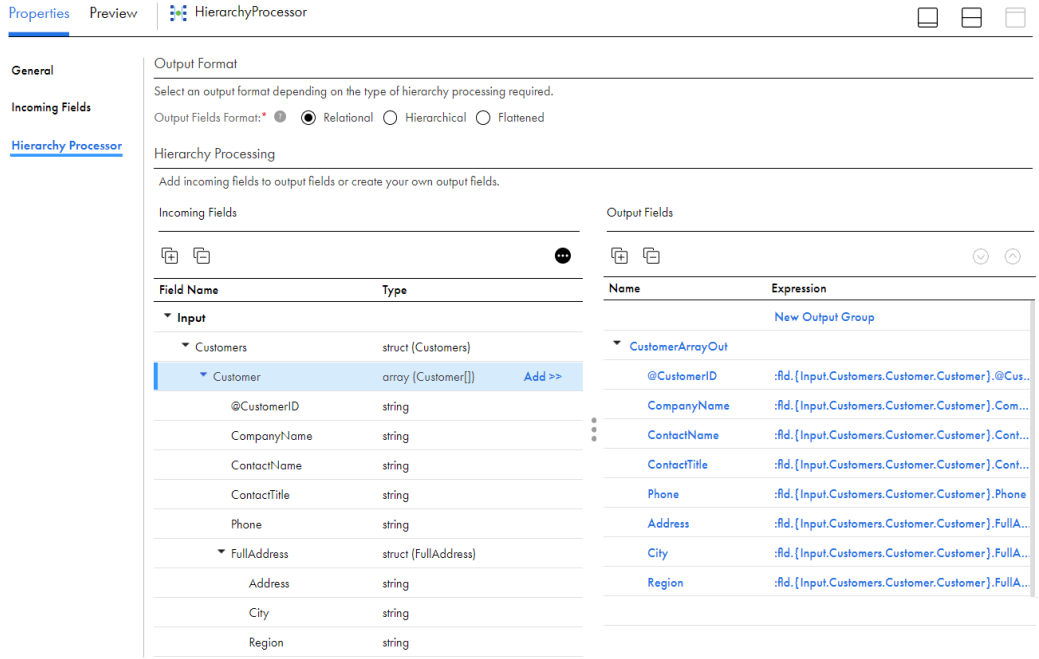
単一オカレンスの子を追加する例

各出力レコードに 1 人の顧客に関する情報が含まれている Customer 配列から顧客情報を抽出する必要があるとします。

［単一オカレンスの子の追加］オプションを選択して、Customer 配列で **［追加］**を選択します。この操作により、FullAddress 構造の下の子を含む、Customer の下のすべての単一オカレンスの子が出力グループにマッピングされます。

Customers 構造レベルで単一オカレンスの子を追加できないことに注意してください。これは、最上位の構造の下に単一オカレンスの子が存在しないためです。

次の図に、受信フィールドと出力フィールドを示します。



すべての子孫の追加

受信構造および配列フィールドを出力に追加し、出力形式がリレーショナルである場合は、すべての子孫を追加できます。[すべての子孫の追加] オプションにより、すべての配列と構造を含む、フィールドの下のすべての子が出力グループに追加されます。

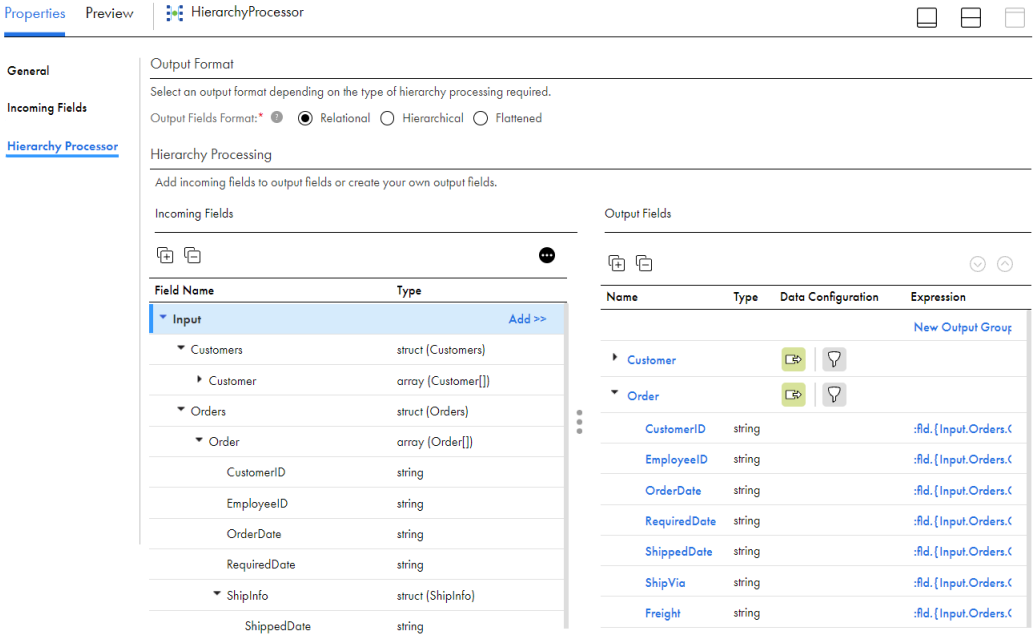
[すべての子孫の追加] オプションを選択した場合、配列を含む階層データを同じ出力グループにマッピングすることはできません。配列を含む階層データを含んだ受信フィールドを出力グループに追加すると、階層プロセッサトランスフォーメーションによって配列ごとに個別の出力グループが作成されます。

すべての子孫を追加する例

すべての受信フィールドをリレーショナル出力グループに追加する必要があるとします。受信フィールドには、配列 Customer と Order が含まれています。

親の受信フィールドの横にある **[追加]** をクリックし、**[すべての子孫の追加]** オプションを選択します。この操作により、Customer 配列のフィールドが最初の出力グループにマッピングされ、Order 配列のすべてのフィールドが 2 番目の出力グループにマッピングされます。

次の図に、入力フィールドと出力フィールドを示します。



リレーショナル出力グループとフィールドの設定

出力グループと出力フィールドを作成および変更できます。出力形式がリレーショナルの場合は、複数の出力グループを作成できます。

受信フィールドを出力グループに追加した後に、**[階層プロセッサ]** タブで出力フィールド名をクリックしてフィールドを変更できます。出力フィールドを手動で追加および定義することもできます。

以下の表に、出力フィールドのプロパティを示します。

プロパティ	説明
子	このフィールドが属する親のフィールドまたはグループ。名前の構造から、グループ、親フィールド、構造名がわかります。例: OUTGROUP.Grandparent.Parent。<struct name>
名前	現在の出力グループまたはフィールドの名前。
タイプ	現在のフィールドのデータ型。プリミティブデータ型を選択できます。
精度	フィールドの合計有効桁数。
スケール	小数点以下の桁数。
配列の要素タイプ	現在の配列要素のデータ型。
配列の要素精度	現在の配列要素の精度。ターゲットデータの作成時に使用されます。
配列の要素スケール	現在の配列要素のスケール。ターゲットデータの作成時に使用されます。
構造名	現在の構造フィールドの構造名。

プロパティ	説明
要素の構造名	構造フィールドの現在の配列における要素の構造名。
説明	フィールドとその用途の説明（オプション）。

注: 表示される出力フィールドプロパティは、データ型によって異なります。

データソースの設定

階層プロセッサトランスフォーメーションでは、データソースは、出力グループまたは出力フィールドのプリミティブ子フィールドにデータを入力する入力グループまたは受信配列を識別します。

出力がリレーショナルの場合、出力グループのデータソースは常に入力グループまたは受信フィールドになります。例えば、出力グループに配列を追加するとします。単一オカレンスの子を追加した場合、出力グループのデータソースは入力グループになります。すべての子孫を追加すると、それぞれの出力グループのデータソースは受信フィールド配列になります。

フィルタ条件の設定

フィルタ条件を定義して、入力データのサブセットを階層プロセッサトランスフォーメーションに投影できます。受信フィールドまたは出力フィールドに基づいてフィルタリングを実行できます。

配列/構造フィールドのデータが出力グループの兄弟フィールドのデータと対応するようにする必要がある場合は、プリミティブフィールドから出力配列/構造フィールドにデータを読み取るようにフィルタ条件を設定します。

フィルタ設定の例

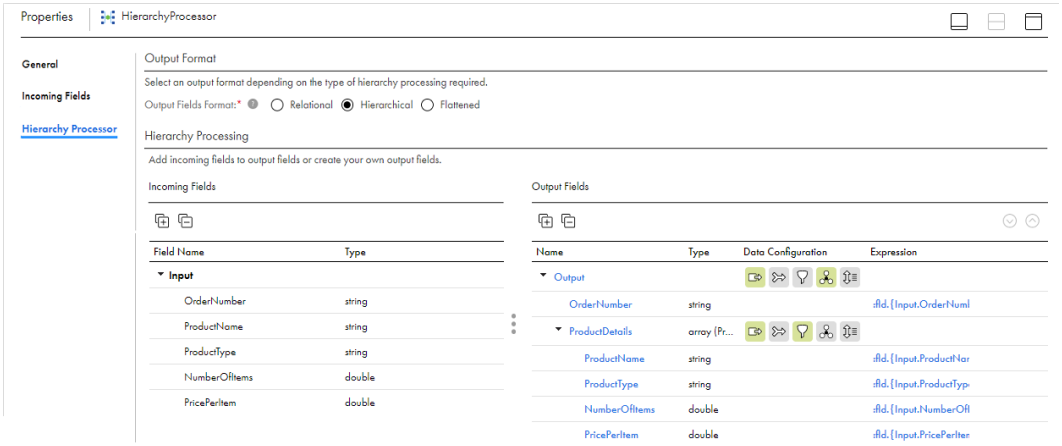
リレーショナルデータを JSON ファイルに変換する必要があるとします。受信データは、注文情報を含むリレーショナルテーブルに含まれています。それぞれの注文には複数の製品が含まれるため、注文テーブルには注文ごとに複数の行が含まれます。

受信データは次のようになります。

```
OrderNumber,ProductName,ProductType,NumberOfItems,PricePerItem
12345,M&Ms Candies Chocolate Peanut Party Size - 38 Oz,Candy,2,14.49
12345,Stella Parm Shredded Cup - 20 Oz,Dairy,1,10.99
12345,AHA Sparkling Water Blueberry Pomegranate - 8-12 Fl. Oz.,Beverages,1,3.33
23456,Weetabix Biscuit Cereal Whole Grain 2 Count - 14 Oz,Breakfast & Cereal,2,4.99
23456,Producers Milk Lowfat 1% - Half Gallon,Dairy,1,2.79
23456,Egglands Best Eggs Cage Free Large Brown - 12 Count,Eggs,1,4.99
```

製品の詳細を、特定の注文番号に関連付けられている配列に読み込む必要があります。

次の図に、受信フィールドと出力フィールドの構造を示します。



［出力フィールド］パネルで［出力］グループのデータソースを Input に設定し、グループ化フィールドを Input.OrderNumber として設定して、出力から重複レコードを削除します。ProductDetails 配列のデータソースを Input に設定します。

ProductDetails 配列の詳細が出力の注文番号に対応するようにするには、配列に次のフィルタ条件を設定します。

```
:fld.{Input.OrderNumber}= :fld.{Output.OrderNumber}
```

レコードをさらに絞り込むには、フィルタで AND 条件を使用します。例えば、「Candy」という製品タイプのレコードを除外するには、次のフィルタ条件を設定します。

```
:fld.{Input.OrderNumber}= :fld.{Output.OrderNumber} AND :fld.{Input.ProductType} ! = 'Candy'
```

出力には注文ごとに 1 つのレコードが含まれ、「Candy」という製品タイプの受信レコードは除外されます。

出力データには、次のようなレコードが含まれます。

```
{
    "OrderNumber":"12345",
    "ProductDetails":[
        {
            "ProductName":"AHA Sparkling Water Blueberry Pomegranate - 8-12 Fl. Oz.",
            "ProductType":"Beverages",
            "NumberOfItems":"1",
            "PricePerItem":"3.33"
        },
        {
            "ProductName":"Stella Parm Shredded Cup - 20 Oz",
            "ProductType":"Dairy",
            "NumberOfItems":"1",
            "PricePerItem":"10.99"
        }
    ]
}
{
    "OrderNumber":"23456",
    "ProductDetails":[
        {
            "ProductName":"Egglands Best Eggs Cage Free Large Brown - 12 Count",
            "ProductType":"Eggs",
            "NumberOfItems":"1",
            "PricePerItem":"4.99"
        },
        {
            "ProductName":"Producers Milk Lowfat 1% - Half Gallon",
            "ProductType":"Dairy",
            "NumberOfItems":"1",
```

```
            "PricePerItem":"2.79"
        },
        {
            "ProductName":"Weetabix Biscuit Cereal Whole Grain 2 Count - 14 Oz",
            "ProductType":"Breakfast & Cereal",
            "NumberOfItems":"2",
            "PricePerItem":"4.99"
        }
    ]
}
```


式の設定

階層プロセッサトランスフォーメーションで式を定義して、カスタマイズされたリレーショナル出力または階層出力を作成できますが、フラット化された出力を作成することはできません。また、式を使用してフィルタ条件を定義します。

階層プロセッサトランスフォーメーションは、さまざまなデータセットからの情報を処理できます。一部のフィールド名は、異なるデータセット間で一意ではない場合があります。その結果、同じフィールド名が別のデータセットまたは同じデータセットの階層内で使用される可能性があるため、フィールドをその名前で単純に参照することはできません。

階層プロセッサトランスフォーメーションの式の構文は、式トランスフォーメーションで使用される構文とは異なります。

階層プロセッサトランスフォーメーションでフィールドを参照するには、次の構文を使用します。

```
:fld.{input_group_name.field_name}.field_name
```

次の表に、構文の詳細を示します。

構文部分	説明
.fld.	階層プロセッサトランスフォーメーションの式の構文を示します。
input_group_name	入力グループまたはデータセットの名前。
field_name	最上位のフィールドでない場合は、フルパス名を含むフィールドの名前。 配列タイプのフィールドがある場合は、配列名を含めます。配列がプリミティブで配列名がない場合は、配列名として elem を使用します。 構造または配列内のフィールドの場合、実際のフィールド名は右中括弧の外側で指定します。
.field_name	構造または配列内のフィールドを参照する場合にのみ、field_name の部分を含めます。以下のガイドラインに従ってください。 - 構造内のフィールドの場合、field_name 部分は次の形式を使用します: .structName.fieldname - 配列内のフィールドの場合、field_name 部分は次の形式を使用します: .fieldName

式の設定

出力形式がリレーショナルまたは階層の場合は、式を設定できます。フラット化された出力の場合、フィールドを追加すると式がデフォルトの形式に固定され、設定ができなくなります。

階層プロセッサトランスフォーメーションで式を設定するには、次の手順を実行します。

1. 次のいずれかのアクションを実行して、出力フィールドを作成します。
   a. 受信フィールドを出力として使用するには、受信フィールドの上にカーソルを合わせると表示される **[追加]** リンクをクリックします。
   b. 新しいフィールドを作成するには、[出力フィールド] パネルで出力グループ名または配列名の近くにある **[新しいフィールド]** をクリックします。
2. [出力フィールド] セクションで式をクリックします。

   新しく作成したフィールドで、このリンクは **[設定]** と表示されます。**[フィールド式]** ダイアログボックスが表示されます。
3. 式エディタで式を入力します。使用するオブジェクトの横にある **[追加]** リンクをクリックすることで、受信フィールド、関数、および変数を式に追加できます。式を手動で入力することも可能です。

   式には、定数、変数、組み込み関数、およびユーザー定義関数を含めることができます。関数をネストすると、複合式を作成することができます。

   注: いずれかのフィールドへの参照を変更すると、式は失敗します。
4. **[検証]** をクリックします。
5. エラーがあれば修正します。
6. **[OK]** をクリックします。

   エディタを閉じて、無効な式のトラブルシューティングを後ほど行うこともできます。

JSON データを使用したマッピングの実行

JSON 形式のデータを含む階層プロセッサ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には、マッピングタスクを使用する必要があります。

JSON 入力の読み取り

JSON データを読み取る場合、入力ファイルは、複数行のスキーマに基づくものと単一行のスキーマに基づくものがあります。

次のサンプルは、JSON スキーマを 1 行で示したものです。

```
{"Name":"Tom","Street":"2100 Seaport Blvd","City":"Redwood City","State":"CA","Country":"USA","Zip":"94063"}
```

次のサンプルは、複数の行にまたがる JSON スキーマを示しています。

```
{
    "Name": "Tom",
    "Surname": "Day",
    "City": "Redwood City",
    "State": "CA",
    "Country": "USA",
    "Zip": "94063"
}
```

デフォルトでは、階層プロセッサトランスフォーメーションは各 JSON スキーマを単一行として読み取ります。複数行にまたがる入力を読み取るには、複数行の JSON ファイルを読み取るようにソーストランスフォーメーションで形式オプションを設定できます。

JSON 出力の書き込み

JSON データを書き込む場合は、各出力レコードを個別のファイルに書き込むか、すべての出力レコードを 1 つのファイルに書き込むことができます。

デフォルトでは、各出力レコードは個別のファイルに書き込まれます。出力レコードを 1 つの JSON 形式のファイルに書き込むには、マッピングタスクで次の Spark セッションプロパティを設定します。

セッションプロパティ名	セッションプロパティ値
spark.sql.shuffle.partitions	1

階層型からリレーショナルへの例

顧客注文ファイルに、現在の顧客連絡先情報と顧客の最近の注文が含まれているとします。この注文ファイルは階層 JSON ファイルであり、会社のクラウドアプリケーションによって生成されています。階層データを処理し、そのデータをターゲットファイルにリレーショナル形式および区切り形式で書き込むことができます。

注文ファイルデータを使用して、マスターデータベースの顧客情報の更新に使用するリレーショナル顧客テーブルを作成するとします。それとは別に、増加中の注文を分析するとします。注文ファイルを使用して、分析用に個別の区切り注文ファイルを作成できます。

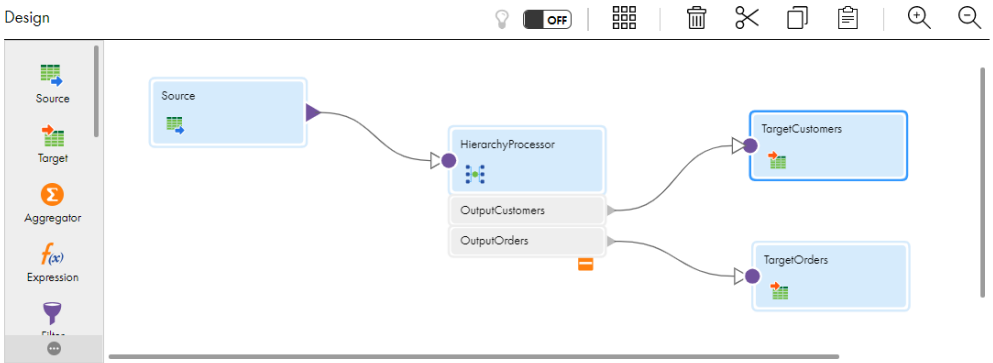
データを、階層入力からリレーショナルおよび区切り出力に変換する必要があるとします。

マッピングを作成して実行するには、次のタスクを実行します。

1. S3 ソースおよびターゲットオブジェクトを使用するため、Amazon S3 V2 コネクタにアクセスできることを確認します。
2. ソース JSON ファイルから階層データを読み取るソーストランスフォーメーションを追加します。
3. ソースオブジェクトに、次のプロパティを設定します。

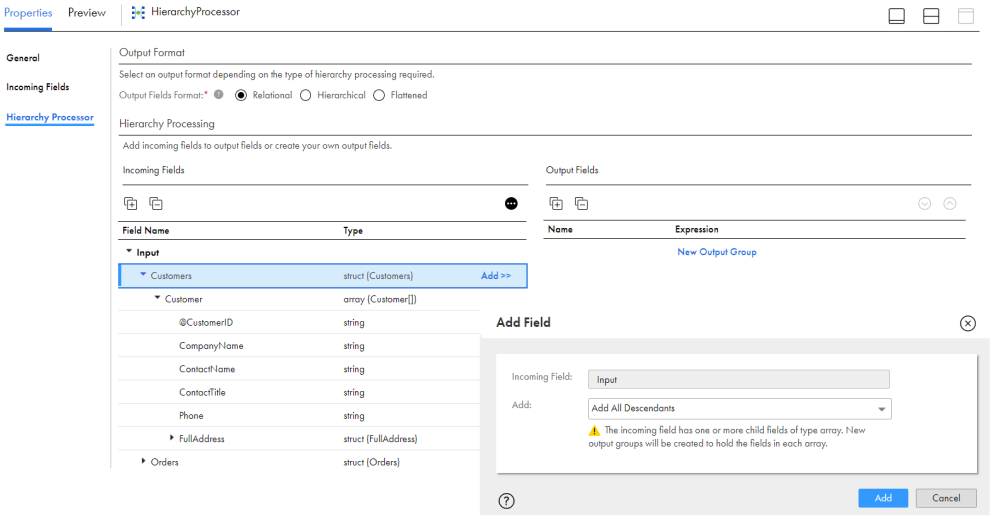
プロパティ	値
接続	Amazon S3 V2
ソースフォーマット	JSON

4. 階層プロセッサトランスフォーメーションを追加します。次の図は、データフローを示しています。



5. OutputCustomers 出力グループを作成して、リレーショナル顧客データファイルを作成します。

次の図は、入力フィールドを追加する方法を示しています。入力フィールドを追加すると、出力グループが作成されます。



6. OutputOrders という出力グループを作成して、区切り注文データファイルを作成します。
7. **[階層プロセッサ]** タブで、**[受信フィールド]** を **[出力フィールド]** にマッピングします。
フィールドを個別に追加したり、構造および配列のフィールドで次のオプションを使用したりすることができます。
   - すべての子孫の追加。フィールド配下のすべての子（すべての配列と構造を含む）を追加します。受信フィールドに配列が含まれている場合、データ統合では、配列ごとに個別の出力グループが作成されます。
   - 単一オカレンスの子の追加。フィールド配下のすべての単一オカレンスの子を、単一オカレンスの子が構造体配下でネストされている場合でも、追加します。
8. ターゲットトランスフォーメーションを追加して、顧客データ出力を書き込みます。
9. ターゲットオブジェクトに対して次のプロパティを設定します。

プロパティ	値
接続	Amazon S3 V2
形式オプション	リレーショナル

10. ターゲットトランスフォーメーションを追加して、注文データ出力を書き込みます。
11. ターゲットオブジェクトに対して次のプロパティを設定します。

プロパティ	値
接続	Amazon S3 V2
形式オプション	区切りファイル

12. OutputCustomers 出力グループを TargetCustomers というターゲットトランスフォーメーションにリンクします。

13. OutputOrders 出力グループを TargetOrders というターゲットトランスフォーメーションにリンクします。
14. マッピングを実行します。

階層出力の処理

階層プロセッサトランスフォーメーションにより、次のタイプの階層出力を処理できます。

- リレーショナルから階層。最大 5 つのリレーショナル入力グループを 1 つの階層出力グループに変換します。
- 階層から階層。1 つ以上の階層入力グループを、異なるスキーマを持つ 1 つの階層出力グループに変換します。

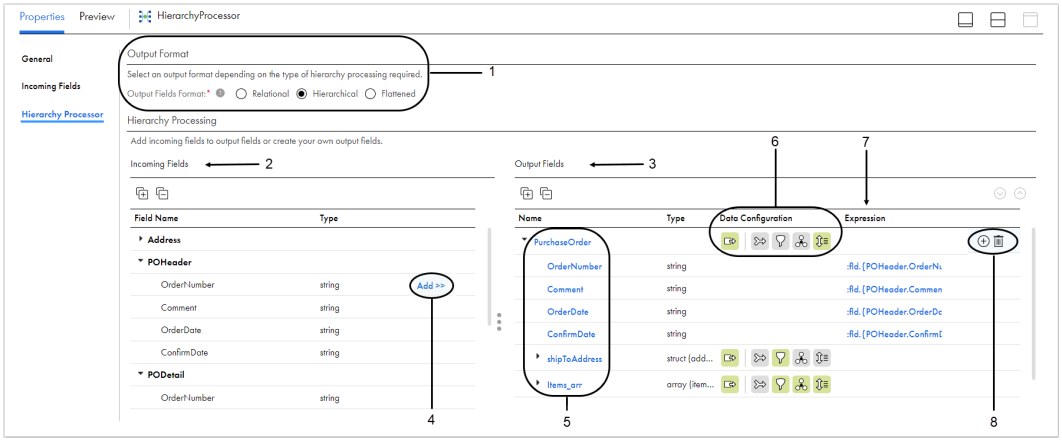
階層プロセッサトランスフォーメーションを使用した階層出力の定義

受信フィールドを出力フィールドにマッピングして、階層出力の出力データ構造を設定できます。

出力データ形式が階層の場合は、次のガイドラインを考慮してください。

- 操作を実行する際に従うべき特定の順序があります。
- 出力グループの名前を変更または削除できます。
- 入力データと出力データの両方を集計できます。

次の図に、リレーショナル入力と階層出力を持つ［**階層プロセッサ**］タブを示します。



1. 出力形式。**［階層］**を選択して、受信フィールドに階層スキーマを構築します。
2. 入力グループ、受信フィールド。これらのフィールドを使用して出力フィールドにマッピングします。
3. 出力フィールドこれらのフィールドを使用して複合出力ファイルを作成します。
4. 出力グループへの受信フィールドの追加。フィールドを出力グループに追加するために使用します。
5. 出力フィールド名。フィールド名をクリックして、フィールド名またはデータ型を変更します。
6. データ設定アイコン。出力グループとフィールドを設定するために使用します。
7. 式。式をクリックして、出力フィールドの式を表示またはカスタマイズします。
8. 出力フィールドの追加または削除。出力フィールドを作成または削除する場合に使用します。

ヒント: ［受信フィールド］パネルまたは［出力フィールド］パネルのサイズを変更して、情報を見やすくすることができます。

階層出力の定義

階層出力を使用して階層プロセッサトランスフォーメーションを定義するには、次のタスクを実行します。

1. **［階層］**出力形式を選択します。
2. 受信フィールドを出力に追加するか、手動でフィールドを出力に追加して、トランスフォーメーション出力を設定します。
3. 出力グループとフィールドを設定します。
    a. データソースを設定します。
    b. 必要に応じて、データソースを結合します。
    c. 必要に応じて、出力フィールド式をカスタマイズします。
    d. 必要に応じて、トランスフォーメーションにフィルタを追加します。
    e. 必要に応じて、ソート順フィールドを使用して出力の順序を定義します。
    f. 必要に応じて、グループ化フィールドを使用して入力データを集計します。
    g. 必要に応じて、出力データを集計します。

操作の順序

階層出力を操作するときは、特定の順序で操作を実行する必要があります。

出力データ形式が階層の場合は、次の順序で操作を実行します。

1. 結合。データソースの結合条件を指定します。
2. フィルタ。必要に応じて、ソースデータのサブセットのみを含むようにフィルタ条件を指定します。
3. グループ化。必要に応じて、集計式の受信フィールドを指定します。
4. ソート順。必要に応じて、ソート済み出力を作成するための受信フィールドを指定します。

出力グループの名前変更と削除

出力データの形式が階層である場合は、入力グループの名前変更または削除を行うことができます。

入力グループを変更すると、入力グループを参照している場合は、次の設定も更新されます。

- 入力データソース名
- 結合データソース名
- ソート順フィールド
- グループ化フィールド

ただし、次の設定は手動で変更する必要があります。

- 式
- 結合条件
- フィルタ条件

注: 出力データの形式が階層である場合は、入力グループ名を変更できません。

階層出力グループへの受信フィールドの追加

個々の受信フィールドまたは入力グループ内のすべてのフィールドを出力グループに追加でき、受信フィールドを出力グループまたは配列あるいは構造フィールドに追加できます。必要に応じて、出力フィールドの名前を変更したり削除したりすることができます。

追加する受信フィールドまたは入力グループの横にある**［階層プロセッサ］**タブの**［追加］**リンクをクリックします。

受信フィールドのデータ型と出力タイプに基づいて、次の方法でフィールドを追加できます。

受信フィールドの追加

選択した受信フィールドを、選択した出力グループまたはフィールドに追加します。

［受信フィールドの追加］オプションは、プリミティブデータ型の受信フィールドを出力に追加した場合に使用できます。

入力グループの追加

グループ内のすべての受信フィールドを、選択した出力グループまたはフィールドに追加します。

［入力グループの追加］オプションは、プリミティブデータ型の受信フィールドを出力に追加した場合に使用できます。

プリミティブ単一オカレンスの子の追加

構造の配下にネストされたプリミティブな単一オカレンスの子フィールドを含んだ、フィールドの下にあるすべてのプリミティブ単一オカレンスの子を出力に追加します。プリミティブ単一オカレンスの子を追加しても、配列の下にネストされた子は追加されません。

プリミティブ単一オカレンスの子は、受信構造または配列フィールドを選択した場合に追加することができ、出力は階層となります。詳細については、「プリミティブ単一オカレンスの子の追加」 (ページ 198)を参照してください。

受信フィールドの保持

選択したフィールドの階層構造を出力に保持します。例えば、構造の配列を出力グループに追加すると、出力グループには同じ構造を持つ構造の配列が含まれます。

［受信フィールドを保持］オプションは、受信構造または配列フィールドを選択した場合に使用することができ、出力は階層となります。詳細については、「受信フィールドの保持」 (ページ 198)を参照してください。

選択した配列のフラット化

プリミティブの配列をプリミティブフィールドにフラット化します。配列内の要素ごとに 1 つの出力レコードを作成します。

［選択した配列をフラット化］オプションは、受信構造または配列フィールドを選択した場合に使用することができ、出力は階層となります。詳細については、「選択した配列のフラット化」 (ページ 199)を参照してください。

選択した配列を構造として追加

構造の配列を構造フィールドにフラット化します。配列の要素ごとに 1 つの出力レコードを作成します。

［選択した配列を構造として追加］オプションは、受信構造または配列フィールドを選択した場合に使用することができ、出力は階層となります。詳細については、「選択した配列を構造として追加」 (ページ 201)を参照してください。

プリミティブ単一オカレンスの子の追加

受信構造および配列フィールドを出力に追加した場合に、プリミティブ単一オカレンスの子を追加できます。受信構造または配列フィールドを階層出力に追加した場合は、［プリミティブ単一オカレンスの子の追加］オプションを使用できます。このオプションにより、構造の配下にネストされたプリミティブな単一オカレンスの子フィールドを含む、フィールドの下にあるすべてのプリミティブ単一オカレンスの子が出力グループに追加されます。

子オブジェクトを持つ受信フィールドに対してのみ、プリミティブ単一オカレンスの子を追加できます。

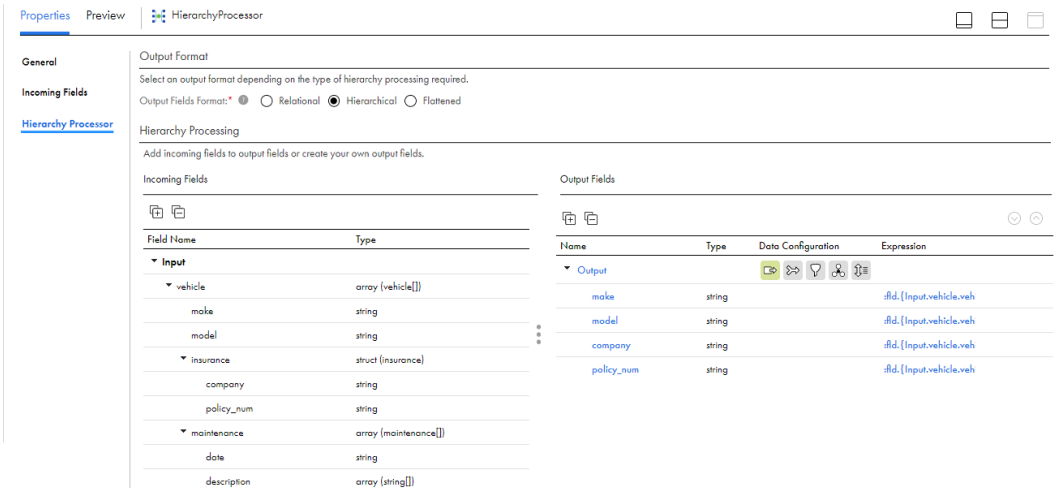
選択したフィールドに配列が含まれている場合、この配列には複数の要素が含まれる可能性があるため、配列とその子は追加されません。

例

車両レコードの配列から、メーカー、モデル、会社、およびポリシー番号を抽出する必要があるとします。それぞれの出力レコードには、1 台の車両に関する情報が含まれている必要があります。

vehicle 配列を出力グループに追加し、**［プリミティブ単一オカレンスの子の追加］**を選択します。

次の図に、受信フィールドと出力フィールドを示します。



日付フィールドは、選択したフィールドの子配列の配下にあり、単一オカレンスではないため、出力グループに追加されないことに注意してください。

受信フィールドの保持

受信構造フィールドと配列フィールドを出力に追加する場合に、受信フィールドを保持することができます。受信フィールドを保持すると、構造を変更せずに受信フィールドが階層出力にコピーされます。

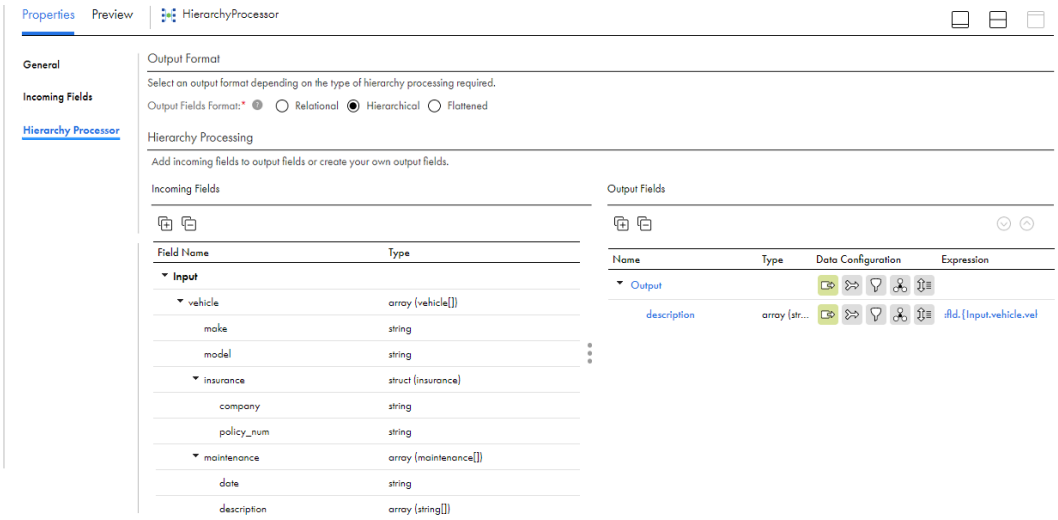
ネストされた配列を追加し、［受信フィールドの保持］を選択した場合、出力グループのデータソース構成によってレコードの作成方法が決まります。

受信フィールドの保持の例

メンテナンスレコードのネストされた配列から説明情報を抽出する必要があるとします。説明情報は文字列の配列に含まれています。出力データも文字列の配列に含める必要があるとします。

説明配列を出力グループに追加し、**［受信フィールドの保持］**を選択します。

次の図に、受信フィールドと出力フィールドを示します。



データ統合が出力配列を作成する際に、説明配列のデータソースが Input.vehicle.vehicle.maintenance.maintenance.desc.elem に設定されます。これは、出力配列の情報が入力グループの desc 配列の要素から取得されていることを示します。出力グループのデータソースによって、出力レコードの構造が決まります。

デフォルトでは、データ統合では、出力グループのデータソースが Input に設定されます。マッピングを実行すると、データ統合ではすべての説明が 1 つの出力レコードに照合されます。それぞれの車両の個別のレコードに説明を書き込むには、データソースを Input.vehicle.vehicle に設定します。メンテナンスレコードごとに個別のレコードに説明を書き込むには、データソースを Input.vehicle.vehicle.maintenance.maintenance に設定します。データソース構成の詳細については、「データソースの設定」 (ページ 205)を参照してください。

選択した配列のフラット化

プリミティブの受信配列を出力に追加すると、選択した配列を同じデータ型のフィールドにフラット化できます。

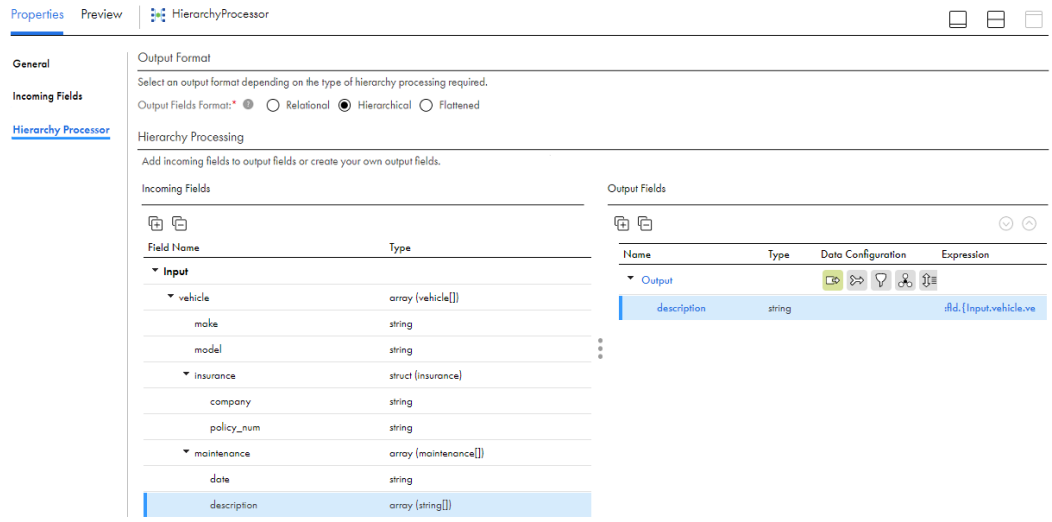
選択した配列をフラット化すると、配列内の要素ごとに 1 つのレコードが作成されます。

配列のフラット化の例

メンテナンスレコードのネストされた配列から説明情報を抽出する必要があるとします。説明情報は文字列の配列に含まれています。出力を文字列フィールドにフラット化する必要があるとします。

description 配列を出力グループに追加し、**[選択した配列のフラット化]** を選択します。

次の図に、受信フィールドと出力フィールドを示します。



出力には、受信データに description が現れるたびに 1 つのレコードが含まれます。

例えば、受信データには次のようなレコードが含まれます。

```
[
  {
    "vehicle": [
      {
        "make": "Toyota",
        "model": "Corolla",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS12876"
        },
        "maintenance": [
          {
            "date": "01/01/2020",
            "description": ["oil filter1", "oil filter2"]
          },
          {
            "date": "01/08/2020",
            "description": ["tire rotation1", "tire rotation2"]
          }
        ]
      },
      {
        "make": "Toyota",
        "model": "RAV4",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS2033"
        },
        "maintenance": [
          {
            "date": "01/02/2020",
            "description": ["air filter replacement1", "air filter replacement2"]
          },
          {
            "date": "01/08/2020",
            "description": ["battery replacement1", "battery replacement2"]
          }
        ]
      }
    ]
  }
]
```

データ統合では、受信データに description が現れるたびに次のような出力レコードが 1 つ作成されます。

```
{"description":"oil filter1"}
{"description":"oil filter2"}
{"description":"tire rotation1"}
{"description":"tire rotation2"}
{"description":"air filter replacement1"}
{"description":"air filter replacement2"}
{"description":"battery replacement1"}
{"description":"battery replacement2"}
```

選択した配列を構造として追加

プリミティブの受信配列を構造として追加すると、配列は構造フィールドにフラット化されますが、子孫配列はフラット化されません。

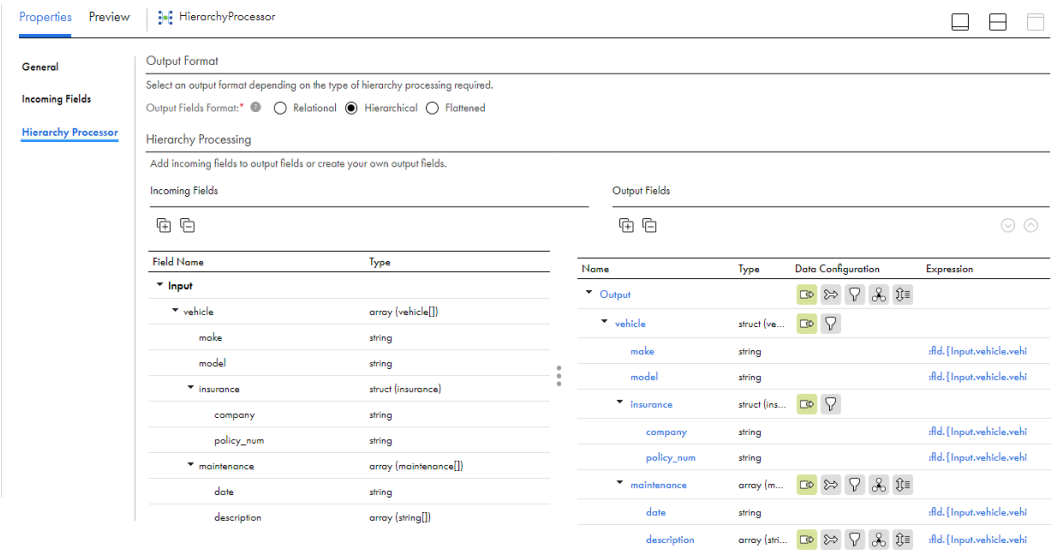
[選択した配列を構造として追加] オプションにより、選択した配列の要素ごとに 1 つのレコードが作成されます。

配列を構造として追加する例

メンテナンスレコードの子配列をフラット化せずに、車両レコードの配列を構造にフラット化する必要があるとします。

vehicle 配列を出力グループに追加し、**[選択した配列を構造として追加]** を選択します。

次の図に、受信フィールドと出力フィールドを示します。



出力には、vehicle 配列の要素ごとに 1 つのレコードが含まれます。

例えば、受信データには、2 台の車両に関するデータを含む次のレコードが含まれます。

```
[
  {
    "vehicle": [
      {
        "make": "Toyota",
        "model": "Corolla",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS12876"
        },
        "maintenance": [
          {
            "date": "01/01/2020",
```

```
            "description": ["oil filter1", "oil filter2"]
          },
          {
            "date": "01/08/2020",
            "description": ["tire rotation1", "tire rotation2"]
          }
        ]
      },
      {
        "make": "Toyota",
        "model": "RAV4",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS2033"
        },
        "maintenance": [
          {
            "date": "01/02/2020",
            "description": ["air filter replacement1", "air filter replacement2"]
          },
          {
            "date": "01/08/2020",
            "description": ["battery replacement1", "battery replacement2"]
          }
        ]
      }
    ]
  }
]
```

階層プロセッサトランスフォーメーションにより、車両ごとに 1 つずつ、次の出力レコードが作成されます。

```
{
    "vehicle":{
        "make":"Toyota",
        "model":"Corolla",
        "insurance":{
            "company":"Allstate",
            "policy_num":"AS12876"
        },
        "maintenance":[
            {
            "date":"01/08/2020",
            "description":["tire rotation2","tire rotation1"]
            },
            {
            "date":"01/01/2020",
            "description":["oil filter2","oil filter1"]
            }
        ]
    }
}
{
    "vehicle":{
        "make":"Toyota",
        "model":"RAV4",
        "insurance":{
            "company":"Allstate",
            "policy_num":"AS2033"
        },
        "maintenance":[
            {
                "date":"01/08/2020",
                "description":["battery replacement2","battery replacement1"]
            },
            {
                "date":"01/02/2020",
                "description":["air filter replacement2","air filter replacement1"]
            }
        ]
```

```
    }
}
```

階層出力グループとフィールドの設定

階層出力の出力グループと出力フィールドを作成および変更できます。

受信フィールドを出力グループに追加した後に、出力フィールド名を選択してフィールドを変更できます。出力フィールドを手動で追加および定義することもできます。

名前とタイプ

以下の表に、階層出力フィールドのプロパティを示します。

プロパティ	説明
子	このフィールドが属する親のフィールドまたはグループ。名前の構造から、グループ、親フィールド、構造名がわかります。例: OUTGROUP.Grandparent.Parent.struct_name
名前	現在の出力グループまたはフィールドの名前。
タイプ	現在のフィールドのデータ型。階層出力の場合は、プリミティブまたは複合データ型を選択できます。
精度	フィールドの合計有効桁数。
スケール	小数点以下の桁数。
配列の要素タイプ	現在の配列要素のデータ型。
配列の要素精度	現在の配列要素の精度。ターゲットデータの作成時に使用されます。
配列の要素スケール	現在の配列要素のスケール。ターゲットデータの作成時に使用されます。
構造名	現在の構造フィールドの構造名。
要素の構造名	構造フィールドの現在の配列における要素の構造名。
説明	フィールドとその用途の説明（オプション）。

注: 表示される出力フィールドプロパティは、データ型によって異なります。

集計オプション

親フィールドを除くすべての出力フィールドには、設定可能な集計オプションがあります。次の表に、集計オプションを示します。

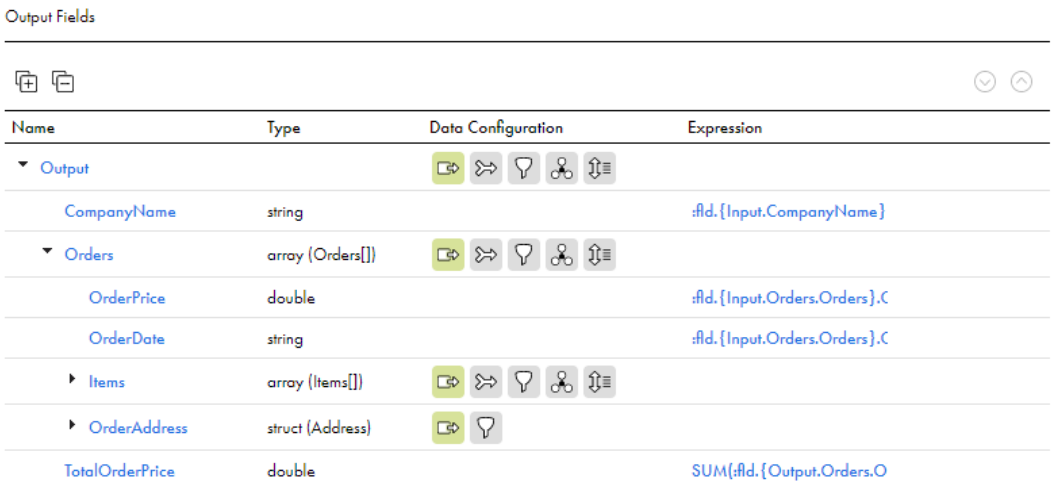
プロパティ	説明
このフィールドを使用して、出力フィールド配列に値を集計します。	このプロパティを選択して、出力データを現在のフィールドに集約します。
出力フィールド	集計する場合は、集計するフィールドを持つ兄弟配列を指定します。配列は、現在の出力フィールドまたはグループの子である必要があります。

出力フィールド配列の値の集計

階層プロセッサトランスフォーメーション出力に配列が含まれている場合は、出力フィールドを追加して配列内の値を集計することをお勧めします。出力フィールドは、配列フィールドの兄弟である必要があります。

例えば、Output グループに注文情報を含む配列が含まれているとします。Orders 配列には、それぞれの注文の価格が格納された OrderPrice フィールドが含まれています。それぞれの会社の合計注文価格を検索する必要があります。

次の図に、出力フィールドを示します。



合計注文価格を見つけるには、TotalOrderPrice というフィールドを出力グループに追加します。

TotalOrderPrice フィールドを編集します。**［このフィールドを使用して、出力フィールド配列に値を集計します］**を選択し、値を集計する出力フィールドとして［Orders］配列を選択します。TotalOrderPrice フィールドに次の式を設定します。

```
SUM(:fld.{Output.Orders.Orders.OrderPrice})
```

データソースの設定

出力グループと、出力内のすべての配列および構造フィールドのデータソースを設定できます。データソースは、出力グループまたは出力フィールドのプリミティブ子フィールドにデータを入力する入力グループまたは受信配列を識別します。

データソースとしてフィールドを選択すると、次のオブジェクトにアクセスできます。

- すべてのプリミティブな子
- すべての祖先のプリミティブな子
- 配列の子の個々の要素
- 祖先の配列の子の個々の要素

出力グループまたは出力フィールドには複数のデータソースを設定できます。この設定を行う場合は、データを結合するための結合条件を設定する必要があります。

特定のレコードを除外するようにフィルタを設定できます。また、データを集計するためのグループ化フィールドと、レコードをソートするためのソート順フィールドを指定することもできます。

階層出力グループおよびフィールドのデータソースは、出力データ構造によって異なります。

親からのデータソースの継承

配列フィールドまたは構造フィールドのデータソースを設定する場合、受信データを使用するか、親のデータソースを継承することで、フィールドの子にデータを入力できます。

受信データを使用する場合、その受信データは配列または構造の子にデータを入力するために使用されます。

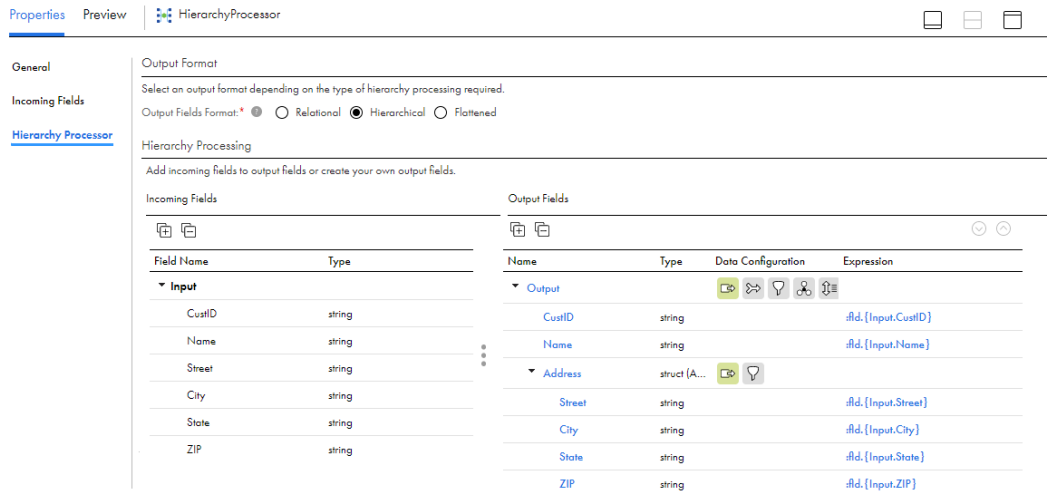
親のデータソースを継承すると、親の出力フィールドに変換されたデータが配列または構造の子に入力されます。これにより、親フィールドのデータトランスフォーメーション（結合やフィルタなど）が保持されます。フィールドにフィルタを適用してデータをさらにフィルタリングすることはできますが、データソース、結合、グループ化フィールド、またはソート順フィールドを設定することはできません。

例えば、顧客 ID が一意である顧客レコードのリレーショナルテーブルからデータを読み取っているとします。受信データには、次のレコードが含まれています。

```
CustID,Name,Street,City,State,ZIP
00234,Ravindra Singh,123 6th St. Apt. 5A,Boston,MA,02134
14416,Melissa Clark,11 Winding Way,Watch Hill,RI,02891
```

顧客の住所フィールドを構造に書き込むとします。

次の図に、受信フィールドと出力フィールドを示します。



[出力フィールド] パネルで Output グループのデータソースを Input に設定し、Address 構造のデータソースを Inherit parent's data sources (Output)に設定します。マッピングを実行すると、階層プロセッサトランスフォーメーションにより、入力データに CustID が現れるたびに 1 つのレコードが作成され、出力の顧客 ID に対応する住所データが構造に生成されます。

```
{
    "CustID":"00234",
    "Name":"Ravindra Singh",
    "Address":{
        "Street":"123 6th St. Apt. 5A",
        "City":"Boston",
        "State":"MA",
        "ZIP":"02134"
    }
}
{
    "CustID":"14416",
    "Name":"Melissa Clark",
    "Address":{
        "Street":"11 Winding Way",
        "City":"Watch Hill",
        "State":"RI",
        "ZIP":"02891"
    }
}
```

Address 構造のデータソースを Input に設定した場合、同じ出力を取得するには、構造に次のフィルタ条件を設定する必要があります: :fld.{Input.CustID} = :fld.{Output.CustID} AND :fld.{Input.Name} = :fld.{Output.Name}.フィルタ条件の設定に関する詳細については、「フィルタ条件の設定」 (ページ 210)を参照してください。

出力フィールドが親のデータを継承する配列である場合は、階層プロセッサトランスフォーメーションにより、1 つの要素を持つ配列が作成されます。

階層出力でのデータソースの設定の例

ネストされた配列を出力に追加し、受信フィールドを保持した場合、作成されるレコードは、出力グループのデータソースの設定方法によって異なります。

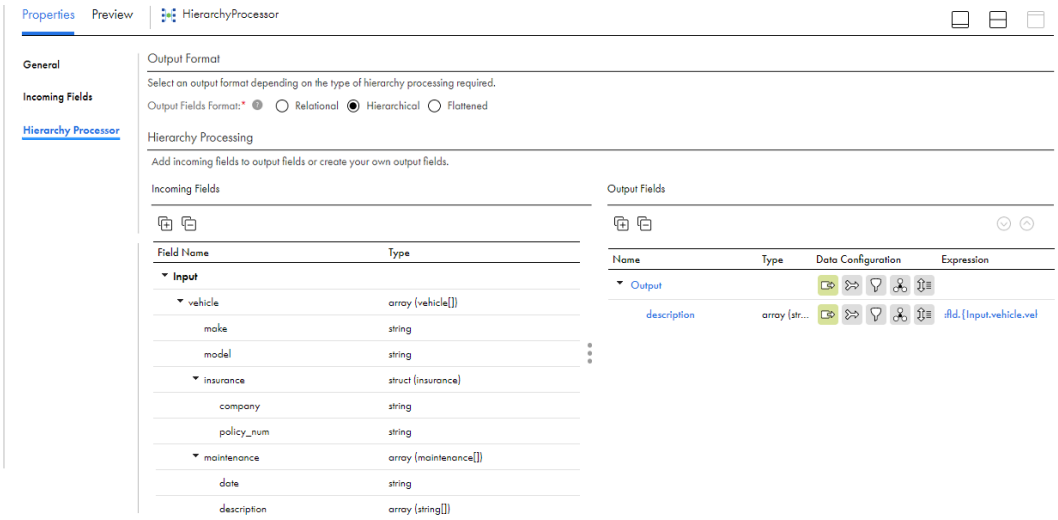
例えば、JSON ファイル内のメンテナンスレコードのネストされた配列から説明情報を抽出するとします。説明情報は文字列の配列に含まれています。出力データも文字列の配列に含める必要があるとします。

次のサンプルデータは、受信レコードを示しています。

```
[
  {
    "vehicle": [
      {
        "make": "Toyota",
        "model": "Corolla",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS12876"
        },
        "maintenance": [
          {
            "date": "01/01/2020",
            "description": ["oil filter1", "oil filter2"]
          },
          {
            "date": "01/08/2020",
            "description": ["tire rotation1", "tire rotation2"]
          }
        ]
      },
      {
        "make": "Toyota",
        "model": "RAV4",
        "insurance": {
          "company": "Allstate",
          "policy_num": "AS2033"
        },
        "maintenance": [
          {
            "date": "01/02/2020",
            "description": ["air filter replacement1", "air filter replacement2"]
          },
          {
            "date": "01/08/2020",
            "description": ["battery replacement1", "battery replacement2"]
          }
        ]
      }
    ]
  }
]
```

階層プロセッサトランスフォーメーションで、description 配列を出力グループに追加し、**［受信フィールドを保持］**を選択します。

次の図に、受信フィールドと出力フィールドを示します。



階層プロセッサトランスフォーメーションが出力配列を作成する際に、説明配列のデータソースが Input.vehicle.vehicle.maintenance.maintenance.desc.elem に設定されます。これは、出力配列の情報が入力グループの desc 配列の要素から取得されていることを示します。デフォルトでは、階層プロセッサトランスフォーメーションによって出力グループのデータソースが Input に設定されます。

出力グループのデータソース設定によって、階層プロセッサトランスフォーメーションが出力レコードを作成する方法が決まります。

すべての説明を 1 つの出力レコードに組み合わせるには、出力グループのデータソースを Input として保持します。これにより、次のような出力が生成されます。

```
{"description":["battery replacement2","battery replacement1","air filter replacement2","air filter
replacement1","tire rotation2","tire rotation1","oil filter2","oil filter1"]}
```

受信データに vehicle が現れるたびに 1 つの出力レコードを作成するには、データソースを Input.vehicle.vehicle に設定します。この場合、出力データには vehicle ごとに 1 つのレコードが含まれます。

```
{"description":["tire rotation2","tire rotation1","oil filter2","oil filter1"]}
{"description":["battery replacement2","battery replacement1","air filter replacement2","air filter
replacement1"]}
```

受信データに maintenance が現れるたびに 1 つの出力レコードを作成するには、データソースを Input.vehicle.vehicle.maintenance.maintenance に設定します。この場合、出力には maintenance レコードごとに 1 つのレコードが含まれます。

```
{"description":["oil filter2","oil filter1"]}
{"description":["tire rotation2","tire rotation1"]}
{"description":["air filter replacement2","air filter replacement1"]}
{"description":["battery replacement2","battery replacement1"]}
```

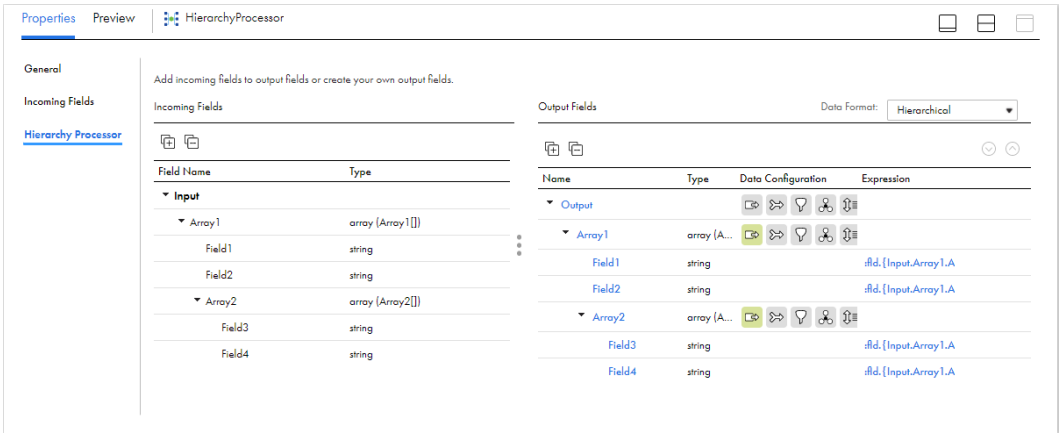
データソースの競合

階層データを階層データに変換し、出力にマルチレベル配列を追加する場合は、出力グループまたは出力フィールドのデータソースが競合しないようにする必要があります。受信配列とその子孫配列の 1 つを同じ出力グループまたは出力フィールドのデータソースとして選択すると、競合が発生します。

データソースの競合がある場合、競合を解決するまでトランスフォーメーションは無効のままとなります。さらに、競合を解決するまで、結合、フィルタ、ソート順フィールド、またはグループ化フィールドを設定することはできません。

データソースの競合の例

次の図で出力グループのデータソースとして Array1 と Array2 を選択すると、データソースの競合が発生します。



Field1 と Field2 のデータを提供するデータソースを特定する方法がないため、競合が発生します。この場合、階層プロセッサトランスフォーメーションでは競合するデータソースエラーが表示されます。

競合を解決するには、出力グループからいずれかのデータソースを削除します。

出力グループまたはフィールドのデータソースの変更

出力グループと、出力内のすべての配列および構造フィールドには、データソースが必要です。入力グループまたは受信フィールドを出力に追加すると、通常は、階層プロセッサトランスフォーメーションによってデータソースが設定されます。データソースは手動で追加または編集できます。

出力グループのデータソースを設定する手順

1. 出力グループの［**データソース**］アイコンをクリックします。
2. 出力グループ用に設定されたデータソースがすでに存在するかどうかを調べます。変更が必要な場合は次の手順を続行し、変更が必要ではない場合はダイアログボックスを終了します。
3. 新しいデータソースを追加するには、［**データソースの追加**］アイコンをクリックします。
4. 省略記号アイコンをクリックして、データソースとして使用する入力グループまたは配列を選択します。
5. 設定を検証します。
6. ［**保存**］をクリックします。

ヒント: ごみ箱アイコンをクリックして、正しくないデータソースまたは不要なデータソースを削除します。

構造または配列フィールドのデータソースを設定する手順

1. 出力グループの［**データソース**］アイコンをクリックします。
2. 構造または配列用に設定されたデータソースがすでに存在するかどうかを調べます。変更が必要な場合は次の手順を続行し、変更が必要ではない場合はダイアログボックスを終了します。
3. 使用するデータソースのタイプを選択します。
   - 親のデータソースを継承する: 構造または配列の親と同じデータソースを使用します。
   - 入力グループまたは受信フィールドを使用する: データソースとして使用する入力グループまたは受信フィールドを選択します。
4. 設定を検証します。
5. ［**Save**］（**保存**）をクリックします。

ヒント: ごみ箱アイコンをクリックして、正しくないデータソースまたは不要なデータソースを削除します。

出力データの設定

階層出力の場合、データソースの結合条件とフィルタ条件、およびグループ化フィールドとソート順フィールドを設定できます。入力データと出力データの両方で集計できます。

結合条件の設定

出力データのフォーマットが階層の場合は、データソースに結合条件を定義できます。出力グループまたは出力フィールドに複数のデータソースがある場合は、結合条件を設定する必要があります。入力グループまたは受信フィールドからのデータを結合するための結合条件を設定します。

[階層プロセッサ] タブで出力グループの結合条件を設定します。

1. 出力グループの **[結合条件]** アイコンをクリックします。
2. 結合条件を追加します。
3. 左側のデータソースを選択します。
4. 結合タイプを選択します。
   - 内部。結合条件に一致する行が含まれます。結合条件に一致しない行は破棄されます。
   - 左外部。右辺パイプラインのすべての行および左辺パイプラインの一致する行が含まれます。一致しない左辺パイプラインの行は破棄します。
   - 右外部。左辺パイプラインのすべての行および右辺パイプラインの一致する行が含まれます。一致しない右辺パイプラインの行は破棄します。
   - 完全外部。結合条件に一致する行および左辺パイプラインと右辺パイプラインのすべての入力データが含まれます。

   注: 大規模なデータセットで外部結合を選択する場合は、マッピングタスクで Spark ドライバのメモリを増やす必要があることがあります。Spark セッションプロパティの詳細については、「*タスク*」を参照してください。
5. 右側のデータソースを選択します。
6. **[結合条件の設定]** をクリックします。
7. フィールドと組み込み関数を選択して式を作成します。
8. 式を検査します。
9. **[保存]** をクリックします。

フィルタ条件の設定

フィルタ条件を定義して、入力データのサブセットを階層プロセッサトランスフォーメーションに投影できます。受信フィールドまたは出力フィールドに基づいてフィルタリングを実行できます。

配列/構造フィールドのデータが出力グループの兄弟フィールドのデータと対応するようにする必要がある場合は、プリミティブフィールドから出力配列/構造フィールドにデータを読み取るようにフィルタ条件を設定します。

フィルタ設定の例

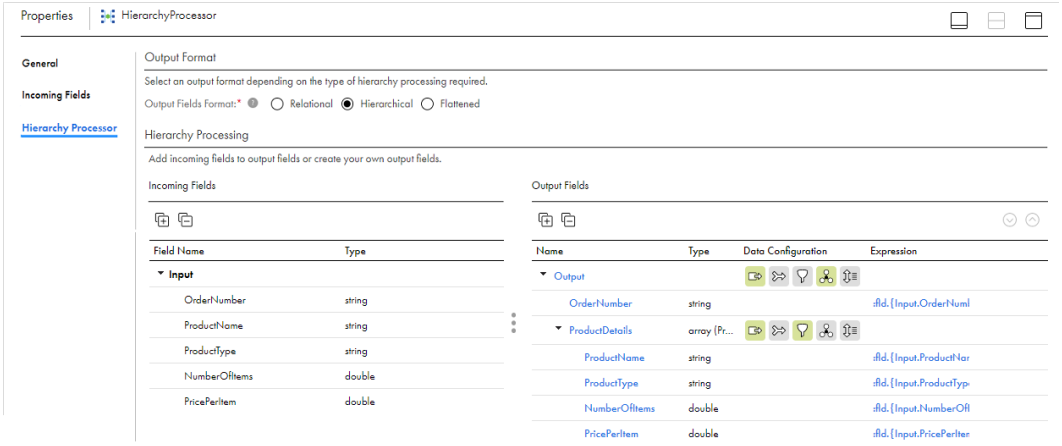
リレーショナルデータを JSON ファイルに変換する必要があるとします。受信データは、注文情報を含むリレーショナルテーブルに含まれています。それぞれの注文には複数の製品が含まれるため、注文テーブルには注文ごとに複数の行が含まれます。

受信データは次のようになります。

```
OrderNumber,ProductName,ProductType,NumberOfItems,PricePerItem
12345,M&Ms Candies Chocolate Peanut Party Size - 38 Oz,Candy,2,14.49
12345,Stella Parm Shredded Cup - 20 Oz,Dairy,1,10.99
12345,AHA Sparkling Water Blueberry Pomegranate - 8-12 Fl. Oz.,Beverages,1,3.33
23456,Weetabix Biscuit Cereal Whole Grain 2 Count - 14 Oz,Breakfast & Cereal,2,4.99
23456,Producers Milk Lowfat 1% - Half Gallon,Dairy,1,2.79
23456,Egglands Best Eggs Cage Free Large Brown - 12 Count,Eggs,1,4.99
```

製品の詳細を、特定の注文番号に関連付けられている配列に読み込む必要があります。

次の図に、受信フィールドと出力フィールドの構造を示します。



[出力フィールド] パネルで [出力] グループのデータソースを Input に設定し、グループ化フィールドを Input.OrderNumber として設定して、出力から重複レコードを削除します。ProductDetails 配列のデータソースを Input に設定します。

ProductDetails 配列の詳細が出力の注文番号に対応するようにするには、配列に次のフィルタ条件を設定します。

```
:fld.{Input.OrderNumber}= :fld.{Output.OrderNumber}
```

レコードをさらに絞り込むには、フィルタで AND 条件を使用します。例えば、「Candy」という製品タイプのレコードを除外するには、次のフィルタ条件を設定します。

```
:fld.{Input.OrderNumber}= :fld.{Output.OrderNumber} AND :fld.{Input.ProductType} ! = 'Candy'
```

出力には注文ごとに 1 つのレコードが含まれ、「Candy」という製品タイプの受信レコードは除外されます。

出力データには、次のようなレコードが含まれます。

```
{
    "OrderNumber":"12345",
    "ProductDetails":[
        {
            "ProductName":"AHA Sparkling Water Blueberry Pomegranate - 8-12 Fl. Oz.",
            "ProductType":"Beverages",
            "NumberOfItems":"1",
            "PricePerItem":"3.33"
        },
        {
            "ProductName":"Stella Parm Shredded Cup - 20 Oz",
            "ProductType":"Dairy",
            "NumberOfItems":"1",
            "PricePerItem":"10.99"
        }
    ]
}
{
    "OrderNumber":"23456",
```

```
    "ProductDetails":[
        {
            "ProductName":"Egglands Best Eggs Cage Free Large Brown - 12 Count",
            "ProductType":"Eggs",
            "NumberOfItems":"1",
            "PricePerItem":"4.99"
        },
        {
            "ProductName":"Producers Milk Lowfat 1% - Half Gallon",
            "ProductType":"Dairy",
            "NumberOfItems":"1",
            "PricePerItem":"2.79"
        },
        {
            "ProductName":"Weetabix Biscuit Cereal Whole Grain 2 Count - 14 Oz",
            "ProductType":"Breakfast & Cereal",
            "NumberOfItems":"2",
            "PricePerItem":"4.99"
        }
    ]
}
```

グループ化フィールドの設定

階層出力の場合、プリミティブ受信フィールドを集計用にグループ化し、入力データのグループごとに 1 つの行を生成できます。

集計のためにフィールドをグループ化する手順

1. 出力グループまたは配列の **[グループ化フィールド]** アイコンをクリックします。
2. グループ化がすでに存在するかどうかを調べます。変更が必要な場合は次の手順を続行し、変更が必要ではない場合はダイアログボックスを終了します。
3. **[新規グループ化フィールドの追加]** アイコンをクリックして、グループ化する新しいフィールドを追加します。
4. 省略記号アイコンをクリックして、グループ化に追加するプリミティブフィールドを選択します。
5. 設定を検証します。
6. **[保存]** をクリックします。

ソート順フィールドの設定

出力データ形式が階層の場合、出力のソート順を定義できます。

注: ソート操作が有効になるためには、次の条件を満たす必要があります。

- 階層プロセッサトランスフォーメーションがターゲットトランスフォーメーションに直接接続されている。
- すべてのソートフィールドがターゲットトランスフォーメーションに接続されている。
- 接続されたソート順フィールドのデータタイプがターゲットフィールドのデータタイプと一致する。

[階層プロセッサ] タブでソート順フィールドを設定します。

1. 出力グループまたは配列/構造フィールドの **[ソート順フィールド]** アイコンをクリックします。
2. ソート基準となる受信データを追加し、データを昇順または降順にソートします。
3. フィールドを並べ替えて、ソート順を調整します。
4. 設定を検証します。
5. **[保存]** をクリックします。

式の設定

階層プロセッサトランスフォーメーションで式を定義して、カスタマイズされたリレーショナル出力または階層出力を作成できますが、フラット化された出力を作成することはできません。また、式を使用してフィルタ条件を定義します。

階層プロセッサトランスフォーメーションは、さまざまなデータセットからの情報を処理できます。一部のフィールド名は、異なるデータセット間で一意ではない場合があります。その結果、同じフィールド名が別のデータセットまたは同じデータセットの階層内で使用される可能性があるため、フィールドをその名前で単純に参照することはできません。

階層プロセッサトランスフォーメーションの式の構文は、式トランスフォーメーションで使用される構文とは異なります。

階層プロセッサトランスフォーメーションでフィールドを参照するには、次の構文を使用します。

```
:fld.{input_group_name.field_name}.field_name
```

次の表に、構文の詳細を示します。

構文部分	説明
.fld.	階層プロセッサトランスフォーメーションの式の構文を示します。
input_group_name	入力グループまたはデータセットの名前。
field_name	最上位のフィールドでない場合は、フルパス名を含むフィールドの名前。 配列タイプのフィールドがある場合は、配列名を含めます。配列がプリミティブで配列名がない場合は、配列名として `elem` を使用します。 構造または配列内のフィールドの場合、実際のフィールド名は右中括弧の外側で指定します。
.field_name	構造または配列内のフィールドを参照する場合にのみ、field_name の部分を含めます。以下のガイドラインに従ってください。 - 構造内のフィールドの場合、field_name 部分は次の形式を使用します: `.structName.fieldname` - 配列内のフィールドの場合、field_name 部分は次の形式を使用します: `.fieldName`

式の設定

出力形式がリレーショナルまたは階層の場合は、式を設定できます。フラット化された出力の場合、フィールドを追加すると、デフォルトの式は設定できなくなります。

階層プロセッサトランスフォーメーションで式を設定するには、次の手順を実行します。

1. 次のいずれかのアクションを実行して、出力フィールドを作成します。
   a. 受信フィールドを出力として使用するには、受信フィールドの上にカーソルを合わせると表示される**［追加］**リンクをクリックします。
   b. 新しいフィールドを作成するには、［出力フィールド］パネルで出力グループ名または配列名の近くにある**［新しいフィールド］**をクリックします。
2. ［出力フィールド］セクションで式をクリックします。

   新しく作成したフィールドで、このリンクは**［設定］**と表示されます。**［フィールド式］**ダイアログボックスが表示されます。
3. 式エディタで式を入力します。使用するオブジェクトの横にある**［追加］**リンクをクリックすることで、受信フィールド、関数、および変数を式に追加できます。式を手動で入力することも可能です。

式には、定数、変数、組み込み関数、およびユーザー定義関数を含めることができます。関数をネストすると、複合式を作成することができます。

注: いずれかのフィールドへの参照を変更すると、式は失敗します。

4. **[検証]** をクリックします。
5. エラーがあれば修正します。
6. **[OK]** をクリックします。

エディタを閉じて、無効な式のトラブルシューティングを後ほど行うこともできます。

JSON データを使用したマッピングの実行

JSON 形式のデータを含む階層プロセッサ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には、マッピングタスクを使用する必要があります。

JSON 入力の読み取り

JSON データを読み取る場合、入力ファイルは、複数行のスキーマに基づくものと単一行のスキーマに基づくものがあります。

次のサンプルは、JSON スキーマを 1 行で示したものです。

```
{"Name":"Tom","Street":"2100 Seaport Blvd","City":"Redwood City","State":"CA","Country":"USA","Zip":"94063"}
```

次のサンプルは、複数の行にまたがる JSON スキーマを示しています。

```
{
    "Name": "Tom",
    "Surname": "Day",
    "City": "Redwood City",
    "State": "CA",
    "Country": "USA",
    "Zip": "94063"
}
```

デフォルトでは、階層プロセッサトランスフォーメーションは各 JSON スキーマを単一行として読み取ります。複数行にまたがる入力を読み取るには、複数行の JSON ファイルを読み取るようにソーストランスフォーメーションで形式オプションを設定できます。

JSON 出力の書き込み

JSON データを書き込む場合は、各出力レコードを個別のファイルに書き込むか、すべての出力レコードを 1 つのファイルに書き込むことができます。

デフォルトでは、各出力レコードは個別のファイルに書き込まれます。出力レコードを 1 つの JSON 形式のファイルに書き込むには、マッピングタスクで次の Spark セッションプロパティを設定します。

セッションプロパティ名	セッションプロパティ値
spark.sql.shuffle.partitions	1

リレーショナルから階層型への例

2 つの発注書テーブルと顧客住所テーブルからの顧客販売データを使用して、階層形式の発注書ファイルを作成する必要があるとします。

階層プロセッサトランスフォーメーションを使用して、階層形式で発注書を作成します。

POHeader テーブルには、顧客からの注文に関する基本情報が含まれています。

OrderNumber	コメント	OrderDate	ConfirmDate
1	AppD for POD4	2020-10-01 00:00:00.0	2020-10-02 00:00:00.0
2	GoJS for IICS	2020-10-12 00:00:00.0	2020-10-12 00:00:00.0

住所テーブルには、各注文の顧客の住所情報が含まれています。

OrderNumber	AddressType	名前	街路	City	状態	国	Zip
1	ShipTo	Tom	2100 Seaport Blvd	Redwood City	CA	USA	94063
1	BillTo	Tom	2100 Seaport Blvd	Redwood City	CA	USA	94063
2	ShipTo	Bill	1630 S Delaware St	San Mateo	CA	USA	94402
2	BillTo	Bill	PO Box 313	San Mateo	CA	USA	94402

PODetail テーブルには、顧客の発注書に関する詳細が含まれています。

OrderNumber	ItemNum	ProductName	Quantity	Price	コメント	ShipDate	PartNum
1	1	AppD Agent for JVM	60	500	JVM エージェント	2020-10-15 00:00:00.0	1
1	3	ELB エージェント	10	200	ELB エージェント	2020-10-15 00:00:00.0	3
1	2	MySQL エージェント	2	120	MySQL エージェント	2020-10-16 00:00:00.0	2
1	4	MySQL エージェント	2	120	MySQL エージェント	2020-10-01 00:00:00.0	2
1	5	MySQL エージェント	2	120	MySQL エージェント	2020-10-01 00:00:00.0	2
2	1	GOJS OEM 版	2	20000	GOJS 開発者	2020-10-19 00:00:00.0	101
2	2	GOJS プロフェッショナルサービス	5	5000	GOJS 開発者	2020-10-19 00:00:00.0	102

階層形式で注文書を作成するには、次の手順を実行します。

1. 手順 1.マッピングを設計します。
2. 手順 2.出力グループを作成し、構造を作成します。

3. 手順 3.構造の配列を作成します。
4. 手順 4.出力データを集計します。
5. 手順 5.構造の配列を作成し、データソースを結合します。
6. 手順 6.マッピングを実行します。

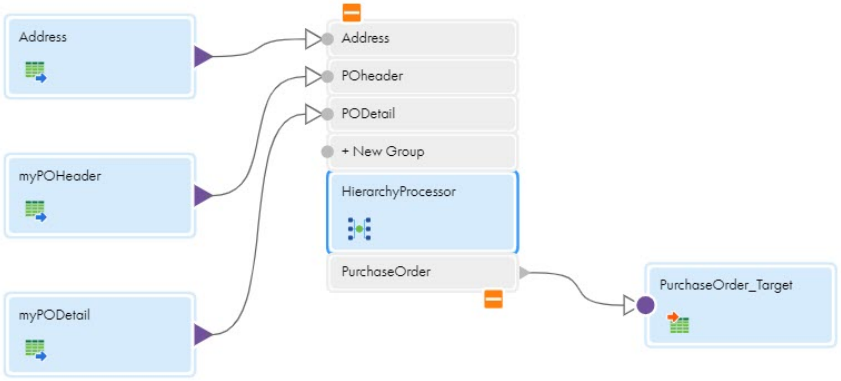
手順 1。マッピングを設計します

最初の手順として、Mapping Designer で、ソースおよびターゲットのトランスフォーメーションを使用して階層プロセッサを設定します。

Mapping Designer で次の手順を実行します。

1. 階層プロセッサトランスフォーメーションを追加し、出力データ形式を階層型に変更します。
2. ［POHeader］、［PODetail］、および［Address］テーブルをソースオブジェクトとして追加します。
3. ソースオブジェクトをデータフローの階層プロセッサトランスフォーメーションに接続します。
4. 階層プロセッサトランスフォーメーションで、［PurchaseOrder］出力グループを追加し、ターゲットオブジェクトをデータフローで接続します。

マッピングは次の図のようになります。



手順 2。出力グループを作成し、構造を作成します

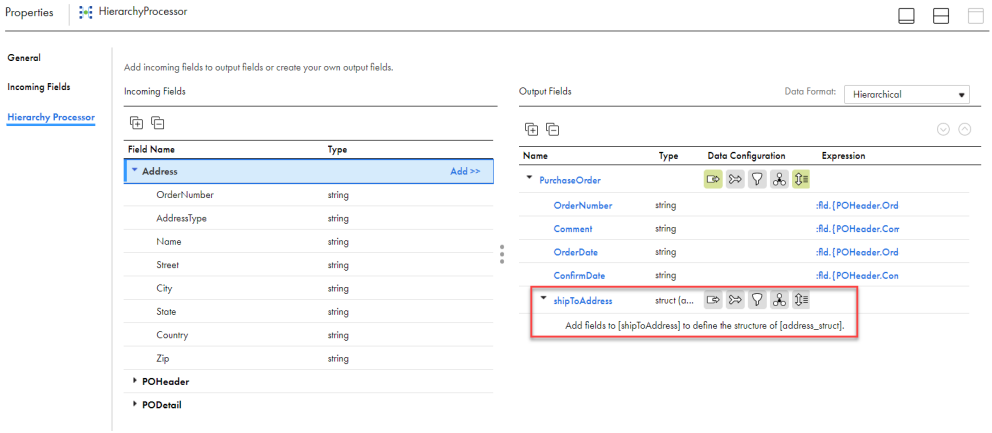
基本的なマッピングを設定した後に、次は出力グループを作成します。

次の手順を実行して、基本的な発注書データを含む出力グループを作成し、配送先住所を追加します。

1. ［POHeader］からのすべての入力フィールドを［PurchaseOrder］出力グループに追加します。
2. 次のプロパティを持つ新しい出力フィールドを追加します。

プロパティ	値
子	PurchaseOrder
名前	shipToAddress
タイプ	struct
構造名	address_struct

3. ［Address］からのすべての入力フィールドを出力グループの［shipToAddress］構造に追加します。



4. 出力グループで不要な次のフィールドを削除します。
   - PurchaseOrder.shipToAddress.OrderNumber
   - PurchaseOrder.shipToAddress.AddressType
5. PurchaseOrder.shipToAddress 構造のフィルタ条件を追加します。
   `:fld.{Address.OrderNumber}=:fld.{PurchaseOrder.OrderNumber} AND :fld.{Address.AddressType}='ShipTo'.`

手順 3。構造の配列を作成します

構造の items 配列に発注書の詳細を追加して、出力の設定を続けます。項目番号で並べ替え、部品番号でグループ化し、入荷数量と入荷価格を集計するようにデータ処理ストラテジを設定します。

以下の手順を実行します

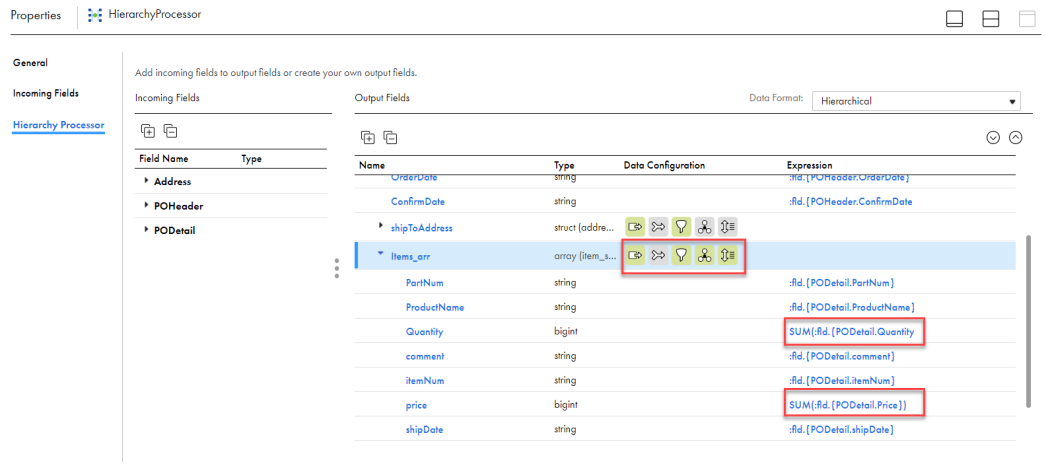
1. 次のプロパティを持つ新しい出力フィールドを追加します。

プロパティ	値
子	PurchaseOrder
名前	Items_arr
タイプ	array
配列の要素タイプ	struct
要素の構造名	item_str

2. ［PODetail］からのすべての入力フィールドを出力グループの［Items_arr］配列に追加します。
3. 出力グループで不要な次のフィールドを削除します: PurchaseOrder.Items_arr.OrderNumber。
4. ［PurchaseOrder.Items_arr］配列のフィルタ条件［`:fld.{PODetail.OrderNumber}=:fld.{PurchaseOrder.OrderNumber}`］を追加します。
5. ［PurchaseOrder.Items_arr］配列のグループ化フィールドを次のように設定します: PODetail.PartNum。
6. ［PurchaseOrder.Items_arr］配列の昇順のソート順フィールドを次のように設定します: PODetail.ItemNum。

7. ［PurchaseOrder.Items_arr］配列の［PODetail.Quantity］のフィールド式を SUM(:fld.{PODetail.Quantity})に更新して、数量を集計します。
8. ［PurchaseOrder.Items_arr］配列の［PODetail.Price］のフィールド式を SUM(:fld.{PODetail.Price})に更新して、価格を集計します。

以下の図は、出力グループの［Items_arr］配列のデータ設定アイコンと式を示しています。



手順 4.出力データを集計します

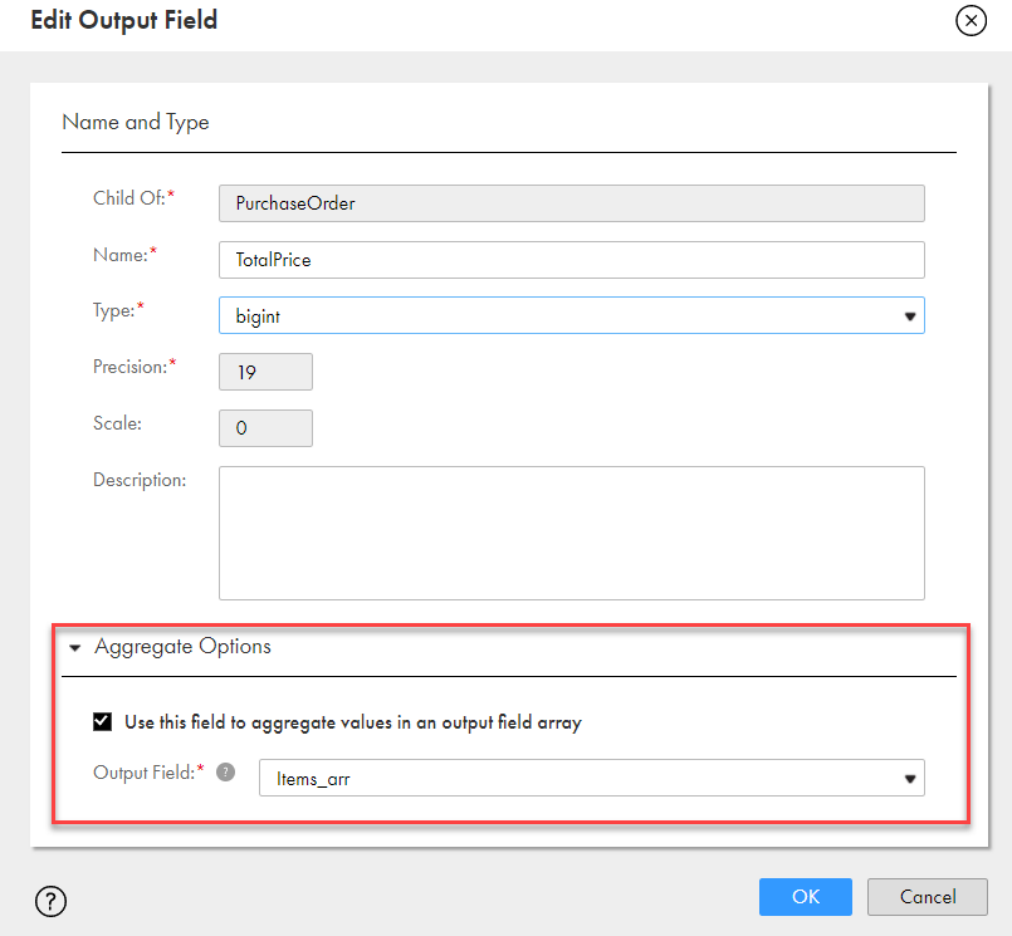
出力データを集計して合計価格を計算します。

次の手順を実行して、特定の発注書のすべての項目を集計します。

1. 次のプロパティを持つ新しい出力フィールドを追加します。

プロパティ	値
子	PurchaseOrder
名前	TotalPrice
タイプ	bigint
集計オプション: このフィールドには、出力フィールド配列の値が集計されます	有効
出力フィールド	Items_arr

以下の図は、［TotalPrice］出力フィールドの集計オプションを示しています。



2. PurchaseOrder.TotalPrice に次のフィールド式を設定して、合計価格を集計します。

```
SUM(:fld.{PurchaseOrder.Items_arr.item_str.Quantity}*:fld.{PurchaseOrder.Items_arr.item_str.Price})
```

手順 5.構造の配列を作成し、データソースを結合します

構造の［SameDayItems］配列を追加して設定します。フィルタ、結合、およびフィールド式を使用して、同じ日に注文および出荷された項目のみを出力します。

以下の手順を実行します。

1. 次のプロパティを持つ新しい出力フィールドを追加します。

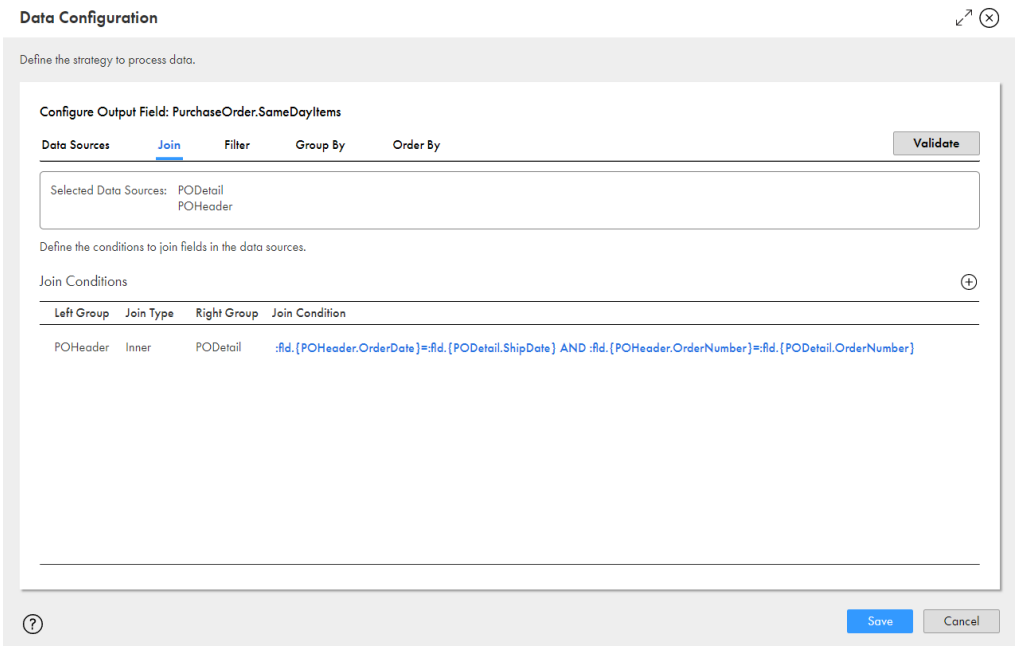
プロパティ	値
子	PurchaseOrder
名前	SameDayItems
タイプ	array

プロパティ	値
配列の要素タイプ	struct
要素の構造名	sameday_str

2. ［PODetail］からのすべての入力フィールドを、出力グループの［SameDayItems］配列に追加します。
3. 出力グループで不要な次のフィールドを削除します: PurchaseOrder.SameDayItems.OrderNumber。
4. ［PurchaseOrder.SameDayItems］配列のデータソースとして［POHeader］を追加します。
5. 次のプロパティを使用して、［PurchaseOrder.SameDayItems］配列の結合条件を追加します。

プロパティ	値
左グループ	POHeader
結合タイプ	内部
右グループ	PODetail
結合条件	:fld.{POHeader.OrderDate}=:fld.{PODetail.ShipDate} AND :fld.{POHeader.OrderNumber}=:fld.{PODetail.OrderNumber}

以下の図は、［SameDayItems］のデータソースと結合条件を示しています。



6. ［PurchaseOrder.SameDayItems］配列にフィルタ条件を追加します。
`:fld.{PODetail.OrderNumber}=:fld.{PurchaseOrder.OrderNumber}`
。

手順 6.マッピングを実行します

最後の手順として、マッピングタスクを作成して実行し、JSON 出力を生成します。

以下の手順を実行します。

1. マッピングタスクを作成します。
2. マッピングタスクを実行します。
3. 出力を確認します。

ヒント: マッピングタスクの詳細については、「*タスク*」ガイドを参照してください。

次の JSON は、マッピングを実行した後の［PurchaseOrder］ターゲット出力を示しています。


```
{
    "OrderNumber": "1",
    "Comment": "AppD for POD4",
    "OrderDate": "2018-10-01 00:00:00.0",
    "ConfirmDate": "2018-10-02 00:00:00.0",
    "address_struct": {
        "Name": "Tom",
        "Street": "2100 Seaport blvd",
        "City": "Redwood City",
        "State": "CA",
        "Country": "USA",
        "Zip": "94063"
    },
    "Items_arr": [{
        "itemNum": "1",
        "ProductName": "AppD Agent for JVM",
        "Quantity": 60,
        "price": 500,
        "comment": "JVM agents",
        "shipDate": "2018-10-15 00:00:00.0",
        "PartNum": "1"
    }, {
        "itemNum": "2",
        "ProductName": "MySQL agents",
        "Quantity": 6,
        "price": 360,
        "comment": "MySQL agents",
        "shipDate": "2018-10-15 00:00:00.0",
        "PartNum": "2"
    }, {
        "itemNum": "3",
        "ProductName": "ELB agents",
        "Quantity": 10,
        "price": 200,
        "comment": "ELB agents",
        "shipDate": "2018-10-16 00:00:00.0",
        "PartNum": "3"
    }],
    "TotalPrice": 34160
} {
    "OrderNumber": "2",
    "Comment": "GoJS for IICS",
    "OrderDate": "2018-10-12 00:00:00.0",
    "ConfirmDate": "2018-10-12 00:00:00.0",
    "address_struct": {
        "Name": "Bill",
        "Street": "23rd Ave",
        "City": "San Mateo",
        "State": "CA",
        "Country": "USA",
        "Zip": "94401"
    },
    "Items_arr": [{
        "itemNum": "1",
        "ProductName": "GOJS OEM Edition",
        "Quantity": 2,
```

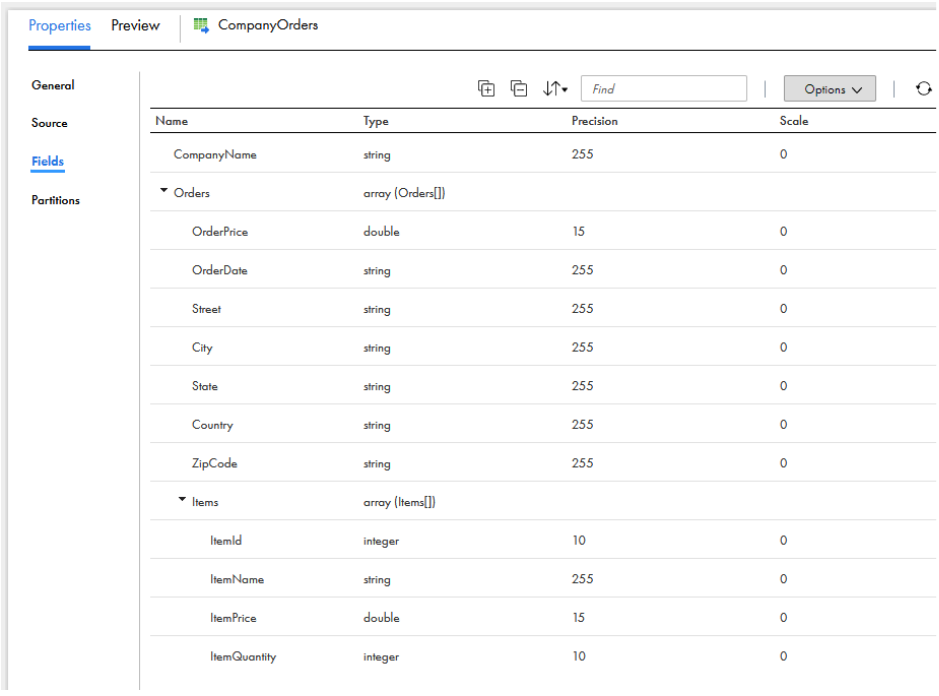
```
        "price": 20000,
        "comment": "GOJS Dev",
        "shipDate": "2018-10-19 00:00:00.0",
        "PartNum": "101"
    }, {
        "itemNum": "2",
        "ProductName": "GOJS Prefessional Service",
        "Quantity": 5,
        "price": 5000,
        "comment": "GOJS Dev",
        "shipDate": "2018-10-19 00:00:00.0",
        "PartNum": "102"
    }],
    "TotalPrice": 65000
}
```


階層型から階層型の例

階層データの既存のファイルを使用して、階層形式の顧客注文ファイルを作成するとします。

既存の顧客注文ファイル CompanyOrders には、注文を行った会社の名前、注文した商品の価格、日付、配送先住所、ID 番号といった各注文に関する情報が含まれています。

次の図に、CompanyOrders ファイルの構造を示します。



配送先住所を構造に再構築し、それぞれの会社のすべての注文の合計価格を計算するフィールドを追加するとします。

次の手順を実行して、ターゲットファイルを作成および設定します。

1. 手順 1.マッピングを設計します。
2. 手順 2.出力グループを設定します。
3. 手順 3.合計価格を集計するための出力フィールドを作成します。
4. 手順 4.注文先住所の出力構造を作成します。
5. 手順 5.マッピングを実行します。

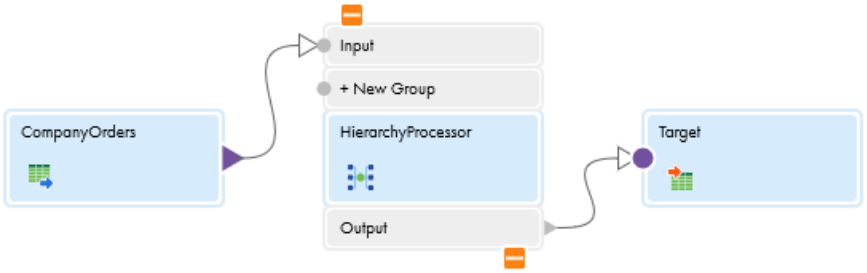
手順 1。マッピングを設計します

最初の手順として、Mapping Designer で、ソースおよびターゲットのトランスフォーメーションを使用して階層プロセッサを設定します。

Mapping Designer で次の手順を実行します。

1. CompanyOrders ファイルをソースオブジェクトとして追加します。
2. 階層プロセッサトランスフォーメーションをマッピングに追加し、CompanyOrders を入力ソースとして接続します。
3. 階層プロセッサトランスフォーメーションで、出力データ形式を階層型に変更します。これにより、出力グループが作成されます。
4. ターゲットトランスフォーメーションをマッピングに追加し、階層プロセッサトランスフォーメーションの出力をこのターゲットオブジェクトに接続します。

次の図は、マッピングがどのようになるかを示しています。

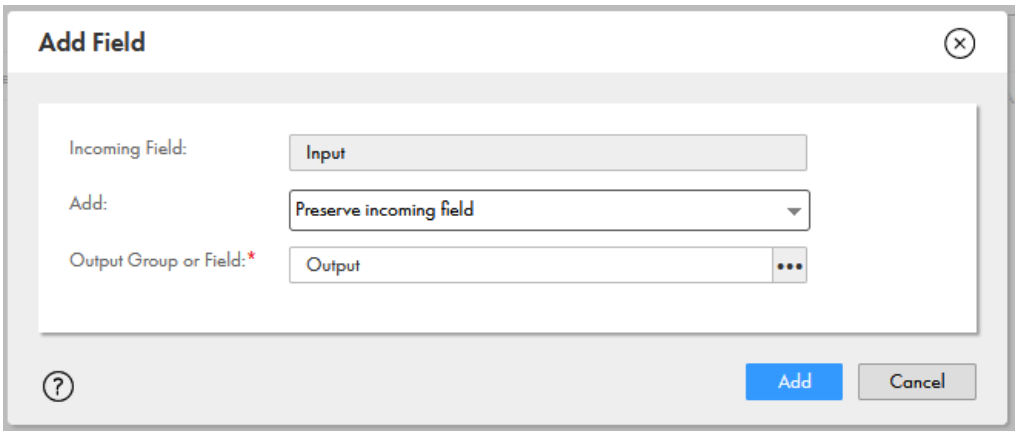


手順 2。出力グループを設定します

基本的なマッピングを設定した後に、すべての受信フィールドを追加して出力グループを作成します。

階層プロセッサトランスフォーメーションで次の手順を実行します。

1. 入力から出力グループにすべての受信フィールドを追加します。*[受信フィールドを保持]* に **[追加]** を設定します。
   次の図に、**[フィールドの追加]** ダイアログを示します。



2. 出力グループのデータソースが *[入力]* に設定されていることを確認します。

手順 3。合計価格を集計するための出力フィールドを作成します

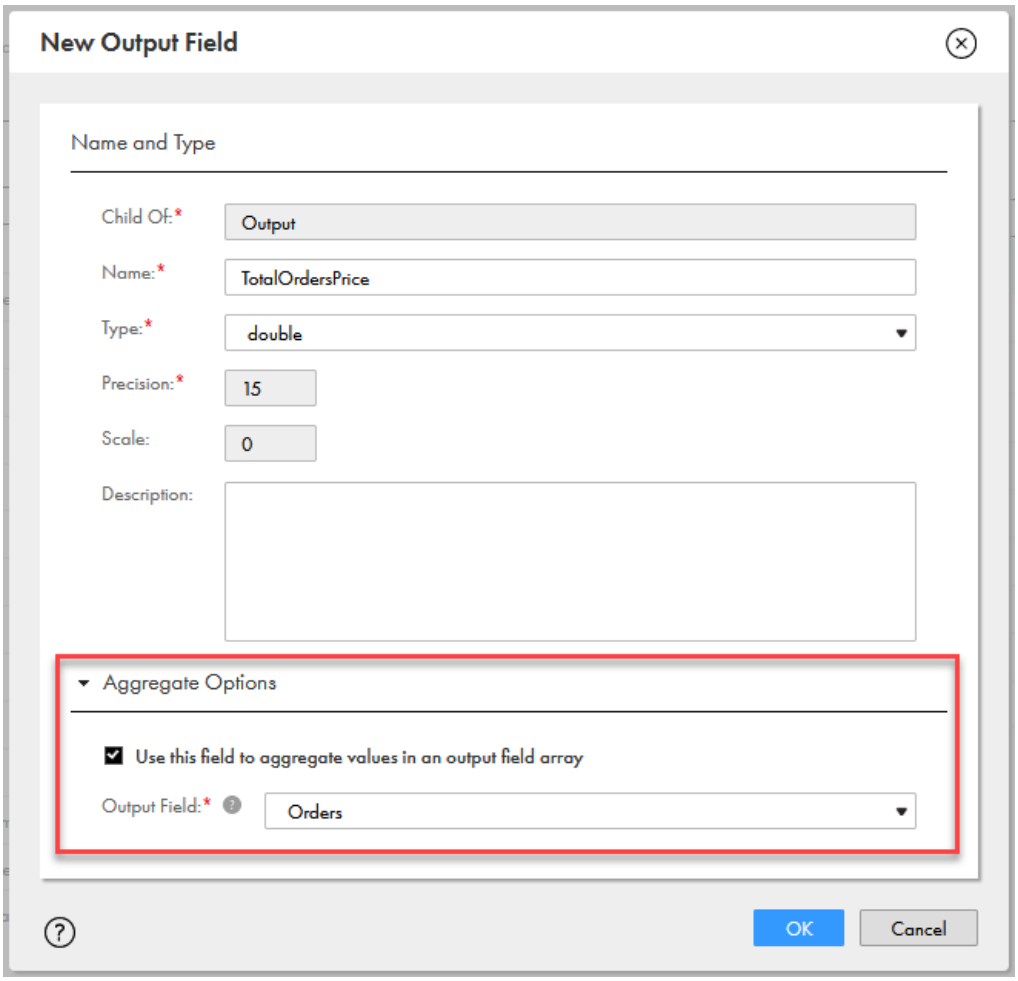
各会社のすべての注文の合計価格を計算する階層プロセッサトランスフォーメーションで出力フィールドを作成します。

以下の手順を実行します。

1. 次のプロパティを持つ新しい出力フィールドを追加します。

プロパティ	値
子	出力
名前	TotalOrdersPrice
タイプ	double
集計オプション	有効
出力フィールド	Orders

以下の図に、[TotalOrdersPrice] 出力フィールドの集計オプションを示します。



2. Output.TotalOrdersPrice に次のフィールド式を設定して、会社のすべての注文の合計価格を集計します。

   ```
   SUM(:fld.{Output.Orders.Orders.OrderPrice})
   ```

手順 4.注文先住所の出力構造を作成します

出力で注文先住所の構造を作成し、不要なフィールドを出力から削除します。

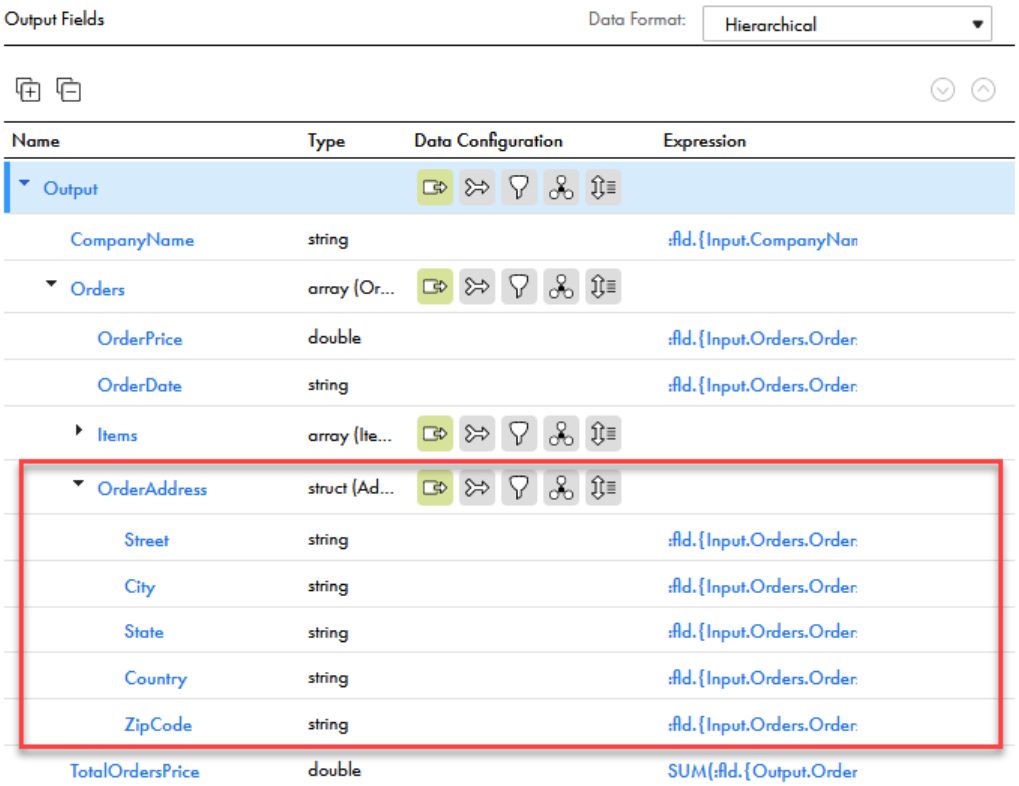
以下の手順を実行します。

1. 次のプロパティを持つ新しい出力フィールドを追加します。

プロパティ	値
子	Output.Orders.Orders
名前	OrderAddress
タイプ	struct
構造名	住所

2. Orders から OrderAddress にすべての受信フィールドを追加します。**[追加]** を *［プリミティブ単一オカレンスの子の追加］* に設定します。
3. OrderAddress のデータソースを *［出力を使用］* に設定します。
4. 不要な次のフィールドを OrderAddress 構造から削除します。
   - Output.Orders.Orders.OrderAddress.OrderPrice
   - Output.Orders.Orders.OrderAddress.OrderDate
5. 不要な次のフィールドを Orders 出力グループから削除します。
   - Output.Orders.Orders.Street
   - Output.Orders.Orders.City
   - Output.Orders.Orders.State
   - Output.Orders.Orders.Country
   - Output.Orders.Orders.ZipCode

次の図に、OrderAddress 構造を示します。



手順 5.マッピングを実行します

最後の手順として、マッピングタスクを作成して実行し、JSON 出力を生成します。

以下の手順を実行します。

1. マッピングタスクを作成します。
2. マッピングタスクを実行します。
3. 出力を確認します。

ヒント: マッピングタスクの詳細については、「*タスク*」ガイドを参照してください。

フラット化された出力の処理

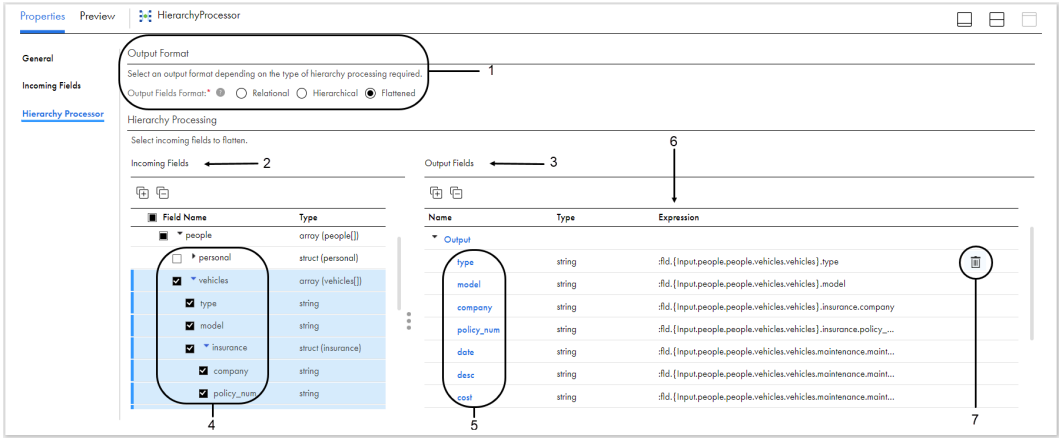
階層プロセッサトランスフォーメーションにより、階層入力グループを、フラット化された非正規化出力グループに変換できます。

例えば、すべての顧客の顧客情報と車両情報を含むショップメンテナンスファイルがあるとします。車両のメンテナンスデータを非正規化し、顧客に関する個人情報を除外するとします。

階層プロセッサトランスフォーメーションを使用したフラット化済みの出力の定義

［階層プロセッサ］タブを使用して、受信フィールドを出力フィールドにマッピングします。フラット化された出力オプションを使用すると、非正規化出力を作成できます。

次の図に、階層入力とフラット化された出力を持つ**［階層プロセッサ］**タブを示します。



1. 出力形式。**［フラット化済み］**を選択して、階層入力を非正規化出力に変換します。
2. 受信フィールド。受信データスキーマとフィールドタイプを表示します。
3. 出力フィールド出力ファイルの作成時に、出力フィールドを表示します。
4. 出力の設定。受信フィールドを選択して、フラット化された出力ファイルスキーマを構築します。
5. 出力フィールド名。フィールド名をクリックして、フィールド名を変更します。
6. 式。フィールド式を表示して、入力から出力へのフィールドマッピングを指定します。
7. 削除。出力フィールドを削除する場合に使用します。

ヒント: 最大化アイコンを使用したり、［受信フィールド］パネルや［出力フィールド］パネルのサイズを変更したりすると、必要な情報を確認できます。

フラット化された出力を使用して階層プロセッサトランスフォーメーションを定義するには、次のタスクを実行します。

1. **［フラット化済み］**出力形式を選択します。
2. 出力に追加する受信フィールドを選択して、トランスフォーメーション出力を設定します。
3. オプション。出力フィールドの名前を変更します。
4. オプション。出力フィールドを削除します。

フラット化された出力グループへの受信フィールドの追加

出力には、受信フィールドを個別に追加することや入力グループ全体を追加することができます。受信フィールドを出力グループに追加すると、必要に応じて、出力フィールドの名前の変更や出力フィールドの削除ができます。

出力フィールドを追加するには、追加する入力フィールドまたは入力グループの横にあるチェックボックスをオンにします。親を選択するとすべての子項目が自動的に選択されますが、不要なエントリを選択解除することもできます。また、出力から不要なエントリを削除することもできます。

階層データのフラット化

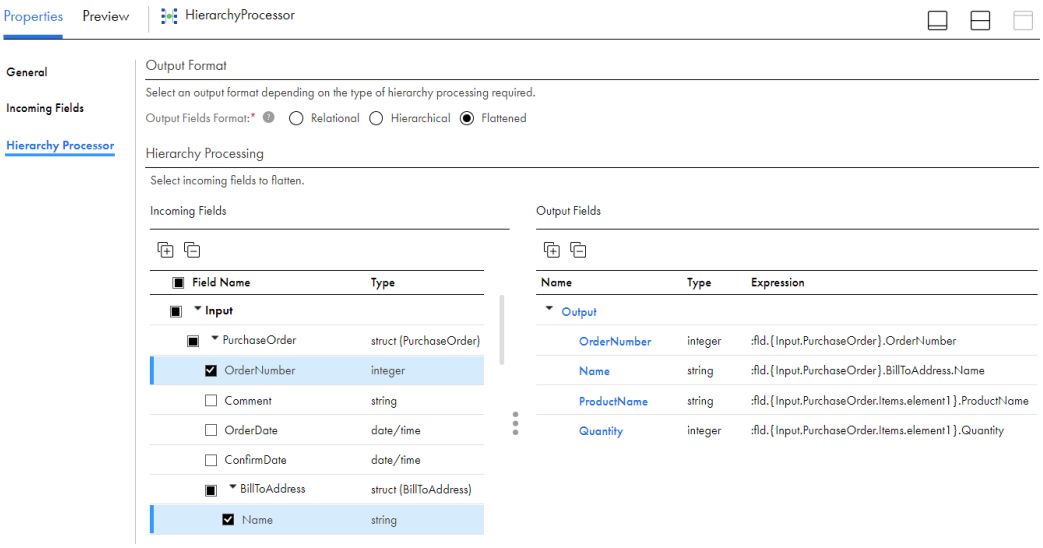
受信階層フィールドを出力に追加すると、出力スキーマのすべてのフィールドが自動的にフラット化されます。

フラット化された階層データの例

受信フィールドが、構造と配列の階層内に含まれています。出力の OrderNumber、Name、ProductName、Quantity の 4 つのフィールドのみが必要です。

受信フィールドを出力に追加するには、適切なチェックボックスをオンにします。

次の図に、入力と出力を示します。



フラット化された出力グループとフィールドの名前変更

フラット化された形式で受信フィールドを出力グループに追加した後に、必要に応じてこれらのフィールドの名前を変更できます。また、出力グループの名前も変更できます。

出力フィールドの名前を変更するには、フィールド名をクリックします。出力グループの名前を変更するには、出力グループ名をクリックします。

以下の表に、出力フィールドのプロパティを示します。

プロパティ	説明
名前	出力フィールドの名前です。
タイプ	現在のフィールドのデータ型。データ型を変更することはできません。
精度	フィールドの合計有効桁数。精度を変更することはできません。
スケール	小数点以下の桁数。スケールを変更することはできません。

式の形式

フラット化された出力を使用すると、式はデフォルトに固定され、変更できなくなります。ただし、デフォルトの式の構文を理解している場合は、各出力フィールドのソースの場所を簡単に特定することができます。

ヒント: 出力グループにフィールドを追加するには、受信フィールド名の横にあるチェックボックスを使用します。これは、リレーショナル出力または階層出力に使用する［追加］リンクとは異なります。

階層プロセッサトランスフォーメーションは、さまざまなデータセットからの情報を処理できます。一部のフィールド名は、異なるデータセット間で一意ではない場合があります。その結果、同じフィールド名が別のデータセットまたは同じデータセットの階層内で使用される可能性があるため、フィールドをその名前で単純に参照することはできません。

階層プロセッサトランスフォーメーションの式の構文は、式トランスフォーメーションで使用される構文とは異なります。

階層プロセッサトランスフォーメーションでフィールドを参照するには、次の構文を使用します。

```
:fld.{input_group_name.field_name}.field_name
```

次の表に、構文の詳細を示します。

構文部分	説明
.fld.	階層プロセッサトランスフォーメーションの式の構文を示します。
input_group_name	入力グループまたはデータセットの名前。
field_name	最上位のフィールドでない場合は、フルパス名を含むフィールドの名前。 配列タイプのフィールドがある場合は、配列名を含めます。配列がプリミティブで配列名がない場合は、配列名として elem を使用します。 構造または配列内のフィールドの場合、実際のフィールド名は右中括弧の外側で指定します。
.field_name	構造または配列内のフィールドを参照する場合にのみ、field_name の部分を含めます。以下のガイドラインに従ってください。 - 構造内のフィールドの場合、field_name 部分は次の形式を使用します: .structName.fieldname - 配列内のフィールドの場合、field_name 部分は次の形式を使用します: .fieldName

JSON データを使用したマッピングの実行

JSON 形式のデータを含む階層プロセッサ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には、マッピングタスクを使用する必要があります。

JSON 入力の読み取り

JSON データを読み取る場合、入力ファイルは、複数行のスキーマに基づくものと単一行のスキーマに基づくものがあります。

次のサンプルは、JSON スキーマを 1 行で示したものです。

```
{"Name":"Tom","Street":"2100 Seaport Blvd","City":"Redwood City","State":"CA","Country":"USA","Zip":"94063"}
```

次のサンプルは、複数の行にまたがる JSON スキーマを示しています。

```
{
    "Name": "Tom",
    "Surname": "Day",
```

```
    "City": "Redwood City",
    "State": "CA",
    "Country": "USA",
    "Zip": "94063"
}
```

デフォルトでは、階層プロセッサトランスフォーメーションは各 JSON スキーマを単一行として読み取ります。複数行にまたがる入力を読み取るには、複数行の JSON ファイルを読み取るようにソーストランスフォーメーションで形式オプションを設定できます。

JSON 出力の書き込み

JSON データを書き込む場合は、各出力レコードを個別のファイルに書き込むか、すべての出力レコードを 1 つのファイルに書き込むことができます。

デフォルトでは、各出力レコードは個別のファイルに書き込まれます。出力レコードを 1 つの JSON 形式のファイルに書き込むには、マッピングタスクで次の Spark セッションプロパティを設定します。

セッションプロパティ名	セッションプロパティ値
spark.sql.shuffle.partitions	1

階層からフラット化済みの例

階層データをリレーショナルデータに変換し、そのデータをターゲットファイルに非正規化形式で書き込むとします。

ショップのメンテナンスファイルに、顧客と顧客の車両情報が含まれています。このファイルは階層 JSON ファイルであり、会社のショップアプリケーションによって生成されています。

次の JSON スクリプトは、マッピングを実行する前のショップメンテナンスソース入力を示しています。

```
{
    "people": [{
        "personal": {
            "age": 20,
            "gender": "M",
            "name": {
                "first": "John",
                "last": "Doe"
            }
        },
        "vehicles": [{
            "type": "car",
            "model": "Honda Civic",
            "insurance": {
                "policy_num": "HA12345"
            },
            "maintenance": [{
                "desc": "oil change",
                "cost": "111.50",
                "summary": [{
                    "line1": "0w20",
                    "line2": "synthetic"
                }, {
                    "line1": "2.0L 4-cyl",
                    "line2": "4.4 quarts"
                }]
            }, {
                "desc": "new tires",
                "cost": "425.00",
                "summary": [{
                    "line1": "235/40R18",
                    "line2": "4 tires"
```

```
            }, {
                "line1": "All Season",
                "line2": "No spare"
            }]
        }]
    }, {
        "type": "truck",
        "model": "Dodge Ram",
        "insurance": {
            "policy_num": "DR12345"
        },
        "maintenance": [{
            "desc": "new tires",
            "cost": "299.99",
            "summary": [{
                "line1": "275/60R20",
                "line2": "2 tires"
            }, {
                "line1": "All Season",
                "line2": "No spare"
            }]
        }, {
            "desc": "oil change",
            "cost": "111.50",
            "summary": [{
                "line1": "5w30",
                "line2": "conventional"
            }, {
                "line1": "5.7L V8",
                "line2": "7.0 quarts"
            }]
        }]
    }],
    "source": "internet"
}, {
    "personal": {
        "age": 24,
        "gender": "F",
        "name": {
            "first": "Jane",
            "last": "Roberts"
        }
    },
    "vehicles": [{
        "type": "car",
        "model": "Toyota Camry",
        "insurance": {
            "policy_num": "TC98765"
        },
        "maintenance": [{
            "desc": "tires rotated",
            "cost": "389.50",
            "summary": [{
                "line1": "4 tires",
                "line2": "leak repairs"
            }]
        }, {
            "desc": "oil change",
            "cost": "59.50",
            "summary": [{
                "line1": "0w20",
                "line2": "special"
            }]
        }]
    }, {
        "type": "car",
        "model": "Honda Accord",
        "insurance": {
            "policy_num": "HA98765"
        },
```

```
            "maintenance": [{
                "desc": "new air filter",
                "cost": "399.50",
                "summary": [{
                    "line1": "17220-6B2-A00",
                    "line2": "rebuild assembly"
                }]
            }, {
                "desc": "new brakes",
                "cost": "799.50",
                "summary": [{
                    "line1": "2-443344586",
                    "line2": "rear brake kit"
                }]
            }]
        }],
        "source": "phone"
    }]
}
```

車両のメンテナンスデータを非正規化し、顧客の個人情報を除外するとします。

次の手順を実行して、ターゲットファイルを作成および設定します。

1. 手順 1.マッピングを設計します。
2. 手順 2.出力グループを設定します。
3. 手順 3.マッピングを実行します。

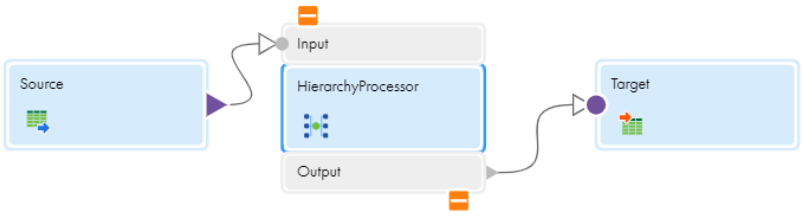
手順 1。マッピングを設計します

最初の手順として、Mapping Designer で、ソースおよびターゲットのトランスフォーメーションを使用して階層プロセッサを設定します。

Mapping Designer で次の手順を実行します。

1. ショップのメンテナンスファイルをソースオブジェクトとして、ソーストランスフォーメーションをマッピングに追加します。
2. 階層プロセッサトランスフォーメーションをマッピングに追加し、ショップメンテナンスを入力ソースとして接続します。
3. 階層プロセッサトランスフォーメーションで、出力形式として**［フラット化済み］**を選択します。
4. ターゲットトランスフォーメーションをマッピングに追加し、階層プロセッサトランスフォーメーションの出力をこのターゲットオブジェクトに接続します。

マッピングは次の図のようになります。

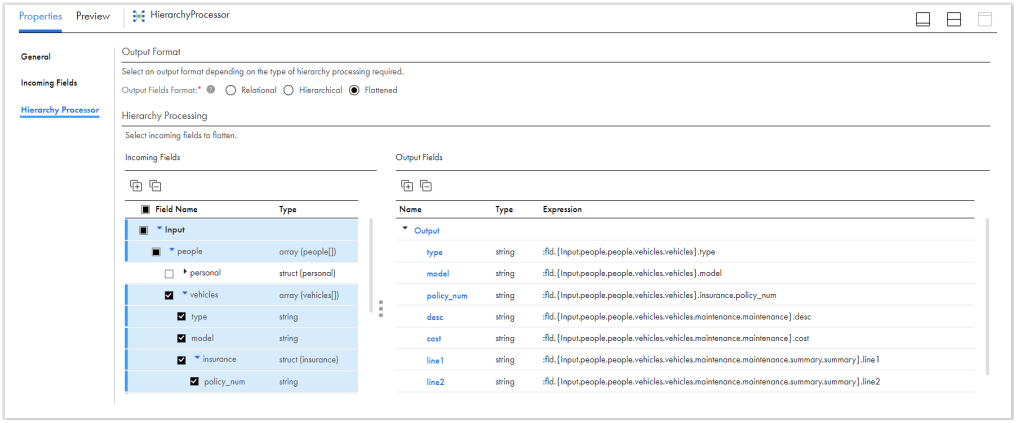


手順 2。出力グループを設定します

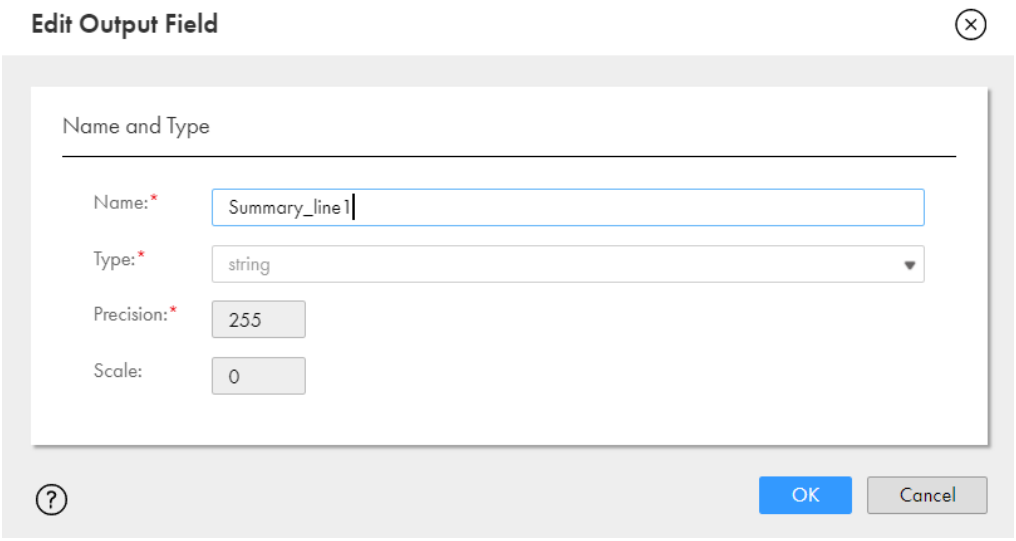
基本的なマッピングを設定した後に、車両ショップのメンテナンスデータを使用して出力グループを作成します。

階層プロセッサトランスフォーメーションで次の手順を実行します。

1. 最上位の入力グループを選択します。これにより、すべての入力フィールドが自動的に選択され、出力に追加されます。
2. personal（struct）をクリアします。これにより、構造内のすべての要素が自動的にクリアされ、出力から削除されます。
   受信フィールドと出力フィールドは、次の図のようになります。



3. summary line1 の出力フィールドをクリックし、名前を「Summary_line1」に変更します。
4. summary line2 についても、同様に名前の変更を行います。
   次の図に、**［出力フィールドの編集］**ダイアログボックスでフィールド名を編集する方法を示します。



5. マッピングを確認します。

手順 3。マッピングを実行します

最後の手順として、マッピングタスクを作成して実行し、JSON 出力を生成します。

以下の手順を実行します。

1. マッピングタスクを作成します。
2. マッピングタスクを実行します。
3. 出力を確認します。

次の表に、マッピングを実行した後の、部分的に非正規化されたターゲット出力を示します。

type	model	policy_num	desc	cost	Summary_line1	Summary_line2	source
car	Honda Civic	HA12345	oil change	111.5	0w20	synthetic	internet
car	Honda Civic	HA12345	oil change	111.5	2.0L 4-cyl	4.4 quarts	internet
car	Honda Civic	HA12345	new tires	425	235/40R18	4 tires	internet
car	Honda Civic	HA12345	new tires	425	All Season	No spare	internet
truck	Dodge Ram	DR12345	new tires	299.99	275/60R20	2 tires	internet
truck	Dodge Ram	DR12345	new tires	299.99	All Season	No spare	internet
truck	Dodge Ram	DR12345	oil change	111.5	5w30	conventional	internet
truck	Dodge Ram	DR12345	oil change	111.5	5.7L V8	7.0 quarts	internet
car	Toyota Camry	TC98765	tires rotated	389.5	4 tires	leak repairs	phone
car	Toyota Camry	TC98765	oil change	59.5	0w20	special	phone
car	Honda Accord	HA98765	new air filter	399.5	17220-6B2-A00	rebuild assembly	phone
car	Honda Accord	HA98765	new brakes	799.5	2-443344586	rear brake kit	phone

第 14 章

入力トランスフォーメーション

入力トランスフォーメーションは、マップレットに渡すデータを設定するために使用するパッシブトランスフォーメーションです。マッピングまたはマップレットに含まれるアップストリームトランスフォーメーションからマップレットがデータを受信するようにするときに、入力トランスフォーメーションを使用します。

入力フィールドを追加して、アップストリームトランスフォーメーションからマップレットに渡すデータフィールドを定義します。

マップレットに複数の入力トランスフォーメーションを追加することができます。マップレットの各入力トランスフォーメーションは、マップレットトランスフォーメーションでマップレットを使用する場合の入力グループになります。少なくとも 1 つの入力グループを、アップストリームトランスフォーメーションに接続する必要があります。

マップレットで入力トランスフォーメーションを使用できますが、マッピングでは使用できません。

入力フィールド

入力トランスフォーメーションに入力フィールドを追加して、マップレットに渡すフィールドを定義します。各入力トランスフォーメーションに、少なくとも 1 つの入力フィールドを追加する必要があります。

[入力フィールド] タブの入力フィールドを追加します。フィールドを追加するには、**[フィールドの追加]** をクリックして、フィールド名、データ型、精度、スケール、および説明（オプション）を入力します。説明には最大 4000 文字を含めることができます。

マップレットトランスフォーメーションでマップレットを使用するときは、アップストリームトランスフォーメーションに少なくとも 1 つの入力フィールドをマッピングします。

第 15 章

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使用して、データ統合機能を拡張します。Java トランスフォーメーションは、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、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素早く定義することができます。Java トランスフォーメーションは、アクティブまたはパッシブなトランスフォーメーションにすることができます。

Secure Agent では、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する必要があります。Azul OpenJDK が Secure Agent とともにインストールされるので、JDK を別途インストールする必要はありません。Azul OpenJDK には、Java Runtime Environment (JRE)が含まれています。

Secure Agent は 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。Java トランスフォーメーションが含まれるマッピングまたはマッピングタスクを実行すると、Secure Agent は JRE を使用してバイトコードを実行して入力行を処理し、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の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ります。
- すべての入力行はトランスフォーメーションによって処理されている。
- トランスフォーメーションがトランザクション通知を受け取ったとき。

Java トランスフォーメーションの式は詳細モードでのみ呼び出すことができます。

Graviton 対応クラスタでは Java トランスフォーメーションは使用できません。Graviton 対応クラスタの詳細については、「Administrator」のヘルプを参照してください。

注: Java トランスフォーメーションを作成する場合は、マッピングタスクでコードを実行する前に Java コードを確認し、クエリ、リモートスクリプト、データ接続などの潜在的な危険性を持つアクティブコンテンツが含まれていないことを確認してください。

Java トランスフォーメーションの定義

Java トランスフォーメーションを定義するには、トランスフォーメーションフィールドおよびプロパティを設定し、**[Java]** タブで Java コードスニペットを入力して、コードをコンパイルします。

注: サードパーティまたはカスタムの Java パッケージを使用する場合、Java トランスフォーメーションを作成する前に、コードのコンパイル時やトランスフォーメーションを含むマッピングの実行時に使用するクラスパスを設定します。

1. Mapping Designer で、トランスフォーメーションパレットからキャンバスに Java トランスフォーメーションをドラッグし、それをアップストリームおよび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。
2. トランスフォーメーションの受信フィールドルールおよび出力フィールドを設定します。
3. トランスフォーメーションのプロパティを設定します。
4. トランスフォーメーション機能を定義する Java コードスニペットを書き込みます。
5. コードをコンパイルします。
6. コンパイルエラーを特定して修正します。

クラスパス設定

Java トランスフォーメーションでサードパーティまたはカスタムの Java パッケージを使用する場合は、クラスパスと少なくとも 1 つのクラス値を設定します。Secure Agent では、Java コードをコンパイルするときのクラスパス内にクラスおよびリソースファイルが含まれています。

例えば、Java トランスフォーメーションに Java パッケージである converter をインポートし、このパッケージを converter.jar で定義したとします。この場合、Java コードをコンパイルする前に、必ず converter.jar をクラスパスに追加する必要があります。ただし、java.io はビルトインの Java パッケージであるため、java.io パッケージをインポートする場合はクラスパスを設定する必要はありません。ビルトインの Java パッケージのみ使用する場合は、クラスパスを設定する必要はありません。

Secure Agent が JAR ファイルまたはクラスファイルにアクセスできるようにするには、Java パッケージに関連付けられた各 JAR ファイルまたはクラスファイルディレクトリへのクラスパスを設定します。複数のクラスパスエントリは、Windows の場合はセミコロン、UNIX の場合はコロンで区切ります。

詳細クラスタ上の JAR ファイルまたはクラスファイル

詳細モードの Java トランスフォーメーションでサードパーティまたはカスタムの Java パッケージを使用する場合は、JAR ファイルまたはクラスファイルを Secure Agent インストールディレクトリに保存します。

JAR ファイルまたはクラスファイルを Secure Agent マシンの次のディレクトリに格納します。

```
<Secure Agent installation directory>/ext/ctjars
```

詳細モードの Java トランスフォーメーションでネイティブライブラリを使用する場合は、ctjars ディレクトリと同じ場所に、.so ファイルと実行可能ファイル用にそれぞれ次の native-lib ディレクトリと native-bin ディレクトリを作成します。

```
<Secure Agent installation directory>/ext/native-lib
<Secure Agent installation directory>/ext/native-bin
```

Secure Agent マシンで JAR ファイルまたはクラスファイルを更新した場合は、次に詳細モードでジョブを実行したときにファイルが有効になります。Secure Agent マシンが予期せず停止してエージェントが別のマシンで再起動した場合は、新しいマシンの同じディレクトリに JAR ファイルまたはクラスファイルを追加します。

注: 本番環境では、ワークロードの実行中に ext フォルダの下のコンテンツ（ctjars、connectors、python、native-lib、native-bin）を頻繁に更新しないでください。長時間実行されているジョブに悪影響を及ぼし、障害を引き起こす可能性があります。必要に応じて、k8s.infarpm.max.installers=x フラグをデフォルトの 5 よりも大きい値に設定できますが、これを行うと、ワーカーノードでディスクプレッシャの問題が発生する可能性があります。

サーバーレスランタイム環境を使用する場合、ファイルは補足ファイルの場所に格納します。補足ファイルの場所の詳細については、Administrator ヘルプを参照してください。

クラスパス値

Java トランスフォーメーションでサードパーティまたはカスタムの Java パッケージを使用する場合は、少なくとも 1 つのクラスパス値を設定します。

次のクラスパス値を設定できます。

Secure Agent **の** JVMClassPath **プロパティ**

Secure Agent は、Java トランスフォーメーションを設計および検証する際にこのクラスパスを使用し、Mapping Designer からマッピングを実行するか、またはマッピングタスクを実行します。このクラスパスは、エージェント上で実行されるすべてのマッピングおよびマッピングタスクに適用されます。

管理者で Secure Agent の JVMClassPath プロパティを設定します。

このプロパティを設定するか、Secure Agent マシンで CLASSPATH 環境変数を設定します。クラスパス値を両方とも設定する必要はありません。

CLASSPATH **環境変数**

Secure Agent は、Java トランスフォーメーションを設計および検証する際にこのクラスパスを使用し、Mapping Designer からマッピングを実行するか、またはマッピングタスクを実行します。このクラスパスは、エージェント上で実行されるすべてのマッピングおよびマッピングタスクに適用されます。

Secure Agent マシン上で、CLASSPATH 環境変数を設定します。

CLASSPATH 環境変数を設定するか、Secure Agent の JVMClassPath プロパティを設定します。クラスパス値を両方とも設定する必要はありません。

設計時クラスパス

Secure Agent は、Java トランスフォーメーションを設計および検証するとき、および Mapping Designer からマッピングを実行するときにこのクラスパスを使用します。このクラスパスは、マッピングタスクによるマッピングを実行するときには使用されません。

トランスフォーメーションをテストするときに、JVMClassPath プロパティにも CLASSPATH 環境変数にも必要なパッケージが含まれない場合は、設計時クラスパスを設定します。JVMClassPath プロパティまたは CLASSPATH 環境変数に必要なパッケージが含まれるように設定した場合は、設計時クラスパスを設定する必要はありません。

Java トランスフォーメーションの詳細プロパティで設計時クラスパスを設定できます。

Java **クラスパスセッションプロパティ**

Secure Agent は、マッピングタスクの実行時にこのクラスパスを使用します。このクラスパスは、プロパティが設定されているマッピングタスクにのみ適用されます。

1 つのマッピングタスクにクラスパスを設定するが、その他のタスクには設定しない場合は、Java クラスパスセッションプロパティを設定します。JVMClassPath プロパティまたは CLASSPATH 環境変数に必要

なパッケージが含まれるように設定した場合は、Java クラスパスセッションプロパティを設定する必要はありません。

マッピングタスクの詳細セッションプロパティで Java クラスパスセッションプロパティを設定します。

複数のクラスパス値を設定すると、Secure Agent は適用されるすべてのクラスパスを使用します。例えば、Secure Agent で JVMClassPath プロパティ、CLASSPATH 環境変数、および Java トランスフォーメーションで設計時クラスパスを設定するとします。Java トランスフォーメーションで Java コードをコンパイルするか、Mapping Designer によるマッピングを実行すると、Secure Agent は 3 つすべてのクラスパスを使用します。マッピングタスクによるマッピングを実行すると、Secure Agent は JVMClassPath および CLASSPATH 環境変数のみ使用します。

警告: 複数のクラスパスを設定する場合は、ランタイムエラーを発生する可能性があるためクラスまたはリソースのコピーが複数作成されないようにしてください。

Secure Agent の JVM クラスパスの設定

管理者の［Secure Agent］ページで、Secure Agent の DTM JVMClassPath プロパティを設定します。Secure Agent は、Java トランスフォーメーションを設計および検証する際にこのクラスパスを使用し、Mapping Designer からマッピングを実行するか、またはマッピングタスクを実行します。このクラスパスを設定後は、Secure Agent を再起動する必要があります。

1. 管理者で、**［ランタイム環境］**を選択します。
2. Secure Agent を選択します。
3. ［Secure Agent］ページで、**［編集］**をクリックします。
4. ［システム構成の詳細］で、以下の値を選択します。

オプション	値
サービス	データ統合サーバー
タイプ	DTM

5. JVMClassPath プロパティの行で、**［編集］**をクリックします。
6. 既存のクラスパス値にクラスパスを付加します。

   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。UNIX の場合、コロン（:）を使用します。

   注: デフォルト値の「pmserversdk.jar」は削除しないでください。
7. **［保存］**をクリックします。

システム構成の詳細を更新したら、Secure Agent を再起動します。

CLASSPATH 環境変数の設定

Secure Agent マシン上で、CLASSPATH 環境変数を設定します。Secure Agent は、Java トランスフォーメーションを設計および検証する際にこのクラスパスを使用し、Mapping Designer からマッピングを実行するか、またはマッピングタスクを実行します。このクラスパスを設定後は、Secure Agent を再起動する必要があります。

Windows および UNIX で異なる CLASSPATH 環境変数を設定します。

Windows での CLASSPATH の設定

Windows では、［コントロールパネル］の［システムの詳細な設定］から CLASSPATH 環境変数を設定します。

1. Windows の［コントロールパネル］から［システムの詳細な設定］を開きます。
2. **［環境変数］**をクリックします。
3. ［システム変数］で、**［新規］**をクリックします。
4. CLASSPATH に変数名、クラスパスに変数値を設定します。
5. **［OK］**をクリックします。

環境変数を設定したら、Secure Agent を再起動します。

UNIX での CLASSPATH の設定

UNIX では、UNIX シェルから CLASSPATH 環境変数を設定します。

▶ 次のいずれかのコマンドを入力します。

- UNIX の C シェル環境では、次のように入力します。

  ```
  setenv CLASSPATH <classpath>
  ```

- UNIX の Bourne シェル環境では、次のように入力します。

  ```
  CLASSPATH = <classpath>
  export CLASSPATH
  ```

環境変数を設定したら、Secure Agent を再起動します。

設計時クラスパスの設定

Java トランスフォーメーションの詳細プロパティで設計時クラスパスを設定します。Secure Agent は、Java トランスフォーメーションを設計および検証するとき、および Mapping Designer からマッピングを実行するときに設計時クラスパスを使用します。

1. Java トランスフォーメーションが含まれるマッピングを開きます。
2. Java トランスフォーメーションを選択し、**［詳細プロパティ］**タブをクリックします。
3. **［設計時クラスパス］**フィールドにクラスパスを入力します。

Java クラスパスセッションプロパティの設定

マッピングタスクの詳細セッションプロパティで Java クラスパスセッションプロパティを設定します。Secure Agent は、マッピングタスクの実行時にこのクラスパスを使用します。

1. マッピングタスクを開きます。
2. **［スケジュール］**ページで、詳細セッションプロパティまで下にスクロールし、**［追加］**をクリックします。
3. **［全般オプション設定］**で**［Java クラスパス］**を選択します。
4. **［セッションプロパティ値の設定］**フィールドにクラスパスを入力します。
5. **［保存］**をクリックします。

Java トランスフォーメーションのフィールド

Java トランスフォーメーションには受信フィールドと出力フィールドがあります。受信フィールドと出力フィールドは、**［Java］**タブで定義している Java コードスニペットの変数として使用します。

受信フィールドは**［受信フィールド］**タブに表示されます。デフォルトでは、Java トランスフォーメーションはアップストリームトランスフォーメーションからすべての受信フィールドを継承します。すべての受信フィールドを使用する必要がない場合は、フィールドルールを定義して、特定のフィールドを含めたり除外したりすることができます。フィールドルールの詳細については、「フィールドルール」 (ページ 25)を参照してください。

［出力フィールド］タブで出力フィールドを追加します。ダウンストリームトランスフォーメーションに渡す出力データの出力フィールドを追加します。フィールドを追加するには、**［フィールドの追加］**をクリックして、フィールド名、データ型、精度、スケール、および説明（オプション）を入力します。説明には最大 4000 文字を含めることができます。Java エディタの**［出力］**タブで**［新しいフィールドの作成］**をクリックして、出力フィールドを作成することもできます。

Java トランスフォーメーションは、以下のフィールドデータ型に基づいて出力フィールドの値を初期化します。

- プリミティブデータ型トランスフォーメーションは、フィールド変数の値を 0 に初期化します。
- 複合データ型トランスフォーメーションはフィールド変数を NULL に初期化します。

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フィールドタイプに基づいて、データ統合トランスフォーメーションのデータ型を 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、入力フィールドデータ型を 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 データ型を出力フィールド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 integer データ型の入力フィールドに対しては、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、入力フィールドの integer 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フィールドの値を 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 integer 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データ統合トランスフォーメーションのデータ型を 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。

トランスフォーメーションデータ型	Java **データ型**
bigint	long
binary	byte[]
date/time	BigDecimal long（1970/01/01 00:00:00.000 GMT 以降のミリ秒数）

トランスフォーメーションデータ型	Java データ型
decimal	double BigDecimal
double	double
integer	整数型
string	ストリング
text	String

入力 Java、String、byte[]、および BigDecimal データ型は複合データ型であり、double、int および long データ型はプリミティブデータ型です。

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。Java エディタの［入力行に達したとき］セクションでは、isNull API メソッドおよび setNull API メソッドを使用して、入力フィールドの NULL 値を出力フィールドの NULL 値に設定できます。例については、「setNull」 (ページ 264)を参照してください。

注: 高精度が有効な場合は、10 進データ型は BigDecimal にマッピングされます。BigDecimal は、+演算子などの一部の演算子と一緒には使用できません。Java コードに 10 進データ型フィールドを使用する式が含まれており、このフィールドでこの演算子のいずれかが使用されていると、Java コードのコンパイルは失敗します。

ソート条件

詳細モードでは、ソート条件を設定してソートフィールドおよびソート順を指定できます。マッピングタスクは、Java コードの実行前にソート条件を使用してデータをソートします。

ソートフィールドとは、ソート基準として使用する 1 つまたは複数のフィールドのことです。ソート順を設定して、データを昇順または降順にソートします。

パーティションにグループ化されているデータのソート条件を設定する場合、マッピングタスクは各パーティションでデータをソートします。

複数のソート条件を指定する場合、マッピングタスクは各条件をシーケンシャルにソートします。マッピングタスクは、連続した各ソート条件を前のソート条件の 2 番目のソートとして扱います。ソート条件の順序は設定が可能です。

ソート条件のパラメータを使用する場合、マッピングの実行時またはマッピングタスクの設定時にソートフィールドとソート順を定義します。

グループ化フィールド

詳細モードでは、Group By フィールドを使用して、Java コードの実行前にデータをパーティションにグループ化する方法を定義できます。

Group By フィールドを設定すると、マッピングタスクにより、同じデータが含まれる行がパーティションにグループ化されます。その後、トランスフォーメーションの各パーティションについて Java コードが実行されます。例えば、入力行の動作は各パーティションおよびパーティション内の各行について処理され、データの終わりに達したときの動作はパーティション内のすべての行を処理した後に各パーティションについて処理されます。

複数の Group By フィールドを選択すると、タスクは Group By フィールドのデータの一意の組み合わせごとにパーティションを作成します。

Group By フィールドを設定しない場合、Java コードはデータのデフォルトのパーティション化スキームに基づいて実行されます。

Group By フィールドのパラメータを使用する場合、マッピングの実行時またはマッピングタスクの設定時に Group By フィールドを定義します。

Java トランスフォーメーションプロパティの設定

Java 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コードおよびトランスフォーメーションのプロパティが両方とも含まれています。**[詳細]** タブで、Java トランスフォーメーションプロパティを設定します。

以下の表では、Java トランスフォーメーションのプロパティについて説明します。

プロパティ	説明
動作	トランスフォーメーション動作。アクティブまたはパッシブ。アクティブの場合、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。パッシブの場合、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 1 つの出力行を生成できます。 デフォルトはアクティブです。
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
トランスフォーメーション範囲	Secure Agent が受信データに対してトランスフォーメーションロジックを適用する方法です。次のオプションを使用します。 - 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。トランスフォーメーション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。トランスフォーメーション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。たとえば、Java コードが単一のトランザクションのデータに対して集計計算を実行する場合に、[Transaction] を選択します。 - 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、Secure Agent はトランザクション境界を削除します。[すべての入力] は、トランスフォーメーションの結果がソース内のデータのすべての行に依存する場合に選択します。たとえば、トランスフォーメーションの Java コードがすべての入力データをソートする場合、[すべての入力] を選択します。 アクティブトランスフォーメーションの場合、デフォルトは [すべての入力] です。パッシブトランスフォーメーションの場合、このプロパティは常に [行] に設定されます。
アップデートストラテジの定義	トランスフォーメーションが出力行に対するアップデートストラテジを定義するか指定します。有効になっている場合、Java コードが出力行のアップデートストラテジを決定します。無効になっている場合、アップデートストラテジはターゲットトランスフォーメーションで設定された操作によって決定されます。 このプロパティは、アクティブな Java トランスフォーメーションに対して設定できます。 デフォルトでは無効になっています。

プロパティ	説明
高精度 10 進演算を有効にする	Java クラスが BigDecimal の 10 進数フィールドを高精度で処理できるようにします。10 進数データ型を 16 以上 28 未満の精度で処理するにはこのオプションを有効にします。 デフォルトでは無効になっています。 詳細モードでは、Java トランスフォーメーションは常に高精度を使用します。
日付/時刻にナノ秒を使用する	生成された Java コードがトランスフォーメーションの date/time データ型を、ナノ秒の精度を持つ Java BigDecimal データ型に変換するかどうかを指定します。 有効にすると、生成された Java コードはトランスフォーメーションの date/time データ型を Java BigDecimal データ型に変換します。無効にすると、コードは date/time データ型をミリ秒の精度を持つ Java long データ型に変換します。 デフォルトでは無効になっています。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。 デフォルトでは有効になっています。
設計時クラスパス	トランスフォーメーションを設計および検証するとき、および Mapping Designer からマッピングを実行するときに、Secure Agent がカスタムまたはサードパーティのパッケージ用に使用するクラスパスです。 このクラスパスは、マッピングタスクによるマッピングを実行するときには使用されません。 トランスフォーメーションをテストするときに、Secure Agent の JVMClassPath プロパティ、Secure Agent マシンの CLASSPATH 環境変数のどちらにも必要なパッケージが含まれない場合は、設計時クラスパスを設定します。JVMClassPath プロパティまたは CLASSPATH 環境変数に必要なパッケージが含まれるように設定した場合は、このプロパティを設定する必要はありません。

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、動作をアクティブまたはパッシブとして定義します。**[詳細]** タブの動作を定義します。

Java トランスフォーメーションは、入力データの各行に対して 1 回ずつ、Java エディタの［入力行に達したとき］セクションで定義した 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、動作に基づいて出力行を処理します。

- 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、あるトランスフォーメーションに開始日と終了日を表す 2 つの入力フィールドが含まれているとします。generateRow メソッドを使用すると、開始日から終了日までの各日に対して出力行を生成できます。
- 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あと、各入力行に対して 1 つの出力行を生成します。

トランスフォーメーションの動作は変更することができます。ただし、動作を変更した場合は Java コードを再コンパイルする必要があります。

アップデートストラテジの定義

アクティブ Java トランスフォーメーションのアップデートストラテジを定義できます。アップデートストラテジを定義するには、**［詳細］**タブの**［アップデートストラテジの定義］**オプションを使用します。

アップデートストラテジは、以下のレベルで定義できます。

Java **コード内**

出力行のアップデートストラテジを設定する Java コードを記述できます。Java コードは、挿入、更新、削除、または拒否のフラグを行に設定できます。Java コード内でアップデートストラテジを定義するには、**［アップデートストラテジの定義］**オプションを有効にして、Java エディタの［入力行に達したとき］セクションで setOutRowType メソッドを使用して行にフラグを設定します。アップデートストラテジの設定の詳細については、「setOutRowType」 (ページ 265)を参照してください。

ターゲットトランスフォーメーション内

ターゲットトランスフォーメーションを設定して、アップデートストラテジを設定できます。ターゲットトランスフォーメーションを設定して、アップデートストラテジを設定するには、**［アップデートストラテジの定義］**オプションを無効にして、ターゲットプロパティでターゲット操作を設定します。

このオプションは、パッシブトランスフォーメーションには適用されません。

高精度 10 進演算の使用

Java トランスフォーメーションで、Decimal データ型の高精度 10 進演算を処理する方法を設定できます。高精度 10 進演算を有効にするには、**［詳細］**タブで**［高精度 10 進演算を有効にする］**オプションを使用します。

デフォルトでは、Java トランスフォーメーションは Decimal タイプのフィールドを Double データ型（精度 15）に変換します。精度が 15 を超える Decimal データ型を処理する場合は、高精度 10 進演算を有効化し、Decimal フィールドを Java の BigDecimal クラスを使用して処理します。

高精度 10 進演算を有効化すると、Decimal フィールドを BigDecimal として 28 までの精度で処理できます。Java トランスフォーメーションは、精度が 28 より大きい Decimal データを Double データ型に変換します。

例えば、Decimal タイプの入力フィールドを持つ Java トランスフォーメーションがあるとします。このトランスフォーメーションは、40012030304957666903 の値を受け取ります。高精度 10 進演算を有効化している場合、このフィールドの値は表示されたとおりに扱われます。高精度 10 進演算が有効化されていない場合、このフィールドの値は 4.00120303049577 x 10^19 になります。

詳細モードでの高精度

詳細モードのマッピングに基づくマッピングタスクを正常に実行するには、Java トランスフォーメーションとマッピングで同じ精度モードを使用する必要があります。Java トランスフォーメーションでは常に高精度が使用されますが、マッピングではマッピング内の階層フィールドのタイプに基づいて高精度または低精度が使用されます。

マッピング内の階層フィールドに 10 進数データ型の子フィールドが含まれている場合、マッピングは低精度を使用して実行されます。同じマッピングに 10 進数フィールドを持つ Java トランスフォーメーションが含まれている場合、マッピングタスクは失敗します。

サブ秒の処理

date/time データ型のサブ秒を Java トランスフォーメーションで処理する方法を設定できます。**[詳細]** タブで、サブ秒の処理を定義します。

デフォルトでは、生成された Java コードによって、トランスフォーメーションの date/time データ型が、ミリ秒単位の精度を持つ Java long データ型に変換されます。

Java コードでは、1 秒以下のデータをナノ秒単位まで処理できます。ナノ秒単位を処理するには、**[日付/時刻にナノ秒を使用する]** オプションを有効にします。このオプションを有効にすると、生成された Java コードによって、トランスフォーメーションの date/time データ型が、ナノ秒単位の精度を持つ Java BigDecimal データ型に変換されます。

詳細モードでは、マイクロ秒まで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。

Java コードの開発

Java トランスフォーメーションの機能を定義するには、**[Java]** タブで Java コードスニペットを入力します。コードスニペットを入力して Java パッケージをインポートし、Helper コードを書き込み、特定のトランスフォーメーションイベントの動作を定義する Java コードを書き込みます。Java エディタでは任意の順序でコードスニペットを開発することができます。

Java エディタの以下のセクションで Java コードスニペットを入力します。

パッケージのインポート

サードパーティ製の Java パッケージ、ビルトイン Java パッケージ、またはカスタム Java パッケージをインポートします。

ヘルパーコード

[パッケージのインポート] セクション以外のすべてのセクションで使用できる変数およびメソッドを定義します。

入力行に達したとき

入力行を受け取ったときのトランスフォーメーションの動作を定義します。

データの終わり

すべての入力データを処理したときのトランスフォーメーションの動作を定義します。

トランザクションを受け取ったとき

トランザクション通知を受け取ったときのトランスフォーメーションの動作を定義します。アクティブな Java トランスフォーメーションで使用します。

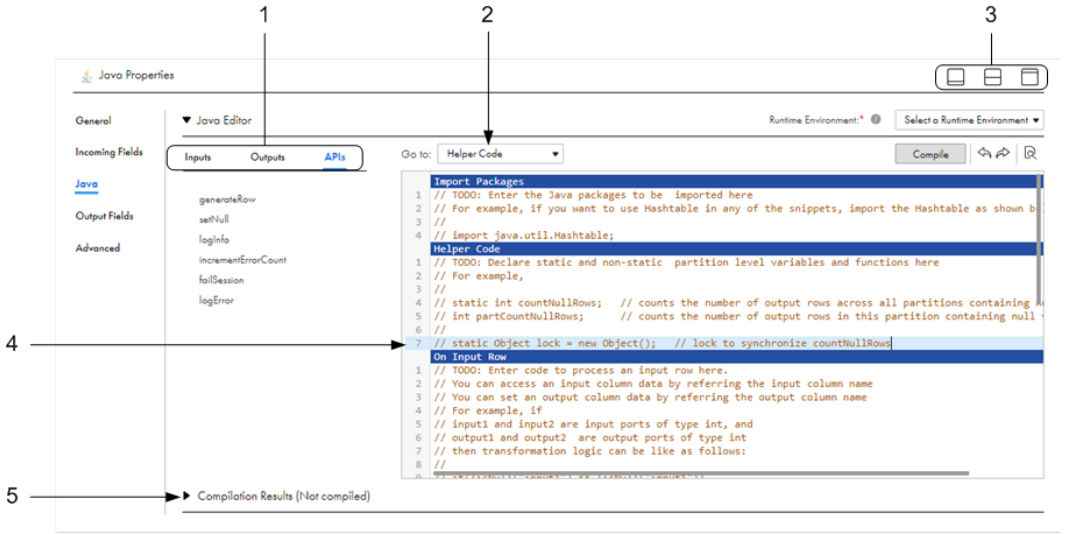
詳細モードでは、[受信トランザクション] セクションを使用できません。

入力データへのアクセスおよび出力データの設定は、[入力行に達したとき] セクションで実行します。アクティブなトランスフォーメーションの場合、[データの終わり] セクションおよび [トランザクションを受け取ったとき] セクションでも、出力データを設定できます。

Java コードスニペットの作成

トランスフォーメーション動作を定義するコードスニペットを作成するには、**[Java]** タブで Java エディタを使用します。

次の画像は、**[Java]** タブと展開されている Java エディタを示しています。



1. [入力]、[出力]、および［API］タブです。入力および出力フィールドを変数として追加し、Java コードスニペットで API メソッドを呼び出すには、これらのタブを使用します。これらのタブに表示されるフィールドおよびメソッドは、コードエントリ領域でどのセクションが選択されているかに基づいて異なります。
2. [移動］リストです。コードエントリ領域でセクション間の切り替えに使用します。
3. [最小化]、[両方とも開く]、および［最大化］アイコンです。[最小化］および［最大化］ボタンを使用すると、トランスフォーメーションのプロパティを最小化および最大化できます。[両方とも開く］アイコンを使用すると、Mapping Designer のキャンバスとトランスフォーメーションのプロパティを同時に開けます。
4. コードエントリ領域です。[パッケージのインポート]、[Helper コード]、[入力行に達したとき]、[データの終わり]、および［トランザクションを受け取ったとき］セクションに Java コードスニペットを入力します。
5. コンパイル結果です。コンパイル結果を展開すると、コンパイル結果の詳細、コンパイルエラー、およびコード全体を表示します。

ヒント: トランスフォーメーションのプロパティを展開してコードエントリ領域を詳しく表示するには、**[最大化]** をクリックします。

1. **[移動]** リストで、コードスニペットを入力するセクションを選択します。
2. スニペットの入力および出力フィールドにアクセスするには、**[入力]** または **[出力]** タブでフィールドを選択し、**[追加]** をクリックします。

   また、**[出力]** タブで出力フィールドを作成し、**[新しいフィールドの作成]** をクリックしても作成できます。
3. スニペットで Java トランスフォーメーションの API メソッドを呼び出すには、**[API]** タブでメソッドを選択し、**[追加]** をクリックします。

   [API] タブに表示されるメソッドは、選択されているセクションによって変わります。例えば、getInRowType メソッドは［入力行に達したとき］セクションでのみ使用でき、他のセクションでは使用できません。このため、このメソッドは、［入力行に達したとき］セクションが選択されている場合にのみ表示されます。
4. 必要に応じて、メソッドの入力値を設定します。
5. セクションに基づいて適切な Java コードを書き込みます。

Java コードスニペットの作成が完了したら、コードをコンパイルして、トランスフォーメーションを検証します。

Java パッケージのインポート

Java パッケージは、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インポートできます。Java エディタの［パッケージのインポート］セクションでパッケージをインポートします。

例えば Java I/O パッケージをインポートする場合、［パッケージのインポート］セクションに以下のコードを入力します。

```
import java.io.*;
```

組み込み、サードパーティ製、またはカスタムの Java パッケージをインポートできます。サードパーティ製またはカスタムの Java パッケージをインポートする場合は、パッケージをクラスパスに追加する必要があります。クラスパスの設定の詳細については、「クラスパス設定」 (ページ 237)を参照してください。

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セクションで使用できます。

［パッケージのインポート］セクションでは、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。

注: Java トランスフォーメーションが含まれるマッピングまたはマッピングタスクをエクスポートするときに、Java トランスフォーメーションに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含まれる jar または class ファイルは、エクスポート XML ファイルに含まれません。そのため、マッピングまたはタスク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含まれる jar または class ファイルを Secure Agent マシンにコピーする必要があります。

Helper コードの定義

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で 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。Java エディタ［Helper コード］セクションで Helper コードを定義します。

［Helper コード］領域で変数およびメソッドを宣言すると、［パッケージのインポート］領域を除く任意のコードエントリ領域でそれらを使用できます。

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。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、次のコードでは、Java 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ための静的変数を宣言します。

```
static int errorThreshold;
```

ユーザー定義の静的メソッドまたはインスタンスメソッド

これらのメソッドは、Java トランスフォーメーションの機能を拡張します。［Helper コード］セクションで宣言した Java メソッドは、出力変数を使用および変更することができます。［Helper コード］セクションの 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、［Helper コード］セクション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;
}
```

入力行の動作の定義

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できます。Java エディタの［入力行に達したとき］セクションで入力行の動作を定義します。このセクションの Java コードは、各入力行に対して 1 回ずつ実行されます。

以下の入出力フィールドのデータ、変数、およびメソッドは、［入力行に達したとき］セクションでアクセスして使用します。

入力フィールドおよび出力フィールドの変数

入力および出力フィールドのデータに変数としてアクセスするには、フィールドの名前を変数の名前として使用します。例えば「in_int」が整数の入力フィールドである場合、Java プリミティブデータ型 int で「in_int」変数として参照することで、このフィールドのデータにアクセスできます。入力および出力フィールドを変数として宣言する必要はありません。

入力フィールド変数に値は割り当てないでください。［入力行に達したとき］セクションの入力変数に値を割り当てると、対応するフィールドの入力データを現在の行では取得できません。

静的変数とユーザー定義メソッド

［Helper コード］セクションで宣言した任意の静的変数またはユーザー定義メソッドを使用します。

例えば、アクティブ Java トランスフォーメーションに、integer データ型の 2 つの入力フィールド（BASE_SALARY と BONUSES）、および integer データ型の 1 つの出力フィールド（TOTAL_COMP）があるとします。また［Helper コード］セクションで、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、［入力行に達したとき］セクションで以下の Java コードを使用し、入力フィールドの合計値を出力フィールド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 つの入力フィールド（BASE_SALARY および BONUSES）の値を加算した値を出力フィールド（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。

データの終わりに達したときの動作の定義

入力データをすべて処理したとき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できます。Java エディタの［データの終わり］セクションでデータの終わりの動作を定義します。

［データの終わり］セクションで出力行を生成するには、**［詳細］**タブで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範囲を［トランザクション］または［すべての入力］に設定します。このセクションでは、入力フィールドの変数にアクセスしたり、値を設定したりすることはできません。

以下の変数およびメソッドは、［データの終わり］セクションでアクセスして使用します。

出力フィールドの変数

アクティブな Java トランスフォーメーションで出力データへのアクセスまたは設定を実行する場合、出力フィールド名を変数として使用します。

ユーザー定義メソッド

［Helper コード］セクションで宣言した任意の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セッションログに情報を書き込みます。

```
logInfo("Number of null rows for partition is: " + partCountNullRows);
```

詳細モードで Java トランスフォーメーションを使用する場合は、次のガイドラインを考慮してください。

- 標準出力に出力を書き出すことはできませんが、ログファイルに表示される標準エラーには出力を書き出すことができます。
- 出力フィールドにバイナリ NULL 文字を渡すことはできません。

  マッピングエラーを避けるには、データを出力フィールドに書き出す前に、バイナリ NULL 文字を代替文字に置き換えるコードを Java トランスフォーメーションに追加します。

トランザクション通知動作の定義

トランザクション通知を受け取る際のアクティブ Java トランスフォーメーションの動作を定義できます。Java エディタの［トランザクションを受け取ったとき］セクションでトランザクション通知動作を定義します。

［トランザクションを受け取ったとき］セクションのコードスニペットは、トランスフォーメーションのトランザクションスコープが［トランザクション］に設定されている場合に限り実行されます。このセクションでは、入力フィールドの変数にアクセスしたり、値を設定したりすることはできません。

［トランザクションを受け取ったとき］セクションでは、以下の出力データ、変数、およびメソッドにアクセスして使用できます。

出力フィールドの変数

出力データへのアクセスまたは設定を実行する場合、出力フィールド名を変数として使用します。

ユーザー定義メソッド

［Helper コード］セクションで宣言した任意の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。

Java コードによるフラットファイルの解析

フラットファイルを解析する Java コードを作成できます。スキーマがさまざまに異なるフラットファイルまたは JMS メッセージから特定のデータカラムを抽出するには、Java コードを使用します。

例えば、区切りフラットファイルから先頭 2 つのデータカラムを読み込むものとします。区切りフラットファイルからデータを読み込んで、1 つ以上の出力フィールドに渡すマッピングを作成します。

マッピングは、以下のようなコンポーネントから構成されています。

ソーストランスフォーメーション

ソースは区切りフラットファイルです。各行を単一文字列として Java トランスフォーメーションに渡すソースを設定します。ソースファイルには下記のデータが含まれています。

```
1a,2a,3a,4a,5a,6a,7a,8a,9a,10a
1b,2b,3b,4b,5b,6b,7b,8b,9b
1c,2c,3c,4c,5c,6c,7c
1d,2d,3d,4d,5d,6d,7d,8d,9d,10d
```

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の機能を Java エディタで定義します。

Java エディタの［入力行に達したとき］セクションを使用して各入力行を読み取り、最初の 2 つのフィールドを［outputRow］出力フィールドに渡します。［入力行に達したとき］セクションで、以下のコードを入力します。

```
// Collect the first two fields of the row and output them into outputRow.
String[] rowsSplit = row.split(",", 3);
if (rowsSplit.length >= 2) {
  outputRow = rowsSplit[0] + "," + rowsSplit[1];
}
generateRow();
```

ターゲットトランスフォーメーション

ターゲットが Java トランスフォーメーションから［outputRow］出力フィールドを受け取るように設定します。マッピングを実行すると、ターゲットファイルには以下のデータが含まれます。

```
1a,2a
1b,2b
1c,2c
1d,2d
```

コードのコンパイル

Java トランスフォーメーションを検証するには、Java エディタで入力する Java コードスニペットをコンパイルする必要があります。データ統合は、Secure Agent を使用して Java コードスニペットをコンパイルします。コードをコンパイルすると **［Java］** タブにコンパイル結果が表示されます。

作成した Java 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の基本的な機能を定義する Java クラスが含まれています。トランスフォーメーションをコンパイルすると、Secure Agent は、Java エディタで入力した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ます。これにより、トランスフォーメーションのクラスコード全体が生成されます。

Secure Agent は、JDK を呼び出してクラスコード全体をコンパイルします。JDK 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。

注: マップレットでは、Java トランスフォーメーションは、データ統合サーバーで実行可能なデータ型と API に基づいてコンパイルされます。invokeJExpression API メソッドなど、詳細クラスタでのみ実行できるデータ型または API がコードに含まれている場合、コードのコンパイルに失敗します。

コードをコンパイルするには、次のタスクを完了する必要があります。

1. 標準ではない Java パッケージを使用する場合は、クラスパスを設定します。

   クラスパスの設定の詳細については、「クラスパス設定」 (ページ 237)を参照してください。

2. コンパイルに使用するランタイム環境を選択します。

   ［Java］ タブでランタイム環境を選択します。Hosted Agent はコンパイル用にサポートされていません。

コードをコンパイルするには、Java エディタで **［コンパイル］** をクリックします。

コンパイル結果に、コンパイルの結果が表示されます。コンパイル結果を使用して Java コードのエラーを特定します。

クラスコード全体の表示

Java トランスフォーメーションのコンパイル時に生成されるクラスコード全体を表示およびダウンロードできます。**[Java]** タブの［コンパイル結果］からクラスコード全体にアクセスできます。

クラスコード全体を表示するには、コンパイル結果で **[完全なコードを表示]** をクリックします。データ統合では、**[コード全体]** ダイアログボックスにクラスコード全体を表示します。

クラスコード全体をダウンロードするには、**[コード全体]** ダイアログボックスで **[コード全体のダウンロード]** をクリックします。

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

[Java] タブのコンパイル結果では、Java トランスフォーメーション用の Java コードエラーの特定、およびそのソースの検出を実行できます。Java エディタのエラー、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のエラーにより、Java トランスフォーメーションのエラーが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のトラブルシューティングを実行するには:

- Java コードスニペット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。
- コンパイル結果を使用したエラーのタイプとエラーの場所を特定します。

エラーのソースとタイプを特定してから、**[Java]** タブで Java コードを修正し、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

Java トランスフォーメーションをコンパイルすると、結果はコンパイル結果に表示されます。コンパイルに成功すると、コンパイル結果には「コンパイルが成功しました」というメッセージが表示されます。コンパイルに失敗すると、コンパイル結果にはコンパイルエラーとエラーの場所が表示されます。

コンパイルエラーは、次の場所で発生する可能性があります。

Java **エディタ**

エラーが Java エディタのコードスニペット内にある場合、データ統合はそのセクションとエラーが含まれる行をリストします。またデータ統合は、Java エディタでエラーの原因を強調表示します。

コード全体

エラーがコード全体にある場合、データ統合はエラーの場所を「コード全体」であると表示し、**[コード全体]** ダイアログボックスにエラーが含まれる行をリストします。

[コード全体] ダイアログボックスでは、エラーを見つけることはできますが、Java コードを編集できません。**[コード全体]** ダイアログボックスで見つけたエラーを修正するには、適切なセクションでコードを編集します。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]** ダイアログボックスを使用する必要があります。

エラータイプの特定

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。

コンパイルエラーには以下のタイプがあります。

ユーザーコードのエラー

エラーは、Java エディタのさまざまなセクション内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、データ統合がユーザーコードをクラスのコード全体に追加するときにも発生することがあります。

例えば、Java トランスフォーメーションには integer データ型の int1 という名前の入力フィールドがあるとします。クラスのコード全体は、以下のコードで入力フィールドの変数を宣言します。

```
int int1;
```

しかし、［入力行に達したとき］セクション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、［入力行に達したとき］セクションの変数名を変更します。

非ユーザーコードのエラー

Java エディタのセクション内にあるユーザーコードが原因で、非ユーザーコードにエラーが発生することがあります。

例えば、Java トランスフォーメーションには integer データ型の入力フィールド int1 と出力フィールド out1 があるとします。ここで、以下のコードを［入力行に達したとき］セクションに入力し、入力フィールド int1 の interest を計算して出力フィールド 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、データ統合は［入力行に達したとき］セクションのコードを、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。

Java トランスフォーメーションの例

ここで例として示される Java コードを使用すると、アクティブな Java トランスフォーメーションの作成およびコンパイルを実行できます。

Java トランスフォーメーションを使用して、架空の会社の従業員データを処理します。Java トランスフォーメーションは、フラットファイルソースから入力行を読み込み、フラットファイルターゲットに出力行を書き込みます。ソースファイルには、従業員 ID 番号、名前、職位、管理職の ID 番号を含む従業員データが格納されています。

トランスフォーメーションは、指定された従業員の管理職の名前を管理職の ID 番号に基づいて検索し、従業員データを含む出力行を生成します。出力データには、従業員 ID 番号、名前、職位、従業員の管理職の名前が含まれています。ソースデータに従業員の管理職が存在しない場合は、トランスフォーメーションはその従業員が組織階層の最上位に位置している人物であると想定します。

注: トランスフォーメーションロジックは、従業員の職位がソースファイルで降順に配置されていると想定します。

この例のマッピングを作成して実行するには、次の手順を実行します。

1. ソースファイルを作成します。

   Secure Agent がアクセスできるディレクトリにカンマ区切りのフラットファイルを作成します。

2. マッピングを設定します。

   ソーストランスフォーメーション、ターゲットトランスフォーメーション、および Java トランスフォーメーションをマッピングに追加し、フィールドを設定します。

3. Java トランスフォーメーションの Java コードスニペットを設定します。

   [パッケージのインポート]、[Helper コード]、および［入力行に達したとき］セクションにコードスニペットを入力します。

4. コードをコンパイルし、マッピングを実行します。

ソースファイルを作成する

ソースファイルをカンマ区切りのフラットファイルとして作成します。Secure Agent がアクセス可能なディレクトリにファイルを保存します。

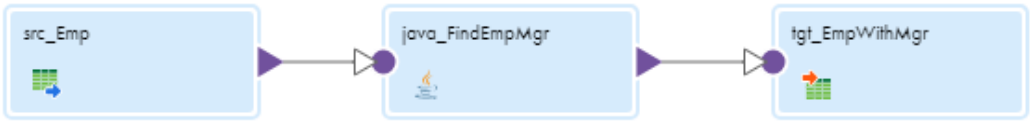
以下のデータを使用します。

```
EMP_ID,EMP_NAME,EMP_AGE,EMP_DESC,EMP_PARENT_EMPID
1,James Davis,50,CEO,
4,Elaine Masters,40,Vice President - Sales,1
5,Naresh Thiagarajan,40,Vice President - HR,1
6,Jeanne Williams,40,Vice President - Software,1
9,Geetha Manjunath,34,Senior HR Manager,5
10,Dan Thomas,32,Senior Software Manager,6
14,Shankar Rahul,34,Senior Software Manager,6
20,Juan Cardenas,32,Technical Lead,10
21,Pramodh Rahman,36,Lead Engineer,14
22,Sandra Patterson,24,Software Engineer,10
23,Tom Kelly,32,Lead Engineer,10
35,Betty Johnson,27,Lead Engineer,14
50,Dave Chu,26,Software Engineer,23
70,Srihari Giran,23,Software Engineer,35
71,Frank Smalls,24,Software Engineer,35
```

マッピングの設定

この例でマッピングを設定するには、新しいマッピングを作成し、Java トランスフォーメーションを追加します。次にソースフィールド、ターゲットフィールド、および Java トランスフォーメーションフィールドを設定します。Java 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動作がアクティブであることを確認します。

次の図に、マッピングを示します。



トランスフォーメーションを次のように設定します。

ソーストランスフォーメーション

ソーストランスフォーメーションで、次の表に示すようにソースフィールドメタデータを更新します。

名前	タイプ	精度	スケール	オリジン
EMP_ID	integer	10	0	<ソースファイル名>
EMP_NAME	string	100	0	<ソースファイル名>
EMP_DESC	string	100	0	<ソースファイル名>
EMP_AGE	integer	10	0	<ソースファイル名>
EMP_PARENT_EMPID	integer	10	0	<ソースファイル名>

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には、ソーストランスフォーメーションからの受信フィールドがすべて含まれます。

以下の出力フィールドを作成します。

名前	タイプ	精度	スケール
EMP_ID_OUT	integer	10	0
EMP_NAME_OUT	string	100	0
EMP_DESC_OUT	string	100	0
EMP_PARENT_EMPNAME	string	100	0

[詳細] タブで、動作がアクティブに設定されていることを確認します。

ターゲットトランスフォーメーション

ターゲットトランスフォーメーションで、以下のターゲットフィールドを作成します。

名前	タイプ	精度	スケール	オリジン
EMP_ID	integer	10	0	<ターゲットファイル名>
EMP_NAME	string	100	0	<ターゲットファイル名>
EMP_DESC	string	100	0	<ターゲットファイル名>
EMP_PARENT_EMPNAME	string	100	0	<ターゲットファイル名>

次の表に示すようにフィールドマッピングを設定します。

ターゲットフィールド名	マッピングされたフィールド
EMP_ID	EMP_ID_OUT
EMP_NAME	EMP_NAME_OUT
EMP_DESC	EMP_DESC_OUT
EMP_PARENT_EMPNAME	EMP_PARENT_EMPNAME

Java コードスニペットの設定

トランスフォーメーション機能を定義する Java コードスニペットを［**Java**］タブで入力します。

Java エディタの以下のセクションで Java コードスニペットを入力します。

パッケージのインポート

java.util.Map および java.util.HashMap パッケージをインポートします。

ヘルパーコード

Java トランスフォーメーションにおいてデータの状態を追跡するために Map オブジェクト、ロックオブジェクト、およびブール変数を作成します。

入力行に達したとき

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コードを入力します。

パッケージのインポート

［パッケージのインポート］セクションでは、サードパーティ製の Java パッケージ、ビルトイン Java パッケージ、カスタム Java パッケージをインポートします。この例では、Map パッケージおよび HashMap パッケージを使用しています。

［パッケージのインポート］セクションに、以下のコードを入力します。

```
import java.util.Map;
import java.util.HashMap;
```

データ統合が、トランスフォーメーションの Java コードにインポート文を追加します。

ヘルパーコード

［Helper コード］セクションで、Java トランスフォーメーションのユーザー定義変数およびメソッドを宣言します。

［Helper コード］セクションは、［入力行に達したとき］セクションの Java コードで使用する以下の変数を定義します。

変数	説明
empMap	ソースからの ID 番号および従業員名を格納する Map オブジェクトです。
lock	パーティション全体で empMap へのアクセスを同期するために使用する Lock オブジェクトです。

変数	説明
generateRow	現在の入力行に対して出力行の生成が必要かどうかを判断するために使用する Boolean 変数です。
isRoot	その従業員が企業の組織図の最上位（ルート）に位置するかどうかを判断するために使用する Boolean 変数です。

［Helper コード］セクションに、以下のコードを入力します。

```
// Static Map object to store the ID and name relationship of an employee.
// If a session uses multiple partitions, empMap is shared across all partitions.
private static Map <Integer, String> empMap = new HashMap <Integer, String> ();

// Static lock object to synchronize the access to empMap across partitions.
private static Object lock = new Object();

// Boolean to track whether to generate an output row based on validity of the
// input data.
private boolean generateRow;

// Boolean to track whether the employee is root.
private boolean isRoot;
```

データ統合が、トランスフォーメーションの Java コードにインポート文を追加します。

入力行に達したとき

Java トランスフォーメーションは、入力行を受け取ると［入力行に達したとき］セクションで Java コードを実行します。この例では、トランスフォーメーションによって出力行を生成することについての可能性が入力行の値に基づいて決定されます。

［入力行に達したとき］セクションで、以下のコードを入力します。

```
// Initially set generateRow to true for each input row.
generateRow = true;

// Initially set isRoot to false for each input row.
isRoot = false;

// Check if input employee id and name is null.
if (isNull ("EMP_ID") || isNull ("EMP_NAME"))
{
    incrementErrorCount(1);

    // If input employee id and/or name is null, don't generate a output row for this
    // input row.
    generateRow = false;

} else {

    // Set the output field values.
    EMP_ID_OUT = EMP_ID;
    EMP_NAME_OUT = EMP_NAME;
}

if (isNull ("EMP_DESC"))
{
    setNull("EMP_DESC_OUT");
} else {
    EMP_DESC_OUT = EMP_DESC;
}

boolean isParentEmpIdNull = isNull("EMP_PARENT_EMPID");

if(isParentEmpIdNull)
```

```
{
    // This employee is the root for the hierarchy.
    isRoot = true;
    logInfo("This is the root for this hierarchy.");
    setNull("EMP_PARENT_EMPNAME");
}

synchronized(lock)

{
    // If the employee is not the root for this hierarchy, get the corresponding
    // parent ID.
    if(!isParentEmpIdNull)
    EMP_PARENT_EMPNAME = (String) (empMap.get(new Integer (EMP_PARENT_EMPID)));

    // Add employee to the map for future reference.
    empMap.put (new Integer(EMP_ID), EMP_NAME);
}

// Generate row if generateRow is true.
if(generateRow)
    generateRow();
```

コードのコンパイルとマッピングの実行

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には、Java エディタで**［コンパイル］**をクリックします。コードを正常にコンパイルすると、マッピングを実行できます。

コンパイル結果には、コンパイルの状況が表示されます。Java コードのコンパイルに失敗した場合、Java エディタでエラーを修正してから Java コードをコンパイルします。トランスフォーメーションを正常にコンパイルしたら、マッピングを保存して実行します。

マッピングを実行するには、Mapping Designer で**［実行］**をクリックします。

ターゲットファイルには次の結果が含まれています。

```
"EMP_ID","EMP_NAME","EMP_DESC","EMP_PARENT_EMPNAME"
1,"James Davis","CEO",
4,"Elaine Masters","Vice President - Sales","James Davis"
5,"Naresh Thiagarajan","Vice President - HR","James Davis"
6,"Jeanne Williams","Vice President - Software","James Davis"
9,"Geetha Manjunath","Senior HR Manager","Naresh Thiagarajan"
10,"Dan Thomas","Senior Software Manager","Jeanne Williams"
14,"Shankar Rahul","Senior Software Manager","Jeanne Williams"
20,"Juan Cardenas","Technical Lead","Dan Thomas"
21,"Pramodh Rahman","Lead Engineer","Shankar Rahul"
22,"Sandra Patterson","Software Engineer","Dan Thomas"
23,"Tom Kelly","Lead Engineer","Dan Thomas"
35,"Betty Johnson","Lead Engineer","Shankar Rahul"
50,"Dave Chu","Software Engineer","Tom Kelly"
70,"Srihari Giran","Software Engineer","Betty Johnson"
71,"Frank Smalls","Software Engineer","Betty Johnson"
```

第 16 章

Java トランスフォーメーション API リファレンス

Java トランスフォーメーションを作成するときは、トランスフォーメーションの動作を定義するために、API メソッドを Java コードスニペットに追加できます。

API メソッドをコードスニペットに追加するには、Java エディタの **[API]** タブをクリックし、追加するメソッドを選択し、**[追加]** をクリックします。または、Java コードスニペットに API メソッドを直接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 コードスニペットに追加できる API メソッドは次のとおりです。

failSession

例外発生時に、エラーメッセージを表示して、セッションを失敗させます。

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

セッションのエラーカウントを増やします。

invokeJExpression

式を呼び出し、式の値を返します。詳細モードでのみ使用します。

isNull

入力カラムの NULL 値の有無を確認します。

logError

セッションログにエラーメッセージを書き込みます。

logInfo

セッションログに情報メッセージを書き込みます。

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。

setOutRowType

出力行の更新戦略を設定します。行に挿入、更新、または削除のフラグを設定できます。

failSession

例外発生時に、エラーメッセージを表示して、セッションを失敗させます。failSession を使用すると、セッションが終了します。

Java エディタの任意のセクション（[パッケージのインポート] を除く）で failSession を使用します。Java エディタの try/catch ブロックで failSession を使用しないでください。

以下の構文を使用します。

```
failSession(String errorMessage);
```

引数	データ型	入出力	説明
errorMessage	文字列型	入力	エラーメッセージの文字列。

以下の Java コードを使用すると、入力フィールド input1 に NULL 値が存在するかどうかについてテストされ、input1 が NULL の場合はセッションに失敗します。

```
if(isNull(”input1”)) {
     failSession(“Cannot process a null value for field 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。

generateRow を呼び出すと、Java トランスフォーメーションは出力フィールド変数の現在の値を使用して出力行を生成します。入力行に対応する複数の行を生成する場合、各入力行で generateRow を繰り返し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 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。

Java エディタの任意のセクション（[パッケージのインポート] を除く）で generateRow を使用します。generateRow は、アクティブなトランスフォーメーションでのみ使用できます。generateRow をパッシブなトランスフォーメーションで使用すると、セッションではエラーが発生します。

以下の構文を使用します。

```
generateRow();
```

以下の Java コードを使用すると、1 つの出力行が生成され、出力フィールドの値が変更され、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。

getInRowType は、Java エディタの［入力行に達したとき］セクションでのみ使用できます。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のみ使用できます。アップデートストラテジを設定するように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使用すると、セッションでエラーが発生します。

以下の構文を使用します。

```
rowType getInRowType();
```

引数	データ型	入出力	説明
rowType	String	アウトプット	更新方式のタイプ。INSERT、UPDATE、DELETE、REJECT を使用できます。

行のタイプが UPDATE または INSERT で、input1 入力フィールドの値が 100 より小さい場合は、以下の Java コードを使用して現在の行の入力タイプをプロパゲートします。input1 の値が 100 よりも大きい場合は、出力タイプを DELETE に設定します。

```
// Set the value of the output field.
output1 = input1;

// Get and set the row type.
String rowType = getInRowType();
setOutRowType(rowType);

// Set row type to DELETE if the output field value is > 100.
if(input1 > 100
     setOutRowType(DELETE);
```

incrementErrorCount

セッションのエラーカウントを増やします。エラー数がセッションのエラーしきい値に達すると、セッションは失敗します。

Java エディタの任意のセクション（［パッケージのインポート］を除く）で incrementErrorCount を使用します。

以下の構文を使用します。

```
incrementErrorCount(int nErrors);
```

引数	データ型	入出力	説明
nErrors	Integer	入力	セッションのエラーカウントとして増やすエラーの数。

以下の Java コードを使用すると、トランスフォーメーションの入力フィールドが NULL 値の場合にエラーカウントが増加します。

```
// Check whether input employee ID or name is null.
if (isNull ("EMP_ID_INP") || isNull ("EMP_NAME_INP"))
{
```

```
    incrementErrorCount(1);

    // If input employee ID or name is null, don't generate an output row for this input row.
    generateRow = false;
}
```

invokeJExpression

式を呼び出し、式の値を返します。詳細モードでのみ使用します。

invokeJExpression は、Java エディタの［パッケージのインポート］と［ヘルパーコード］を除く任意のセクションで使用します。

以下の構文を使用します。

```
(dataType)invokeJExpression(
                    String expression,
                    Object[] paramMetadataArray);
```

次の表に、引数を示します。

引数	データ型	入出力	説明
dataType	-	出力	戻り値のキャスト先のデータ型。デフォルトでは、戻り値のデータ型は object です。 戻り値は integer、double、string、または byte[]データ型にキャストできます。
expression	文字列	入力	呼び出す式を表す文字列。 文字「x」を使用して、パラメータに連続した番号を付ける必要があります。例えば、呼び出す式に 3 つのパラメータが必要な場合は、パラメータに x1、x2、および x3 という名前を付けます。
paramMetadataArray	Object[]	入力	呼び出す式の入力パラメータを含むオブジェクトの配列。

以下の Java コードを使用すると、文字列 John および Smith を連結する concat()メソッドが呼び出されます。

```
(String)invokeJExpression("concat(x1,x2)", new Object [] { "John ", "Smith" });
```

次の文字列が返されます。

```
John Smith
```

invokeJExpression メソッドを使用する場合は、以下のルールとガイドラインを考慮してください。

- デフォルトでは、戻り値のアップデートストラテジは INSERT です。別のアップデートストラテジを使用するには、Java コードでアップデートストラテジを定義する必要があります。
- 引数、パラメータ、または戻り値が NULL の場合、値は NULL インジケータとして扱われます。

  例えば、呼び出す式の戻り値が NULL で戻り値のデータ型が string の場合、invokeJExpression メソッドは NULL の値を含む文字列を返します。
- 呼び出す式の入力パラメータが date/time データ型の場合は、パラメータを文字列として渡し、TO_DATE 関数を使用して文字列を date/time データ型に変換する必要があります。

例えば、次の引数を使用して、呼び出す式に日付/時刻値を渡します。

```
new Object [] { "TO_DATE("01/22/98", "MM/DD/YY")" }
```

- invokeJExpression メソッドが date/time データ型を返す場合は、戻り値を文字列にキャストする必要があります。

isNull

入力カラムの値を調べ、NULL 値の有無を確認します。isNull は、入力カラムを値として使用する前にそのカラムのデータが NULL かどうかをチェックします。

isNull は、Java エディタの［入力行に達したとき］セクションで使用します。

以下の構文を使用します。

```
Boolean isNull(String satrColName);
```

引数	データ型	入出力	説明
strColName	文字列型	入力	入力カラムの名前。

以下の Java コードを使用すると、SALARY 入力カラムを出力フィールド TOTAL_SALARIES に追加する前に、その入力カラムの値がチェックされます。

```
// If value of SALARY is not null
if (!isNull("SALARY")) {

     // Add to TOTAL_SALARIES.
     TOTAL_SALARIES += SALARY;
}
```

または

```
// If value of SALARY is not null
String strColName = "SALARY";
if (!isNull(strColName)) {

     // Add to TOTAL_SALARIES.
     TOTAL_SALARIES += SALARY;
}
```

logError

セッションログにエラーメッセージを書き込みます。

Java エディタの任意のセクション（[パッケージのインポート］を除く）で logError を使用します。

以下の構文を使用します。

```
logError(String errorMessage);
```

引数	データ型	入出力	説明
errorMessage	文字列型	入力	エラーメッセージの文字列。

以下の Java コードを使用すると、入力フィールドが NULL のエラーが記録されます。

```
// Log an error if BASE_SALARY is null.
if (isNull("BASE_SALARY")) {
     logError("Cannot process a null salary field.");
}
```

以下のメッセージがメッセージログに表示されます。

```
[JTX_1013] [ERROR] Cannot process a null salary field.
```

logInfo

セッションログに情報メッセージを書き込みます。

Java エディタの任意のセクション（[パッケージのインポート] を除く）で logInfo を使用します。

以下の構文を使用します。

```
logInfo(String logMessage);
```

引数	データ型	入出力	説明
logMessage	文字列型	入力	情報メッセージの文字列。

以下の Java コードを使用すると、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 1000 行を処理した後に、セッションログにメッセージが書き込まれます。

```
if (numRowsProcessed == messageThreshold) {
     logInfo("Processed " + messageThreshold + " rows.");
}
```

以下のメッセージがセッションログに表示されます。

```
[JTX_1012] [INFO] Processed 1000 rows.
```

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。出力カラムを NULL に設定すると、出力行を生成するまで値は変更できません。

[パッケージのインポート] を除く Java エディタのすべてのセクションで setNull を使用します。

以下の構文を使用します。

```
setNull(String strColName);
```

引数	データ型	入出力	説明
strColName	文字列型	入力	出力カラムの名前。

以下の Java コードを使用すると、入力カラムの値がチェックされ、出力カラムに対応する値が NULl に設定されます。

```
// Check the value of Q3RESULTS input column.
if(isNull("Q3RESULTS")) {
```

```
     // Set the value of output column to null.
     setNull("RESULTS");
}
```

または

```
// Check the value of Q3RESULTS input column.
String strColName = "Q3RESULTS";
if(isNull(strColName)) {

     // Set the value of output column to null.
     setNull(strColName);
}
```

setOutRowType

出力行の更新方式を設定します。setOutRowType メソッドは、挿入、更新、または削除を実行する行にフラグを設定できます。

Java エディタの［入力行に達したとき］セクションで setOutRowType を使用します。setOutRowType 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のみ使用できます。アップデートストラテジを設定するように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 setOutRowType を使用すると、セッションでエラーが発生してセッションに失敗します。

以下の構文を使用します。

```
setOutRowType(String rowType);
```

引数	データ型	入出力	説明
rowType	文字列型	入力	更新方式のタイプ。INSERT、UPDATE、DELETE を使用できます。

行のタイプが UPDATE または INSERT で、input1 入力フィールドの値が 100 より小さい場合は、以下の Java コードを使用してカレント行の入力タイプをプロパゲートします。input1 の値が 100 よりも大きい場合は、出力タイプを DELETE に設定します。

```
// Set the value of the output field.
output1 = input1;

// Get and set the row type.
String rowType = getInRowType();
setOutRowType(rowType);

// Set row type to DELETE if the output field value is > 100.
if(input1 > 100)
     setOutRowType(DELETE);
```

第 17 章

ジョイナトランスフォーメーション

ジョイナトランスフォーメーションは、2 つの関連する異種ソースからのデータを結合できます。例えば、ジョイナトランスフォーメーションを使用して、Salesforce の取引先オブジェクトのデータが含まれるフラットファイルから取引先情報を結合できます。

ジョイナトランスフォーメーションは、結合条件および結合タイプに基づいてデータを結合できます。結合条件では、2 つのソース間でフィールドを照合します。複数の結合条件を作成できます。結合タイプでは、結果に含まれるデータセットを定義します。

トランスフォーメーションをジョイナトランスフォーメーションにリンクする場合、トランスフォーメーションをマスタグループまたは明細グループにリンクします。ジョブのパフォーマンスを高めるには、小さいデータセットを表すトランスフォーメーションをマスタグループに接続します。

マッピング内の 3 つ以上のソースを結合する場合は、複数のジョイナトランスフォーメーションを使用できます。ジョイナトランスフォーメーションからの出力を別のソースパイプラインと結合できます。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できます。

一致するフィールド名があるソースを結合する場合、フィールド名の競合が発生する可能性があります。競合は次のいずれかの方法で解決できます。

- フィールド名の競合の解決を作成する。
- アップストリームトランスフォーメーションの一致するフィールドの名前を変更する。
- 式トランスフォーメーションを介してデータを渡し、フィールドの名前を変更する。

結合条件

結合条件では、入力行を結合するタイミングを定義します。結合条件には、ソース行を結合するために一致している必要がある両方のソースのフィールドが含まれます。

マスタデータと明細データが等しいかどうかに基づいて、1 つまたは複数の条件を定義します。例えば、2 つの従業員データセットに従業員 ID 番号が含まれている場合、次の条件は両方のデータセットの同じ従業員 ID がある行に一致します。

```
EMP_ID1 = EMP_ID2
```

1 つ以上の結合条件を使用します。結合条件を追加すると、データの結合に必要な時間が増加します。複数の結合条件を使用する場合、マッピングタスクは指定した順に条件を評価します。

条件の両方のフィールドは、同じデータ型を持つ必要があります。データ型が一致しない 2 つのフィールドを条件で使用しなければならない場合、データ型が一致するように変換します。

例えば、Char データと Varchar データを結合する場合、Char 値に埋め込まれている空白は文字列の一部に含まれます。両方のフィールドに「Shoes」という値が含まれていても、Char(40)フィールドには 35 個の末尾のスペースが含まれているため、値が一致しません。値が確実に一致するようにするには、一方のフィールドのデータ型をもう一方に合わせて変更します。

詳細モードでは、結合条件にバイナリデータ型を含めたり、結合条件でバイナリデータ型を評価したりすることはできません。

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。NULL 値を持つ行を結合するには、NULL 値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。

結合タイプ

結合タイプにより、残りのマッピングに渡される結果セットが決まります。

次の結合タイプを使用できます。

Normal **ジョイン**

結合条件に一致する行が含まれます。結合条件に一致しない行は破棄されます。

マスタ外部

明細パイプラインのすべての行およびマスタパイプラインの一致する行が含まれます。一致しないマスタパイプラインの行は破棄されます。

明細外部

マスタパイプラインのすべての行および明細パイプラインの一致する行が含まれます。一致しない明細パイプラインの行は破棄されます。

完全外部

結合条件に一致する行およびマスタパイプラインと明細パイプラインのすべての入力データが含まれます。

詳細プロパティ

ジョイナトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベル、キャッシュ設定、NULL の順序付け、トランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
キャッシュディレクトリ	マスタ行または明細行、およびこれらの行のインデックスをキャッシュするためのディレクトリを指定します。 デフォルトでは、データ統合は、データ統合サーバーで Secure Agent $PMCacheDir プロパティに入力されたディレクトリを使用します。新しいディレクトリを入力する場合は、そのディレクトリが存在していて、かつキャッシュファイルを格納するための十分なディスク領域があることを確認します。マップされたドライブまたはマウントされたドライブを指定できます。
マスタにおける NULL の順序付け	マスターパイプラインでの Null の順序付け。[NULL は最上位] または [NULL は最下位] を選択します。
明細における NULL の順序付け	明細パイプラインでの Null の順序付け。[NULL は最上位] または [NULL は最下位] を選択します。
データキャッシュサイズ	トランスフォーメーションのデータキャッシュサイズ。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
インデックスキャッシュサイズ	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。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
ソート済み入力	データがソート済みであることを指定します。このオプションを選択すると、ソート済みデータが結合され、パフォーマンスが向上します。
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になっている場合は、[昇順] を選択します。[昇順] を選択すると、ソート済み入力が有効になります。デフォルトは [Auto] です。

プロパティ	説明
トランスフォーメーション範囲	データ統合が受信データにトランスフォーメーションロジックを適用する方法を指定します。 - 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。データの行が同一トランザクション内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。 - 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、データ統合は受信トランザクションの境界を削除します。データの行がソース内のすべての行に依存している場合は、[すべての入力] を選択します。 - 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。データの行が他の行に依存していない場合は [Row] を選択します。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。

注: 階層フィールドを結合条件で使用することはできません。

ジョイナトランスフォーメーションの作成

ジョイナトランスフォーメーションで、2 つの関連する異種ソースからのデータを結合します。

ジョイナトランスフォーメーションを作成する前に、ソースデータを表すためにソーストランスフォーメーションをマッピングに追加します。使用する他のアップストリームトランスフォーメーションを追加します。

2 つのパイプラインのデータに、一致するフィールド名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からのトランスフォーメーションアップストリームの一方のフィールドセットの名前を変更します。

1. **[トランスフォーメーション]** パレットで、ジョイナトランスフォーメーションをマッピングキャンバスにドラッグします。
2. 一方のデータセットを表すアップストリームトランスフォーメーションをジョイナトランスフォーメーションのマスタグループに接続します。

   ジョブのパフォーマンスを高めるには、小さいデータセットを表すトランスフォーメーションを使用します。

3. もう一方のデータセットを表すアップストリームトランスフォーメーションをジョイナトランスフォーメーションの明細グループに接続します。
4. **［全般］**タブで、トランスフォーメーションの名前と、説明（省略可能）を入力します。
5. **［受信フィールド］**タブで、トランスフォーメーションに入力されるデータを定義するフィールドルールを設定します。
6. **［結合条件］**タブで、結合タイプを選択します。
7. 結合条件を設定するには、結合条件で［簡易］を選択します。**［新規結合条件の追加］**をクリックし、使用するマスターフィールドと詳細フィールド、および演算子を選択します。複数の結合条件を作成できます。

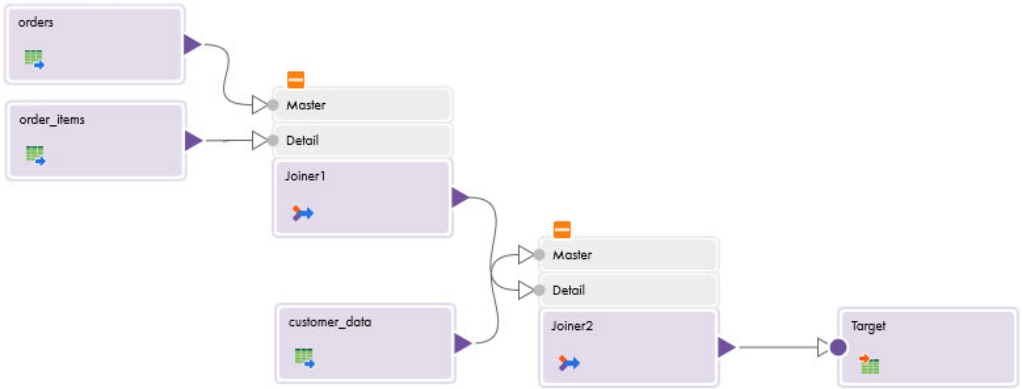
   または、結合条件にパラメータを使用するには、結合条件に［完全にパラメータ化］を選択します。

ダウンストリームトランスフォーメーションをマッピングに追加して設定できます。マッピングが完了したら、マッピングを検証して保存できます。

ジョイナトランスフォーメーションの例

あなたはオンライン小売業者のマーケティングマネージャであり、注文データをさまざまな Amazon S3 ソースからの製品データおよび顧客データとマージして、顧客が何を購入しているかを理解したいと考えています。ジョイナトランスフォーメーションを使用して、ソースからのデータを結合します。

Amazon S3 バケットには、*orders*、*order_items*、*customer_data* という 3 つのソースデータテーブルがあります。次の図に、これらのソースからのデータを結合するマッピングを示します。



マッピングには次の要素が含まれます。

***orders* のソーストランスフォーメーション**

orders データテーブルには、各オンライン注文の注文番号、日付、価格、顧客 ID のフィールドが含まれています。

次の表に *orders* の一部を示します。

Order_id	order_date	customer_id	order_price
1005	2023-01-20	789	78.25

Order_id	order_date	customer_id	order_price
1006	2023-01-24	268	150.09
1007	2023-02-07	268	30.20

order_items のソーストランスフォーメーション

order_items データテーブルには、数量や価格など、各注文の商品に関する詳細が含まれます。

次の表に *order_items* の一部を示します。

Order_id	item_id	qty	price
1005	5063	2	34.99
1006	2389	3	19.99
1006	5063	1	34.99
1007	9871	2	10.99

ソーストランスフォーメーションでは、*order_items* を *orders* と結合するときにフィールド名の競合を避けるために、フィールド order_id の名前を items_order_id に変更します。

customer_data のソーストランスフォーメーション

customer_data テーブルには、名前、生年月日、電話番号など、顧客が提供する情報のフィールドが含まれています。

次の表に *customer_data* の一部を示します。

c_id	c_name	c_dob
789	Kelcy Almeida	1969-07-20
268	Chidi Donalds	1972-12-07

orders と *order_items* のジョイナトランスフォーメーション

最初のジョイナトランスフォーメーションは、*orders* と *order_items* の間の通常の結合を実行します。*orders* ソーストランスフォーメーションはマスターグループであり、*order_items* ソーストランスフォーメーションは詳細グループであるため、注文された各品目に注文情報が追加されます。

ジョイナトランスフォーメーションは、次の結合条件を使用して、注文 ID でデータを照合します: order_id = items_order_id。

customer_data のジョイナトランスフォーメーション

2 番目のジョイナトランスフォーメーションは、*customer_data* と最初のジョイナトランスフォーメーションからの出力の間の詳細な外部結合を実行します。このトランスフォーメーションでは、*customer_data* ソーストランスフォーメーションがより小さなデータセットであるため、マスターグループとして使用されます。

2 番目のジョイナトランスフォーメーションは、次の結合条件を使用して、顧客 ID でデータを照合します: customer_id = c_id。

ターゲットトランスフォーメーション

ターゲットトランスフォーメーションは、データを Amazon S3 の新しいファイルに書き込みます。受信フィールドを設定して、結合の結果として生じる重複フィールドを除外できます。

次の表に、出力データの一部を示します。

Order_id	order_date	order_price	item_id	qty	price	c_id	c_name	c_dob
1005	2023-01-20	78.25	5063	2	34.99	789	Kelcy Almeida	1969-07-20
1006	2023-01-24	150.09	2389	3	19.99	268	Chidi Donalds	1972-12-07
1006	2023-01-24	150.09	5063	1	34.99	268	Chidi Donalds	1972-12-07
1007	2023-02-07	30.20	9871	2	10.99	268	Chidi Donalds	1972-12-07

第 18 章

ラベラトランスフォーメーション

ラベラトランスフォーメーションによって、Data Quality で作成されたラベラアセットをマッピングに追加します。

ラベラアセットによって、入力フィールドの情報のタイプを評価し、検出されたそれぞれのタイプのデータにラベルを割り当てる一連の操作を定義します。

トランスフォーメーションを設定する場合は、ラベラアセットにリンクするターゲットフィールドに入力フィールドをマッピングします。マッピングを実行すると、トランスフォーメーションでは、アセットで定義されたラベリング条件に一致する入力フィールドの値が検索されます。トランスフォーメーションは、ラベルリング操作の結果を 2 つの出力フィールドに書き込みます。出力フィールドの詳細については、「ラベラトランスフォーメーションの出力フィールド」 (ページ 276)を参照してください。

ラベラトランスフォーメーションは、別の場所で設計したデータトランスフォーメーションロジックをマッピングに追加できるという点において、マップレットトランスフォーメーションと類似しています。マップレットと同様に、ラベラアセットは再利用可能なアセットです。

ラベラートランスフォーメーションでは、受信フィールドと発信フィールドが表示されます。ラベラアセットに含まれるロジックは表示されず、ラベラアセットを編集することもできません。ラベラアセットを編集するには、Data Quality でラベラアセットを開きます。

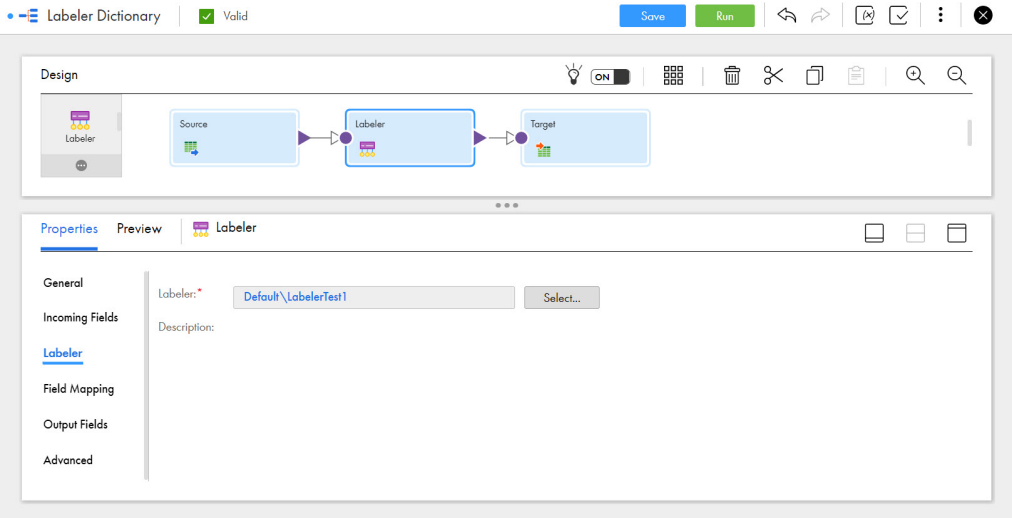
ラベラトランスフォーメーションの設定

マッピングのラベラトランスフォーメーションを設定する場合は、最初にトランスフォーメーションに含めるロジックが含まれているラベラアセットを選択します。次に、ラベラアセットで指定されたターゲットフィールドにトランスフォーメーションの受信フィールドをマッピングします。

トランスフォーメーションを設定するには、以下のタスクを実行します。

1. ラベラトランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. **[ラベラ]** タブで、トランスフォーメーションに含めるラベラアセットを選択します。

次の図に、ラベラアセットを選択するために使用するオプションを示します。



3. **［受信フィールド］**タブで、受信フィールドを検証します。

   デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。

4. **［フィールドマッピング］**タブで、アップストリームオブジェクトのフィールドをターゲットフィールドに接続します。

   ラベラアセットのフィールド名には、入力フィールドの名前が反映されている場合があります。この場合、**［自動マップ］**オプションを使用してフィールドの接続を行うことができます。

5. **［出力フィールド］**タブの出力フィールドのプロパティを確認します。

   トランスフォーメーションの出力フィールドの詳細については、「ラベラトランスフォーメーションの出力フィールド」 (ページ 276)を参照してください。

6. **［全般］**タブでは、必要に応じて、ラベラトランスフォーメーションの名前の変更や説明の追加ができます。

データ品質アセットの同期

アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。

アセットのバージョンを同期するには、マッピングでトランスフォーメーションを開き、［プロパティ］パネルでトランスフォーメーショ名を選択します。例えば、クレンジングトランスフォーメーションの［プロパティ］パネルで**［クレンジング］**を選択します。同期が必要な場合は、アセットを同期するように求めるメッセージがデータ統合に表示されます。

ラベラトランスフォーメーションのフィールドマッピング

フィールドマッピングを設定して、データをアップストリームトランスフォーメーションからラベラトランスフォーメーションに移動する方法を定義します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

注: 受信フィールドをラベラアセットの単一のフィールドにマッピングします。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドがあれば、そのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。
トランスフォーメーションのラベラアセットがパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、［すべてパラメータを使用します］オプションを選択します。
- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。［すべてパラメータを使用します］または［部分的にパラメータを使用します］をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは*フィールドマッピング*型である必要があります。

1 つのマッピング内の複数のラベラトランスフォーメーションで同じフィールドマッピングパラメータを使用しないでください。

オプション

［受信フィールド］リストと**［ターゲットフィールド］**リストでフィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。［手動］または［部分的にパラメータを使用します］フィールドマップオプションを選択すると、［自動マップ］オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。

- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合では Cust_Name フィールドと Customer_Name フィールドが自動的にリンクされます。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似する名前のフィールドをマッピングできます。

［自動マップ］ > **［自動マップを取り消す］**をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで**［アクション］** > **［マップ解除］**をクリックします。選択した 1 つ以上のフィールドのマッピングを解除するには、［ターゲットフィールド］コンテキストメニューで**［選択項目をマップ解除］**をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、［ターゲットフィールド］コンテキストメニューで**［マッピングのクリア］**をクリックします。

ラベラトランスフォーメーションの出力フィールド

ラベラトランスフォーメーションは、ラベル適用操作の結果を含む 1 つ以上の出力フィールドを作成します。フィールドの数は、トランスフォーメーションのラベラアセットがトークンラベル適用操作と文字ラベル適用操作のどちらを指定するのかによって異なります。出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。タブには、出力フィールドの名前、タイプ、精度、およびスケールが表示されます。

トークンラベル適用の出力

ラベラトランスフォーメーションでは、トークンラベル適用モードで次の出力フィールドを作成します。

LabeledOutput

ラベリング条件に一致する値が、アセットで指定されたラベルに置き換えられた入力フィールドデータのコピー。

TokenizedData

入力フィールドデータのコピー。ラベラアセットがディクショナリを読み取る場合、トランスフォーメーションによって、ディクショナリ値に一致する入力値がディクショナリの代替値に置き換えられる場合があります。Data Quality ユーザーは、入力値を代替値に置き換えるようにラベラアセットを設定できます。

トランスフォーメーションは、ラベル適用操作で識別されない入力フィールドの値を各出力フィールドにコピーします。

文字ラベル適用の出力

ラベラトランスフォーメーションでは、文字ラベル適用モードで次の出力フィールドを作成します。

LabeledOutput

ラベル適用条件に一致する文字が、アセットで指定されたラベルに置き換えられた入力フィールドデータのコピー。

このフィールドには、文字ラベルと、ラベルが適用されない入力データの文字を含めることができます。

第 19 章

ルックアップトランスフォーメーション

指定されたルックアップ条件に基づいてデータを取得するには、ルックアップトランスフォーメーションを使用します。例えば、ルックアップトランスフォーメーションを使用すると、ソースデータで使用されているコードの値をデータベーステーブルから取得できます。

マッピングタスクに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、タスクはルックアップフィールドとルックアップ条件に基づいてルックアップソースを照会します。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。単一の行または複数の行を返すように、ルックアップトランスフォーメーションを設定することができます。単一の行を返す用に設定されたルックアップ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。複数の行を返す用に設定されたルックアップ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します。

- 関連する値を取得する。ソースの値に基づいてルックアップテーブルから値を取得します。たとえば、ソースに従業員 ID があるものとします。ルックアップテーブルから従業員名を取得します。
- 複数の値を取得する。ルックアップテーブルから複数の行を取得します。例えば、部門内のすべての従業員を返します。
- 緩やかに変化する次元テーブルを更新する。行がターゲットに存在するかどうかを判断します。

実行できるルックアップのタイプは次のとおりです。

コネクトされたルックアップまたはコネクトされていないルックアップ

接続されたルックアップトランスフォーメーションは、ソースデータを受け取り、ルックアップを実行し、データを返します。

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は、ソースにもターゲットにも接続されません。トランスフォーメーションは、ルックアップ式を使用して、ルックアップトランスフォーメーションを呼び出します。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は、呼び出し元のトランスフォーメーションに 1 つのカラムを返します。

キャッシュを使用するルックアップおよびキャッシュを使用しないルックアップ

パフォーマンスを最適化するため、ルックアップソースをキャッシュします。ルックアップソースをキャッシュする場合、静的キャッシュまたは動的キャッシュを使用できます。また、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。

デフォルトでは、ルックアップキャッシュは静的なままであり、マッピングタスクが実行されても変更されません。動的キャッシュの場合、ターゲットテーブルが変更されるときにキャッシュの行が挿入または更新されます。ターゲットテーブルをルックアップソースとしてキャッシュすると、キャッシュ内の値をルックアップして値がターゲットに存在するかどうかを調べることができます。

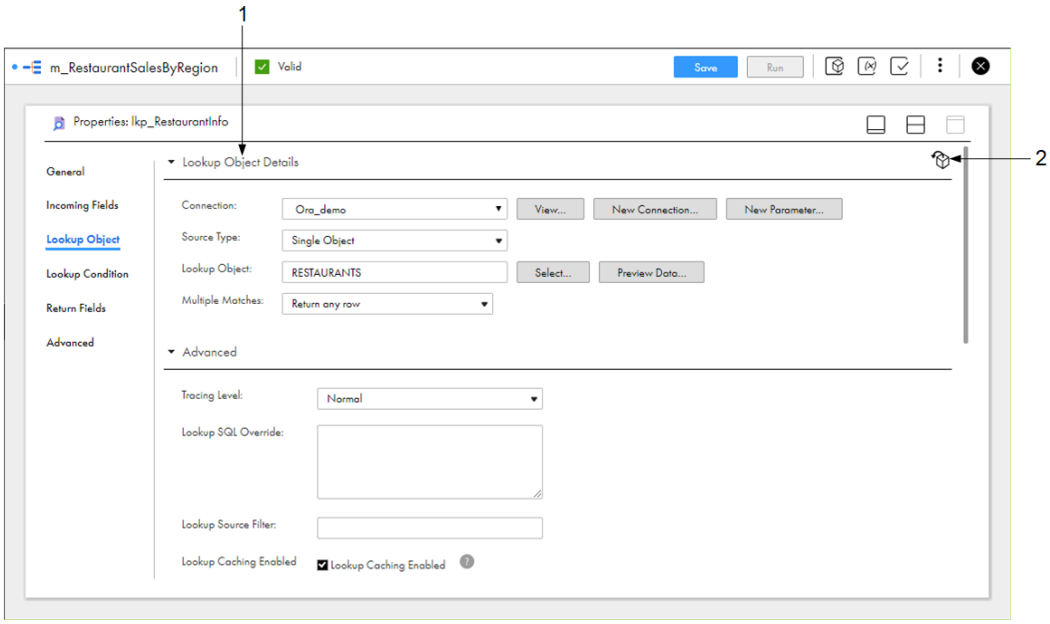
デフォルトでは、ルックアップキャッシュは非永続キャッシュでもあります。したがって、データ統合によってマッピングタスクの完了後にキャッシュファイルは削除されます。マッピングの実行間でルックアップテーブルが変更されない場合は、永続キャッシュを使用するとパフォーマンスを向上できます。

ルックアップオブジェクト

ルックアップオブジェクトは、データ統合がルックアップの実行時に照会するソースオブジェクトです。ルックアップオブジェクトは、ルックアップソースとも呼ばれます。

［プロパティ］パネルの**［ルックアップオブジェクト］**タブで、ルックアップソースを選択します。ルックアップソースに設定するプロパティは、接続タイプに応じて異なります

次の図に、リレーショナルルックアップの**［ルックアップオブジェクト］**タブを示します。



1. 接続、ソースタイプ、ルックアップオブジェクト、および複数一致の動作を設定するルックアップオブジェクトの詳細。
2. マッピングインベントリからルックアップソースを選択します。

以下の方法でルックアップソースを選択できます。

接続とルックアップオブジェクトを選択する。

［ルックアップオブジェクトの詳細］領域で、接続、ソースタイプ、およびルックアップオブジェクトを選択します。また、新しい接続を作成することもできます。

マッピングインベントリからルックアップソースを選択します。

組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されており、**［データカタログ］**ページでオブジェクトをマッピングに追加した場合、**［インベントリ］**パネルからルックアップソースを選択できます。組織の管理者によって Enterprise Data Catalog 統合プロパティが設定されていない、またはデータカタログの検出が実行されていない場合、**［インベントリ］**パネルには何も表示されません。データカタログの検出に関する詳細については、「*マッピング*」を参照してください。

パラメータを使用する。

マッピングタスクを実行するときは、入力パラメータを使用して接続やルックアップオブジェクトを定義できます。パラメータの詳細については、「*マッピング*」を参照してください。

カスタムクエリを使用する。。

カスタムクエリを使用して、クエリするカラムの数を減らすことができます。ソースオブジェクトのサイズが大きい場合は、カスタムクエリを使用することをお勧めします。

また、ルックアップで複数一致が返される場合は、トランスフォーメーションの動作も指定する必要があります。

ルックアップオブジェクトのプロパティ

ルックアップを設定するときには、ルックアップ接続とルックアップオブジェクトを選択します。ルックアップ条件で複数の一致が返されるときの動作も定義します。

次の表に、ルックアップオブジェクトのプロパティを示します。

プロパティ	説明
接続	ルックアップ接続の名前。
ソースタイプ	ソースタイプ。データベースルックアップの場合、ソースタイプには単一のパラメータまたはクエリを指定できます。フラットファイルルックアップの場合、ソースタイプには、単一のオブジェクト、ファイルリスト、コマンド、またはパラメータを指定できます。
ルックアップオブジェクト	ソースタイプが単一のオブジェクトの場合、このプロパティには、ルックアップファイル、テーブル、またはオブジェクトを指定します。 ソースプロパティがファイルリストの場合、このプロパティには、ファイルリストを含むテキストファイルを指定します。 ソースタイプがコマンドの場合、このプロパティにより、データ統合が戻りフィールドをインポートするサンプルファイルを指定します。
パラメータ	ソースタイプがパラメータの場合、このプロパティには、パラメータを指定します。
クエリの定義	ソースタイプがクエリの場合、**[カスタムクエリの編集]** ダイアログボックスを表示します。有効なカスタムクエリを入力して、[OK] をクリックします。
複数一致	ルックアップ条件で複数の一致が返されるときの動作。すべての行、任意の行、最初の行、最後の行、またはエラーを返すことができます。 すべての行を選択して複数の一致がある場合、ルックアップ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になります。任意の行、最初の行、または最後の行を選択して、複数の一致がある場合、ルックアップ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になります。

プロパティ	説明
形式オプション	ファイル形式のオプションであり、ルックアップオブジェクトがフラットファイルである場合に適用可能です。 **［形式オプション］**ダイアログボックスを開き、ファイルの形式を定義します。次のファイル形式オプションを設定します。 - 区切り文字。区切り文字。 - テキスト修飾子。テキストを修飾する文字。 - エスケープ文字。エスケープ文字。 - フィールドラベル。タスクがフィールドラベルを生成するのか、ソースファイルからラベルをインポートするのかを指定します。 - 最初のデータ行。データの最初の行。タスクは、入力した行番号で読み取りを開始します。
コマンド	ソースタイプがコマンドの場合、このプロパティには、ソースファイルリストを生成するコマンドを指定します。

ファイルリストとコマンドの詳細については、「ファイルリスト」 (ページ 44)を参照してください。パラメータとファイル形式の詳細については、「*マッピング*」を参照してください。

複数一致ポリシーの制限

ルックアップの設定時に、ルックアップ条件が複数一致を返したときの動作を定義します。一部のタイプのルックアップには、複数一致ポリシーに制限があります。

次のタイプのルックアップには、複数一致ポリシーの制限があります。

キャッシュを使用しないルックアップ

一部のコネクタタイプは、キャッシュされていないルックアップでの複数一致ポリシー**［最初の行を返す］**および**［最後の行を返す］**をサポートしていません。これらのポリシーのいずれかを選択し、コネクタでキャッシュされていないルックアップのポリシーがサポートされていない場合、データ統合では詳細プロパティ**［ルックアップキャッシュを有効にする］**が有効になり、編集することはできません。

動的キャッシュを使用するルックアップ

ルックアップトランスフォーメーションが動的キャッシュを使用した場合にエラーを返すように複数一致ポリシーを設定する必要があります。他の複数一致ポリシーはサポートされていません。

Salesforce **ルックアップ**

Salesforce オブジェクトに対してルックアップを実行した場合に、任意の行を返すか、エラーを返すようにすることができます。

詳細モードのルックアップ

詳細モードでルックアップトランスフォーメーションを使用した場合に、すべての行または任意の行を返すか、エラーを返すようにすることができます。複数一致ポリシー**［最初の行を返す］**および**［最後の行を返す］**はサポートされていません。

詳細モードで複数の一致がある場合にエラーを返す動作を定義すると、ルックアップトランスフォーメーションによって重複行が削除され、重複行はログファイルに含まれません。

さまざまなコネクタでサポートされている複数一致ポリシーの詳細については、該当するコネクタのヘルプを参照してください。

カスタムクエリ

データベースルックアップ用のカスタムクエリを作成できます。カスタムクエリを作成することで、クエリを実行するカラムの数を減らすことができます。

カスタムクエリをルックアップソースとして使用するには、ソースタイプとして**［クエリ］**を選択し、クエリを定義します。クエリを定義する場合は、SQL SELECT 文を入力して、使用するソースカラムを選択します。データ統合は、SQL 文を使用してソースカラムの情報を取得します。

ルックアップトランスフォーメーションでカスタムクエリを使用する場合は、SQL 文に次の形式を使用します。

- リレーショナルデータベース接続の場合は、次の例のように SQL 文の各カラムにエイリアスを使用します。

  ```
  SELECT COL1 AS COL1, COL2 AS COL2, COL3 AS COL3 from TABLE_NAME
  ```
- その他のタイプのデータベース接続の場合は、ソースデータベースに有効な SQL を使用します。クエリではデータベース固有の関数を使用できます。

カスタムクエリをルックアップソースとして使用するには、ルックアップキャッシュを有効にする必要があります。

ヒント: カスタムクエリを作成する前に、ソースデータベースで使用する SQL 文をテストします。データ統合では、無効な SQL 文に特定のエラーメッセージが表示されることはありません。

ルックアップ条件

ルックアップ条件では、ルックアップによってルックアップオブジェクトから値をいつ返すかを定義します。ルックアップ条件を設定するときは、データフローの 1 つ以上のフィールドの値をルックアップオブジェクトの値と比較します。

ルックアップ条件には、データフローの受信フィールド、ルックアップオブジェクトのフィールド、演算子が含まれます。フラットファイルとデータベース接続については、ルックアップ条件で次の演算子を使用できます。

=（等しい）

<（より小さい）

>（より大きい）

<=（以下）

>=（以上）

!=（等しくない）

その他の接続の場合、および動的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場合、ルックアップ条件で=（等しい）演算子を使用できます。

ルックアップ条件については、次の情報に注意してください。

- 複数の条件を入力すると、マッピングタスクにより、AND 論理演算子が使用されてルックアップ条件が評価されて、条件が結合されます。返されるのは、すべてのルックアップ条件と一致する行です。
- 複数の条件を含めるときは、パフォーマンスを最適化するため、次の順序で条件を入力してください。

  1. =（等しい）
  2. <（より小さい）、<=（以下）、>（より大きい）、>=（以上）
  3. !=（等しくない）

- ルックアップ条件では NULL 値が照合されます。入力フィールドが NULL であるとき、マッピングタスクでは、ルックアップの NULL 値と等しい NULL が評価されます。
- 詳細モードでは、ルックアップ条件にバイナリデータ型が含まれている場合、マッピングは無効になります。

ルックアップの戻りフィールド

［プロパティ］パネルの **［戻りフィールド］** タブで、ルックアップオブジェクトから返すフィールドを選択します。

［戻りフィールド］ タブには、選択したルックアップオブジェクトのすべてのフィールドが表示されます。デフォルトでは、データフローのリストのすべてのフィールドがマッピングに含まれます。使用しないフィールドを削除してください。

動的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場合、タスクは NewLookupRow 戻りフィールドを返します。このフィールドは削除できません。NewLookupRow 戻りフィールドの詳細については、「動的キャッシュの更新」 (ページ 291)を参照してください。

フィールドの名前を編集し、フィールドのメタデータを編集できます。フィールドメタデータを編集すると、名前、ネイティブデータ型、ネイティブ精度、ネイティブスケールを変更できます。

一部のリレーショナル接続タイプでは、ルックアップの戻りフィールドのデフォルトのカラム値を指定できます。デフォルト値には文字列または式を使用できます。文字列を使用する場合は、'ABC'のように文字列を一重引用符で囲みます。ルックアップの戻りフィールドのデフォルト値を設定できるかどうかを確認するには、該当するコネクタのヘルプを参照してください。

注: フィールドメタデータを変更した場合、変更を自動的に元に戻すことはできません。タスクの実行時にエラーを引き起こす可能性のある変更は行わないようにしてください。

フィールドをフィールドリストに追加できるのは、そのフィールドがルックアップオブジェクトに存在する場合です。フィールドを追加するには、フィールド名、データ型、精度、スケールなど、正確なフィールドの詳細が必要です。

ルックアップオブジェクトの元のフィールドをリストアするには、同期アイコンを使用します。同期により、削除されたフィールドがリストアされて新しいフィールドが追加され、追加されたフィールドのうちルックアップオブジェクトで対応するフィールドがあるものが保持されます。同期すると、追加されたフィールドのうち、ルックアップオブジェクトで対応するフィールドがないものは削除されます。同期化を行ってもフィールドメタデータへのローカルな変更は元に戻りません。

次の表に、**［戻りフィールド］** タブで使用できるオプションを示します。

フィールドオプション	説明
フィールドの追加アイコン	選択したルックアップオブジェクトからフィールドを追加します。リストに表示されないオブジェクトからフィールドを取得する際に使用します。 **［新しいフィールド］** ダイアログボックスが開きます。正しいフィールド名、データ型、精度、スケールを入力して［OK］をクリックします。
削除アイコン	データフローからフィールドを削除して、リストからフィールドを削除します。
ソートアイコン	ネイティブ順、昇順、または降順にフィールドをソートします。

フィールドオプション	説明
［検索］フィールド	検索文字列を入力して、その文字列を含む名前のフィールドを検索します。
［オプション］メニュー	以下のオプションが含まれます。 - ［フィールドの技術名を使用］。フィールドの技術名でフィールド名を表示します。 - ［ラベルを使用］。ラベルでフィールド名を表示します。 - ［メタデータの編集］。名前、ネイティブタイプ、ネイティブ精度、またはネイティブスケールを変更します（データ型に該当する場合）。一部のリレーショナル接続タイプでは、ルックアップの戻りフィールドのデフォルト値を設定できます。 メタデータを編集する場合は、フィールドの技術名またはラベルでネイティブ名を表示することができます。
同期アイコン	フィールドのリストをルックアップオブジェクトと同期します。 **注**: このオプションを選択した場合、戻りフィールドのメタデータに加えた変更はすべて失われます。
比較で無視する	トランスフォーメーションで動的キャッシュが使用されている場合、デフォルトでは、データ統合は関連する受信フィールドの値とすべてのルックアップフィールドの値を比較することで、ルックアップキャッシュ内の行を更新するかどうかを決定します。 データ統合が行の更新前に値を比較する際にフィールドを無視するようにする場合は、このプロパティを有効にします。トランスフォーメーションでは、少なくとも 1 つ以上のフィールドを比較する必要があります。 このプロパティは、ルックアップトランスフォーメーションで動的キャッシュが使用される場合に各フィールドに表示されます。
更新に対する NULL 入力を無視する	有効にすると、関連付けられた受信フィールドの値に NULL 値が含まれる場合でも、データ統合によってキャッシュ内のカラムは更新されません。 ルックアップ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、このプロパティは NewLookupRow フィールドを除く各フィールドに表示されます。 デフォルトでは無効になっています。
実行時に既存のフィールドを保持	マッピングの保存後にフィールドメタデータが変更された場合、データ統合では、マッピングの実行時に更新されたフィールドメタデータが使用されます。通常、これは適切な動作です。ただし、マッピングでネイティブフラットファイル接続を使用しており、設計時に使用したメタデータを保持する場合は、**［実行時に既存フィールドを保持］**オプションを有効にします。このオプションを有効にすると、データ統合のマッピングタスクでは、マッピングの作成時に使用したフィールドメタデータが使用されます。

詳細プロパティ

ルックアップトランスフォーメーションに対して詳細プロパティを設定できます。ルックアップトランスフォーメーションで使用できる詳細プロパティは、接続タイプとマッピングタイプで決まります。

以下のプロパティを設定できます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
ルックアップソースファイルディレクトリ	フラットファイルルックアップソースのディレクトリの名前。デフォルトでは、データ統合は、ルックアップソース接続ディレクトリからファイルを読み取ります。 入力パラメータを使用して、ソースファイルディレクトリを指定することもできます。 サービスプロセス変数ディレクトリ$PMLookupFileDir を使用した場合、タスクはシステム変数への設定済みパスにターゲットファイルを書き込みます。システム変数の設定済みパスを見つけるには、次のディレクトリにある pmrdtm.cfg ファイルを参照してください。 <Secure Agent installation directory>\apps\Data_Integration_Server\<Data Integration Server version>\ICS\main\bin\rdtm また、$PMLookupFileDir 変数への設定済みパスは、Administrator のデータ統合サーバーシステム設定の詳細にあります。
ルックアップソースファイル名	ファイル名、またはルックアップソースファイルのファイル名とパス。
ルックアップ SQL のオーバーライド	ルックアップテーブルにクエリを実行するデフォルトの SQL 文をオーバーライドします。ルックアップ値のクエリに使用する SQL 文を指定します。 ルックアップキャッシュを有効にして使用します。
ルックアップソースフィルタ	ルックアップトランスフォーメーションの任意のフィールドのデータ値に基づいてルックアップを制限します。 ルックアップキャッシュを有効にして使用します。
ルックアップキャッシュを有効にする	ランタイムセッション中にルックアップデータをキャッシュするかどうかを決定します。キャッシュを有効にすると、データ統合はルックアップソースを一度クエリし、セッション中に使用する値をキャッシュします。これにより、パフォーマンスを向上できます。キャッシュを無効にすると、トランスフォーメーションに行が渡されるたびに、SELECT 文によってルックアップ値が取得されます。 キャッシュが有効になっている場合、次の状況では編集できません。 - キャッシュされていないルックアップをルックアップソースタイプがサポートしていない場合。 - 複数一致ポリシーを選択していても、キャッシュされていないルックアップのポリシーをルックアップソースタイプがサポートしていない場合。例えば、Amazon Redshift V2 ソースに対するルックアップの複数一致ポリシーとして **[最初の行を返す]** または **[最後の行を返す]** を選択した場合、キャッシュを無効にすることはできません。 デフォルトでは有効になっています。 フラットファイルルックアップは常にキャッシュされるため、このプロパティはフラットファイルルックアップには表示されません。
ルックアップキャッシュのディレクトリ名	[ルックアップキャッシュを有効にする] を選択した場合にキャッシュされたルックアップデータを保存するディレクトリを指定します。 ディレクトリ名には環境変数を指定できます。

プロパティ	説明
ルックアップキャッシュの永続性	ルックアップキャッシュファイルを保存して、キャッシュを使用するように設定されたルックアップトランスフォーメーションをデータ統合が次に処理するときにそのキャッシュを再利用するかどうかを指定します。
キャッシュファイル名のプレフィックス	永続ルックアップキャッシュで使用します。永続ルックアップキャッシュファイルに使用するファイル名の接頭語を指定します。データ統合は、ディスクに保存する永続キャッシュファイルの名前としてファイル名の接頭語を使用します。 名前付き永続キャッシュファイルが存在する場合、データ統合はそれらのファイルからメモリキャッシュを構築します。名前付き永続キャッシュファイルが存在しない場合、データ統合は永続キャッシュファイルを再構築します。 接頭語を入力します。.idx や.dat などのファイル拡張子は含めません。
ルックアップソースからの再キャッシュ	永続ルックアップキャッシュで使用します。選択すると、データ統合は、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を最初に呼び出したときにルックアップソースから永続ルックアップキャッシュを再構築します。
データキャッシュサイズ	トランスフォーメーションのデータキャッシュサイズ。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
インデックスキャッシュサイズ	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。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
動的ルックアップキャッシュ	静的キャッシュの代わりに動的キャッシュを使用するかどうかを決定します。動的キャッシュを有効にすると、キャッシュとターゲットの同期を維持するようにキャッシュがターゲット内に行を挿入または更新するため、タスクはキャッシュを更新します。 ルックアップキャッシュが有効の場合に使用します。
更新時に古い値を出力	このプロパティを有効にすると、タスクがキャッシュ内の行を更新したときに、行の更新前にルックアップキャッシュに存在していた値が出力されます。行を挿入すると、NULL 値が返されます。 動的ルックアップキャッシュが有効の場合に使用します。
動的キャッシュの同期	このプロパティを有効にすると、タスクはルックアップソースから最新の値を取得し、動的キャッシュを更新します。これは、同じルックアップソースを使用する複数のタスクを同時に実行している場合に役に立ちます。 動的ルックアップキャッシュが有効の場合に使用します。 一部の接続タイプでは、キャッシュ同期を使用できません。詳細については、該当するコネクタのヘルプを参照してください。
挿入でなければ更新	行のタイプが「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。有効にすると、マッピングタスクはキャッシュの行を挿入して既存の行を更新します。無効にすると、マッピングタスクは既存の行を更新しません。 動的ルックアップキャッシュが有効な場合に使用します。

プロパティ	説明
ルックアップソースは静的です	このプロパティを有効にすると、タスクの実行時にルックアップソースは変更されません。
日時形式	日時形式とフィールド幅を設定します。ミリ秒、マイクロ秒、ナノ秒の形式のフィールド幅は 29 です。ここで日時形式を指定しない場合、フィールドへの入力に任意の日時形式を使用できます。デフォルトは、「YYYY-MM-DD HH24:MI:SS」です。形式を設定しても、フィールドのサイズは変わりません。 デフォルトは、「YYYY-MM-DD HH24:MI:SS」です。日時形式を設定しても、フィールドのサイズは変わりません。
桁区切り記号	桁区切り記号を指定します。カンマ（,）、ピリオド（.）または［なし］を指定します。 デフォルトは［なし］です。
小数点記号	小数点記号を指定します。カンマ（,）またはピリオド（.）を入力します。 デフォルトはピリオドです。
大文字小文字を区別した文字列比較	フラットファイルの文字列カラムで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文字列比較を有効にするかどうかを決定します。キャッシュを使用しないリレーショナルルックアップの場合、大文字小文字を区別した比較をサポートするカラムタイプはデータベースによって異なります。 詳細モードのルックアップでは、大文字小文字の区別は自動的に有効になります。
NULL の順序付け	NULL 値の順序付けを決定します。NULL 値を上位にするか下位にするかを選択できます。デフォルトでは、NULL 値は上位でソートされます。これは、比較演算子における NULL 値の扱い（上位、下位、または NULL）の設定より優先されます。リレーショナルルックアップでは、NULL の順序はデータベースのデフォルト値によって決まります。
ソート済み入力	ルックアップファイルのデータがソートされているかどうかを示します。このオプションを選択すると、ファイルのルックアップパフォーマンスが向上します。ソート済み入力を有効にした場合に条件カラムがグループ化されていないと、セッションは失敗します。条件カラムがグループ化されているがソートはされていない場合、ルックアップはソート済み入力が設定されていない場合と同様に処理されます。
ルックアップキャッシュの事前作成	ルックアップトランスフォーメーションにデータが渡る前に、ルックアップキャッシュを作成できるようにします。同時に複数のルックアップキャッシュファイルを作成してパフォーマンスを向上できます。

プロパティ	説明
サブ秒の精度	日時フィールドにサブ秒の精度を設定します。リレーショナルルックアップの場合、日時データの位取りを編集できるデータベースについては、精度を変更できます。サブ秒の精度を変更できるデータ型は、Oracle Timestamp、Informix Datetime、および Teradata Timestamp です。 0 から 9 までの正の整数値を入力します。デフォルトは 6 マイクロ秒です。 タスクでプッシュダウンの最適化を有効にすると、データベースはサブ秒の精度の設定に関係なく、完全な日時値を返します。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

ルックアップ SQL オーバーライド

マッピングに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、マッピングタスクでは、ルックアップトランスフォーメーションで設定するフィールドおよびプロパティに基づいてルックアップオブジェクトを照会します。マッピングタスクは、データの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。ルックアップトランスフォーメーションでリレーショナルルックアップを実行すると、デフォルトのクエリをオーバーライドできます。

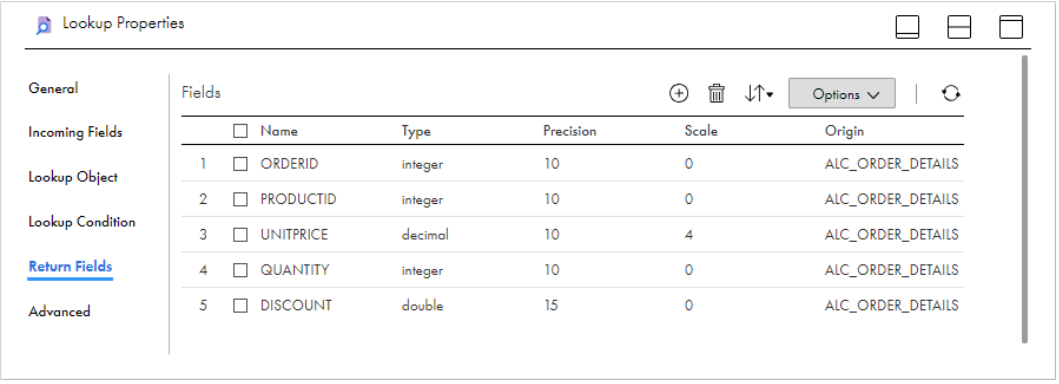
デフォルトのクエリには、マッピング内のすべてのルックアップフィールドを含む SELECT 文が含まれています。また、SELECT 文には、すべてのカラムをルックアップトランスフォーメーションの表示と同じ順序で並べた ORDER BY 句も含まれています。デフォルトのクエリを表示するには、マッピングタスクを実行します。デフォルトのクエリがログファイルに表示されます。

ORDER BY 句を変更する場合は、WHERE 句を追加するかルックアップデータをキャッシュする前に変換すると、デフォルトのクエリをオーバーライドできます。例えば、データベース関数を使用して、マッピングで使用されるフィールドのデータ型およびフォーマットと一致するようにルックアップテーブルのデータ型またはフォーマットを調整できます。または、デフォルトのクエリをオーバーライドして複数のテーブルを照会できます。

デフォルトのクエリは、ルックアップトランスフォーメーションの **[詳細]** タブでオーバーライドします。**[ルックアップ SQL オーバーライド]** フィールドに、SELECT 文をすべて入力します。クエリの各カラムにはエイリアスを使用します。ORDER BY 句を変更する場合は、クエリの末尾に「--」を追加してマッピングタスクで生成する ORDER BY 句を抑止する必要があります。

例

ルックアップトランスフォーメーションは、Microsoft SQL Server のテーブル ALC_ORDER_DETAILS から次のフィールドを返します。



トランスフォーメーションは、以下のルックアップ条件を使用します。

```
ORDERID=in_ORDERID
```

マッピングタスクを実行すると、ログファイルに次のクエリが表示されます。

```
LKPDP_1> DBG_21097 [2018-11-07 14:11:33.509] Lookup Transformation [lkp_ALC_ORDER_DETAILS]: Default sql to create lookup cache: SELECT PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT,ORDERID FROM "icsauto"."ALC_ORDER_DETAILS" ORDER BY ORDERID,PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT
```

ORDER BY 句をオーバーライドして PRODUCTID の順に並べ替えるには、**[詳細]** タブの **[ルックアップ SQL オーバーライド]** フィールドに次のクエリを入力します。

```
SELECT PRODUCTID AS PRODUCTID, UNITPRICE AS UNITPRICE, QUANTITY AS QUANTITY, DISCOUNT AS DISCOUNT, ORDERID AS ORDERID FROM "icsauto"."ALC_ORDER_DETAILS" ORDER BY PRODUCTID --
```

マッピングタスクを再度実行すると、ログファイルに次のクエリが表示されます。

```
LKPDP_1> DBG_21312 [2018-11-07 14:14:36.734] Lookup Transformation [lkp_ALC_ORDER_DETAILS]: Lookup override sql to create cache: SELECT PRODUCTID AS PRODUCTID, UNITPRICE AS UNITPRICE, QUANTITY AS QUANTITY, DISCOUNT AS DISCOUNT, ORDERID AS ORDERID FROM "icsauto"."ALC_ORDER_DETAILS" ORDER BY PRODUCTID -- ORDER BY ORDERID,PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT
```

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オーバーライドするときに、一定のルールとガイドラインが適用されます。

次のガイドラインを使用します。

- リレーショナルルックアップの場合は、ルックアップ SQL クエリをオーバーライドできます。
- ルックアップクエリをオーバーライドする場合は、トランスフォーメーションのルックアップキャッシュも有効にする必要があります。
- データベースで必要とされる構文を使用して、SELECT 文をすべて入力します。
- データベース予約語はすべて引用符で囲んでください。
- SELECT 文のルックアップフィールドと戻りフィールドをすべて含めます。

  SELECT 文のフィールド数が増減すると、マッピングタスクに失敗します。
- クエリの各カラムにはエイリアスを使用します。

カラムのエイリアスを使用しないと、マッピングタスクに失敗して以下のエラーが表示されます。

```
Failed to initialize transformation [<Lookup Transformation Name>]
```

- ORDER BY 句をオーバーライドするには、クエリの末尾に「--」を追加します。

  オーバーライドで ORDER BY 句を入力しても、マッピングタスクは ORDER BY 句を生成します。そのため、クエリの末尾にダッシュを 2 つ（--）入力することで生成した ORDER BY 句を抑止する必要があります。
- ORDER BY 句に複数のカラムが含まれている場合は、ルックアップ条件のフィールドと同じ順序でカラムを入力します。
- マッピングタスクでプッシュダウンの最適化を使用する場合は、ORDER BY 句をオーバーライドしたり、生成した ORDER BY 句にコメント表示して抑止したり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。
- すべての行を返すルックアップトランスフォーメーションを設定する際に、マッピングタスクはソートされたキーを使用して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がルックアップのすべての行を取得する際に、マッピングタスクはソートされた順序でキーを使用してデータキャッシュを構築します。行がソートされていなければ、マッピングタスクはキャッシュからすべての行を取得できません。データがキーに対してソートされていない場合は、予期しない結果になる可能性があります。
- ルックアップ SQL オーバーライドにパラメータを含めることはできません。
- ルックアップ SQL オーバーライドとルックアップソースフィルタを同じトランスフォーメーションに設定する場合、マッピングタスクはフィルタを無視します。

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、マッピングタスク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できます。ルックアップソースフィルタを設定すると、マッピングタスク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。

ルックアップソースフィルタを設定するには、ルックアップトランスフォーメーションの **[詳細]** タブを開いて **[ルックアップソースフィルタ]** フィールドにフィルタを入力します。フィルタ条件には WHERE キーワードを含めないでください。

例えば、ID が 510 より大きい従業員の姓をすべて取得する必要があるとします。

ルックアップトランスフォーメーションの［EmployeeID］フィールドで次のルックアップソースフィルタを設定します。

```
EmployeeID >= 510
```

マッピングタスクでソース行を読み取るときに、EmployeeID の値が 510 より大きい場合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使用するマッピングタスク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、マッピングタスクは SQL オーバーライドを示すビューを作成します。マッピングタスク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。

注: ルックアップ SQL オーバーライドとルックアップソースフィルタを同じトランスフォーメーションで設定する場合、マッピングタスクはフィルタを無視します。

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使用してルックアップキャッシュがターゲットに同期し続けるようにします。

ルックアップキャッシュを有効にすると、マッピングタスクは、最初のルックアップ要求を処理するときにルックアップキャッシュをビルドします。キャッシュは、静的または動的にできます。キャッシュが静的の場合、ルックアップキャッシュ内のデータは、マッピングタスクの実行時に変更されません。タスクがキャッシュを複数回使用する場合、タスクは同じデータを使用します。キャッシュが動的の場合、タスクはタスク内のアクションに基づいてキャッシュを更新します。このためタスクがルックアップを複数回使用する場合、ダウンストリームトランスフォーメーションは更新されたデータを使用できます。

動的キャッシュはほぼすべてのタイプのルックアップソースに使用できます。フラットファイルまたは Salesforce ルックアップに動的キャッシュを使用することはできません。特定のタイプのルックアップソースで動的キャッシュを使用する方法に関する詳細については、該当するコネクタのヘルプを参照してください。

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、マッピングタスクは、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

キャッシュに行がない場合、マッピングタスクは行を挿入します。マッピングタスクは、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

キャッシュに行が存在する場合、マッピングタスクは行を更新します。マッピングタスクは、入力フィールドに基づいてキャッシュ内の行を更新します。マッピングタスクは、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

キャッシュに行があり変更がない場合、マッピングタスクは変更を行いません。マッピングタスクは、行に変更なしのフラグを設定します。

動的ルックアップトランスフォーメーションには、タスクがキャッシュ内の各行に行った変更を示す戻りフィールド、NewLookupRow が含まれます。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。

パラメータ化されたソース、ターゲット、またはルックアップを、動的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とともに使用することはできません。

静的ルックアップと動的ルックアップの比較

ソースに重複するプライベートキーが含まれている場合、静的キャッシュではなく動的キャッシュを使用する必要があります。また、ソースにパフォーマンスを最適化するための大きなデータテーブルが含まれている場合も、動的キャッシュを使用する必要があります。

データ統合は、ルックアップトランスフォーメーションを設定して静的または動的キャッシュを使用するかどうかに基づいて、異なるルックアップ条件で処理を行います。

次の表で、静的キャッシュを使用したルックアップトランスフォーメーションと動的キャッシュを使用したルックアップトランスフォーメーションを比較します。

静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
タスクの実行中は、キャッシュは変更されません。	タスクでは、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。
ルックアップには、フラットファイル、リレーショナルデータベース、および Salesforce のようなその他の接続タイプを使用できます。	フラットファイルまたは Salesforce 接続タイプは使用できません。
ルックアップ条件が True の場合、タスク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。 条件が true ではない場合、タスクはデフォルト値を返します。	ルックアップ条件が true の場合、タスクはキャッシュおよびターゲット内の行を更新するか、またはキャッシュを変更せずにそのままとします。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。 ルックアップ条件が true ではない場合、タスクはキャッシュおよびターゲットに行を挿入するか、または行タイプに基づいてキャッシュを変更せずにそのままとします。これは、キャッシュまたはターゲットテーブルに行がなかったことを示しています。

動的キャッシュの更新

マッピングタスクは、ルックアップクエリの結果および定義した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応じて、行を読み込む際にルックアップキャッシュを変更します。マッピングタスクは、実行するアクションを示す NewLookupRow 戻りフィールドに値を割り当てます。

以下の表に、有効な NewLookupRow 値を示します。

NewLookupRow 値	説明
0	マッピングタスクではキャッシュ内の行の更新、またはキャッシュへの行の挿入が行われません。
1	マッピングタスクはキャッシュに行を挿入します。
2	マッピングタスクはキャッシュ内の行を更新します。

ダウンストリームトランスフォーメーションで NewLookupRow 値を使用できます。

挿入行の挿入と更新

マッピングタスクが、挿入行のキャッシュへの挿入と更新を処理する方法を設定できます。行タイプが挿入の場合に動的ルックアップキャッシュの既存の行を更新するには、トランスフォーメーションの **[挿入でなければ更新]** の詳細プロパティを有効にします。

注: 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]** プロパティはマッピングタスク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、マッピングタスクは、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、マッピングタスクはデータキャッシュで行を更新します。

[挿入でなければ更新] を有効にせず、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、マッピングタスクは、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、マッピングタスク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**
無効 - 挿入のみ	○	-	変更なし	0
無効 - 挿入のみ	×	-	挿入	1
有効	○	はい	更新	2
有効	○	×	変更なし	0
有効	×	-	挿入	1

動的キャッシュおよびルックアップソースの同期

ルックアップトランスフォーメーションでは、ターゲットに渡した行を追跡するために、動的ルックアップキャッシュが保持されます。複数のタスクが同じターゲットを更新する場合は、各タスクのルックアップトランスフォーメーションを設定して、ターゲットではなく同じルックアップソースに動的ルックアップキャッシュを同期させることができます。

キャッシュをルックアップソースと同期させるには、ルックアップトランスフォーメーションの **[動的キャッシュの同期]** プロパティを有効にします。

キャッシュがルックアップソースと同期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すると、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ソース上でルックアップを実行します。ルックアップソース内にデータが存在しない場合、ルックアップトランスフォーメーションは動的ルックアップキャッシュを更新する前にルックアップソースに行を挿入します。

別のタスクで行が挿入された場合には、ルックアップソース内にデータが存在する可能性があります。ルックアップキャッシュをルックアップソースに同期させるために、タスクはルックアップソースから最新の値を取得します。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ソースから取得した値を動的ルックアップキャッシュに挿入します。

例えば、複数のタスクが同時に実行されているとします。各タスクは新しい製品名の製品番号を生成します。あるタスクが製品番号を生成した場合、他のタスクは同じ製品番号を使用してその製品を識別する必要があります。製品番号は 1 回生成され、ルックアップソースに挿入されます。その製品を含む行を別のタスクが処理する場合は、ルックアップソース内の製品番号を使用する必要があります。各タスクはルックアップソース上でルックアップを実行し、すでに生成されている製品番号を特定します。

キャッシュがルックアップソースと同期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すると、タスクは挿入行の動的ルックアップキャッシュ上でルックアップを実行します。動的ルックアップキャッシュ内にデータが存在しない場合、タスクはルックアップソース上でルックアップを実行します。次のいずれかのタスクを完了します。

- ルックアップソース内にデータが存在する場合、タスクはルックアップソースからのカラムがある動的ルックアップキャッシュに行を挿入します。ソース行でキャッシュを更新することはありません。

- データがルックアップソース内に存在しない場合、タスクはデータをルックアップソースに挿入し、行をキャッシュに挿入します。

ルックアップソースには、ルックアップキャッシュと同じフィールドがあります。カラムがルックアップトランスフォーメーションから射影されるか、フィールドがルックアップ条件の一部でない限り、タスクがルックアップキャッシュに行を挿入することはありません。

動的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、マッピングタスク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。タスク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。

フィールドマッピング

動的ルックアップキャッシュを使用するときは、**[フィールドマッピング]** タブで受信フィールドをルックアップキャッシュフィールドにマッピングします。**[フィールドマッピング]** タブは、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するときのみ使用できます。

動的キャッシュを使用するときはすべての受信フィールドをマッピングして、タスクの実行時にキャッシュを更新できるようにする必要があります。必要に応じて、ターゲットオブジェクトのフィールドに対して生成されたキーを作成する場合は、受信フィールドの代わりにシーケンス ID フィールドをマッピングできます。

生成されたキーフィールド

動的ルックアップキャッシュを設定すると、ターゲットオブジェクトのフィールド用に生成キーを作成できます。

ターゲットオブジェクトのフィールド用に生成されたキーを作成するには、シーケンス ID フィールドを **[フィールドマッピング]** タブのルックアップキャッシュフィールドにマッピングします。受信フィールドの代わりにシーケンス ID フィールドを、整数または Bigint データ型のキャッシュフィールドにマッピングできます。整数ルックアップフィールドの場合、生成キーの最大値は 2,147,483,647 です。Bigint ルックアップフィールドの場合、生成キーの最大値は 9,223,372,036,854,775,807 です。

シーケンス ID フィールドをマッピングすると、データ統合では、ルックアップキャッシュに行を挿入するときにキーが生成されます。

データ統合は、次のプロセスを使用してシーケンス ID を生成します。

1. データ統合が動的ルックアップキャッシュを作成するときは、動的ルックアップキャッシュ内のシーケンス ID のあるフィールドごとに値の範囲を追跡します。
2. データ統合がデータの行をキャッシュに挿入するときは、最大のシーケンス ID の値に 1 を加えることでフィールドのキーを生成します。

3. データ統合は、生成されるシーケンス ID の最大値に達すると、再び 1 から生成を開始します。そして、データ統合は、既存の最小値から 1 を引いた値に達するまで、各シーケンス ID に 1 を加え続けます。データ統合で一意のシーケンス ID 番号が不足すると、マッピングタスクは失敗します。

データ統合は、キャッシュに挿入する行ごとにシーケンス ID を生成します。

比較でのフィールドの無視

動的ルックアップキャッシュを使用する場合、データ統合が関連する受信フィールドの値とルックアップフィールドの値を比較する際にフィールドを無視するように設定できます。比較で一部のフィールドを無視すると、マッピング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を使用するマッピングを実行すると、デフォルトでは、データ統合は関連する受信フィールドの値とすべてのルックアップフィールドの値を比較します。データ統合はこれらの値を比較することで、ルックアップキャッシュ内の行を更新するかどうかを決定します。受信フィールドの値がルックアップフィールドの値と異なる場合、データ統合はキャッシュ内の行を更新します。

比較する必要がないフィールドがある場合は、フィールドを比較するときにデータ統合に無視させるフィールドを選択できます。例えば、更新する必要のあるデータが行に含まれているかどうかを示すカラムがソースデータに含まれているとします。キャッシュおよびターゲットテーブル内の行を更新するかどうかを示しているフィールド以外のすべてのルックアップフィールドについて**［比較で無視する］**プロパティを有効にします。

フィールドを無視するための設定は、ルックアップトランスフォーメーションの**［戻りフィールド］**タブで行います。フィールドを無視するには、そのフィールドについて**［比較で無視する］**プロパティを有効にします。

トランスフォーメーションでは、少なくとも 1 つ以上のフィールドを比較する必要があります。

動的ルックアップクエリのオーバーライド

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、Lookup 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、マッピングタスクはルックアップ 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。

マッピングタスク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、マッピングタスクは 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。マッピングタスク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、マッピングタスク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。

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
```

永続ルックアップキャッシュ

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できます。永続キャッシュを使用すると、データ統合ではキャッシュファイルが保存され、マッピングの実行ごとに再利用されます。

ルックアップトランスフォーメーションでキャッシュを有効にした場合、データ統合ではデフォルトで非永続キャッシュが使用されます。非永続キャッシュを使用すると、データ統合では、マッピングの実行終了時にキャッシュファイルが削除されます。次にマッピングを実行したときに、データ統合はデータベースからメモリキャッシュを作成します。

マッピングの実行間でルックアップテーブルが変更されない場合は、永続キャッシュを使用できます。永続キャッシュを使用すると、ルックアップテーブルの読み取りにかかる時間がなくなるため、マッピングのパフォーマンスが向上します。永続ルックアップキャッシュに設定している場合、データ統合が最初のマッピングを実行したときに、キャッシュファイルがディスクに保存されます。データ統合が次のマッピングを実行するときに、そのキャッシュファイルからメモリキャッシュが作成されます。

ルックアップトランスフォーメーションで永続ルックアップキャッシュを使用するための設定は、トランスフォーメーションの詳細プロパティで行います。永続キャッシュを使用するには、**[ルックアップキャッシュパーシステント]**プロパティを有効にします。

永続キャッシュを使用する場合は、以下のオプションを設定することができます。

キャッシュファイルの名前の指定

永続ルックアップキャッシュを使用する場合、キャッシュファイルの名前を指定できます。

名前を指定するには、ルックアップトランスフォーメーションの**[詳細]**タブにある**[キャッシュファイル名のプレフィックス]**フィールドにファイル名のプレフィックスを入力します。.idx や.dat などのサフィックスは入力しません。

ルックアップキャッシュの再構築

ルックアップテーブルが変更されることがある場合は、ルックアップキャッシュを再構築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できます。このように設定すると、データ統合は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を最初に呼び出すときに、ルックアップソースからルックアップキャッシュを再構築します。

キャッシュを再構築するようにトランスフォーメーションを設定するには、ルックアップトランスフォーメーションの**[詳細]**タブで**[ルックアップソースからの再キャッシュ]**プロパティを有効にします。

ルックアップキャッシュの再構築

データ統合による最後の永続キャッシュの作成時以降にルックアップソースが変更されていると思われる場合は、ルックアップキャッシュを再構築することができます。

キャッシュを再構築すると、データ統合は新しいキャッシュファイルを作成し、既存の永続キャッシュファイルを上書きします。データ統合は、キャッシュの再構築時にセッションログにメッセージを書き込みます。

キャッシュを再利用できない場合、データ統合はキャッシュを再構築するか、マッピングタスクが失敗します。動作はキャッシュが名前付きか名前なしかによって異なります。

次の表は、マッピングが実行間で変更された場合に、データ統合で名前付き永続キャッシュと名前なし永続キャッシュがどのように処理されるかをまとめたものです。

実行間でのマッピングの変更	名前付きキャッシュ	名前なしキャッシュ
データ統合がキャッシュファイルを見つけられません。例えば、ファイルがすでに存在していない。	キャッシュの再構築	キャッシュの再構築
マッピングタスクの詳細セッションプロパティで［高精度を有効にする］オプションを有効化または無効化。	マッピングタスクに失敗	キャッシュの再構築
Mapping Designer または Mapplet Designer でトランスフォーメーションを編集（トランスフォーメーションの説明の編集は除く）。	マッピングタスクに失敗	キャッシュの再構築
マッピングを編集（ルックアップトランスフォーメーションは除く）。	キャッシュの再利用	キャッシュの再構築
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パイプラインでパーティションの数を変更。	マッピングタスクに失敗	キャッシュの再構築
ルックアップテーブル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。	マッピングタスクに失敗	キャッシュの再構築

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、マッピング内の他のトランスフォーメーションに関連付けられていないルックアップトランスフォーメーションです。マッピングパイプラインのトランスフォーメーションが、:LKP 式でルックアップトランスフォーメーションを呼び出します。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は、呼び出し元のトランスフォーメーションに 1 つのカラムを返します。

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使用して、次のタイプのデータオブジェクトにルックアップを実行できます。

- フラットファイル
- リレーショナルデータベース
- Amazon Redshift V2
- Amazon S3 V2
- Google BigQuery V2
- Microsoft Azure Synapse SQL
- Snowflake Data Cloud

次の表に、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と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の違いを示します。

機能	接続されたルックアップ	未接続のルックアップ
入力値	入力値をマッピング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。
キャッシュ	キャッシュには、マッピングで使用されるすべてのルックアップカラムが格納されます。これには、ルックアップ条件のカラム、および出力フィールドとして他のトランスフォーメーションにリンクされるカラムが含まれます。 静的キャッシュまたは動的キャッシュを使用できます。	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/出力フィールドおよびルックアップ/戻りフィールドが格納されます。 動的キャッシュは使用できません。
戻り値	同じ行の複数の値を返します。	各行の特定のフィールドを返します。
ルックアップ条件	ルックアップ条件に一致するものがない場合、データ統合はすべての出力フィールドにデフォルト値を返します。 一致するものがある場合、データ統合は、すべてのルックアップ/出力フィールドにルックアップ条件の結果を返します。	ルックアップ条件に一致するものがない場合、データ統合は NULL を返します。 一致するものがある場合、データ統合は、ルックアップ条件の結果を戻り値フィールドに返します。
出力値	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/出力フィールド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渡します。ルックアップ/出力/戻りフィールドは、次の:LKP 式を含むトランスフォーメーションに値を渡します。

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の設定

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、**[接続されていないルックアップ]** オプションを選択し、入力フィールドを追加し、ルックアップ条件を設定して、戻り値を指定します。次に、別のトランスフォーメーションでルックアップ式を構成します。

1. ルックアップトランスフォーメーションの **[全般]** タブで、**[接続されていないルックアップ]** オプションを有効にします。
2. 受信フィールドを作成します。

   ルックアップトランスフォーメーションの **[受信フィールド]** タブで、:LKP 式の各引数に対して入力フィールドを作成します。作成したいルックアップ条件のそれぞれに対して、受信フィールドをルックアップトランスフォーメーションに追加しなければなりません。各条件で異なるポートを作成できます。また、複数の条件で同じ受信フィールドを使用することもできます。
3. 戻り値を指定します。

   ルックアップトランスフォーメーションに複数の入力値を渡し、データのカラムを 1 つ返すことができます。データ統合は、ルックアップクエリから 1 つの値を返します。戻りフィールドを使用して、戻り値を指定します。
4. 別のトランスフォーメーションのルックアップ式を設定します。

   式、アグリゲータ、フィルタ、またはルータトランスフォーメーションなどの式を使用するトランスフォーメーションの:LKP 式から、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の入力値を指定します。引数は、ルックアップ条件に使用されているルックアップトランスフォーメーションのローカル入力フィールドと一致するローカル入力ポートです。

別のトランスフォーメーションからの接続されていないルックアップの呼び出し

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別のトランスフォーメーションで、:LKP 式から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の入力値を指定します。1 つのマッピングで同じルックアップを複数回呼び出すことができます。接続されていないルックアップはジョイナまたは Java トランスフォーメーションから呼び出せません。

:LKP 式には以下の構文を使用します。

```
:LKP.<Lookup transformation name> (<argument>, <argument>, ...)
```

引数は、ルックアップ条件に使用されているルックアップトランスフォーメーションのローカル入力フィールドと一致するローカル入力ポートです。

例えば、次の式は、ITEM_ID と PRICE フィールドを lkp_ItemPrices という名前のないルックアップトランスフォーメーションに渡します。

```
:LKP.lkp_ItemPrices (ITEM_ID, PRICE)
```

コネクトされていない Lookup トランスフォーメーションを呼び出す式を記述するときは、以下のガイドラインに従ってください。

- 各引数を記述する順序は、Lookup 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。
- 式のフィールドのデータタイプは、ルックアップトランスフォーメーションの入力ポートのデータタイプと一致しなければなりません。
- 式の引数（フィールド）は、ルックアップ条件の入力ポートと同じ順に並んでいなければなりません。
- :LKP 式で接続されたルックアップトランスフォーメーションを呼び出すと、データ統合では、そのマッピングが無効としてマークされます。

接続されているルックアップの例

次の例で、ルックアップトランスフォーメーションは Orders テーブルに対してルックアップを実行し、特定の顧客に対するすべての注文を返します。また、顧客の最初の注文または最後の注文だけを返すように、ルックアップを定義することもできます。

最初に、ルックアップ条件を設定します。この条件は、ルックアップテーブルから返す行を特定する式です。例えば、ルックアップフィールド CUSTOMER_ID が受信フィールド CUSTOMER_IN と等しいすべてのレコードを探す、シンプルなルックアップ条件を作成します。

この条件に基づいて、ルックアップでは、顧客 ID が、ルックアップトランスフォーメーションに渡された顧客番号と等しいすべての行が検索されます。

複数の条件を追加することもできます。例えば、次の条件を追加すると、金額が$100.00 を超える注文のみが返されます。

O_ORDER_AMT > 100.00

すべての条件に該当する場合にのみ、データが返されます。

動的ルックアップの例

ルックアップトランスフォーメーションを動的になるように設定するには、動的ルックアップキャッシュを使用します。

動的キャッシュは、ソーステーブルに大量のデータが含まれる場合や重複したプライマリキーが含まれる場合に便利です。

次の例では、ソーステーブルに重複したプライマリキーが含まれる場合に静的キャッシュではなく動的キャッシュを使用するメリットを示しています。

人事部からのデータを含む給与支払い名簿テーブルを更新するとします。給与支払い名簿テーブルには、次のデータが含まれています。ここで、ID はプライマリキーです。

ID	名前	場所
1	Abhi	USA
2	Alice	UK

ルックアップトランスフォーメーションでマッピングを作成し、ターゲットの給与支払い名簿テーブルとルックアップキャッシュを使用します。人事部のテーブルには重複キーが含まれることが予期されるため、キャッシュが動的になるように設定します。

マッピングで、人事部のテーブルをソースに指定します。ソーステーブルには、以下のデータが含まれます。

ID	名前	場所
1	Abhi	India
2	Alice	UK
3	James	Japan
3	James	USA

マッピングタスクを作成して給与支払い名簿テーブルを更新します。マッピングタスクを開始すると、ターゲットテーブルの行を含むキャッシュファイルが作成されます。タスクが行を処理するときに最初の行に更新というフラグを設定し、キャッシュを更新します。3 番目の行に挿入というフラグを設定し、キャッシュに行を挿入します。キャッシュに行が存在するため、4 番目の行には更新というフラグを設定します。

静的キャッシュを使用して同じシナリオに従った場合、タスクは 4 番目の行に挿入というフラグを設定します。タスクが行を処理するときに更新されないため、キャッシュに James の行は含まれません。同じプライマリキーを持つ 2 つの行を処理することはできないため、ターゲットデータベースでエラーが発生します。

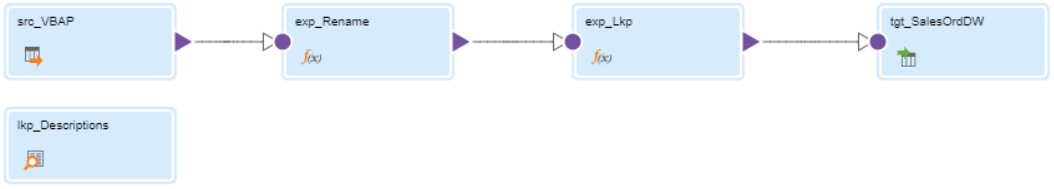
接続されていないルックアップの例

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使用して、テーブル内の暗号化または数値 ID 値を、ルックアップテーブルのわかりやすい名前で置き換えることができます。

例えば、SAP トランザクションテーブルからデータウェアハウスのリレーショナルテーブルに一部の受注データをロードする必要があるとします。SAP テーブルには、販売グループや営業所などの値の数値 ID が含まれて

います。データウェアハウステーブルでは、数値 ID をローカル言語の対応する名前に置き換える必要があります。各 ID に関連付けられている名前は、参照テーブルに格納されます。参照テーブルから名前を取得するには、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使用します。

次の図に、マッピングを示します。



次の方法でトランスフォーメーションを設定します。

ソーストランスフォーメーション

ソーストランスフォーメーションを使用して、データの抽出元のテーブルを指定します。

[ソース] タブで、ソース接続を構成し、データを抽出するテーブルを選択します。

最初の式トランスフォーメーション（省略可能）

必要に応じて、式トランスフォーメーションを使用してフィールドの名前を変更し、NULL 値を置換します。

[受信フィールド] タブで、**[名前付きフィールド]** フィールドの選択条件を使用して、ターゲットテーブルに読み込むフィールドを選択します。必要に応じて、選択したフィールドの名前を変更してわかりやすい名前を付けます。

[式] タブで、NULL 値を置き換える出力フィールドを作成します。例えば、販売グループコードおよび営業所コードの NULL 値をスペースに置き換えるには、次の出力フィールドを作成します。

出力フィールド	式
in_sales_group	IIF(ISNULL(sales_group_code),' ',sales_group_code)
in_sales_office	IIF(ISNULL(sales_office_code),' ',sales_office_code)

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

参照テーブルから説明を取得するには、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使用します。

[全般] タブで、**[接続されていないルックアップ]** オプションを有効にします。

[受信フィールド] タブで、ルックアップトランスフォーメーションに渡す必要がある各値の受信フィールドを作成して、必要なデータを取得します。例えば、ドメイン名、言語、およびコード値をルックアップトランスフォーメーションに渡すには、in_domain_name、in_language、および in_lookup_code フィールドを作成します。

[ルックアップオブジェクト] タブで、ルックアップ接続を構成し、ルックアップテーブルとして使用する参照テーブルを選択します。

[ルックアップ条件] タブで、各受信フィールドのルックアップ条件を指定します。以下に例を示します。

ルックアップフィールド	演算子	受信フィールド
domain_name	=	in_domain_name
language_code	=	in_language
lookup_code	=	in_lookup_code

[戻りフィールド] タブで、返される参照テーブルのフィールドを選択します。例えば、説明を返すには、戻りフィールドとして lookup_description を選択できます。

2 番目の式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を使用して、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呼び出し、各 ID 値に関連付けられている名前を取得します。

[受信フィールド] タブで、アップストリームトランスフォーメーションのすべてのフィールドを含めます。

[式] タブで、ルックアップトランスフォーメーションから各説明を取得する出力フィールドを作成します。:LKP 式を使用して、ルックアップトランスフォーメーションを呼び出します。例えば、英語の適切なドメインから販売グループおよび営業所名を取得するには、次の出力フィールドを作成します。

出力フィールド	式
sales_group	`:LKP.lkp_Descriptions('sales_group','en',in_sales_group)`
sales_office	`:LKP.lkp_Descriptions('sales_office','en',in_sales_office)`

ターゲットトランスフォーメーション

[ターゲット] タブで、ターゲット接続を構成し、データをロードするリレーショナルテーブルを選択します。

[フィールドマッピング] タブで、アップストリームトランスフォーメーションから適切なターゲットフィールドに出力フィールドをマッピングします。例えば、2 番目の式トランスフォーメーションから SALES_GROUP および SALES_OFFICE ターゲットフィールドに sales_group および sales_office 出力フィールドをマッピングするには、次のフィールドマッピングを構成します。

ターゲットフィールト	マッピングされたフィールド
SALES_GROUP	sales_group
SALES_OFFICE	sales_office

第 20 章

機械学習トランスフォーメーション

詳細モードでは、機械学習トランスフォーメーションは機械学習モデルを実行し、予測を返します。また、REST API を使用して受信データを機械学習モデルに渡し、予測をダウンストリームトランスフォーメーションに渡します。

次のビデオは、機械学習トランスフォーメーションを使用して、組織のデータ統合ジョブに機械学習モデルを組み込む方法に関するユースケースを示しています。

Consuming a Machine Learning Model in Informatica Cloud Data Integration Elastic

機械学習トランスフォーメーションを使用する前に、次の要件を満していることを確認してください。

- 機械学習モデルが Amazon SageMaker や Azure Machine Learning などの機械学習プラットフォームにデプロイされており、REST エンドポイントを使用してモデルから予測を取得できる。
- API コレクションに、REST V3 接続を使用して REST エンドポイントにアクセスするための POST 要求がある。

API コレクションの詳細については、「*コンポーネント*」を参照してください。

REST エンドポイントとしてのモデルのデプロイ

機械学習モデルは、REST エンドポイントとしてデプロイする必要があります。機械学習トランスフォーメーションは、エンドポイントを使用してモデルと通信します。

次のような機械学習プラットフォームに従って、モデルを REST エンドポイントとしてデプロイします。

Amazon SageMaker

Amazon SageMaker で、Amazon API Gateway と AWS Lambda を使用して、モデルをエンドポイントとしてデプロイします。

詳細については、次の AWS Machine Learning のブログ記事にある手順を参照してください。

https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-aws-lambda/

Azure Machine Learning

Azure Machine Learning で、モデルをリアルタイムエンドポイントとしてデプロイします。

リアルタイムエンドポイントの詳細については、Microsoft Azure のドキュメントを参照してください。

モデルを REST エンドポイントとしてデプロイした後に、API コレクションを作成し、エンドポイントにアクセスするための REST API 要求を設定します。

機械学習モデルへのアクセス

[モデル] タブで、機械学習トランスフォーメーションが機械学習モデルにアクセスする方法を定義します。

API コレクションから REST API 要求を選択します。API コレクションと同じ REST V3 接続を使用するか、別の REST V3 接続を選択することができます。

次の認証方法は、機械学習トランスフォーメーションで使用できないため、REST V3 接続では使用できません。

- OAuth 2.0 認証コード
- OAuth 2.0 クライアント資格情報

サーバーレスランタイム環境で機械学習トランスフォーメーションを実行し、トランスフォーメーションと機械学習モデルの間でサーバー認証用に TLS を設定する場合の詳細については、Administrator ヘルプを参照してください。非サーバーレスランタイム環境用に TLS を設定するには、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。

機械学習モデルにアクセスするには、API コレクションに POST 要求が必要です。要求タイプを変更して API コレクションを同期すると、機械学習トランスフォーメーションによって **[モデル]**、**[要求マッピング]**、および **[応答フィールド]** タブがクリアされます。別の POST 要求を選択して、トランスフォーメーションを再設定する必要があります。

要求スキーマへのフィールドのマッピング

[要求マッピング] タブで、受信フィールドを要求スキーマフィールドにマッピングします。機械学習トランスフォーメーションは、要求スキーマフィールドを使用して、REST エンドポイントの機械学習モデルにデータを送信します。

要求スキーマフィールドは、API コレクションの REST API 要求の要求スキーマから取得されます。作業フィールド名は、機械学習トランスフォーメーションが REST API 要求で渡すキーです。REST API がフィールド名の特殊文字を処理できない場合、作業フィールド名は特殊文字に置き換えられます。例えば、アンダースコアはピリオド（.）に置き換えられます。

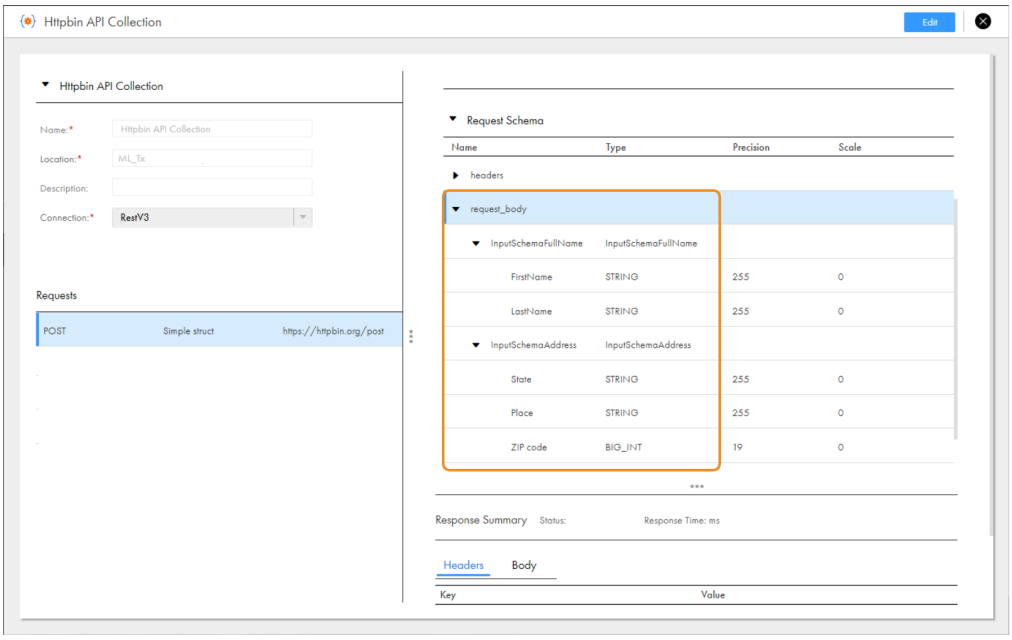
受信フィールドと要求スキーマフィールドの名前は異なる場合がありますが、データ型と階層は一致している必要があります。あるフィールドが REST API で必須ではない場合でも、すべての要求スキーマフィールドにはマッピングされたフィールドが必要です。

要求マッピングには親フィールドのみが表示されることに注意してください。子フィールドを確認するには、機械学習トランスフォーメーションの受信フィールドと API コレクションの要求スキーマフィールドを参照してください。

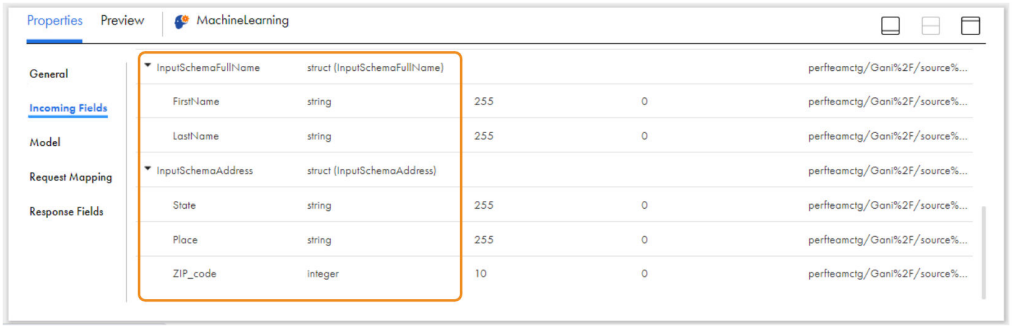
階層フィールドのマッピング

要求マッピングで階層フィールドをマッピングするには、受信フィールドと要求スキーマフィールドの階層が一致している必要があります。

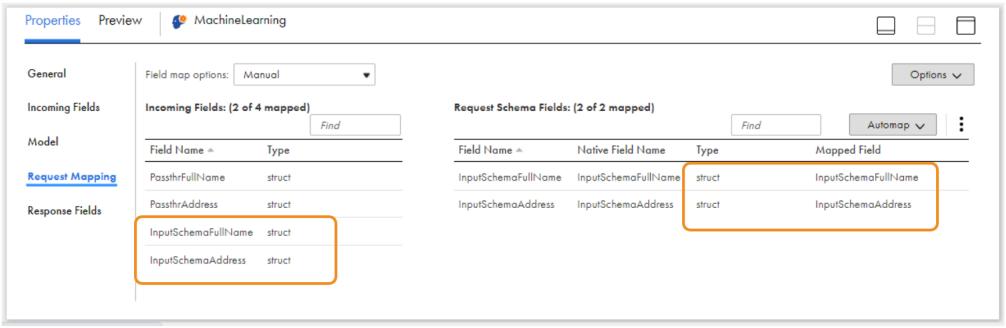
例えば、次の図は API コレクションの要求スキーマを示しています。



次の図に、要求スキーマに一致する階層を持った機械学習トランスフォーメーションの入力フィールドを示します。



次の図に、受信フィールドと要求スキーマフィールド間の要求マッピングを示します。



階層が一致しない場合は、機械学習トランスフォーメーションを実行する前に、階層プロセッサトランスフォーメーションを使用して要求スキーマフィールドに従って受信データを再構築します。詳細については、「階層プロセッサトランスフォーメーションの概要」 (ページ 180)を参照してください。

要求マッピングのオプション

要求マッピングオプションを設定して、受信フィールドを要求スキーマフィールドにマッピングする方法を定義します。

次のような要求マッピングオプションを設定できます。

フィールドマップオプション

フィールドをターゲットにマッピングする方法。次のいずれかのオプションを使用します。

- 手動。受信フィールドをターゲットフィールドに手動でリンクします。このオプションを選択すると、自動的にマッピングされたフィールドのリンクが削除されます。フィールドを手動でマッピングするには、フィールドを［受信フィールド］リストからドラッグして、［ターゲットフィールド］リストの適切なフィールドの横に配置します。または、**［アクション］**メニューを使用して、選択されたフィールドをマッピングまたはマッピング解除したり、すべてのマッピングをクリアしたりすることができます。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。タスクで、すべてのフィールドマッピングを設定できます。
- 一部パラメータ化。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。
  フィールドマッピングパラメータの詳細については、「*マッピング*」を参照してください。

パラメータ

フィールドマッピングに使用するパラメータ。

新しいパラメータ

フィールドマッピングパラメータを作成します。

フィールドの表示

［受信フィールド］リストに表示されるフィールドを制御します。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示します。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

アクションメニュー

追加フィールドリンクオプション。次のオプションが用意されています。

- 選択項目をマップ。選択した受信フィールドと選択したターゲットフィールドをリンクします。
- 選択項目をマップ解除。選択したフィールドのリンクをクリアします。
- マッピングのクリア。すべてのフィールドマッピングをクリアします。

表示

ターゲットフィールドリストにフィールド名を表示する方法を指定します。技術フィールド名またはラベルを使用します。

応答フィールドの表示

［応答フィールド］タブで、REST API によって返された応答スキーマフィールドを表示します。応答スキーマフィールドは、ダウンストリームトランスフォーメーションに渡すことができる予測をキャプチャします。

応答スキーマフィールドは、API コレクションからの REST API 要求の応答スキーマから取得されます。作業フィールド名は、機械学習トランスフォーメーションが REST API 応答から読み取るキーです。作業フィールド名では、REST API が処理できない特殊文字が置き換えられる場合があります。フィールド名は、機械学習トランスフォーメーションがマッピングの応答データを表すために使用する、対応する出力フィールドです。

さらに、ダウンストリームトランスフォーメーションに渡される出力フィールドには、次の文字以外の特殊文字を含めることはできません: @ # ( )。特殊文字は、機械学習トランスフォーメーションによってアンダースコア（_）に置き換えられます。

一括要求の設定

機械学習トランスフォーメーションが機械学習モデルに送信する 1 つの要求に複数の要求をまとめることで、最適なパフォーマンスが得られるように一括要求を設定します。一括要求は、オーバーヘッドの処理とモデルとの通信にかかる時間を削減することで、パフォーマンスを向上させることができます。

一括要求を作成するために、機械学習トランスフォーメーションは要求スキーマの最上位の配列フィールドを選択します。一括要求の JSON 要求本文で、トランスフォーメーションは、選択した配列フィールドの要素として要求行を結合し、1 つの JSON 要求本文に複数の要求のデータが含まれるようにします。一括要求オプションを設定して、各一括要求に含まれるデータの量を指定できます。

例えば、要求スキーマの構造は次のようになります。

Field Name	Working Field Name	Type
▼ instances	▼ instances	struct (instances)
▼ data	▼ data	array (data [])
▼ features	▼ features	array (features [])
keys	keys	struct (data_keys)
values	values	struct (data_values)
index	index	integer
pages	pages	integer

機械学習トランスフォーメーションは、要求を結合するための最上位の配列フィールドとして data 配列を選択します。2 MB のデータを機械学習モデルに送信するように各一括要求を設定した場合、機械学習トランスフォーメーションにより、2 MB の要求行のデータを含むように JSON 要求本文の data 配列が設定されます。

最上位の配列に兄弟配列フィールドを含めることはできません。最上位配列にプリミティブデータ型の兄弟フィールドがある場合、兄弟フィールドのデータは結合されません。その代わり、兄弟フィールドのランダムなレコードが機械学習モデルに送信されます。

一括応答を作成するために、機械学習エンドポイントは応答行を応答スキーマの最上位配列の要素としてまとめる必要があります。機械学習トランスフォーメーションは配列を解析し、出力行に結果が表示されます。機械学習トランスフォーメーションが解析するフィールドを確かめるために、一括要求オプションを確認します。

機械学習トランスフォーメーションで一括要求を使用するには、一括要求を受け入れて一括応答を送信するように機械学習エンドポイントを設定する必要があります。

一括要求オプション

一括要求オプションにより、機械学習トランスフォーメーションが要求の結合と応答の解析に使用するフィールドを決定します。また、それぞれの一括要求に含まれるデータの量も決定します。

次の表に、一括要求のオプションを示します。

オプション	説明
フィールドの要求をまとめる	トランスフォーメーションは要求行をこの配列フィールドの要素としてまとめて、1 つの要求を機械学習モデルに送信します。このフィールドは要求スキーマによって自動的に決定されます。
要求サイズ	各要求の最大サイズ。サイズを選択するには、機械学習プラットフォームのベストプラクティスを確認してください。
フィールドの要求を解析する	機械学習エンドポイントは応答行をこの配列フィールドの要素としてまとめます。次に、トランスフォーメーションが配列を解析し、出力行に結果が表示されます。このフィールドは応答スキーマによって自動的に決定されます。

API プロキシの設定

実行時に API のセキュリティを強化するように API プロキシを設定できます。

次の表に、使用可能なプロキシのタイプを示します。

プロキシタイプ	説明
なし	エージェント、Spark、または接続レベルで設定されたプロキシサーバーをバイパスします。
Spark	Spark エンジンに設定したプロキシが考慮されます。
カスタム	接続レベルで設定したプロキシが考慮されます。

注: プラットフォームプロキシは使用できないため、代わりに Spark プロキシを設定する必要があります。プラットフォームプロキシでは、エージェントレベルで設定したプロキシが考慮されますが、機械学習トランスフォーメーションを使用するマッピングは、プロキシの詳細について Spark エンジンを参照します。

機械学習トランスフォーメーションがサーバーレスランタイム環境で実行されている場合、API プロキシは適用されません。

プロキシサーバーのバイパス

プロキシサーバーをバイパスするには、REST V3 接続のプロキシタイプとして **[なし]** を選択します。エージェントまたは Spark レベルでプロキシサーバーを設定しないでください。詳細については、該当するコネクタのヘルプを参照してください。

Spark プロキシの設定

プラットフォームプロキシは、Spark プロキシに置き換わりました。マッピングタスクで Spark プロキシを設定する必要があります。詳細については、該当するコネクタのヘルプを参照してください。

1. REST V3 接続で、プロキシタイプとして［**プラットフォーム**］を選択します。
2. マッピングタスクで、Spark セッションプロパティ spark.driver.extraJavaOptions を次の値に設定します: -Dhttp.proxyHost=<host> -Dhttp.proxyPort=<port> -Dhttp.proxyUser=<user> -Dhttp.proxyPassword=<password>

カスタムプロキシの設定

REST V3 接続でプロキシの詳細を使用するようにカスタムプロキシを設定します。エージェントまたは Spark レベルでプロキシサーバーを設定しないでください。詳細については、該当するコネクタのヘルプを参照してください。

1. REST V3 接続で、プロキシタイプとして［**カスタム**］を選択します。
2. ホスト、ポスト、ユーザー、およびパスワードを設定します。
3. 接続を保存します。

トラブルシューティング

Monitor を使用してログファイルにアクセスし、機械学習トランスフォーメーションでの REST API 要求と応答のトラブルシューティングを行います。詳細を表示するには、トレースレベルを［詳細 - データ］に設定します。

次の表に、各ログファイルで使用可能な詳細を示します。

ログファイル	**詳細**
Spark ドライバログ	- 入力スキーマ - 出力スキーマ - URL - HTTP メソッド
Spark エグゼキュータログ	マッピングが失敗した場合は、失敗した行ごとに次のような詳細を確認できます。 - 応答コード* - URL、ヘッダー、要求本文などの要求の詳細 - ヘッダーや応答本文などの応答の詳細 成功した行ごとに、次のような詳細を確認できます。 - 応答コード - 応答を受け取るまでの時間
セッションログ	マッピングが失敗した場合のスタックトレース*
** ログレベルが INFO に設定されている場合にも確認できます。*	

詳細モードのログファイルの詳細については、Monitor のヘルプを参照してください。

エラー処理

機械学習トランスフォーメーションは、発生した問題のタイプと、REST API から受け取った応答コードに基づいてエラーを処理します。Spark エグゼキュータログで応答の詳細を確認できます。

次のようなタイプの問題が発生すると、マッピングは失敗します。

- 接続タイムアウトがある場合、ルートが見つからない場合、または不明なホストがある場合の接続の問題。
- URL の構文の問題。
- サーバーとの SSL ハンドシェイク中の証明書の問題などの SSL ハンドシェイクの問題。

次の表に、各応答コードで発生するアクションを示します。

HTTP **ステータスコード**	**アクション**
100 Continue	行はスキップされます。 部分的な要求は許可されません。
101,102,103 Informational, interim	マッピングは失敗します。
3XX Redirection	行はスキップされます。 リダイレクトされた URL に到達できない場合、またはリダイレクトされた URL によって、マッピングに失敗したコードが返された場合、マッピングは失敗します。
400 Bad request	行はスキップされます。
401 Unauthorized	マッピングは失敗します。
403 Forbidden	マッピングは失敗します。
404 Resource not found	マッピングは失敗します。
405 Method not allowed	マッピングは失敗します。
407 Proxy authentication required	要求が再試行されます。
408 Request timeout	マッピングは失敗します。
409 Conflict	行はスキップされます。
415 Unsupported media type	マッピングは失敗します。
418 I'm a teapot	マッピングは失敗します。
421 Misdirected request	マッピングは失敗します。
423 Resource is locked	行はスキップされます。
429 Too many requests	マッピングは失敗します。
500 Internal server error	マッピングは失敗します。
501 Not implemented	マッピングは失敗します。

HTTP **ステータスコード**	**アクション**
502 Bad gateway	マッピングは失敗します。
503 Service unavailable	マッピングは失敗します。
504 Gateway timeout	マッピングは失敗します。
505 HTTP version not supported	マッピングは失敗します。
506 Server internal configuration error	マッピングは失敗します。
511 Network authentication required	マッピングは失敗します。

第 21 章

マップレットトランスフォーメーション

マップレットトランスフォーメーションは、データ統合で作成したマップレット、PowerCenter からインポートしたマップレット、または SAP アセットから生成したマップレットをマッピングに挿入します。各マップレットトランスフォーメーションに含めることができるマップレットは 1 つのみです。複数のマップレット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できます。

マップレットトランスフォーメーションがアクティブであるかパッシブであるかは、マップレット内のトランスフォーメーションロジックに基づいて決まります。アクティブなマップレットには、少なくとも 1 つのアクティブなトランスフォーメーションが含まれます。アクティブなマップレットによって返される行の数は、マップレットに渡されるソース行の数と異なることがあります。パッシブなマップレットには、パッシブなトランスフォーメーションのみが含まれます。パッシブなマップレットで返される行の数は、ソースから渡される行の数と同じです。

マップレットトランスフォーメーションを使用すると、次のことが実現できます。

データ統合のデータトランスフォーメーション機能を拡張します。

例えば、顧客が信用調査に合格した場合に、顧客レコードをターゲットに渡すマッピングを作成するとします。Web サービストランスフォーメーションを作成して、各顧客の信用調査を実行します。マップレットに Web サービストランスフォーメーションを含め、このマップレットをマッピングで使用して信用調査を行います。

トランスフォーメーションロジックをさまざまなマッピングで再利用する。

例えば、さまざまなファクトテーブルがあり、一連の次元キーが必要となるとします。一連のルックアップトランスフォーメーションを含むマップレットを作成し、各次元キーを検索します。各マッピングでルックアップロジックを再作成する代わりに、マップレットを異なるファクトテーブルマッピングに含めます。

複合トランスフォーメーションロジックを隠す。

マップレットトランスフォーメーションには、マップレットの受信フィールドと出力フィールドが表示されます。マップレットに含まれるトランスフォーメーションは表示されません。

マップレットの詳細については、「*コンポーネント*」を参照してください。

マップレットトランスフォーメーションの設定

マップレットトランスフォーメーションをマッピングに追加する場合、最初に、マッピングに含めるトランスフォーメーションロジックが含まれるマップレットを選択する必要があります。マップレットに 1 つ以上の入

力グループが含まれる場合、受信フィールドとフィールドマッピングを設定します。マップレットに 1 つ以上の出力グループが含まれる場合は、出力フィールドを検証します。

マップレットトランスフォーメーションを設定するには、次のタスクを実行します。

1. トランスフォーメーションに組み込むマップレットを選択します。
2. マップレットに 1 つ以上の入力グループが含まれる場合、受信フィールドを設定します。

   デフォルトでは、トランスフォーメーションはアップストリームトランスフォーメーションからすべての受信フィールドを継承します。受信フィールドの制限や名前変更を行えるようにフィールドルールを定義できます。マップレットに複数の入力グループが含まれている場合は、各入力グループに受信フィールドを設定します。

   トランスフォーメーションの受信フィールドの設定に関する詳細については、「受信フィールド」 (ページ 23)を参照してください。
3. マップレットに 1 つ以上の入力グループが含まれる場合は、フィールドマッピングを設定して、データがアップストリームトランスフォーメーションからマップレットトランスフォーメーションに移動する方法を定義します。

   マップレットに複数の入力グループが含まれている場合は、各入力グループにフィールドマッピングを設定します。
4. マップレットに 1 つ以上の出力グループが含まれる場合は、**[出力フィールド]** タブでマップレットの出力フィールドを確認します。少なくとも 1 つの出力グループをダウンストリームトランスフォーメーションに接続します。

マップレットの選択

[プロパティ] パネルの **[マップレット]** タブで、マップレットトランスフォーメーションに使用するマップレットを選択します。データ統合に作成またはインポートしたマップレットを選択するか、インストール済みバンドルに含まれるマップレットを選択することができます。

1. マップレットトランスフォーメーションの **[プロパティ]** パネルで **[マップレット]** タブをクリックします。
2. **[選択]** をクリックします。
3. マップレットが含まれるプロジェクトとフォルダを開き、**[選択]** をクリックします。

   インストールされているバンドルのマップレットは、アドオンバンドルプロジェクトに格納されています。

   選択したマップレットは **[プロパティ]** パネルに表示されます。

選択したマップレットにソースが含まれていない場合、マップレットの選択後に受信フィールドとフィールドマッピングを設定します。

選択したマップレットにターゲットが含まれていない場合、マップレットの選択後に出力フィールドとフィールドマッピングを設定します。

マップレットトランスフォーメーションのフィールドマッピング

データがアップストリームトランスフォーメーションからマップレットトランスフォーメーションに移動する方法を定義するには、マップレットトランスフォーメーションでフィールドマッピングを設定します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

注: マッピングされたフィールド名は、最長で 72 文字にできます。

次のフィールドマッピングオプションを設定できます。

マップレット入力グループ

フィールドマッピングを設定する入力グループ。マップレットトランスフォーメーションに入力グループが複数ある場合、このオプションが表示されます。

フィールドマップオプション

マップレットトランスフォーメーションにフィールドをマッピングする方法。次のいずれかのオプションを選択します。

- 手動。受信フィールドをマップレットトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。

オプション

［受信フィールド］と**［マップレット入力フィールド］**リストでフィールドを表示する方法を制御します。以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

アクションメニュー

追加フィールドマッピングオプション。次のオプションが用意されています。

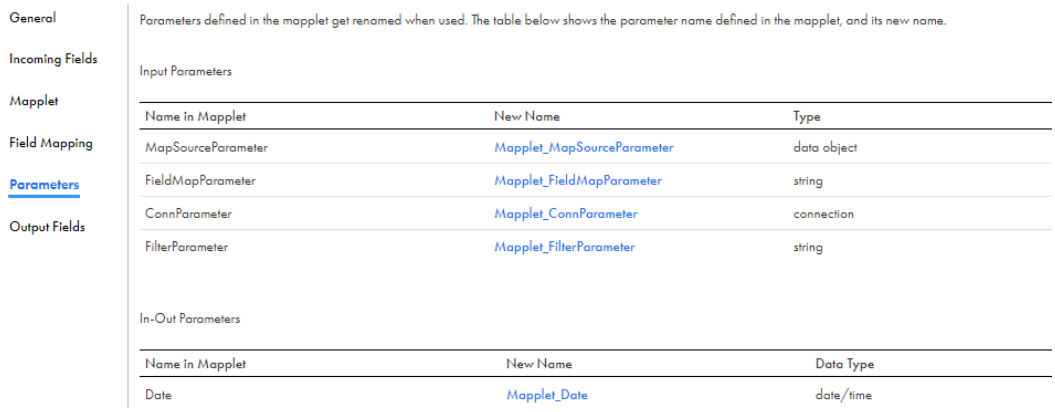
- 選択項目をマップ。選択した受信フィールドを、選択したマップレットマップレット入力フィールドにリンクします。
- 選択項目をマップ解除。選択したフィールドのリンクをクリアします。
- マッピングのクリア。すべてのフィールドマッピングをクリアします。

マップレットパラメータ

パラメータを含むマップレットを選択すると、パラメータの名前がマップレットトランスフォーメーションで変更されます。**［プロパティ］**パネルの**［パラメータ］**タブで対応するパラメータ名を参照できます。

マップレットトランスフォーメーションで、マップレットパラメータ名にプレフィックスとしてマップレットトランスフォーメーションの名前が付加されます。

次の図は、［パラメータ］タブを示しています。



マップレットパラメータのプロパティは編集できますが、パラメータタイプを変更したり、パラメータを削除したりすることはできません。パラメータを削除するには、マップレットを開いてパラメータを削除します。

パラメータのプロパティを編集するには、新しいパラメータの名前を**［パラメータ］**タブまたは**［パラメータ］**パネルでクリックします。マップレットトランスフォーメーションでパラメータのプロパティを変更した場合、この変更はマップレットには影響しません。

マッピング内の他のトランスフォーメーションに含まれるパラメータを再利用できます。

マップレットトランスフォーメーションの出力フィールド

マップレットトランスフォーメーションで使用する、出力トランスフォーメーションを含むマップレットを選択すると、マップレットの出力フィールドは**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。

Mapping Designer には、各出力グループの各出力フィールドについて、名前、タイプ、精度、スケール、基点が表示されます。トランスフォーメーション出力フィールドを編集することはできません。出力フィールドをデータフローから除外したり、出力フィールドの名前を変更したりしてから、出力フィールドをダウンスト

リームトランスフォーメーションに渡す場合は、ダウンストリームトランスフォーメーションでフィールドルールを設定します。

マップレットトランスフォーメーション名

マップレットトランスフォーメーションでマップレットを使用すると、データ統合では、マップレットのトランスフォーメーションの名前が変更されます。ダウンストリームトランスフォーメーションのマップレットでトランスフォーメーションを参照する場合は、必ず更新された名前を使用してください。

マップレットトランスフォーメーションで使用するマップレットには、マッピングのトランスフォーメーションの名前と競合する名前を持つトランスフォーメーションが含まれている可能性があります。名前がマッピングのトランスフォーメーションと競合しないようにするために、データ統合は、実行時にマップレットのトランスフォーメーション名の先頭にマップレットトランスフォーメーション名を付与します。

例えば、マップレットに Expression_1 という名前の式トランスフォーメーションが含まれているとします。マッピングを作成してマップレットトランスフォーメーション Mapplet_Tx_1 でこのマップレットを使用した場合、マッピングを実行すると、式トランスフォーメーションの名前が Mapplet_Tx_1_Expression_1 に変更されます。

また、マップレットに別のマップレットトランスフォーメーションが含まれる場合、データ統合は、トランスフォーメーション名の先頭に 2 つ目のマップレットトランスフォーメーション名を付与します。例えば、前の例のマップレットに MappletTx という名前のマップレットトランスフォーメーションがあり、これに FilterTx_1 が含まれているとします。マッピングでは、FilterTx_1 は、Mapplet_Tx_1_MappletTx_FilterTx_1 に変更されます。

ほとんどの場合、データ統合では、80 文字を超えるトランスフォーメーション名が四捨五入されます。データ統合では、階層ビルダトランスフォーメーション、階層パーサートランスフォーメーション、または構造パーサートランスフォーメーションを含むマップレットトランスフォーメーションの名前は四捨五入されません。マップレットで階層ビルダトランスフォーメーション、階層パーサートランスフォーメーション、構造パーサートランスフォーメーションを使用する場合は、実行時エラーを防ぐために、結合されたマップレットトランスフォーメーション名が 80 文字を超えないようにしてください。

注: 2022 年 4 月リリース以降に作成されたマッピングでマップレットが使用されている場合にのみ、このマップレットのトランスフォーメーション名がデータ統合によって変更されます。

マップレットの同期

マップレットのインタフェースが、マップレットトランスフォーメーションに追加された後で変更された場合は、そのマップレットを同期して変更を反映する必要があります。マップレットの同期は、**[マップレット]** タブで実行します。

マップレットを使用するマッピングとマップレットは、マップレットが同期されるまでは無効です。変更されたマップレットを含むマッピングタスクを実行すると、タスクは失敗します。

マップレットを同期すると、更新によって、マッピングまたはマップレット内の他のトランスフォーメーションで検証エラーが発生する可能性があります。

PowerCenter または SAP からデータ統合にインポートしたマップレットを同期することはできません。

マップレットを同期するには、次の手順を実行します。

1. マップレットを使用するマッピングまたはマップレットを開きます。

2. マップレットトランスフォーメーションを選択します。
3. **［マップレット］**タブで、**［同期］**をクリックします。
4. トランスフォーメーションロジックでエラーを修正します。

第 22 章

ノーマライザ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は 1 つの入力行を複数の出力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、複数回出現データの各インスタンスについて 1 行を返します。

例えば、リレーショナルソースに、四半期の売上データが格納された 4 つのフィールドがあるとします。この場合、四半期ごとに別個の出力行を生成するように、ノーマライザトランスフォーメーションを設定することができます。

ノーマライザトランスフォーメーションによって入力行から複数の行が返される場合、1 回出現入力カラムについては重複する（同じ）データが返されます。

ノーマライザトランスフォーメーションを設定する際には、**［プロパティ］**パネルの次の各タブでノーマライザプロパティを定義します。

- **［正規化されたフィールド］**タブ。複数回出現フィールドを定義し、マッピングに使用する追加フィールドを指定します。
- **［フィールドマッピング］**タブ。受信フィールドを正規化されたフィールドに接続します。

正規化されたフィールド

［正規化されたフィールド］タブで、正規化対象フィールドを定義します。マッピングに使用する他の受信フィールドも指定できます。

正規化されたフィールドを定義する際には、フィールドを手動で作成するか、受信フィールドのリストからフィールドを選択できます。正規化されたフィールドを作成する際には、データ型を String または Number に設定して、精度とスケールを指定できます。詳細モードでは、任意のプリミティブデータ型を使用できます。

詳細モードでは、ノーマライザトランスフォーメーションによって複数の出力グループが生成されます。それ以外の場合、ノーマライザトランスフォーメーションは 1 つの出力グループのみを生成します。

受信フィールドに複数回出現フィールドが含まれているが、対応するカテゴリフィールドがない場合は、カテゴリフィールドを作成してデータの出現回数を定義できます。例えば、3 つの異なるタイプの所得を 3 つのフィールドで表すには、収入カテゴリフィールドを作成して出現回数値を 3 に設定します。

出現回数の設定

正規化されたフィールドの出現回数値を設定して、入力データに当該フィールドのインスタンスが出現する回数を定義します。

複数回出現フィールドを定義するには、フィールドの出現回数値を 2 以上の整数の値に設定します。出現回数値を 2 以上の値に設定すると、ノーマライザトランスフォーメーションによって、そのフィールドの生成カラム ID フィールドが作成されます。ノーマライザトランスフォーメーションは、すべての正規化対象データについて、生成キーフィールドも作成します。

また、ノーマライザトランスフォーメーションは、出現回数値を使用して、対応する出力フィールドのセットも作成します。出力フィールドは、ノーマライザトランスフォーメーションの **[フィールドマッピング]** タブに表示されます。出力フィールドの命名規則は、「<出現フィールド名>_<出現回数>」です。

1 回出現フィールドを定義するには、当該フィールドの出現回数値を 1 に設定します。1 回出現フィールドを定義することで、正規化されたフィールドリストに正規化する必要がない受信フィールドを含めることができます。

出現回数値の異なる複数回出現フィールドのグループ

ノーマライザトランスフォーメーションで、複数回出現フィールドの複数のグループを正規化できます。複数のグループを含める場合で、出現回数値が一致しない場合は、検証エラーが発生しないようにマッピングを設定します。

出現回数値の異なる複数回出現フィールドのグループを処理するには、次のいずれかの方法を使用します。

正規化されたデータを異なるターゲットに書き込む

正規化されたデータを異なるターゲットに書き込む場合は、出現回数値の異なる複数回出現フィールドを使用できます。

例えば、ソースデータに出現回数が 4 回の費用フィールドと出現回数が 3 回の収入フィールドが含まれているとします。このような場合は、正規化された費用データと正規化された収入データを、それぞれ別々のターゲットに書き込むようにマッピングを設定します。

異なる複数回出現フィールドに同じ出現回数を使用する

同じ出現回数を使用するように複数回出現フィールドを設定し、必要な生成フィールドを使用します。複数回出現フィールドに同じ出現回数を使用すると、正規化されたデータを同じターゲットに書き込むことができます。

例えば、ソースデータに出現回数が 4 回の費用フィールドと出現回数が 3 回の収入フィールドが含まれている場合に、出現回数を 4 回として両フィールドを設定します。

ノーマライザフィールドマッピングを設定する際に、4 回の費用フィールドと 3 回の収入フィールドを連結できます（1 つの不要な収入の出力フィールドは未使用のままになります）。これで、すべての正規化されたデータを同じターゲットに書き込むようマッピングを設定できます。

生成キー

ノーマライザトランスフォーメーションは、正規化されたデータのキー値を生成します。

生成されたキーフィールドは、そのフィールドが複数回出現するよう設定されている場合、**[正規化されたフィールド]** タブに表示されます。

マッピングタスクは、正規化されたデータについて、次のフィールドを生成します。

生成キー

タスクが入力行を処理するたびに生成するキー値。タスクを実行すると、生成キーは 1 から開始され、1 行処理するたびに 1 だけ増分されます。

ノーマライザトランスフォーメーションでは、正規化されるすべてのデータに対して 1 つの生成キーフィールドを使用します。

ノーマライザの生成キーの命名規則は、「GK_<再定義されるフィールド名>」です。

注: 生成されたキーは、詳細モードでは適用できません。

生成カラム ID

複数回出現データのインスタンスを表すカラム ID 値。例えば、4 回出現する費用フィールドの場合、1 から 4 の値を使用して出現データの各インスタンスが表現されます。

ノーマライザトランスフォーメーションは、複数回出現するよう設定されている各フィールドについて生成カラム ID を使用します。

ノーマライザの生成カラム ID の命名規則は、「GCID_<再定義されるフィールド名>」です。

詳細クラスタは、生成されたカラム ID フィールドを bigint として処理します。データ統合サーバーは ID を integer として処理します。

ノーマライザフィールドマッピング

受信フィールドを正規化されたフィールドにマッピングするには、ノーマライザトランスフォーメーションの **[フィールドマッピング]** タブを使用します。

ノーマライザのフィールドマッピングを設定するには、次の手順を実行します。

1. 正規化する複数回出現受信フィールドを、ノーマライザトランスフォーメーションが作成した対応する出力フィールドにマッピングします。

   注: 各正規化されたフィールドのセットについて最低 1 つの出力フィールドをマッピングしてください。
2. 受信フィールドを 1 回出現のすべての正規化されたフィールドにマッピングします。

ノーマライザフィールドマッピングのオプション

ノーマライザの **[フィールドマッピング]** タブでは次のフィールドマッピングオプションを使用できます。

フィールドの表示

[受信フィールド] リストに表示されるフィールドを制御します。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示します。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

[正確なフィールド名] と [スマートマップ] を同じフィールドマッピングで使用できます。例えば、[正確なフィールド名] を使用して同じ名前のフィールドを照合してから、[スマートマップ] を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

アクションメニュー

追加フィールドリンクオプション。次のオプションが用意されています。

- 選択項目をマップ。選択した受信フィールドと選択したターゲットフィールドをリンクします。
- 選択項目をマップ解除。選択したフィールドのリンクをクリアします。
- マッピングのクリア。すべてのフィールドマッピングをクリアします。

詳細プロパティ

ノーマライザトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルやトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

ノーマライザトランスフォーメーションのターゲット設定

マッピングにノーマライザトランスフォーメーションを使用する場合は、ターゲットおよびターゲットトランスフォーメーションで次の設定ガイドラインに留意してください。

- 正規化されたデータをターゲットに書き込む場合は、ターゲットトランスフォーメーションのフィールドマッピングで、複数回出現フィールドをターゲットフィールドにマッピングします。
- 生成キーまたは生成カラム ID をターゲットデータに含めるには、必要に応じて追加ターゲットフィールドを作成してから、ターゲットトランスフォーメーションのフィールドマッピングでフィールドをマッピングします。

注: 生成されたキーは、詳細モードでは適用できません。

パラメータ化されたソースのノーマライザフィールドルール

パラメータをソースオブジェクトとして使用してノーマライザトランスフォーメーションのデータを提供するには、ノーマライザで **[名前付きフィールド]** フィールドルールを使用して受信フィールドを定義します。

ノーマライザトランスフォーメーションを設定する際には、受信フィールドと正規化されたフィールド間のフィールドマッピングを定義します。ソースオブジェクトにパラメータを使用した場合は、[名前付きフィールド] ルールを使用してフィールドを追加するまで、フィールド名が受信フィールドのリストに表示されません。

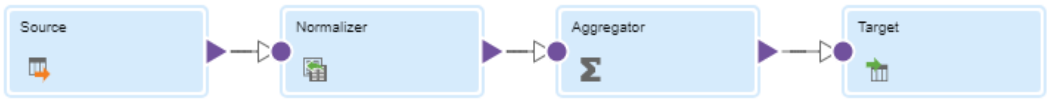
フィールドを追加するには、ノーマライザトランスフォーメーションの **[受信フィールド]** タブで、**[名前付きフィールド]** ルールを作成します。**[名前付きフィールドを含める]** ダイアログボックスで **[追加]** をクリックして、受信フィールドの名前を入力します。

ノーマライザトランスフォーメーションで使用するすべてのソースフィールドを表すフィールドを作成します。

ノーマライザとアグリゲータのマッピング例

あるリレーショナルテーブルに、各店舗の四半期の販売台数が格納されています。各店舗の年間販売台数を計算するには、ノーマライザトランスフォーメーションを使用して各四半期の行を作成します。次に、アグリゲータトランスフォーメーションを使用して各ストアの四半期の販売台数を合計します。

次の図は、作成するマッピングを示しています。



ソースデータには次の行が含まれています。

StoreNo	Q1	Q2	Q3	Q4	Year
1001	30	50	48	80	2014
1022	100	120	125	140	2014
1190	80	108	123	134	2014

データがアグリゲータトランスフォーメーションに渡される前に、次のようにノーマライザトランスフォーメーションを使用してソースデータをピボット化します。

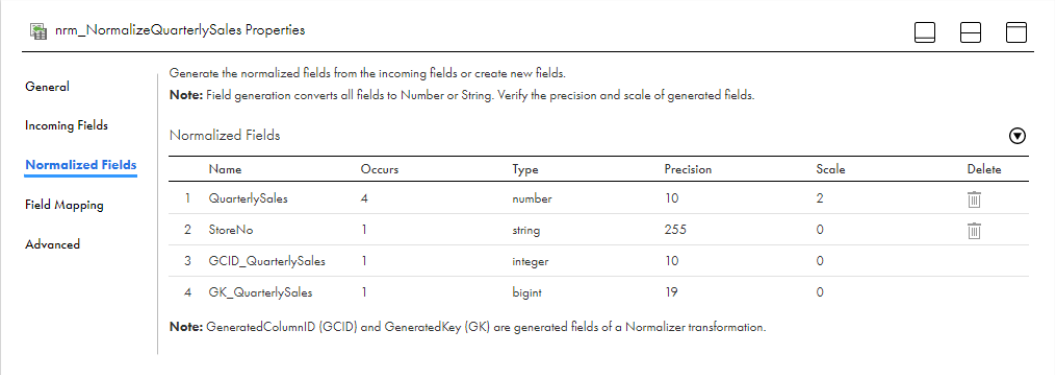
```
StoreNo QuarterlySales
1001          30
1001          50
1001          48
1001          80
1022         100
1022         120
1022         125
1022         140
1190          80
1190         108
1190         123
1190         134
```

正規化されたフィールドの定義

［正規化されたフィールド］タブで、「QuarterlySales」という名前の正規化されたフィールドを作成します。このフィールドが 4 つのフィールドであることを示すために、出現回数値を 4 に設定します。

マッピングに店舗番号データを含めるため、メニューから**［受信フィールドから生成］**を選択し、**［StoreNo］**を選択します。このフィールドには複数回出現データは含まれないため、デフォルトの出現回数である 1 を使用します。

次の画像は、両方のフィールドを追加した後の**［正規化されたフィールド］**タブを示しています。



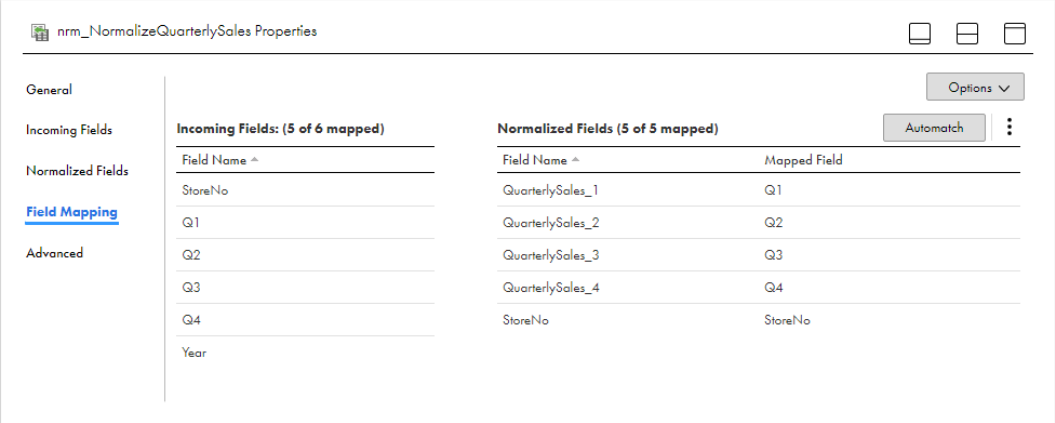
QuarterlySales の出現回数を 4 に設定すると、ノーマライザによって生成カラム ID フィールドと生成キーフィールドが作成されます。

ノーマライザフィールドマッピングの設定

ノーマライザトランスフォーメーションの**［フィールドマッピング］**タブで、受信フィールドを正規化されたフィールドに接続します。

［正規化されたフィールド］リストで、ノーマライザによって、複数回出現フィールド QuarterlySales が、正規化されたデータを保持する対応する各フィールド QuarterlySales_1、QuarterlySales_2、QuarterlySales_3、QuarterlySales_4 で置換されます。このリストには、StoreNo フィールドも含まれています。

次の画像に示すとおり、各受信フィールドを正規化されたフィールド StoreNo および QuarterlySales に接続します。



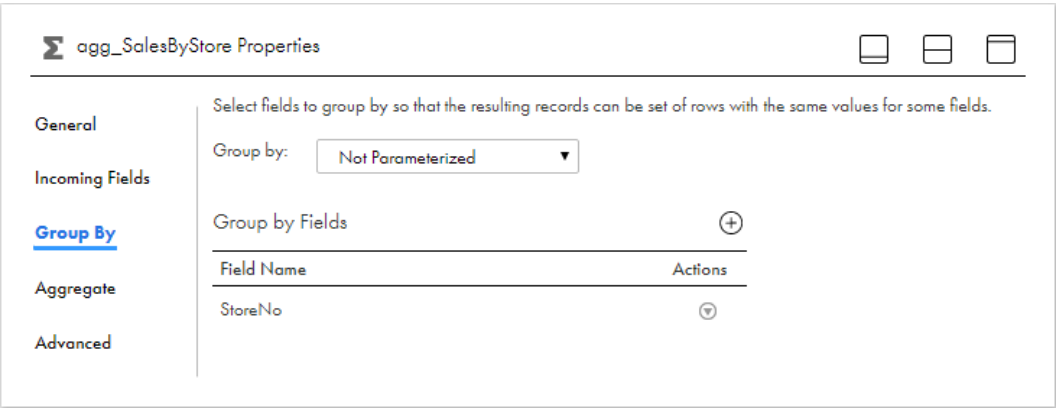
アグリゲータトランスフォーメーションの設定

店舗別の年間売上高を計算するには、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追加して、ノーマライザをアグリゲータに接続します。

アグリゲータトランスフォーメーションで、デフォルトの［すべてのフィールド］ルールを使用して、ノーマライザからアグリゲータにすべてのフィールドを渡します。

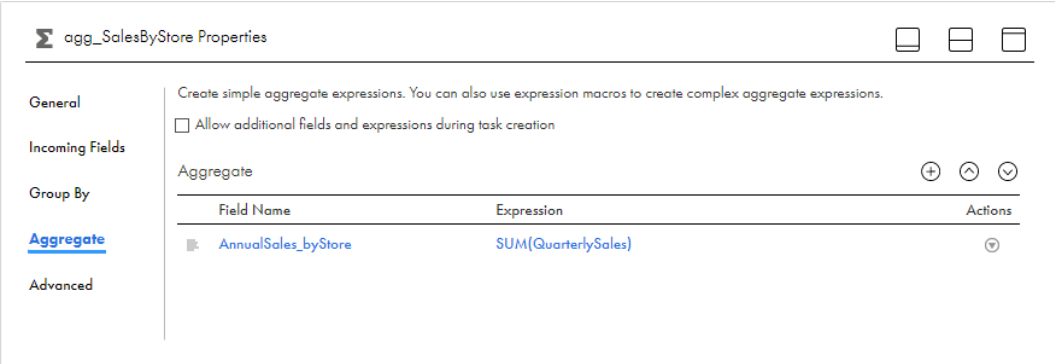
店舗番号でデータをグループ化するには、**[Group By]** タブで Group By フィールドを追加して、**[StoreNo]** フィールドを選択します。

次の画像は、**[Group By]** タブと StoreNo の Group By フィールドを示しています。



[集計] タブで、AnnualSales_byStore という名前の 10 進数出力フィールドを作成します。出力フィールドを設定するには、次の集計式で QuarterlySales フィールドを使用します。`SUM(QuarterlySales)`QuarterlySales フィールドは、すべての正規化された四半期データを表しています。

次の画像は、**[集計]** タブと AnnualSales_byStore 出力フィールドを示しています。



ターゲットの設定

ターゲットトランスフォーメーションを追加し、アグリゲータトランスフォーメーションをターゲットトランスフォーメーションに接続します。

デフォルトの［すべてのフィールド］ルールを使用して、アグリゲータトランスフォーメーションからターゲットトランスフォーメーションにすべてのフィールドを渡します。

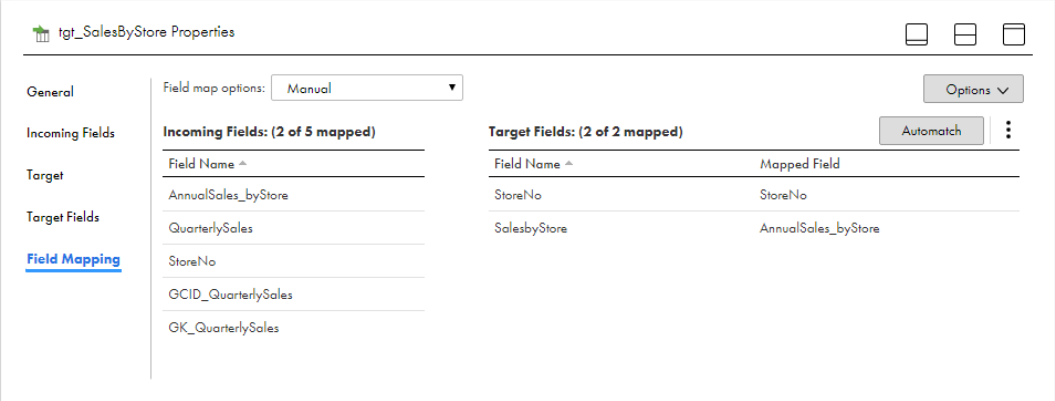
[ターゲット] タブで、ターゲット接続とターゲットオブジェクトを選択します。

[フィールドマッピング] タブの受信フィールドリストには、アグリゲータトランスフォーメーションによって作成された AnnualSales_byStore フィールドと、ソースからマッピングを介して渡された StoreNo フィールドが含まれています。

受信フィールドリストには、QuarterlySales フィールドとノーマライザによって作成された生成キーカラムも含まれています。これらのフィールドはターゲットに書き出す必要はありません。

StoreNo と AnnualSales_byStore フィールドを対応するターゲットフィールドに接続します。

次の画像は、設定済みの **[フィールドマッピング]** タブを示しています。



タスク結果

タスクを実行すると、マッピングタスクによってソースデータが正規化され、四半期ごとに 1 行が作成されます。正規化されたデータは店舗別にグループ化され、各店舗の四半期ごとの販売台数が集計されます。

次のデータがターゲットに書き出されます。

StoreNo	SalesbyStore
1001	208
1022	485
1190	445

第 23 章

出力トランスフォーメーション

出力トランスフォーメーションは、マップレットから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すために使用するパッシブトランスフォーメーションです。

出力トランスフォーメーションに出力フィールドを追加して、マップレットから渡すデータフィールドを定義します。各出力トランスフォーメーションに、少なくとも 1 つの出力フィールドを追加する必要があります。マップレットに複数の出力トランスフォーメーションを追加することができます。各出力トランスフォーメーションは、マップレットトランスフォーメーションでマップレットを使用する場合の出力グループになります。少なくとも 1 つの出力グループをダウンストリームトランスフォーメーションに接続する必要があります。1 つの出力グループを複数のダウンストリームトランスフォーメーションに接続できます。

マップレットで出力トランスフォーメーションを使用できますが、マッピングでは使用できません。

出力フィールド

出力トランスフォーメーションに出力フィールドを追加して、マップレットからダウンストリームトランスフォーメーションに渡すデータフィールドを定義します。各出力トランスフォーメーションに、少なくとも 1 つの出力フィールドを追加する必要があります。

出力フィールドをプロパティパネルの **[出力フィールド]** タブに追加します。フィールドを追加するには、**[フィールドの追加]** をクリックして、フィールド名、データ型、精度、およびスケールを入力します。

マップレットトランスフォーメーションでマップレットを使用するときは、ダウンストリームトランスフォーメーションに少なくとも 1 つの出力フィールドをマッピングします。

フィールドマッピング

フィールドをマッピングして、アップストリームトランスフォーメーションから出力トランスフォーメーションにデータがどのように移動するかを設定します。

[フィールドマッピング] タブでフィールドマッピングを設定します。

[フィールドマッピング] タブには、受信フィールドおよびターゲットフィールドのリストが含まれています。

注: マッピングされたフィールド名は、最長で 72 文字にできます。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

マップレットトランスフォーメーションにフィールドをマッピングする方法。次のいずれかのオプションを選択します。

- 手動。受信フィールドをマップレットトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。

オプション

［受信フィールド］リストと**［出力フィールド］**リストでフィールドが表示される方法を制御します。以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

アクションメニュー

追加フィールドマッピングオプション。次のオプションが用意されています。

- 選択項目をマップ。選択した受信フィールドを、選択したマップレットマップレット入力フィールドにリンクします。
- 選択項目をマップ解除。選択したフィールドのリンクをクリアします。
- マッピングのクリア。すべてのフィールドマッピングをクリアします。

第 24 章

解析トランスフォーメーション

解析トランスフォーメーションによって、Data Quality で作成した解析アセットがマッピングに追加されます。

解析アセットは、入力フィールドのトークンをその内容または構造に基づいて識別する一連の操作を定義します。解析操作におけるトークンは、個別の単語または文字列です。

解析トランスフォーメーションは、トークンをアセットで指定された出力フィールドに解析します。トランスフォーメーションを設定する際に、入力フィールドをトランスフォーメーションの適切なターゲットフィールドにマッピングします。マッピングの実行時に、トランスフォーメーションは入力フィールドで解析条件を満たすトークンを検索し、そのトークンを関連する出力フィールドに書き込みます。トランスフォーメーションで入力データ値を特定できるが、定義された出力フィールドが使用できない場合、オーバーフローデータの定義済みのフィールドにその値が書き込まれることがあります。トランスフォーメーションで入力データの値を特定できない場合、値が未解析データの定義済みのフィールドに書き込まれることがあります。トランスフォーメーションに追加するアセットによって、トランスフォーメーションが作成するオーバーフローデータフィールドと未解析データフィールドの数が決まります。

解析トランスフォーメーションは、別の所で設計したデータトランスフォーメーションロジックをマッピングに追加できるという点において、マップレットトランスフォーメーションと類似しています。マップレットと同様に、解析アセットは再利用可能なアセットです。

解析トランスフォーメーションでは、受信フィールドと発信フィールドが示されます。解析アセットに含まれるロジックは表示されず、解析アセットを編集することもできません。解析アセットを編集するには、解析アセットを Data Quality で開きます。

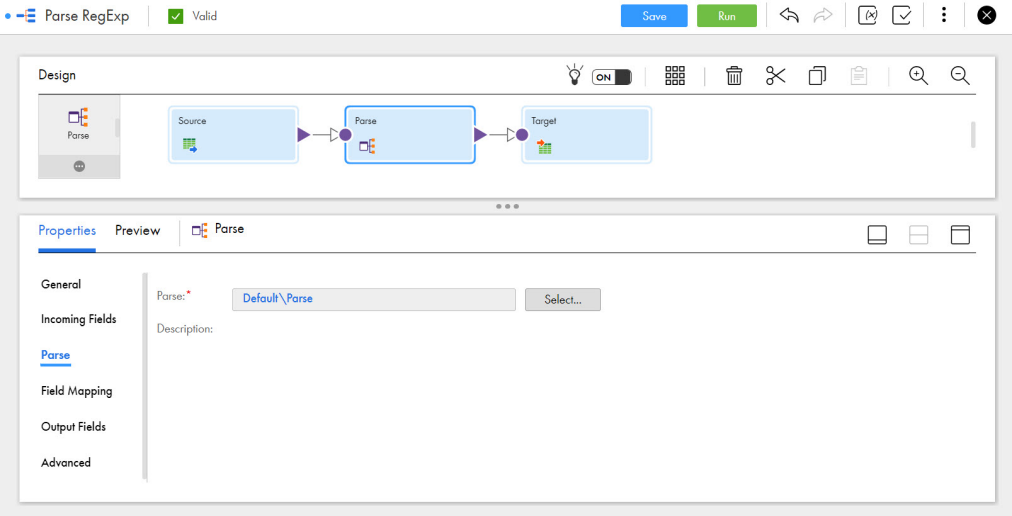
解析トランスフォーメーションの設定

マッピングの解析トランスフォーメーションを設定する場合は、最初に、トランスフォーメーションに含めるロジックが含まれている解析アセットを選択します。次に、トランスフォーメーションの受信フィールドを解析アセットで指定されたターゲットフィールドにマッピングします。

トランスフォーメーションを設定するには、以下のタスクを実行します。

1. 解析トランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. **［解析］**タブで、トランスフォーメーションに含める解析アセットを選択します。

次の図は、解析アセットを選択するために使用するオプションを示しています。



3. **[受信フィールド]** タブで、受信フィールドを検証します。

   デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。

4. **[フィールドマッピング]** タブで、受信フィールドをターゲットフィールドに接続します。

   解析アセットの入力名に入力フィールドの名前が反映される場合があります。この場合、**[自動マップ]** オプションを使用してフィールドの接続を行うことができます。

5. **[出力フィールド]** タブの出力フィールドのプロパティを確認します。

   トランスフォーメーションの出力フィールドの詳細については、「解析トランスフォーメーションの出力フィールド」 (ページ 331)を参照してください。

6. **[全般]** タブでは、オプションとして、解析トランスフォーメーションの名前を変更したり、説明を追加したりすることもできます。**[詳細]** タブで、トランスフォーメーションのトレースレベルを更新することもできます。デフォルトのトレースレベルはノーマルです。

注: アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。データ品質アセットの同期の詳細については、「「データ品質アセットの同期」 (ページ 102)」を参照してください。

解析トランスフォーメーションのフィールドマッピング

データがアップストリームトランスフォーメーションから解析トランスフォーメーションに移行する方法を定義するように、フィールドマッピングを設定します。**[プロパティ]** パネルの **[フィールドマッピング]** タブでフィールドマッピングを設定します。

注: 受信フィールドを解析アセットの 1 つのフィールドにマッピングします。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドがあれば、そのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。
  トランスフォーメーションの解析アセットがパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、[すべてパラメータを使用します] オプションを選択します。
- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。[すべてパラメータを使用します] または [部分的にパラメータを使用します] をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは *フィールドマッピング* 型である必要があります。

1 つのマッピング内の複数の解析トランスフォーメーションに同じフィールドマッピングパラメータを使用しないでください。

オプション

[受信フィールド] リストと **[ターゲットフィールド]** リストでフィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。[手動] または [部分的にパラメータを使用します] フィールドマップオプションを選択すると、[自動マップ] オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。
- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合では Cust_Name フィールドと Customer_Name フィールドが自動的にリンクされます。

[自動マップ] > **[自動マップを取り消す]** をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで **[アクション]** > **[マップ解除]** をクリックします。選択した 1 つ以上のフィールドのマッピングを解除するには、[ターゲットフィールド] コンテキストメニューで **[選択項目をマップ解除]** をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、[ターゲットフィールド] コンテキストメニューで **[マッピングのクリア]** をクリックします。

解析トランスフォーメーションの出力フィールド

解析トランスフォーメーションは、解析アセットで指定された出力フィールドを作成します。出力フィールドのタイプと数は、ユーザーが解析アセットで設定する解析操作によって異なります。出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。

解析トランスフォーメーションは、次のタイプの出力フィールド（一部またはすべて）を作成できます。

解析済みデータフィールド

アセットで定義されている解析条件を満たすデータ値が含まれます。

オーバーフローフィールド

解析条件を満たしているものの、対応する出力フィールドが使用できないデータ値が含まれます。値用の適切なすべての出力フィールドがすでに入力されている場合、解析トランスフォーメーションはその値をオーバーフローフィールドに書き込みます。

未解析フィールド

アセットで定義されている解析条件を満たさない値が含まれるフィールドです。

解析トランスフォーメーションの出力フィールドのルールおよびガイドライン

トランスフォーメーションの出力フィールドを確認するときは、次のルールとガイドラインを考慮してください。

- 出力フィールドのタイプと数は、ユーザーが解析アセットで設定する解析操作によって異なります。
- アセットで正規表現またはディクショナリが指定されている場合は、トランスフォーメーションにより、各正規表現またはディクショナリが正常に解析されたデータ用の 1 つ以上の出力フィールドが作成されます。

  アセットを設定するユーザーが、各正規表現またはディクショナリの操作用の出力フィールド数を決定します。各正規表現またはディクショナリの操作は、アセット設定の*ステップ*と呼ばれます。
- アセットでパターンベースの解析が指定されている場合は、トランスフォーメーションにより、パターンロジックで検出される可能性のある情報のタイプを表す一連の出力フィールドが作成されます。

  パターンベースの解析操作では、次の情報のタイプの出力フィールドを生成できます。

  - 名、姓、名前の接頭語、名前の接尾辞などの人名。
  - 性別、正式なあいさつ、略式のあいさつなどの派生情報。
  - 解析操作により入力データ行で識別されたパターンを表すラベル値。Data Quality ユーザーは、ラベルを使用してアセットのパターンロジックを強化できます。

  解析されたデータの出力フィールドの数は、アセットロジックにより決定されます。
- アセットで正規表現またはディクショナリが指定されている場合は、トランスフォーメーションにより、すべてのオーバーフローデータ用の単一のオーバーフローフィールドが作成されることがあります。または、トランスフォーメーションにより、アセットで定義されている正規表現またはディクショナリごとにオーバーフローフィールドが作成されることもあります。ユーザーはアセットプロパティを更新して、オーバーフローフィールドのポリシーを決定できます。

  アセットでパターン解析操作が指定されている場合は、トランスフォーメーションにより、単一のオーバーフローフィールドが作成される場合と作成されない場合があります。オーバーフローフィールドの有無は、アセットで入力データに指定されているロケールによって異なります。例えば、アセットでロケールがポルトガルまたはブラジルとして指定されている場合は、解析トランスフォーメーションによりパターン解析操作用のオーバーフローフィールドが作成されます。このロケールは Data Quality ユーザーが設定します。

- アセットで正規表現またはディクショナリが指定されていない場合は、トランスフォーメーションにより、すべての未解析データ用の単一のフィールドが作成されます。

  アセットでパターン解析操作が指定されている場合は、トランスフォーメーションにより、未解析データフィールドが作成される場合と作成されない場合があります。パターンベースの解析における未解析データフィールドの有無は、アセットで入力データに指定されているロケールによって異なります。

詳細プロパティ

解析トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。

第 25 章

Python トランスフォーメーション

詳細モードでは、Python トランスフォーメーションを使用することで、Python 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できます。Python トランスフォーメーションは、アクティブまたはパッシブなトランスフォーメーションにすることができます。

Python トランスフォーメーションを使用して、単純なトランスフォーメーション機能または複雑なトランスフォーメーション機能を定義できます。Python トランスフォーメーションを機械学習の実装に使用することもできます。例えば、トレーニング済みモデルをリソースファイルからロードし、そのモデルを使用して入力データを分類したり、予測を作成したりすることができます。

Python トランスフォーメーションを作成するには、次のタイプの Python コードスニペットを記述します。

- 入力行を処理する前に 1 回実行される、パーティション化前の Python コード。
- トランスフォーメーションが入力行を受け取ったときに実行される、メイン Python コード。
- 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後に実行される、パーティション化後の Python コード。

Python トランスフォーメーションを使用するには、組織が適切なライセンスを所有している必要があります。

Graviton 対応クラスタでは Python トランスフォーメーションは使用できません。Graviton 対応クラスタの詳細については、「Administrator」のヘルプを参照してください。

注: Python トランスフォーメーションを作成する場合は、マッピングタスクでコードを実行する前に Python コードを確認し、クエリ、リモートスクリプト、データ接続などの潜在的な危険性を持つアクティブコンテンツがないことを確認してください。

Python のインストールと設定

Python コードをインストールして設定し、Secure Agent マシンで実行します。詳細セッションプロパティの設定や Python パッケージのインストールなどのタスクを設定します。

注: サーバーレスランタイム環境を使用する場合はカスタム Python インストールを作成できないため、これらの手順は適用されません。

Python をインストールして設定するには、以下の手順を実行します。

1. Python トランスフォーメーションでサードパーティ製ライブラリを使用できるようにするには、ランタイム環境が Python のインストールと参照されるリソースファイルにアクセスできることを確認してください。
2. 必須パッケージを Secure Agent マシンにインストールします。
3. Secure Agent マシンにインストールする Python のバージョンをダウンロードし、ファイルを抽出します。

4. インストールディレクトリを準備します。
5. インストールディレクトリをコンパイルおよび構築します。
6. 環境変数を設定して、Secure Agent マシンがインストールにアクセスできるようにします。
7. インストールに正しいフォルダが含まれていることを確認します。
8. インストールでサードパーティ製ライブラリを使用している場合は、それらのライブラリをインストールします。
9. インストールを Secure Agent マシンにコピーします。
10. インストールでサードパーティ製ライブラリを使用している場合は、それらのインストールを確認します。

手順 1。サードパーティ製ライブラリへのアクセスの有効化

Python トランスフォーメーションをサードパーティ製ライブラリとともに使用するには、ランタイム環境から Python インストールと Python コードで参照するリソースファイルにアクセスできる必要があります。

Python をインストールするときに、アクセスするサードパーティ製ライブラリを Python コードに含めることができます。

マッピングタスクで、詳細セッションプロパティを使用して、Python インストールディレクトリと Python 実行可能ファイルの場所を指定します。

1. マッピングタスクの詳細セッションプロパティで、**［追加］**をクリックします。
2. セッションプロパティ名として**［advanced.custom.property］**を選択します。
3. セッションプロパティの値として、次のテキストを入力します。

   ```
   infaspark.pythontx.executorEnv.PYTHONHOME=$INFA_HOME/python&:infaspark.pythontx.exec=$INFA_HOME/python/bin/python3
   ```
4. **［完了］**をクリックします。

手順 2。必須パッケージのインストール

Python をインストールする前に、Secure Agent マシンに必要なパッケージがあることを確認してください。

次の表に、Secure Agent マシンにパッケージをインストールするためのコマンドを示します。

パッケージ	コマンド
OpenSSL	`sudo yum install openssl-devel`
zlib	`sudo yum install zlib-devel`
libffi	`sudo yum install libffi-devel`

手順 3。Python ディストリビューションのダウンロード

Secure Agent マシンにインストールする Python のバージョンを見つけて、ファイルを抽出します。

1. ホームディレクトリに移動します。
2. Python ディストリビューションを Secure Agent マシンにダウンロードします。例えば、次のコマンドを実行して、インターネットから Python 3.6.5 のインストールを取得できます。

   ```
   wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
   ```
   このコマンドにより、ホームディレクトリに Python ディストリビューションをダウンロードします。

3. この tar ファイルを解凍するには、以下のコマンドを実行します。

   ```
   tar -xvf Python-3.6.5.tgz
   ```

   抽出したファイルがディレクトリ~/Python-3.6.5 に表示されます。

手順 4.Python をインストールするディレクトリの準備

Python インストールを含める場所を指定します。インストールディレクトリを準備するには、コマンドを実行してディレクトリを作成し、そのディレクトリを作業ディレクトリとして設定します。

1. ホームディレクトリの下に Python3 ディレクトリを作成するには、次のコマンドを実行します。

   ```
   mkdir Python3
   ```

2. $workingdir という環境変数を作成し、その値を現在の作業ディレクトリに設定するには、次のコマンドを実行します。

   ```
   export workingdir=`pwd`
   ```

3. $workingdir 環境変数を確認するには、次のコマンドを実行します。

   ```
   echo $workingdir
   ```

手順 5.Python をインストールします。

コマンドを実行して、Python インストールをコンパイルおよび構築します。

1. Python ディストリビューションを抽出した~/Python-3.6.5 ディレクトリに移動します。

   ```
   cd Python-3.6.5
   ```

2. $workingdir/Python3 の下に Python をコンパイルするには、次のコマンドを実行します。

   ```
   ./configure --enable-shared --prefix=$workingdir/Python3
   ```

3. Python インストールを構築するには、次のコマンドを実行します。

   ```
   make
   make install
   ```

手順 6.環境変数の設定

Python インストールにアクセスするように Secure Agent マシンを設定します。

1. PYTHONHOME 環境変数を設定するには、次のコマンドを実行します。

   ```
   export PYTHONHOME=$workingdir/Python3
   ```

   $PYTHONHOME は Python をインストールしたディレクトリであることに注意してください。

2. LD_LIBRARY_PATH 環境変数を設定するには、次のコマンドを実行します。

   ```
   export LD_LIBRARY_PATH=$PYTHONHOME/lib
   ```

3. PATH 環境変数を設定するには、次のコマンドを実行します。

   ```
   export PATH=$PYTHONHOME/bin:$PATH
   ```

手順 7.Python インストールの確認

インストールフォルダの場所を確認します。

Python インストールディレクトリに正しいフォルダが含まれていることを確認します。

1. ディレクトリを$PYTHONHOME に変更するには、次のコマンドを実行します。

   ```
   cd $PYTHONHOME
   ```

2. $PYTHONHOME に次のフォルダが含まれていることを確認するには、ls コマンドを実行します。

```
bin
include
lib
share
```

手順 8。サードパーティ製ライブラリのインストール（オプション）

Python インストールで必要な追加ライブラリを特定してインストールします。

Python のインストールでサードパーティ製ライブラリを使用している場合は、pip コマンドを使用して $PYTHONHOME にインストールします。

例えば、次のコマンドを実行して、numpy と scikit-learn をインストールできます。

```
bin/pip3 install numpy
bin/pip3 install scikit-learn
```

手順 9.Python を Secure Agent マシンにコピーします

Python インストールを Secure Agent マシンにコピーします。

Python のインストールを完了するには、次のコマンドを実行します。

1. 次のディレクトリが存在しない場合は、Secure Agent マシンに作成します。

   ```
   <Secure Agent installation directory>/ext/python/
   ```

2. Python インストールを Secure Agent マシンにコピーするには、次のコマンドを実行します。

   ```
   cp -r $PYTHONHOME/* <Secure Agent installation directory>/ext/python/
   ```

3. Informatica ドメインのコンテンツを確認するには、次のコマンドを実行します。

   ```
   ll <Secure Agent installation directory>/ext/python/
   ```

4. コンテンツに次のフォルダが含まれていることを確認します。

   ```
   bin
   include
   lib
   share
   ```

手順 10.必要に応じて、サードパーティ製ライブラリのインストールを確認します

Python のインストールでサードパーティ製ライブラリを使用している場合は、それらのライブラリが Secure Agent マシンにインストールされていることを確認します。

1. ディレクトリを Secure Agent マシンに変更するには、次のコマンドを実行します。

   ```
   cd <Secure Agent installation directory>/ext/python/
   ```

2. Python インタプリターを呼び出すには、次のコマンドを実行します。

   ```
   bin/python3
   ```

3. 次の Python コードを実行します。

   ```
   import numpy
   import sklearn
   ```

4. インストールした Python サードパーティ製ライブラリごとに import 文が成功することを確認します。

   Ctrl+D を押して、Python インタプリターを終了します。

Python トランスフォーメーションのフィールド

Python トランスフォーメーションには受信フィールドと出力フィールドがあります。受信フィールドと出力フィールドは、**[Python]** タブで定義している Python コードスニペットの変数として使用します。

[出力フィールド] タブで出力フィールドを追加します。

ダウンストリームトランスフォーメーションに渡す出力データの出力フィールドを追加します。フィールドを追加するには、**[フィールドの追加]** をクリックして、フィールド名、データ型、精度、およびスケールを入力します。Python エディタの **[出力]** タブで **[新しいフィールドの作成]** をクリックして、出力フィールドを作成することもできます。

トランスフォーメーションにフィールドを追加したら、フィールド名を Python コード内で変数として使用できます。

出力フィールド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。

- フィールド名には ASCII 文字のみを含むことができます。
- フィールド名に Python キーワードを使用しないでください。例えば、`import`、`global`、`class` などのフィールド名は使用しないでください。
- フィールド名に次の予約済み用語を使用しないでください: `resourceJepFile` または `resourceFilesArray`。
- ユーザー定義のデフォルト値は Python コードで設定します。
  例えば、`output_field = 'value'` と入力して、出力フィールド `output_field` のデフォルト値 `'value'` を設定できます。

1 つ以上の入力フィールドをパーティションキーとして設定できます。データ統合は、パーティションキーを使用して、コードの実行前にデータの再パーティション化を行います。受信フィールドをパーティションキーとして追加しない場合、データはデフォルトのパーティションスキームを使用して処理されます。

データ型の変換

Python トランスフォーメーションは、Python トランスフォーメーションのフィールドタイプに基づいて、トランスフォーメーションのデータ型を Python データ型に変換します。

Python トランスフォーメーションは、入力行を読み込むと、受信フィールドデータ型を Python データ型に変換します。Python トランスフォーメーションは、出力行を書き込むと、Python データ型を出力フィールドデータ型に変換します。

例えば、次の処理は、double データ型である入力フィールドに対して発生します。

1. Python トランスフォーメーションは受信フィールドの double データ型を Python の float データ型に変換します。
2. この Python トランスフォーメーションでは、受信フィールドの値を Python の float データ型の値として使用します。
3. トランスフォーメーションは、出力行を生成すると、Python の float データ型を double データ型に変換します。

Python 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のデータ型を Python のデータ型にマッピングする方法を次の表に示します。

トランスフォーメーションデータ型	Python **のデータ型**
Integer	Int
Decimal	Float

トランスフォーメーションデータ型	Python のデータ型
Double	Float
Timestamp	Datetime
String	Str
テキスト	テキスト

入出力フィールドのデータ型

Python トランスフォーメーションでは、対応する受信フィールドと出力フィールドのデータ型を同じにする必要があります。データ型が異なる場合は、Python コードでデータ型を変換します。

例えば、integer データ型の受信フィールドと string データ型の出力フィールドを作成します。受信フィールドのデータを処理し、そのデータを出力フィールドに書き込む Python コードを定義します。Python コードで str()関数を使用して、受信フィールドの integer データ型を変換し、その出力を string データ型として出力フィールドに書き込むことができます。

パーティションキー

パーティションキーを使用して、Python コードを実行する前にデータをパーティションにグループ化する方法を定義できます。

1 つ以上の入力フィールドをパーティションキーとして設定できます。Python トランスフォーメーションは、パーティションキーを使用して、コードの実行前にデータの再パーティション化を行います。受信フィールドをパーティションキーとして追加しない場合、データはデフォルトのパーティションスキームを使用して処理されます。

アクティブ Python トランスフォーメーションとパッシブ 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では、動作をアクティブまたはパッシブとして定義することにより、出力行の生成方法を指定します。**[詳細]** タブの動作を定義します。デフォルトで、Python トランスフォーメーションはアクティブです。

Python トランスフォーメーションは、以下のように、動作に基づいて出力行を処理します。

- アクティブ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できます。出力の行数を定義するには、コード内で generateRow()メソッドを呼び出して、各出力行を定義します。1 つの入力行から複数の出力行を生成するか、複数の入力行から 1 つの出力行を生成するかを選択することもできます。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 つの受信フィールドが含まれている場合は、generateRow()メソッドを呼び出して、開始日と終了日の間の各日付に出力行を生成できます。
- 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の前後で行数を変更できません。このトランスフォーメーションでは、各入力行を処理した後に generateRow()メソッドを呼び出して、出力行を生成します。

リソースファイル

Python トランスフォーメーションでは、リソースファイルと Python コードを使用して、トランスフォーメーション機能を定義します。トレーニング済みモデルを使用する場合は、そのトレーニング済みモデルをリソースファイルとして Python トランスフォーメーションで指定します。

Python トランスフォーメーションには次のコンポーネントがあります。

リソースファイル

Python コードでアクセスするリソースが含まれるファイル。

このファイルは、データ統合の外部で大規模なデータセットに対してトレーニングしたトレーニング済みモデルにすることができます。トレーニング済みモデルを使用してデータを分類したり、Python トランスフォーメーションに渡すデータに基づいて予測を作成したりすることができます。トレーニング済みモデルには Python コードからアクセスします。

ランタイム環境

ランタイム環境のタイプに基づいてリソースファイルを追加します。Python コードでリソースファイルを参照する場合は、リソースファイルを同じディレクトリに追加します。一貫性を維持するため、python_resources という名前の専用フォルダにリソースファイルを保存できます。

次のガイドラインを考慮します。

- Secure Agent マシンが予期せず停止してエージェントが別のマシンで再起動した場合は、新しいマシンの同じディレクトリに Python インストールおよびリソースファイルを追加する必要があります。
- Secure Agent マシンで Python インストールまたはリソースファイルを更新した場合は、次に詳細モードでジョブを実行したときにファイルが有効になります。
- 長時間実行するジョブの失敗を防ぐために、Secure Agent マシン上のファイルはジョブの実行中に 5 回以上更新しないようにしてください。

サーバーレスランタイム環境

補足ファイルの場所にリソースファイルを追加します。

Python インストールファイルまたはリソースファイルを更新した場合、変更を有効にするにはサーバーレスランタイム環境を再デプロイする必要があります。

補足ファイルの場所の詳細については、「Administrator」のヘルプにある「*Administrator*」を参照してください。

Python **コード**

Python 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渡すデータを処理するために使用する Python コード。Python コードを記述するときは、入力変数を再構築するか、トレーニング済みのモデルをロードするか、出力変数を定義します。

Python コードの開発

Python トランスフォーメーションの機能を定義するには、**[Python]** タブで Python コードスニペットを入力します。コードスニペットを入力して、入力変数を再構築し、トレーニング済みのモデルをロードし、出力変数を定義し、追加のトランスフォーメーション機能を定義します。

Python エディタの以下のセクションで Python コードスニペットを入力します。

Pre-Partition Python **コード**

一度解釈されすべてのデータ行間で共有されるコードを定義します。

次のタスクを実行する場合は［**Pre-Partition Python コード**］セクションを使用します。

- import 文の宣言。
- 変数の宣言。
- 変数の初期化。
- ヘルパーメソッドの定義。

メイン Python **コード**

パーティションの処理中に入力行を受信した場合の Python トランスフォーメーションの動作方法を定義します。Python トランスフォーメーションは、［**メイン Python コード**］セクションのコードをパーティションごと、行ごとに処理します。

Post-Partition Python **コード**

パーティション内のすべての入力データを処理した後に Python トランスフォーメーションの動作方法を定義します。`generateRow()`メソッドを呼び出して出力行を生成します。

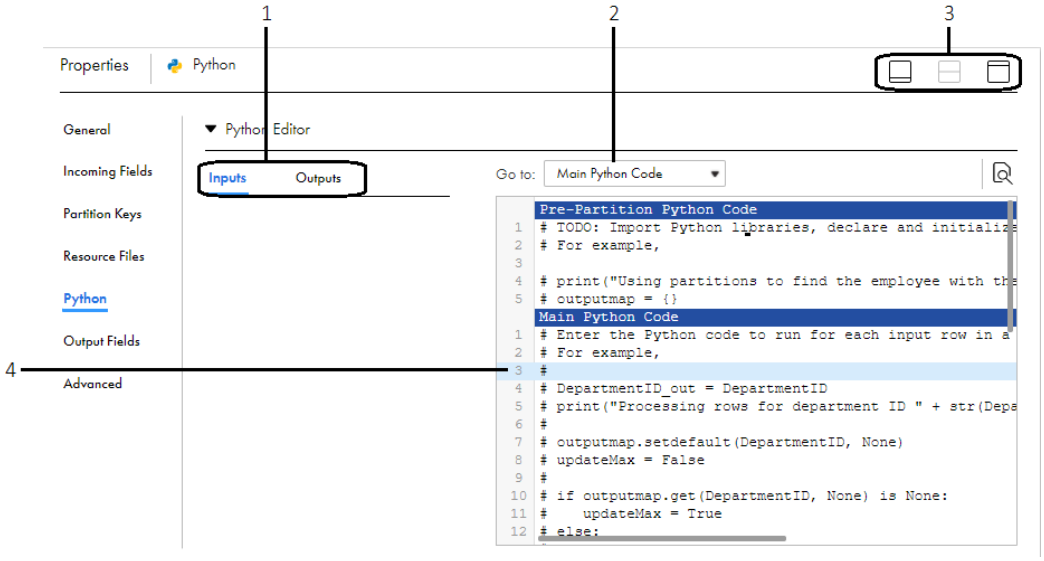
Python コードを記述するときには、以下のガイドラインに従ってください。

- 使用する前に変数を定義します。例えば、変数が［**メイン Python コード**］セクションで定義されている場合、［**Pre-Partition Python コード**］セクションではその変数を参照できません。
- 受信フィールドにアクセスするには、受信フィールド名を呼び出します。
- Python コードは、値を各出力フィールドに割り当てる必要があります。
- トランスフォーメーションで受信フィールドから出力フィールドにデータを書き込む方法を定義するには、出力フィールドに受信フィールドの値を設定します。
  例えば、受信フィールド `incoming_field` のデータを出力フィールド `output_field` に書き込むには、`output_field = incoming_field` と記述します。
- リソースファイルにアクセスするには、`resourceFilesArray` 変数を使用します。リソースファイルを指定するときは、`resourceFilesArray[0]`のようにインデックスを使用します。
- Mapping Designer は Python コードを検証しません。

Python コードスニペットの作成

トランスフォーメーション機能を定義する Python コードスニペットを作成するには、**[Python]** タブで Python エディタを使用します。

次の画像は、**[Python]** タブと展開されている Python エディタを示しています。



1. ［入力］および［出力］タブ。これらのタブを使用して、受信フィールドおよび出力フィールドを変数として Python コードスニペットに追加します。これらのタブに表示されるフィールドおよびメソッドは、コードエントリ領域でどのセクションが選択されているかに基づいて異なります。
2. ［移動］リストです。コードエントリ領域でセクション間の切り替えに使用します。
3. ［最小化］、［両方とも開く］、および［最大化］アイコンです。［最小化］および［最大化］ボタンを使用すると、トランスフォーメーションのプロパティを最小化および最大化できます。［両方とも開く］アイコンを使用すると、Mapping Designer のキャンバスとトランスフォーメーションのプロパティを同時に開けます。
4. コードエントリ領域です。Python コードスニペットは、**[Pre-Partition Python コード]**、**[メイン Python コード]**、**[Post-Partition Python コード]** セクションに入力します。

ヒント: トランスフォーメーションのプロパティを展開してコードエントリ領域を詳しく表示するには、**[最大化]** をクリックします。

Python コードスニペットを作成するには、次のタスクを実行します。

1. **[移動]** リストで、コードスニペットを入力するセクションを選択します。
2. スニペットの受信フィールドまたは出力フィールドにアクセスするには、**[入力]** または **[出力]** タブでフィールドを選択し、**[追加]** をクリックします。

   また、**[出力]** タブで出力フィールドを作成し、**[新しいフィールドの作成]** をクリックしても作成できます。
3. セクションに基づいて適切な Python コードを書き込みます。

リソースファイルの参照

リソースファイルには、Python コードで使用するリソースが含まれています。

Python トランスフォーメーションでは、リソースファイルの相対パスを入力します。例えば、リソースファイルが<Secure Agent installation directory>/ext/python/folder1/myfile に保存されている場合、相対パスは/folder1/myfile になります。

Python コードでリソースファイルにアクセスするには、配列 resourceFilesArray[index]を参照します。特定のリソースファイルにアクセスするには、リソースファイルパスのリストにリソースファイルが表示される順序に従って、リソースファイルを識別するためのインデックスを指定します。

複数のリソースファイルを指定する場合、例えば、Python コードの最初のリソースファイルを参照するには resourceFilesArray[0]を使用します。2 番目のリソースを参照するには resourceFilesArray[1]を使用します。

例: ID カラムの非パーティション化データへの追加

センサを使用してシステムの健全性をモニタする太陽熱発電を組織によって運営します。現在、各センサはその位置で識別されます。これに代わって、ID を使用して各センサを識別し、今後のデータ分析を簡略化します。

センサ出力から次のデータを収集します。

SensorLocation	LastReadingTime
Area A	7/9/2019 11:36:09
Area B	7/9/2019 16:43:42
Area C	7/9/2019 13:23:53

ID カラムを追加して、ID 値を各センサに割り当てるには、次のタスクを実行します。

手順 1.Python **トランスフォーメーションを作成します。**

Python トランスフォーメーションを作成します。**[詳細]** タブで、動作を［パッシブ］に設定します。

手順 2.**データを** Python **トランスフォーメーションに渡します。**

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 Python トランスフォーメーションにデータを渡します。

データを Python トランスフォーメーションに渡すと、次の受信フィールドが含まれています。

- SensorLocation
- LastReadingTime

手順 3.**出力フィールドを作成します。**

Python トランスフォーメーションの **[出力フィールド]** タブを使用して、ID カラムを表す出力フィールド SensorID_out を作成します。

また、次の出力フィールドを作成して、受信データをダウンストリームトランスフォーメーションに渡します。

- SensorLocation_out

- LastReadingTime_out

手順 4.各行の ID 値を設定します。

[メイン Python コード] セクションで、次のコードを使用して処理される各行の ID 値を設定し、データを出力フィールドに書き込みます。

```
SensorID="".join(str(x) for x in map(ord, SensorLocation))

SensorID_out = SensorID
SensorLocation_out = SensorLocation
LastReadingTime_out = LastReadingTime
```

手順 5.マッピングを実行します。

Python トランスフォーメーションの出力フィールドが書き込みトランスフォーメーションに直接リンクされている場合、ターゲットにはマッピング実行後に次のデータが含まれます。

SensorID_out	SensorLocation_out	LastReadingTime_out
65114101973265	Area A	7/9/2019 11:30:00
65114101973266	Area B	7/9/2019 11:35:00
65114101973267	Area C	7/9/2019 11:40:00

例: 最も高い給与を検索するパーティションの使用

組織に HR スタッフメンバがいます。この HR スタッフメンバは、従業員が重要と考える人生の側面と従業員の給与の関連性をモデル化するプロジェクトに従事しています。このプロジェクトは、組織のウエルネス制度の一環です。この情報を使用して、ウエルネス制度を個人に合わせてカスタマイズしようとしています。

Python トランスフォーメーションを使用して、自部門で最も高い給与を得ている従業員を特定できます。

以下の表に、組織が収集できるデータを示します。

DepartmentName	DepartmentID	EmployeeName	SalaryIndex	EmployeeSince
HR	1	Jane Smith	500	2/16/2010
R&D	2	Ellioth Consar	150	3/29/2018
Finance	3	Concor Valashe	230	11/22/2007
Marketing	4	Manchini Voliore	800	5/17/2009
HR	1	Blaze Concave	501	8/25/2016
R&D	2	Janet Encarr	890	1/26/2019
HR	1	Chelsea Blanch	389	9/3/2018
R&D	1	Samuel Coin	10	1/26/2005

Python トランスフォーメーションを使用して、自部門で最も高い給与を得ている従業員を特定するには、次のタスクを実行します。

手順 1.Python **トランスフォーメーションをマッピングに追加します。**

Python トランスフォーメーションを作成します。**[詳細]** タブで、動作を［アクティブ］に設定します。

手順 2.**データを** Python **トランスフォーメーションに渡します。**

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 Python トランスフォーメーションに次のフィールドを渡します。

- DepartmentName
- DepartmentID
- EmployeeName
- SalaryIndex
- EmployeeSince

手順 3.**データを部門別にパーティション化します。**

データを部門別にパーティション化して、各部門の最も高い給与を追跡します。データを部門別にパーティション化するには、**[パーティションキー]** タブで受信フィールド DepartmentID をパーティションキーとして追加します。

手順 4.**出力フィールドを作成します。**

[出力フィールド] タブで次の出力フィールドを作成し、データをダウンストリームトランスフォーメーションに渡すようにします。

- DepartmentName_out
- DepartmentID_out
- EmployeeName_out
- SalaryIndex_out
- EmployeeSince_out

手順 5.**マップを初期化します。**

各部門 ID とその部門の最も高い給与を得ている従業員を関連付けるマップ変数 outputmap を宣言します。

[Pre-Partition Python コード] セクションで次のコードを追加します。

```
print("Using partitions to find the employee with the highest salary")
outputmap = {}
```

手順 6.**データを処理するコードを定義します。**

Python トランスフォーメーションを通過する各入力行について、従業員の給与が処理済みの行の最高給与よりも高いかどうかを確認するコードを定義します。この従業員の給与の方が高い場合は、部門の最高給与を得ている従業員を更新します。

[メイン Python コード] セクションで次のコードを追加します。

```
DepartmentID_out = DepartmentID
print("Processing rows for department ID " + str(DepartmentID_out))

outputmap.setdefault(DepartmentID, None)
updateMax = False

if outputmap.get(DepartmentID, None) is None:
    updateMax = True
else:
    max_salary = outputmap[DepartmentID]['SalaryIndex']
    if max_salary is None:
        updateMax = True
    if SalaryIndex > max_salary:
        updateMax = True
```

```
if updateMax == True:
    employee_data = {'SalaryIndex':SalaryIndex,'EmployeeName':EmployeeName,
                     'EmployeeSince':EmployeeSince,'DepartmentName':DepartmentName}

    outputmap[DepartmentID] = employee_data
```

手順 7.データを出力ファイルに書き込みます。

[Python] タブの **[Post-Partition Python コード]** セクションで、マップ変数 outputmap のデータを使用し、各部門で最も高い給与を得ている従業員の行を生成します。

[Post-Partition Python コード] セクションで次のコードを追加します。

```
for x in outputmap:
    DepartmentID_out = x
    smap = outputmap[x]
    SalaryIndex_out = smap["SalaryIndex"]
    EmployeeName_out = smap["EmployeeName"]
    DepartmentName_out = smap["DepartmentName"]
    EmployeeSince_out = smap["EmployeeSince"]

    ## Generate the output row
    generateRow()
```

手順 8.マッピングを実行します。

Python トランスフォーメーションの出力フィールドがターゲットトランスフォーメーションに直接リンクされている場合、ターゲットにはマッピング実行後に次のデータが含まれます。

DepartmentName	DepartmentID	EmployeeName	SalaryIndex	EmployeeSince
Finance	3	Concor Valashe	230	11/22/2007
Marketing	4	Manchini Voliore	800	5/17/2009
HR	1	Blaze Concave	501	8/25/2016
R&D	2	Janet Encarr	890	1/26/2019

例: トレーニング済みモデルの運用可能化

あなたは製薬会社に勤めていて、心疾患の治療を改善できるよう、ジギタリスの花成に関するデータを調査しています。一般的なジギタリス *Digitalis purpurea* とケジギタリス *Digitalis lanata* のどちらが良好な予後になるのかを見つけようとしています。

研究を実施するには、花のガクと花弁の長さと幅に関するデータを花の種別に分類する必要があります。データを分類するため、データ統合の外部でトレーニング済みモデルを開発しました。

トレーニング済みモデルを運用化するには、次のタスクを完了します。

1. パッシブ Python トランスフォーメーションを含むマッピングを作成し、トレーニング済みモデルをリソースファイルとしてリストします。
2. トレーニング済みモデルにアクセスする Python スクリプトを記述します。
3. 花のガクと花弁に関するデータを Python トランスフォーメーションに渡して、ジギタリスの種別にデータを分類します。

次の表に、Python トランスフォーメーションに渡すことができるガクと花弁のサンプルデータを示します。

名前	タイプ	精度
sepal_length	decimal	10
sepal_width	decimal	10
petal_length	decimal	10
petal_width	decimal	10
true_class	string	50

パッシブ Python トランスフォーメーションでは次のコンポーネントを使用します。

リソースファイル

トレーニング済みモデルのパスをリソースファイルとして指定します。

例えば、次のパスのファイル foxgloveDataMLmodel.pkl に保存されたトレーニング済みモデルを使用します。

- Secure Agent マシン上の場所を基準にしたパス。

  例えば、リソースファイルが<Secure Agent インストールディレクトリ>/ext/python/folder1/foxgloveDataMLmodel.pkl にある場合、相対パスは/folder1/foxgloveDataMLmodel.pkl になります。
- サーバーレスランタイム環境の補足ファイルの場所。

/data/home/dtmqa/data/foxgloveDataMLmodel.pkl

Python **コード**

[Pre-Partition Python コード] および **[メイン Python コード]** セクションで Python コードを指定します。

[Pre-Partition Python コード] セクションを使用してライブラリをインポートし、リソースファイルをロードし、変数を初期化します。

例えば、次のコードを **[Pre-Partition Python コード]** セクションに入力します。

```
from sklearn import svm
from sklearn.externals import joblib
import numpy as np
clf = joblib.load(resourceFileArrays[0])
classes = ['common', 'woolly']
```

[メイン Python コード] セクションを使用して、Python トランスフォーメーションがトレーニング済みモデルを使用して各データ行を評価する方法を定義します。

例えば、次のコードを **[メイン Python コード]** セクションに入力します。

```
input = [sepal_length, sepal_width, petal_length, petal_width]
input = np.array(input).reshape(1,-1)
pred = clf.predict(input)
predicted_class = classes[pred[0]]
sepal_length_out = sepal_length
sepal_width_out = sepal_width
petal_length_out = petal_length
petal_width_out = petal_width
true_class_out = true_class
```

第 26 章

ランクトランスフォーメーション

ランクトランスフォーメーションで、データ範囲の上下限を選択します。ランクトランスフォーメーションを使用して、グループ内で最大または最小の数値を返します。また、ランクトランスフォーメーションを使用して、マッピングソート順の最上位または最下位にある文字列を返すこともできます。

例えば、ランクトランスフォーメーションを使用して地域別に上位 10 名の顧客を選択できます。または、給与やオーバーヘッドの費用が最も低い部署を 3 つ特定することもできます。

ランクトランスフォーメーションは 1 つの値ではなく値のグループを返すため、トランスフォーメーション関数 MAX および MIN とは異なります。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。

ランクトランスフォーメーションでは、パススルーする行数を変更できるため、これはアクティブなトランスフォーメーションです。例えば、トランスフォーメーションを設定して 100 行あるソースから上位 10 行を選択します。この場合、100 行をトランスフォーメーションに渡しますが、10 行のみランクトランスフォーメーションからダウンストリームトランスフォーメーションまたはターゲットに渡されます。

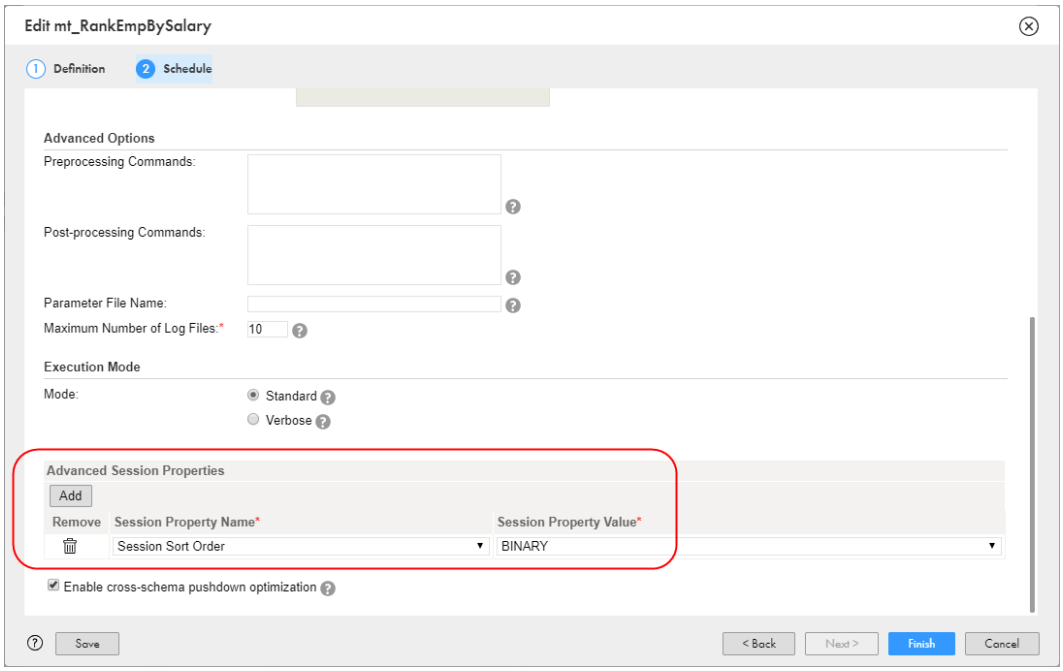
ランクトランスフォーメーションが含まれるマッピングを実行した場合、データ統合は、ランク計算を実行できるようになるまで入力データをキャッシュします。

文字列値のランク付け

文字列フィールド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。文字列フィールドをソートするには、データ統合でマッピングタスクに設定されているセッションのソート順を使用します。

マッピングタスクの詳細セッションプロパティで **[セッションソート順]** プロパティを設定します。バイナリまたはデンマーク語やスペイン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、データ統合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、データ統合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。

次の図に、マッピングタスクの詳細セッションプロパティの［**セッションソート順**］プロパティを示します。



ランクキャッシュ

ランクトランスフォーメーションが含まれるマッピングを実行すると、データ統合はデータおよびインデックスのキャッシュファイルを作成し、トランスフォーメーションを実行します。デフォルトでは、データ統合は、データ統合サーバーで Secure Agent $PMCacheDir プロパティに入力されたディレクトリにキャッシュファイルを格納します。

キャッシュディレクトリおよびキャッシュサイズは、ランクトランスフォーメーションの［**詳細**］タブで変更できます。

データ統合では、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 フィールドに基づいて行データを格納するデータキャッシュ。
- Group By フィールド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。

ランクトランスフォーメーションが含まれるマッピングを実行すると、データ統合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。複数のグループにわたってランク付けを実行するようにランクトランスフォーメーションを設定した場合、データ統合は、検出した各グループごとに段階的にランク付けを行います。

ソーストランスフォーメーション内に複数のパーティションを作成した場合、データ統合はパーティションごとの別々のキャッシュを作成します。

ランクトランスフォーメーションの定義

ランクトランスフォーメーションを定義するには、トランスフォーメーションを Mapping Designer にドラッグして、フィールドを設定し、トランスフォーメーションのプロパティを設定します。

1. Mapping Designer で、トランスフォーメーションパレットからキャンバスにランクトランスフォーメーションをドラッグし、それをアップストリームおよび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。
2. トランスフォーメーションフィールドを設定します。

   デフォルトでは、トランスフォーメーションはアップストリームトランスフォーメーションからすべての受信フィールドを継承します。すべての受信フィールドを使用する必要がない場合、フィールドルールを設定して、特定のフィールドを含めるまたは除外することができます。
3. ランクプロパティを設定します。

   ランク基準、ランク順序、およびランク付けする行数にするフィールドを選択します。
4. 必要に応じて、ランクグループを設定します。

   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。
5. 必要に応じて、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。

   キャッシュのプロパティ、ログメッセージのトレースレベル、トランスフォーメーション範囲、文字列の比較における大文字小文字の区別、およびトランスフォーメーションをオプションにするかどうかを更新できます。

ランクトランスフォーメーションのフィールド

ランクトランスフォーメーションは、アップストリームトランスフォーメーションから受信フィールドを継承します。ランクトランスフォーメーションを作成すると、データ統合によって［RANKINDEX］出力フィールドも作成されます。

ランクトランスフォーメーションでは、次のフィールドを使用します。

受信フィールド

受信フィールドは**［受信フィールド］**タブに表示されます。デフォルトでは、ランクトランスフォーメーションはアップストリームトランスフォーメーションからすべての受信フィールドを継承します。すべての受信フィールドを使用する必要がない場合は、フィールドルールを定義して、特定のフィールドを含めたり除外したりすることができます。フィールドルールの詳細については、「フィールドルール」 (ページ 25)を参照してください。

RANKINDEX

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。データ統合によって［RANKINDEX］フィールドが作成され、グループの各行のランクインデックス値が保存されます。

例えば、ランクトランスフォーメーションを作成して、月間総売上高の最も高い会社の小売店舗を 5 店特定します。売上高の最も高い店舗にはランクインデックス 1 が与えられます。次に売上高の高い店舗はランクインデックス 2、という順になります。総売上高が同じ店舗が 2 店ある場合は、ランクインデックスも同じとなり、トランスフォーメーションは次のランクインデックスをスキップします。

例えば、次のデータセットでは、ロングビーチとアナハイムの店舗の総売上高が同じため、同じランクインデックスが割り当てられています。

RANKINDEX	STORE	GROSS_SALES
1	Long Beach	100000
1	Anaheim	100000
3	Riverside	90000
4	Chula Vista	80050

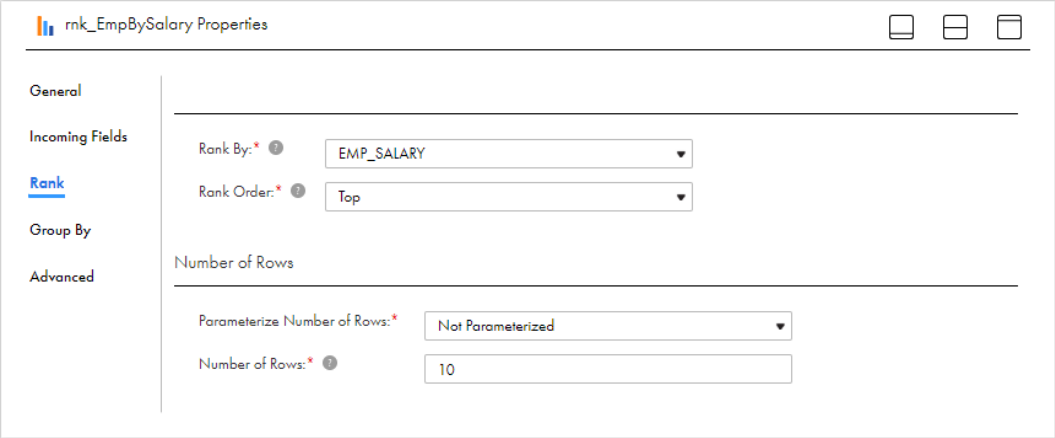
例えば目録内で値段の安い 10 個の製品といった最下位ランクを求める場合、ランクトランスフォーメーションは最下位から最上位の順にランクインデックスを割り当てます。したがって、最も値段の安い商品のランクインデックスには 1 が与えられます。

RANKINDEX は出力フィールドです。このフィールドは、ダウンストリームトランスフォーメーションの **[受信フィールド]** タブに表示されます。

ランクプロパティの定義

ランクトランスフォーメーションのランクプロパティを定義する場合、ランク基準にするフィールドを選択し、ランク順序を選択し、ランク基準にする行数を指定します。**[ランク]** タブでランクプロパティを定義します。

次の図は、**[ランク]** タブを示しています。



以下のプロパティを設定します。

ランク基準

[ランク基準] フィールドで、ランク付けに使用するフィールドを指定します。

例えば、各部門で給与支給額に基づいて上位 10 名の従業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ます。[EMP_SALARY] フィールドには、各従業員の給与が含まれています。**[ランク基準]** フィールドとして [EMP_SALARY] を選択します。

詳細モードでは、**[ランク基準]** フィールドをバイナリデータ型にすることはできません。

ランク順序

［**ランク順序**］フィールドでランク順序を指定します。［**上**］または［**下**］を選択します。

行数

［**行数**］フィールドで各ランクグループに含める行数を指定します。例えば、各部門で給与支給額に基づいて上位 10 名の従業員をランク付けするには、［**行数**］フィールドで 10 を入力します。

パラメータを使用して行数を指定することができます。パラメータを使用するには、［**行数のパラメータ化**］フィールドで［**パラメータを使用します**］を選択し、［**パラメータ**］フィールドにパラメータを入力します。

ランクグループの定義

ランク付けされた行のグループを定義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 個選択する場合は、製造業者のランクグループを定義します。［**グループ別**］タブでランクグループを定義します。

ランクグループを定義するには、**グループ化フィールド**として 1 つ以上の受信フィールドを選択します。ランクグループ内の一意の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順位）に該当する行のグループを作成します。

注: 大量のデータを処理する詳細モードのマッピングのパフォーマンスを向上させるように、ランクグループを定義します。ランクグループを定義した場合、処理が複数のワーカーノードに分散されます。ランクグループを定義しない場合は、データは 1 つのワーカーノードで処理されます。データのボリュームによっては、パフォーマンスが影響を受け、ワーカーノードに関連付けられている EBS ボリューム上のストレージ領域不足のためにマッピングが失敗する可能性があります。

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するとします。ランクインデックスは、以下のように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	Alexandra B.	10000	1
2	Boris M.	9000	1
3	Chanchal R.	8000	1
4	Dong T.	7000	1
5	Elias M.	6000	1
1	Elias M.	11000	2
2	Boris M.	10000	2
3	Alexandra B.	9050	2
4	Dong T.	7500	2
5	Frances Z.	6900	2

複数のランクグループを定義する場合、ランクトランスフォーメーションは、［**グループ化フィールド**］リストでフィールドが選択されている順序でランク付けされた行をグループ化します。

詳細プロパティ

ランクトランスフォーメーションがデータを処理する方法を定義するには、詳細プロパティを設定します。**[詳細]** タブで詳細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。

プロパティ	説明
キャッシュディレクトリ	データ統合がデータキャッシュおよびインデックスキャッシュファイルを作成するディレクトリ。デフォルトでは、データ統合は、データ統合サーバーで Secure Agent $PMCacheDir プロパティに入力されたディレクトリにキャッシュファイルを格納します。 キャッシュディレクトリを変更する場合は、指定するディレクトリが存在し、キャッシュファイルのための十分なディスクスペースがあることを確認してください。 キャッシュのパーティション化中のパフォーマンスを向上させるには、複数のディレクトリをセミコロンで区切って入力します。キャッシュのパーティション化により、トランスフォーメーションを処理する各パーティションに個別のキャッシュが作成されます。 デフォルトは$PMCacheDir です。
ランクのデータキャッシュサイズ	トランスフォーメーションのデータキャッシュサイズ。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
ランクのインデックスキャッシュサイズ	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。次のいずれかのオプションを選択します。 - 自動。データ統合はキャッシュサイズを自動的に設定します。[自動] を選択した場合は、データ統合の最大メモリ量をキャッシュに割り当てるように設定することもできます。 - 値。キャッシュサイズをバイト単位で入力します。 デフォルトは [Auto] です。
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
トランスフォーメーション範囲	データ統合が受信データに対してトランスフォーメーションロジックを適用する方法。次のいずれかの値を選択します。 - 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。トランスフォーメーション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、[トランザクション] を選択します。 - 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、データ統合はトランザクションの境界を削除します。[すべての入力] は、トランスフォーメーションの結果がソース内のデータのすべての行に依存する場合に選択します。 デフォルトは [すべての入力] です。

プロパティ	説明
大文字小文字を区別した文字列比較	データ統合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。文字列で大文字と小文字を区別しないようにするには、このオプションを無効にします。デフォルトでは有効になっています。 詳細クラスタでは、比較では常に大文字と小文字が区別されます。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。 デフォルトでは有効になっています。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。

階層フィールドをランクトランスフォーメーションに渡す場合は、次のガイドラインを考慮してください。

- 階層フィールドを使用してデータをランク付けすることはできません。
- 階層フィールドを使用してランクグループを定義することはできません。

ランクトランスフォーメーションの例

顧客データをリレーショナルデータベーステーブルに保存します。昇格を各層の上位 3 名の顧客に送信します。ランクトランスフォーメーションを使用して、各層の顧客を注文額順にランク付けします。

ソース、マッピング、ターゲットを以下のように設定します。

ソースデータ

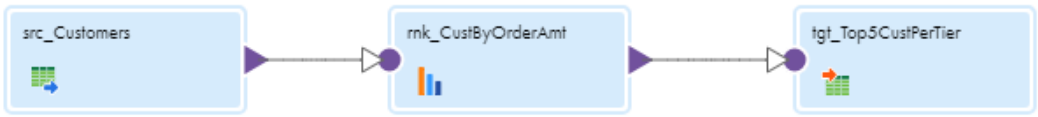
以下の表に、ソースデータを示します。

CUST_ID	CUST_TIER	CUST_NAME	ORDER_AMT
10110102	Gold	Brosseau, Derrick	63508.12
10110109	Platinum	Acheson, Jeff	139824.15
10110143	Silver	Cudell, Bob	49614.00
10110211	Silver	Amico, Paul	47677.30

CUST_ID	CUST_TIER	CUST_NAME	ORDER_AMT
10110215	Platinum	Bergeron, Kim	148871.25
10110224	Silver	Madison, Shelley	40497.10
10110235	Gold	Anderson, Rick	50429.27
10110236	Silver	Tucker, Paul	42585.00
10110237	Silver	Smith, Robert	38563.98
10110393	Gold	Washington, Rochelle	73767.96
10110425	Gold	Nguyen, Trang	65522.25
10110434	Silver	Keane, Thomas	38055.40
10110436	Platinum	Catherwood, Jennifer	117107.44
10110442	Platinum	Charest, Walter	126618.60
10110458	Gold	Coutts, Sylvain	70646.32
10110497	Platinum	Zheng, Wei	191422.00
10110506	Gold	Gonzales, Roberto	79342.90
10110526	Gold	Vanelo, Susan	81978.06
10110528	Platinum	Abedini, John	136506.32
10110530	Silver	Sousa, Maria	10155.42

マッピング設定

次の図に示されるように、マッピングを設定します。



ランクトランスフォーメーションを以下のように設定します。

[ランク] タブ

以下のプロパティを設定します。

フィールド	値
ランク基準	ORDER_AMT
ランク順序	上

フィールド	値
行数のパラメータ化	パラメータを使用しません
行数	3

[グループ化] タブ

［グループ化］フィールドに［CUST_TIER］を選択します。

ターゲットデータ

以下の表に、マッピングの実行時にターゲットに書き込まれるデータを示します。

RANKINDEX	CUST_ID	CUST_TIER	CUST_NAME	ORDER_AMT
1	10110526	Gold	Vanelo, Susan	81978.06
2	10110506	Gold	Gonzales, Roberto	79342.90
3	10110393	Gold	Washington, Rochelle	73767.96
1	10110497	Platinum	Zheng, Wei	191422.00
2	10110215	Platinum	Bergeron, Kim	148871.25
3	10110109	Platinum	Acheson, Jeff	139824.15
1	10110143	Silver	Cudell, Bob	49614.00
2	10110211	Silver	Amico, Paul	47677.30
3	10110236	Silver	Tucker, Paul	42585.00

第 27 章

ルータトランスフォーメーション

ルータトランスフォーメーションは、受信データに条件を適用するために使用できるアクティブなトランスフォーメーションです。

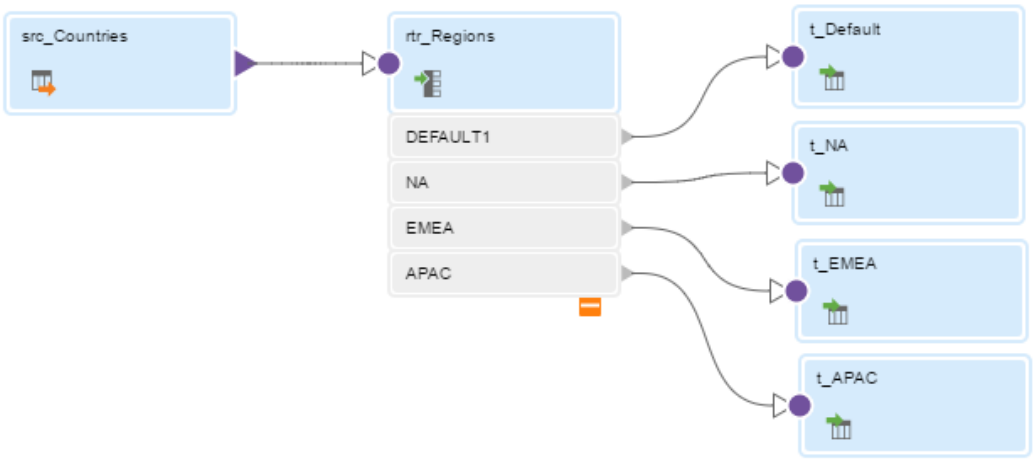
ルータトランスフォーメーションでは、データ統合は、フィルタ条件を使用して受信データの各行を評価します。各ユーザー定義グループの条件をテストしてから、デフォルトグループを処理します。行が複数のグループフィルタ条件を満たす場合、データ統合はこの行を複数回渡します。どの条件も満たさない行は破棄するか、デフォルトの出力グループにルーティングできます。

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、同じタスク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。

次の表は、ルータトランスフォーメーションとフィルタトランスフォーメーションの比較です。

オプション	ルータ	フィルタ
条件	単一のルータトランスフォーメーション内の複数の条件をテストする	フィルタトランスフォーメーションごとに 1 つの条件をテストする
条件を満たさない行の処理	デフォルト出力グループにルーティングするか、破棄する	破棄する
受信データ	1 つのルータトランスフォーメーションで 1 回処理する	各フィルタトランスフォーメーションで処理する

次の図は、地域に基づいてデータをフィルタ処理し、北米、EMEA、またはアジア太平洋地域の異なるターゲットにルーティングするルータトランスフォーメーションを使用したマッピングを示しています。トランスフォーメーションは、他の領域のデータをデフォルトのターゲットにルーティングします。



グループに関する作業

ルータトランスフォーメーションでグループを使用して、入力データをフィルタリングします。

データ統合は、フィルタ条件を使用して、受信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。行が複数のグループフィルタ条件を満たす場合、データ統合はこの行を複数のグループに渡します。

Router 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

データ統合は、入力グループフィールドからプロパティをコピーして、各出力グループのフィールドを作成します。

出力

次の 2 種類の出力グループがあります。

- ユーザー定義グループ。ユーザー定義グループを作成して、受信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。指定する各条件について、それぞれ 1 つのユーザー定義グループを作成します。データ統合は、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ているユーザー定義グループを処理します。
- デフォルトグループ。デフォルトグループは、グループ条件を満たさない行を取得します。デフォルトグループのグループフィルタ条件は、編集、削除、定義できません。すべての条件が FALSE と評価された場合、データ統合は行をデフォルトグループに渡します。どのグループ条件も満たさない行を削除する場合は、デフォルトグループを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でください。

ユーザー定義の出力グループ名を変更できます。行をクリックして **[出力グループの編集]** ダイアログボックスを開きます。

出力グループを接続するためのガイドライン

ルータトランスフォーメーションの出力グループをダウンストリームトランスフォーメーションに接続するときは、次のガイドラインに注意してください。

- 各出力グループは、1 つ以上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。
- 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、複数の出力グループをダウンストリームトランスフォーメーションに接続できます。
- データ統合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でください。

グループフィルタ条件

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。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。詳細モードでは、フィルタ条件が数値結果として評価される必要があります。

グループフィルタ条件は、行が指定された条件を満たすかどうかに基づいて、トランスフォーメーションを通過する行ごとに TRUE または FALSE を返します。ゼロ（0）は FALSE に相当し、ゼロ以外の値はすべて TRUE に相当します。

タスクを実行すると、データ統合は次の方法でデータを処理します。

- TRUE と評価されたデータ行は、各ユーザ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されます。

  ルータトランスフォーメーションでは、複数の出力グループを介してデータを渡すことができます。例えば、データが 3 つの出力グループの条件を満たす場合、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。
- すべての条件が FALSE と評価された行はデフォルトグループに渡されます。

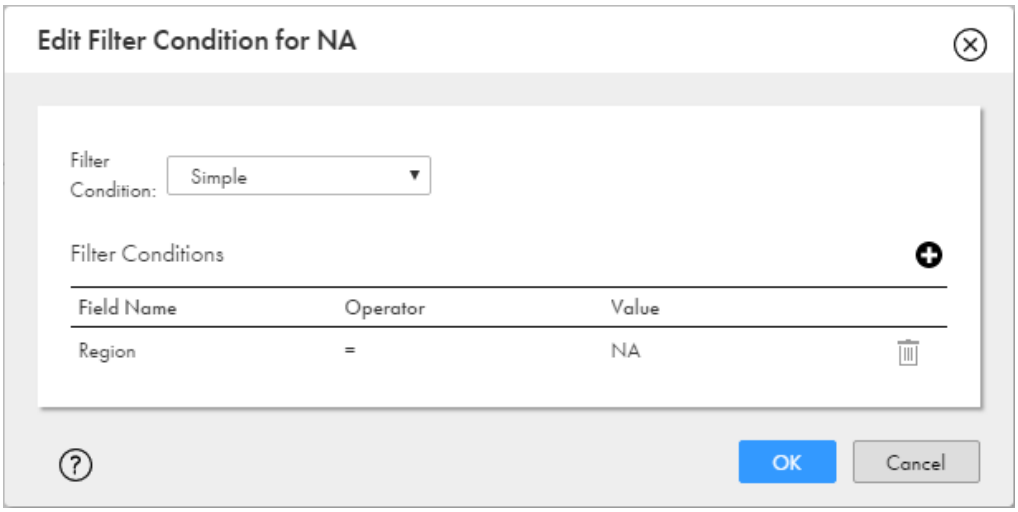
ユーザーはデフォルトグループのグループフィルタ条件を設定できません。ただし、式トランスフォーメーションを追加して計算を実行し、デフォルトグループ内の行を処理することができます。

グループフィルタ条件の設定

ユーザー定義の出力グループごとにグループフィルタ条件を設定します。

1. ソースをルータトランスフォーメーションに接続します。
2. **［出力グループ］**タブをクリックします。
3. プラス（+）記号をクリックし、設定するグループを追加します。
4. **［設定］**をクリックします。
5. **［フィルタ条件の編集］**ダイアログボックスで、次のいずれかのフィルタ条件タイプを選択します。
   - **簡易**: 利用可能なフィールド名と演算子を使用して、単純な条件を指定します。
   - **詳細**: このオプションを選択すると式エディタが開き、複雑な条件を指定してその構文を検証できます。
   - **すべてパラメータを使用します**: 入力パラメータに基づいてフィルタ条件を定義するには、このオプションを選択します。

次の図は、**［フィルタ条件の編集］**ダイアログボックスを示しています。



6. プラス（+）記号をクリックし、このグループに適用する各条件に行を追加します。
7. 各条件について、フィールド名、演算子、値を選択します。
8. **［OK］**をクリックして条件を保存します。

詳細プロパティ

ルータトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルやトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。また、階層フィールドを詳細フィルタ条件で使用したり、階層フィールドを複合演算子とともに使用して、フィルタ条件のプリミティブ子フィールドにアクセスすることもできます。複合演算子の詳細については、*関数リファレンス*の説明を参照してください。

階層フィールドをフィルタ条件で使用する場合は、次のガイドラインを考慮してください。

- 階層フィールドを簡易フィルタ条件で使用することはできません。
- フィルタ条件のパラメータは使用しないでください。

ルータトランスフォーメーションの例

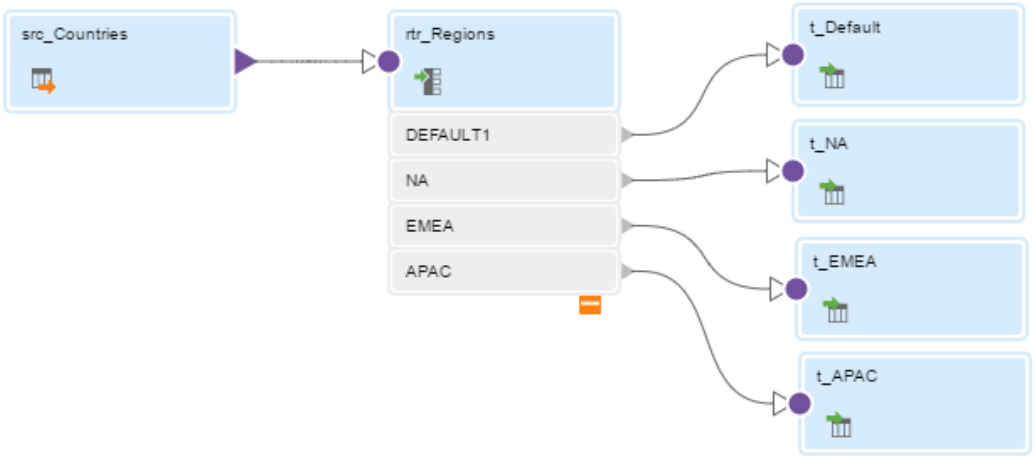
ルータトランスフォーメーションを使用して、次のタスクを完了できます。

- 地域をテストする条件に基づいて、データを国の属性別にグループ化し、各グループをそれぞれのターゲットテーブルにルーティングします。
- 低、中、高の各価格をテストする条件に基づいて、在庫品目を価格カテゴリ別にグループ化し、各グループをそれぞれのターゲットテーブルにルーティングします。

例 1: 地域別の異なるターゲットにデータをルーティングします。

ソースには、さまざまな地域の顧客のデータが含まれます。北米地域、欧州、中東、およびアフリカ地域、およびアジア太平洋地域の顧客のバリアントを使用して、マーケティングキャンペーンを構成する場合。その他の地域の顧客には、標準の広告キャンペーンを表示します。ルータトランスフォーメーションを使用して、4 つの異なるターゲット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。

次の図は、これら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。



次の表に示すように、3 つの出力グループを作成し、**[出力グループ]** タブでグループフィルタ条件を指定します。

グループ名	条件
NA	region = 'NA'
EMEA	region = 'EMEA'
APAC	region = 'APAC'

デフォルトグループには、NA、EMEA、またはアジア太平洋地域にないすべての顧客のデータが含まれます。

例 2: 複数の出力グループに一部の行を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、フィルタ条件を満たすすべての出力グループを通じてデータを渡します。次の例では、条件は価格しきい値をテストしますが、2 つの出力グループのフィルタ条件が重複しています。

グループ名	条件
PriceGroup1	item_price > 100
PriceGroup2	item_price > 500

ルータトランスフォーメーションで item_price=510 の入力行が処理されると、行は両方の出力グループにルーティングされます。

1 つの出力グループを介してデータを渡したい場合は、重複しないようにフィルタ条件を定義します。例えば、PriceGroup1 のフィルタ条件を item_price < = 500 に変更することができます。

第 28 章

ルール仕様トランスフォーメーション

ルール仕様トランスフォーメーションによって、Data Quality で作成したルール仕様アセットがマッピングに追加されます。

ルール仕様は、定義したビジネス条件に従ってデータを分析する 1 つ以上の論理演算のセットです。ルール仕様によって、データがビジネス条件を満たすかどうかを示す出力が生成されます。ルール仕様は、分析するデータを更新することもできます。Data Quality で論理演算を IF/THEN/ELSE 文として定義します。

各ルール仕様トランスフォーメーションには、1 つのルール仕様を含めることができます。複数のルール仕様トランスフォーメーションをマッピングに追加できます。

ルール仕様トランスフォーメーションを使用すると、次のことを実現できます。

- ビジネスデータセットに含まれているデータの種類を定義する。
- ビジネスデータが満たす必要がある条件を定義する。
- データがビジネスルールの条件を満たす場合に実行するアクションを定義する。
- データがビジネスルールの条件を満たさない場合に実行するアクションを定義する。

ルール仕様トランスフォーメーションは、別の所で設計したデータ分析やデータトランスフォーメーションロジックをマッピングに追加できるという点において、マップレットトランスフォーメーションと類似しています。マップレットと同様に、ルール仕様は再利用可能なアセットです。ルール仕様トランスフォーメーションでは、受信フィールドと発信フィールドが示されます。ルール仕様に含まれるロジックは表示されず、ルール仕様を編集することもできません。ルール仕様を編集するには、ルール仕様を Data Quality で開きます。

ルール仕様トランスフォーメーションを使用するには、適切なライセンスが必要です。

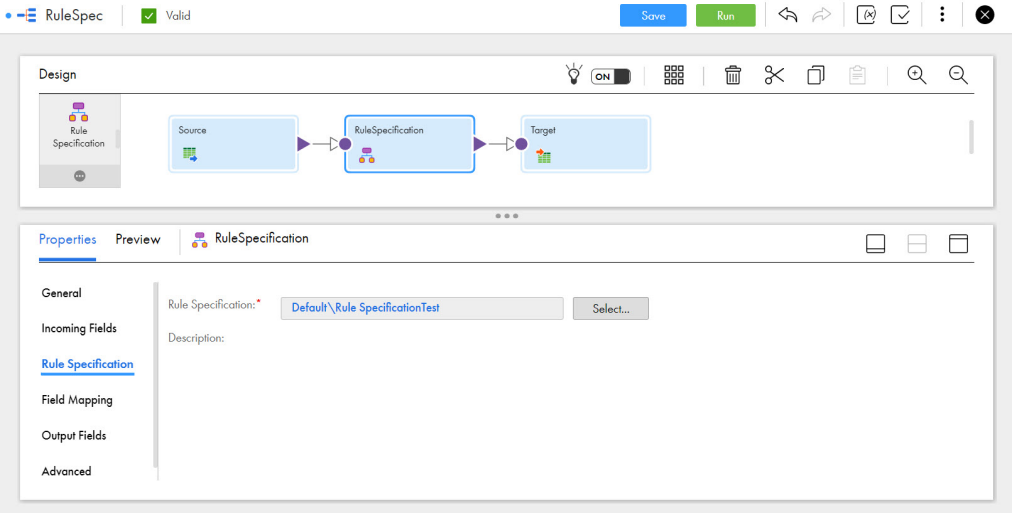
ルール仕様トランスフォーメーションの設定

マッピングのルール仕様トランスフォーメーションを設定する場合は、最初に、マッピングに含めるロジックが含まれているルール仕様アセットを選択します。次に、受信フィールドおよびフィールドマッピングを設定します。その後、出力フィールドを確認します。

トランスフォーメーションを設定するには、以下のタスクを実行します。

1. ルール仕様トランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. **［ルール仕様トランスフォーメーション］**タブで、トランスフォーメーションに含めるルール仕様を選択します。

次の図は、ルール仕様を選択するために使用するオプションを示しています。



3. **[受信フィールド]** タブで、受信フィールドを設定します。

   デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。

4. **[フィールドマッピング]** タブでフィールドマッピングオプションを設定して、受信フィールドからのデータをルール仕様アセットのターゲットフィールドに接続します。

   ルール仕様入力は、データ内の入力フィールドの名前をすでに反映している可能性があります。この場合、**[自動マップ]** オプションを使用してフィールドの接続を行うことができます。アップストリームオブジェクトフィールドに接続する方法については、「ルール仕様トランスフォーメーションのフィールドマッピング」 (ページ 364)を参照してください。

5. **[出力フィールド]** タブのルール仕様出力フィールドを確認します。

6. **[全般]** タブでは、オプションとして、ルール仕様トランスフォーメーションの名前を変更したり、説明を追加したりすることもできます。**[詳細]** タブで、トランスフォーメーションのトレースレベルを更新することもできます。デフォルトのトレースレベルはノーマルです。

注: トランスフォーメーションが使用するルール仕様を識別するためにパラメータを使用する場合は、次のルールとガイドラインを検討してください。

- パラメータを使用してルール仕様を識別する場合、**[フィールドマッピング]** タブで、パラメータを使用してフィールドマッピングを定義する必要があります。パラメータは*文字列*型である必要があります。
- パラメータを使用してルール仕様を識別する場合、ルール仕様トランスフォーメーションに接続するマッピングのダウンストリームオブジェクトはすべてパラメータを使用している必要があります。

注: アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。データ品質アセットの同期の詳細については、「「データ品質アセットの同期」 (ページ 102)」を参照してください。

ルール仕様トランスフォーメーションのフィールドマッピング

データがアップストリームトランスフォーメーションからルール仕様トランスフォーメーションに移行する方法を定義するように、フィールドマッピングを設定します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。タスクで、すべてのフィールドマッピングを設定できます。
  トランスフォーメーションのルール仕様がパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、［すべてパラメータを使用します］オプションを選択します。
- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。［すべてパラメータを使用します］または［部分的にパラメータを使用します］をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは*フィールドマッピング*型である必要があります。

1 つのマッピング内の複数のルール仕様トランスフォーメーションに同じフィールドマッピングパラメータを使用しないでください。

オプション

［受信フィールド］リストと**［ルール仕様の入力フィールド］**リストで、フィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。［手動］または［部分的にパラメータを使用します］フィールドマップオプションを選択すると、［自動マップ］オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。

- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合では Cust_Name フィールドと Customer_Name フィールドが自動的にリンクされます。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似する名前のフィールドをマッピングできます。

［自動マップ］ > **［自動マップを取り消す］**をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで**［アクション］** > **［マップ解除］**をクリックします。選択した 1 つ以上のフィールドをマップ解除するには、［ルール仕様の入力フィールド］コンテキストメニューで**［選択項目をマップ解除］**をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、［ルール仕様の入力フィールド］コンテキストメニューで**［マッピングのクリア］**をクリックします。

ルール仕様トランスフォーメーションの出力フィールド

ルール仕様トランスフォーメーションによって、ルール仕様内のルールセットごとに出力フィールドが表示されます。出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。トランスフォーメーション出力には、ルール仕様内の各ルールセットからの出力が含まれます。

タブには、各出力フィールドの名前、タイプ、精度、およびスケールが表示されます。出力フィールド名は、ルール仕様内のルールセットの名前です。

ルール仕様トランスフォーメーションの出力フィールドプロパティを編集することはできません。プロパティを編集するには、ルール仕様を Data Quality で開きます。

詳細プロパティ

ルール仕様トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。

第 29 章

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、数値を生成する、接続されたパッシブトランスフォーメーションです。一意なプライマリキー値の作成、欠落しているプライマリキーの置き換え、連続した数値のでサイクル動作を実行する場合、Sequence Generator 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にはパススルーフィールドと 2 つの出力フィールド（NEXTVAL および CURRVAL）があります。詳細モードでは、トランスフォーメーションには NEXTVAL という 1 つの出力フィールドがあります。出力フィールドは 1 つ以上のダウンストリームトランスフォーメーションに接続できます。

マッピングタスクにより、マッピングされたフィールドが、接続されたトランスフォーメーションに入力されるたびに値の数値シーケンスが生成されます。Mapping Designer で数値の範囲を設定します。タスクの実行時にシーケンスの最初の数値を変更できます。

タスクの完了後に、マッピングタスクの詳細に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と初期値が表示されます。

注: AWS 環境で実行される詳細モードの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場合は、Spark ドライバが Secure Agent と通信できることを確認してください。詳細については、Administrator のヘルプを参照してください。

シーケンスジェネレータトランスフォーメーションの用途

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、次のように数値のシーケンスを生成します。

一意のシーケンス番号を生成する。

シーケンスは、指定した［初期値］で開始されます。

循環シーケンス番号を生成する。

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 の値を生成します。シーケンスは、10 から再び開始されます。

詳細モードでは適用されません。

既存のシーケンス番号を継続する。

マッピングタスクを実行するたびに、タスクによって値が更新され、最後に生成された値に［増分］の値が追加されます。タスクを実行するたびに付番を最初からやり直すには、［リセット］設定プロパティを有効にします。

同じシーケンスジェネレータを使用して、複数のパーティションに対して連続した一意の番号を生成する。

マッピングタスクで、各パーティションに対して同じ番号が生成されないようにするために、一度にキャッシュされるシーケンス値の数を指定できます。複数のパーティションに同じシーケンスジェネレータを使用する場合は、循環するシーケンス番号を生成できません。

詳細モードでは適用されません。

シーケンスジェネレータ出力フィールド

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、NEXTVAL と CURRVAL という 2 つの出力フィールドがあります。詳細モードでは、このトランスフォーメーションには NEXTVAL という 1 つの出力フィールドがあります。これらのフィールドを編集したり削除したりすることはできません。

シーケンスジェネレータトランスフォーメーションは任意のトランスフォーメーションに接続できます。マッピングに両方のフィールドが含まれている場合、両方の出力フィールドをマッピングする必要はありません。いずれかの出力フィールドをマッピングしない場合、マッピングタスクはマッピングされていないフィールドを無視します。

NEXTVAL フィールド

NEXTVAL フィールドは、［初期値］および［増分］プロパティに基づいてシーケンス番号を生成する場合に使用します。

NEXTVAL フィールドをターゲットトランスフォーメーションまたは他のダウンストリームトランスフォーメーションの入力フィールドにマッピングし、シーケンス番号を生成します。シーケンスでサイクル動作を実行するようにシーケンスジェネレータを設定しない場合、NEXTVAL フィールドは設定された終了値までシーケンス番号を生成します。

NEXTVAL フィールドを複数のトランスフォーメーションにマッピングする場合、マッピングタスクは、マッピングタイプと受信フィールドが無効かどうかに基づいて、各ダウンストリームトランスフォーメーションに対して同じシーケンス番号または一意のシーケンス番号を生成します。

次の表に、シーケンスジェネレータトランスフォーメーションが同じシーケンス番号を生成する状況と一意のシーケンス番号を生成する状況を示します。

マッピングタイプ	受信フィールド	同じシーケンスか一意のシーケンスか
マッピング	無効でない	同じシーケンス
マッピング	無効	一意のシーケンス*
マッピング(詳細モード)	-	同じシーケンス

** 受信フィールドが無効になっている場合に同じシーケンス番号を生成するには、シーケンスジェネレータとトランスフォーメーションの間に式トランスフォーメーションを配置して、シーケンス番号をステージングします。*

CURRVAL フィールド

CURRVAL フィールド値は、NEXTVAL 値に増分値を追加した値です。例えば、［初期値］が 1、［増分］が 1 の場合、マッピングタスクは NEXTVAL と CURRVAL に対して次の値を生成します。

NEXTVAL	CURRVAL
1	2
2	3
3	4
4	5
5	6

通常、マップで NEXTVAL フィールドがすでにダウンストリームトランスフォーメーションにマッピングされている場合、CURRVAL フィールドをマッピングします。NEXTVAL フィールドをマッピングせずに CURRVAL フィールドをマッピングすると、マッピングタスクは各行に同じ数値を生成します。

シーケンスジェネレータのプロパティ

シーケンスジェネレータのプロパティを設定し、シーケンスジェネレータが数値を生成する方法を定義します。

［シーケンス］タブで次のシーケンスジェネレータのプロパティを設定します。

プロパティ	説明
共有シーケンスの使用	共有シーケンスを使用してシーケンス値を生成する場合に有効にします。有効にした場合、シーケンスは共有シーケンスの［現在の値］で開始されます。 共有シーケンスの詳細については、「*コンポーネント*」を参照してください。 デフォルトでは無効になっています。
増分	生成されるシーケンスでの 2 つの連続する値の差。例えば、［増分］が 2 で既存の値が 4 の場合、シーケンスで生成される次の値は 6 になります。 デフォルトは 1 です。 最大値は 2,147,483,647 です。
終了値	マッピングタスクが生成する最大値。タスク実行中にシーケンスがこの値に到達し、シーケンスのサイクルが設定されていない場合、その実行は失敗します。 最大値は 9,223,372,036,854,775,807 です。 NEXTVAL フィールドをダウンストリーム整数フィールドに接続する場合、終了値は、その最大整数値より大きな値には設定しないでください。NEXTVAL がダウンストリームフィールドのデータ型の最大値を超えると、そのマッピングの実行は失敗します。 詳細モードでは、終了値は処理する最大行数以上に設定します。

プロパティ	説明
初期値	マッピングタスクがシーケンスの最初の値として使用する値。一連の値間でサイクルする場合、この値は、［開始値］以上、［終了値］未満である必要があります。 デフォルトは 1 です。
サイクル	有効にすると、マッピングタスクはシーケンスの範囲でサイクル動作を実行します。無効にすると、タスクは設定されている終了値でシーケンスを停止します。タスク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、セッションが失敗します。 デフォルトでは無効になっています。
サイクル開始値	［サイクル］オプションを使用する場合に、マッピングタスクが使用する生成シーケンスの開始値。シーケンス値が終了値に達するとこの値に戻ります。 デフォルトは 0 です。 最大値は 9,223,372,036,854,775,806 です。
キャッシュされる値の数	マッピングタスクによって各実行でキャッシュされるシーケンス値の数。実行ごとに新しい値のバッチが使用されます。タスクはバッチの未使用のシーケンスを破棄します。マッピングタスクは、それぞれの値をキャッシュするたびにリポジトリを更新します。0 に設定した場合、タスクは値をキャッシュに格納しません。 複数のパーティションが同時に同じシーケンスジェネレータを使用する場合に、このオプションを使用して、各パーティションが一意の値を受信するようにします。 デフォルトは 0 です。 このオプションは、［サイクル］プロパティが有効な場合は使用できません。 詳細モードでは、キャッシュされる値の数を設定できません。ただし、組織の管理者は値のキャッシュ方法を最適化できます。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。(Ref 619019)
リセット	有効にすると、マッピングタスクは各実行で元の初期値に基づいて値を生成します。 デフォルトでは無効になっています。

[詳細] タブで次のシーケンスジェネレータのプロパティを設定します。

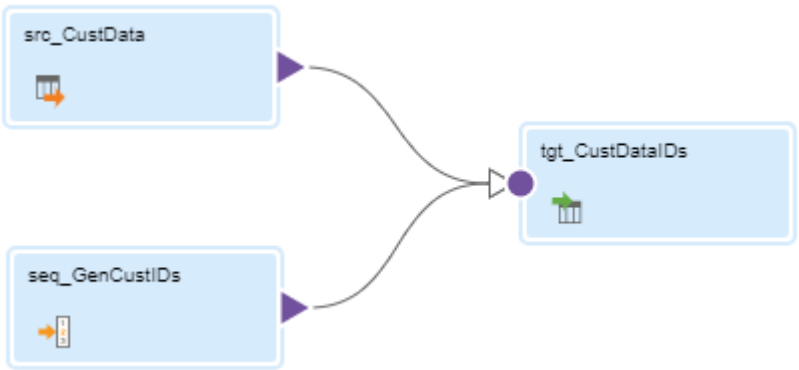
プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[簡易]、[通常]、[詳細 - 初期化]、[詳細 - データ] から選択できます。デフォルトは [通常] です。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。
受信フィールドの無効化	生成されるシーケンスのみをダウンストリームトランスフォーメーションに接続するには、受信フィールドを無効にします。受信フィールドを無効にする場合は、少なくとも 1 つのフィールドを別のトランスフォーメーションからダウンストリームトランスフォーメーションに接続する必要があります。

受信フィールドの無効化

受信フィールドを無効にすると、出力フィールド NEXTVAL および CURRVAL の生成シーケンスのみを 1 つ以上のダウンストリームトランスフォーメーションに接続できます。受信フィールドを無効にした場合、シーケンスジェネレータトランスフォーメーションをアップストリーム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。

受信フィールドを無効にする場合は、別のトランスフォーメーションの少なくとも 1 つのフィールドを、シーケンスジェネレータフィールドとともにダウンストリームトランスフォーメーションに接続する必要があります。例えば、マッピングにシーケンスジェネレータトランスフォーメーションとソーストランスフォーメーションが含まれており、シーケンスジェネレータトランスフォーメーションがターゲットトランスフォーメーションに接続されている場合は、ソーストランスフォーメーションの少なくとも 1 つのフィールドをターゲットトランスフォーメーションに接続する必要があります。

次の図は、シーケンスジェネレータトランスフォーメーションで受信フィールドが無効になっているマッピングを示しています。



注: 詳細モードでは、受信フィールドを無効にすることはできません。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。

シーケンスジェネレータトランスフォーメーションのルールおよびガイドライン

シーケンスジェネレータトランスフォーメーションを作成する場合、次のガイドラインを考慮してください。

- NEXTVAL または CURRVAL 出力フィールドをマッピングする場合、マッピングされるフィールドのデータ型が適切であることを確認します。
- Mapping Designer でマッピングを実行すると、現在の値が保存されないため、マッピングを実行するたびに初期値で開始されます。
- マッピングタスクウィザードでタスクを実行すると、指定した値でシーケンスが開始されるように現在の値を編集できます。
- シーケンスジェネレータトランスフォーメーションをマップレットで使用することはできません。
- 詳細モード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には、Secure Agent を仮想マシンで実行する必要があります。
- 詳細モードでは、生成された値が単調に増加しない場合があります。値は、Spark エンジンがデータを処理する順序に基づいて生成されます。
- セルフサービスクラスタが断続的にエージェントへの接続に失敗する場合があり、これによりマッピングが失敗する可能性があります。これに関連するエラーメッセージがセッションログに表示されます。

シーケンスジェネレータトランスフォーメーションの例

次の例は、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て、プライマリキーを生成する方法を示しています。

顧客データを収集し、顧客 ID を各顧客に割り当てる必要があります。CustomerData.csv フラットファイルには、ソース顧客データが格納されています。次のプロセスを使用して、シーケンスジェネレータトランスフォーメーションが含まれるマッピングを作成し、顧客 ID を作成します。

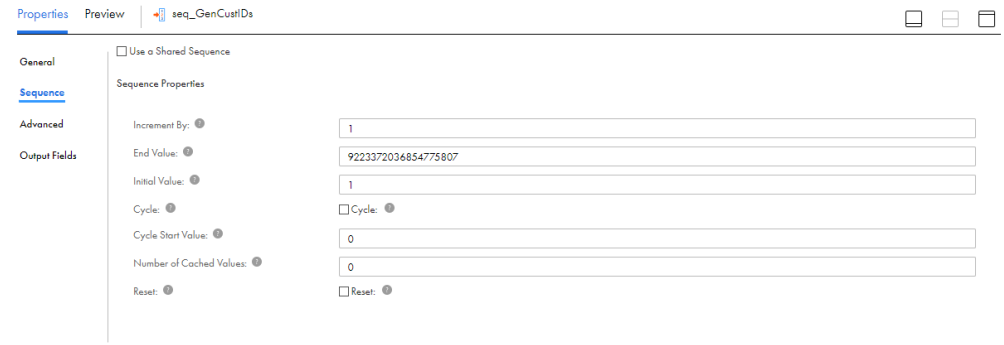
1. ターゲットとして使用する CustomerData.csv ファイルのコピーを作成し、生成された顧客 ID 値を格納するファイルに cust_id フィールドを追加します。CustomerData_IDs.csv ファイルの名前を付けます。

last_name	city	state	country_c	cust_id
Smith	Los Angel	California	USA	
Jones	Chicago	Illinois	USA	
Winston	Houston	Texas	USA	
Leesom	Philadelp	Pennsylva	USA	
Marks	Phoenix	Arizona	USA	
Weston	San Diego	California	USA	
West	Dallas	Texas	USA	
Arlington	San Anton	Texas	USA	
Book	Detroit	Michigan	USA	
Ferris	San Jose	California	USA	
Richards	Indianapo	Indiana	USA	
Benton	San Franci	California	USA	
Martinez	Jacksonvil	Florida	USA	
Alton	Columbus	Ohio	USA	

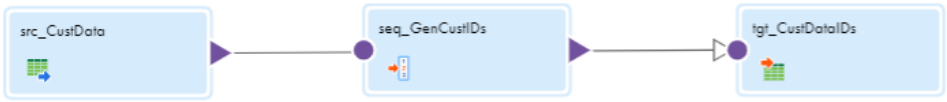
2. CustomerData.csv ファイルと CustomerData_IDs.csv ファイルにアクセスできる接続を作成します。
3. Mapping Designer でマッピングを作成し、ソーストランスフォーメーションをマッピングに追加します。CustomerData.csv ファイルを使用するようにトランスフォーメーションを設定します。
4. シーケンスジェネレータトランスフォーメーションをマッピングに追加します。

   トランスフォーメーションパレットでは、シーケンスジェネレータトランスフォーメーションに「Sequence」というラベルが付けられます。
5. 1 から始まる単純なシーケンスにするには、［シーケンス］タブで **［初期値］** を 1 に設定し、**［増分］** の値を 1 に設定します。この設定では、シーケンスが 1 から始まり、1、2、3 のように値が 1 つずつ増加します。

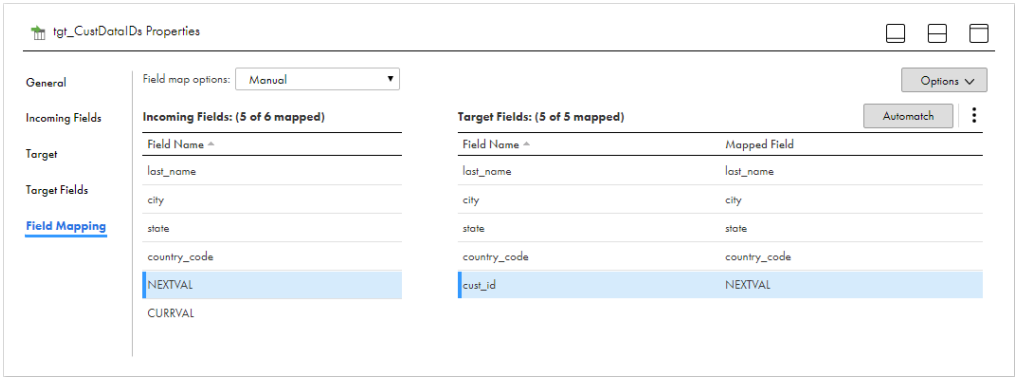
他のプロパティはデフォルト値のままにします。



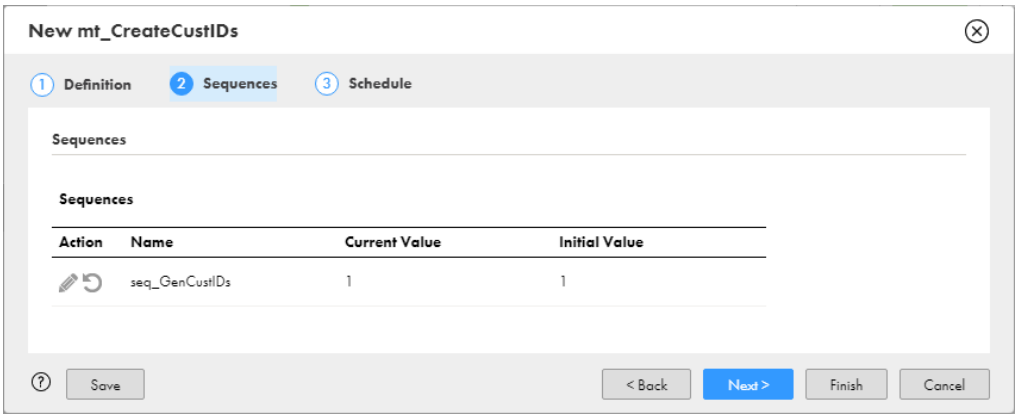
6. ターゲットトランスフォーメーションをマッピングに追加し、作成した CustomerData_IDs.csv ファイルを使用するようにトランスフォーメーションを設定します。
7. ソーストランスフォーメーションをシーケンスジェネレータトランスフォーメーションに、シーケンスジェネレータトランスフォーメーションをターゲットトランスフォーメーションに接続します。



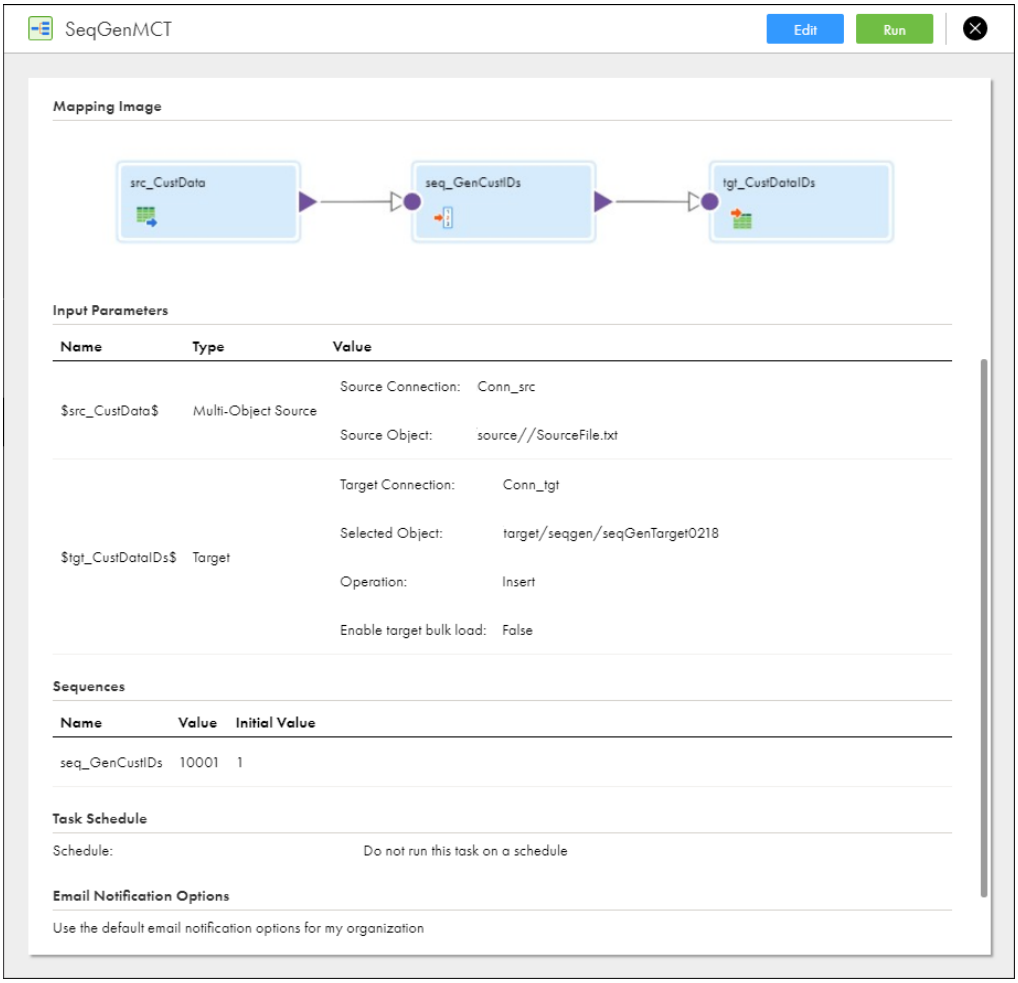
8. ターゲットトランスフォーメーションで、NEXTVAL 出力フィールドを cust_id フィールドにマッピングします。



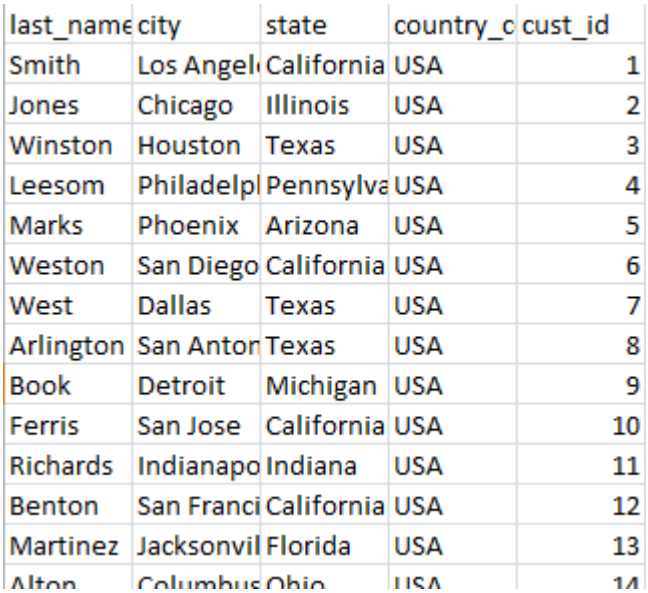
9. マッピングを保存し、マッピングタスクウィザードでマッピングタスクを作成します。マッピングをまだ実行しておらず、**[初期値]** が 1 であるため、**[現在の値]** は 1 です。



10. マッピングタスクを実行すると、マッピングタスクの詳細が表示されます。タスクの詳細には、シーケンスの現在の値と初期値が表示されます。



11. CustomerData_IDs.csv ファイルを開くと、cust_id フィールドに数値シーケンスが取り込まれていることがわかります。



last_name	city	state	country_c	cust_id
Smith	Los Angel	California	USA	1
Jones	Chicago	Illinois	USA	2
Winston	Houston	Texas	USA	3
Leesom	Philadelpl	Pennsylva	USA	4
Marks	Phoenix	Arizona	USA	5
Weston	San Diego	California	USA	6
West	Dallas	Texas	USA	7
Arlington	San Anton	Texas	USA	8
Book	Detroit	Michigan	USA	9
Ferris	San Jose	California	USA	10
Richards	Indianapo	Indiana	USA	11
Benton	San Franci	California	USA	12
Martinez	Jacksonvil	Florida	USA	13
Alton	Columbus	Ohio	USA	14

第 30 章

ソータートランスフォーメーション

指定されたソート条件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、パッシブ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使用して、他の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。例えば、ソート済み受信フィールドを使用するように設定されたルックアップトランスフォーメーションまたはアグリゲータトランスフォーメーションを経由するデータをソートできます。

ソータートランスフォーメーションを作成する場合、フィールドをソート条件として指定し、昇順または降順でソートを行うように各ソートフィールドを設定します。マッピングタスクを設定する場合、ソート条件のパラメータを使用して、パラメータの値を定義できます。

注: 詳細モードでソータートランスフォーメーションを有効にするには、次の条件が満たされていることを確認します。

- Sorter トランスフォーメーションが、ターゲットトランスフォーメーションに直接接続されている。
- ソート条件で使用されるすべてのソートフィールドが、ターゲットトランスフォーメーションに接続されている。
- 接続されたソートフィールドのデータ型が、ターゲットフィールドのデータ型と一致している。

ソート条件

ソート条件を設定して、ソートフィールドおよびソート順を指定します。マッピングタスクは、ソート条件を使用してデータをソートします。

ソートフィールドとは、ソート基準として使用する 1 つまたは複数のフィールドのことです。ソート順を設定して、データを昇順または降順にソートします。詳細モードでは、マッピングタスクをスケジュールするときに、詳細セッションプロパティを使用してソート順をオーバーライドできます。

複数のソート条件を指定する場合、マッピングタスクは各条件をシーケンシャルにソートします。マッピングタスクは、連続した各ソート条件を前のソート条件の 2 番目のソートとして扱います。ソート条件の順序は設定が可能です。

ソート条件のパラメータを使用する場合、マッピングの実行時またはマッピングタスクの設定時にソートフィールドとソート順を定義します。

ソーターキャッシュ

マッピングタスクでは、ソート操作の実行前に、すべての受信データがソータートランスフォーメーションに渡されます。マッピングタスクではキャッシュメモリを使用して、ソータートランスフォーメーションが処理されます。マッピングタスクが十分なメモリを割り当てることができない場合は、マッピングが失敗します。

デフォルトで、マッピングタスクは実行時にキャッシュサイズを決定します。マッピングタスクは、ソート操作を開始する前に、ソーターキャッシュサイズで設定されたメモリ量を割り当てます。

Secure Agent をホストするマシンの利用可能な物理 RAM 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 16 MB（16,777,216 バイト）の物理メモリを割り当てます。メモリをソーターキャッシュに割り当てる場合、マッピングの他のトランスフォーメーションやマッピングタスクのデータ量を考慮します。

受信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、マッピングタスクはデータを一時的にワークディレクトリに保存します。ソータートランスフォーメーションのワークディレクトリにデータを保存する場合、マッピングタスクには最低でも受信データ量の 2 倍のディスク領域が必要です。

トレースレベルを［ノーマル］に設定すると、マッピングタスクは、ソータートランスフォーメーションが使用するメモリ量をセッションログに書き込みます。

詳細プロパティ

ソーター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。マッピングタスクは、プロパティをすべてのソートフィールドに適用します。また、ソータートランスフォーメーションのプロパティは、マッピングタスク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。

ソーター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できます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。
ソーターキャッシュサイズ	ソート操作の実行に必要なメモリの最大量。マッピングタスクでは、ソート操作の実行前に、すべての入力データがソータートランスフォーメーションに渡されます。マッピングタスクが十分なメモリを割り当てることができない場合は、マッピングが失敗します。 ソーターキャッシュには、数値を設定できます。少なくとも 16 MB の物理メモリを割り当てます。デフォルトは［Auto］です。
大文字小文字の区別	マッピングタスク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。大文字と小文字を区別するソートを有効にすると、マッピングタスクは大文字を小文字よりも上位にソートします。デフォルトは［大文字小文字の区別］です。 詳細クラスタでは、ソートでは常に大文字と小文字が区別されます。
作業ディレクトリ	マッピングタスクは、作業ディレクトリを使用してデータのソート中に一時ファイルを作成します。マッピングタスクはデータをソートした後で、一時ファイルを削除します。 Secure Agent マシンの任意のディレクトリを作業ディレクトリとして指定できます。少なくとも 16 MB（16,777,216 バイト）の物理メモリを作業ディレクトリ用に割り当てます。このフィールドには、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設定できます。デフォルトは TempDir システムパラメータです。

プロパティ	説明
個別	出力行を重複しないものとして扱います。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、マッピングタスクはすべてのフィールドをソート条件の一部として設定します。マッピングタスクは、ソート操作時に比較した重複行を破棄します。
NULL を下として扱う	NULL 値を最下位にソートします。例えば、降順のソート条件を設定した場合、ソートフィールドが NULL 値の行は他のどの行よりも後に表示されます。 詳細クラスタでは、NULL 値は上位として扱われます。
トランスフォーメーション範囲	トランザクションは、コミットポイントまたはロールバックポイントによって決まります。トランスフォーメーション範囲は、マッピングタスクで入力データにトランスフォーメーションロジックをどう適用するかを指定します。 - 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。トランスフォーメーション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。 - 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、マッピングタスクは入力トランザクションの境界を削除します。[すべての入力] は、トランスフォーメーションの結果がソース内のデータのすべての行に依存する場合に選択します。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

詳細モードの階層データ

詳細モードでは、配列、マップ、または構造体を表す階層フィールドを使用できます。

階層フィールドはパススルーフィールドとして使用できます。

注: 階層フィールドをソート条件で使用することはできません。

ソータートランスフォーメーションの例

顧客売上の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。顧客売上オブジェクトのソータートランスフォーメーションを使用し、注文番号に従ってデータを昇順にソートします。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。[ソート済み受信

フィールド］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。

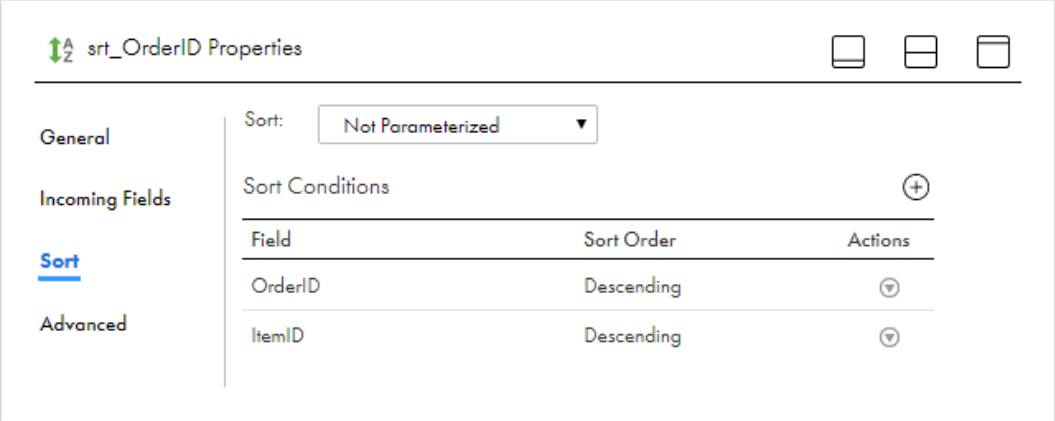
Product_Orders テーブルには、顧客が行ったすべての注文に関する情報が含まれています。

OrderID	ItemID	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
43	NULL	ItemE	1	0.75
41	123456	ItemA	4	3.04
45	123456	ItemA	5	3.04
45	456789	ItemB	3	12.02

Mapping Designer で、Product_Orders をソースオブジェクトとして追加します。

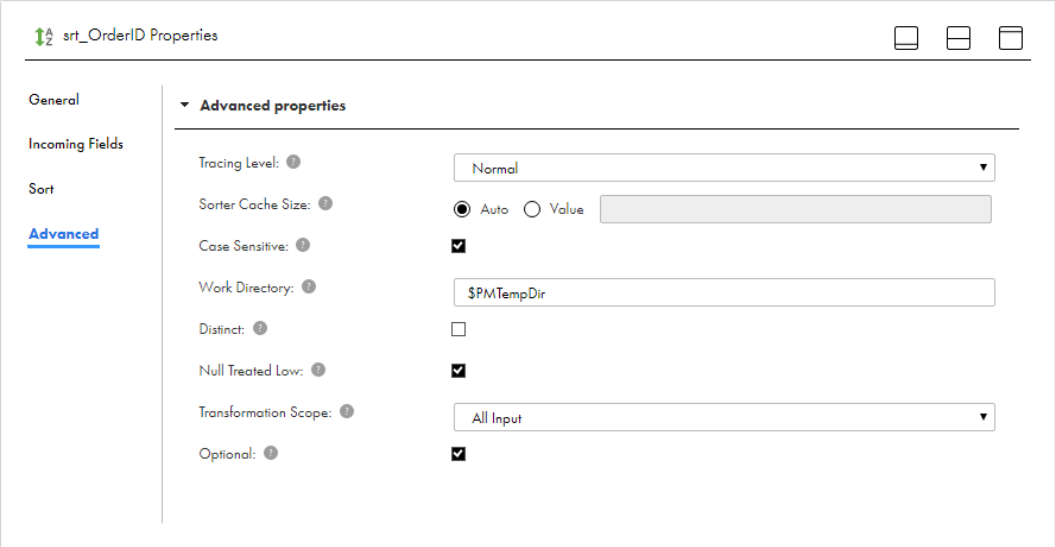
ソータートランスフォーメーションをマッピングキャンバスに追加し、それをデータフローに接続します。オーダー ID と項目 ID で製品の注文をソートします。

次の図に、オーダー ID フィールドと項目 ID フィールドが降順にソートされるよう設定されているソート条件を示します。



マッピングタスクで NULL 値が他の値よりも低い値として扱われるように、NULL を下として扱うソートを有効にします。

次の図に、［NULL を下として扱う］オプションが選択されているソーター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。



マッピングタスクは、データをソートした後で次の行をソータートランスフォーメーションから渡します。

OrderID	ItemID	Item	Quantity	Price
45	456789	ItemB	3	12.02
45	123456	ItemA	5	3.04
45	000468	ItemD	5	0.56
43	123456	ItemA	3	3.04
43	000246	ItemC	6	34.55
43	NULL	ItemE	1	0.75
41	456789	ItemB	2	12.02
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し、パフォーマンスを向上させることができます。アグリゲータトランスフォーメーションをマッピングに追加し、アグリゲータトランスフォーメーションをデータフローに接続します。アグリゲータトランスフォーメーションのフィールドをオーダー ID でグループ化し、注文金額を合計する式を追加します。

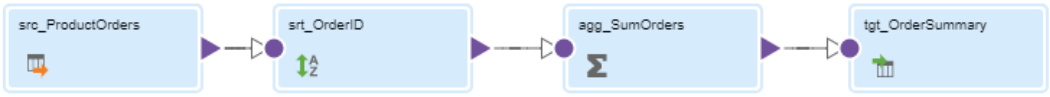
ソータートランスフォーメーションからデータを渡すと、アグリゲータトランスフォーメーションはオーダー ID をグループ化して、注文ごとの合計金額を計算します。

OrderID	Sum
45	54.06
43	217.17
41	36.2

すべての注文の合計を格納するデータウェアハウスのオブジェクトに注文の概要を書き込みます。

マッピングタスクは、ソースから注文データを読み取り、注文をソートして金額を合計し、データをターゲットに書き込みます。

次の図に、データフローのマッピングを示します。



第 31 章

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を使用して、リレーショナルデータベース内のストアドプロシージャまたはストアド関数を呼び出すか、パイプライン内の SQL クエリミッドストリームを処理します。トランスフォーメーションにより、ストアドプロシージャまたはストアド関数の呼び出し、保存済みクエリの処理、または SQL トランスフォーメーションエディタで作成したクエリの処理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は、次のタイプの SQL 文を処理できます。

ストアドプロシージャまたはストアド関数

ストアドプロシージャは、データベースプロシージャ文と、オプションのフロー制御文（実行可能スクリプトに類似）の、コンパイル済みのコレクションです。ストアドプロシージャはデータベース内にあり、データベース内で実行されます。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、ストアドプロシージャまたはストアド関数を処理する場合、入力パラメータをストアドプロシージャまたはストアド関数に渡します。ストアドプロシージャまたはストアド関数は、戻り値をトランスフォーメーションの出力フィールドに渡します。

保存済みクエリまたはユーザー入力クエリ

データ統合で作成した保存済みクエリを処理するように SQL トランスフォーメーションを設定するか、SQL エディタにクエリを入力することができます。SQL トランスフォーメーションでクエリが処理され、行およびデータベースエラーが返されます。

文字列またはパラメータをクエリに渡し、動的クエリを定義するか、または選択パラメータを変更できます。クエリに SELECT 文がある場合、複数の行を出力できます。

ストアドプロシージャまたはストアド関数の処理

Microsoft SQL Server、MySQL、ODBC、または Oracle のデータベース内のストアドプロシージャまたはストアド関数を呼び出すには、SQL トランスフォーメーションを使用します。ストアドプロシージャまたはストアド関数は、SQL トランスフォーメーションを作成する前にデータベースに存在している必要があります。

次のタイプの SQL トランスフォーメーションを使用して、ストアドプロシージャまたはストアド関数を呼び出すことができます。

接続された SQL **トランスフォーメーション**

トランスフォーメーションがマッピングパイプラインに接続されています。ストアドプロシージャまたはストアド関数は行ごとに実行され、単一の出力パラメータまたは複数の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の受信フィールドをストアドプロシージャの入力フィールドにマッピングできます。SQL トランスフォーメーションの出力フィールドは、ストアドプロシージャの出力パラメータまたは戻り値で構成されます。

戻り値は、ストアドプロシージャに定義したコードまたはテキスト文字列です。例えば、ストアドプロシージャは実行された日付を示す値を返すことができます。ストア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フィールドがあります。

未接続の 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がマッピングパイプラインに接続されていません。式トランスフォーメーションによりストアドプロシージャの式を使用して SQL トランスフォーメーションが呼び出されるか、ストアドプロシージャがマッピングの前または後に実行されます。

ストアドプロシージャの出力を式の出力フィールドと変数に返すように式を設定できます。複数の式からストアドプロシージャを呼び出し、ストアドプロシージャをネストすることができます。

未接続の SQL トランスフォーメーションを使用してストアド関数を処理することはできません。

ストアドプロシージャは、次のタスクを実行する場合などに使用します。

- 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ステータスをチェックする。
- 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。
- 一時テーブルを削除する。
- テーブルがデータベースに存在することを確認する。

ストアドプロシージャを使用して、マッピングの一部で行うような計算を実行できます。例えば、消費税を計算するストアドプロシージャがある場合、式トランスフォーメーションで計算を再作成せずに、SQL トランスフォーメーションで計算を実行できます。

マッピングを実行すると、SQL トランスフォーメーションは入力パラメータをストアドプロシージャに渡します。ストアドプロシージャは戻り値をトランスフォーメーションの出力フィールドに渡します。

接続された SQL トランスフォーメーションの例

マッピングには、データフローのユーザー ID が含まれます。ユーザー ID に加えてユーザー名を含める必要がある場合があります。

ユーザー ID をデータベースのユーザー名とマッチングするストアドプロシージャがあります。マッピングに SQL トランスフォーメーションを追加し、そのストアドプロシージャを選択して、ユーザー ID 受信フィールドをストアドプロシージャのユーザー ID 入力フィールドとマッピングします。**[出力フィールド]** タブに SQL トランスフォーメーションがあるかチェックし、SQL トランスフォーメーションにユーザー名フィールドが含まれていることを確認します。マッピングを実行すると、ユーザー名の値がユーザー ID とともに返されます。

接続されていない SQL トランスフォーメーションの例

マッピングに従業員の給与データが含まれており、各従業員の給与を昇給で更新する必要があります。

従業員の昇給を計算するストアドプロシージャがあります。このストアドプロシージャは、新しい給与と増加率を返します。接続されていない SQL トランスフォーメーションを追加し、ストアドプロシージャを選択します。

次に、式トランスフォーメーションをマッピングパイプラインに追加します。式トランスフォーメーションで、新しい給与を取得するための変数フィールドを追加します。出力フィールドを追加し、ストアドプロシージャの関数を使用して式を設定します。出力フィールドが増加率を返すように引数を設定し、新しい給与を返す 2 番目の出力フィールドを作成します。その後、新しい出力フィールドをダウンストリームトランスフォーメーションにマッピングします。

ストアドプロシージャの処理のための接続済または未接続 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でストアドプロシージャを呼び出すときは、接続済または未接続の SQL トランスフォーメーションを使用できます。

入力フィールドのデータを入力パラメータとしてストアドプロシージャに渡す必要がある場合、またはストアドプロシージャの結果を別のトランスフォーメーションに出力パラメータとして渡す必要がある場合は、接続済の SQL トランスフォーメーションを使用してストアドプロシージャを処理します。

マッピングの前または後にストアドプロシージャを実行する必要がある場合、ネストしているストアドプロシージャを実行する場合、またはストアドプロシージャを複数回呼び出す場合は、未接続の SQL トランスフォーメーションを使用してストアドプロシージャを処理します。

以下の表に、接続済または未接続の SQL トランスフォーメーションを使用してストアドプロシージャを処理するタイミングを示します。

シナリオ	SQL **トランスフォーメーションのタイプ**
ストアドプロシージャをマッピングの前または後で実行する	未接続
ストアドプロシージャをマッピング時に 1 回実行する	未接続
行が SQL トランスフォーメーションを通過するごとにストアドプロシージャを実行する	接続済または未接続
マッピングを通過するデータに基づいてストアドプロシージャを実行する（例えば特定のフィールドが NULL 値を含んでいないとき）	コネクトされていないモード
パラメータをストアドプロシージャに渡し、1 つの出力パラメータを受け取る	接続済または未接続
パラメータをストアドプロシージャに渡し、複数の出力パラメータを受け取る **注:** 接続されていない SQL トランスフォーメーションから複数の出力パラメータを取得するには、出力パラメータそれぞれに対して変数を作成する必要があります。	接続済または未接続
ネストしているストアドプロシージャを実行する	コネクトされていないモード
マッピング内でストアドプロシージャを複数回呼び出す	未接続

接続されていない SQL トランスフォーメーション

接続されていない SQL トランスフォーメーションとは、マッピングパイプラインに接続されていない SQL トランスフォーメーションです。接続されていない SQL トランスフォーメーションは、ストアドプロシージャを呼び出すために使用します。

式トランスフォーメーションを使用して、:SP 式で接続されていない SQL トランスフォーメーションを呼び出します。または、マッピングの実行の前または後にストアドプロシージャを呼び出すように SQL トランスフォーメーションを設定します。例えば、接続されていない SQL トランスフォーメーションを使用して、マッピングがソースからデータを受信した後に一時的なソーステーブルを削除できます。

また、マッピングでストアドプロシージャを複数回呼び出す場合も、接続されていない SQL トランスフォーメーションを使用できます。

式からの接続されていない SQL トランスフォーメーションの呼び出し

:SP 式を使用した式トランスフォーメーションから、接続されていない SQL トランスフォーメーションを呼び出します。

式からストアドプロシージャを呼び出すときは、ストアドプロシージャの出力値を式のフィールドに返すように式を設定します。次のいずれかの方法を使用して、出力値を返します。

- 出力値をローカル変数フィールドに割り当てる。
- 出力値をシステム変数 PROC_RESULT に割り当てる。

PROC_RESULT 変数を使用する場合、データ統合によってリターンパラメータの値が出力フィールドに直接割り当てられるため、その値をターゲットに書き込むことができます。ある出力パラメータを PROC_RESULT に、別のパラメータを別の変数に割り当てることもできます。

式の変数を使用して、ストアドプロシージャの OUT または INOUT パラメータにアクセスできます。ストアドプロシージャが複数の出力パラメータを返す場合は、それぞれの出力パラメータのための変数を作成しなければなりません。

式でストアドプロシージャを呼び出すには、次の構文を使用します。

```
:SP.<SQL transformation name> (arg1, arg2, PROC_RESULT)
```

ストアドプロシージャが 1 つの出力パラメータまたは戻り値を返すのであれば、予約変数 PROC_RESULT を出力変数として使用します。

例えば、次の式は GET_NAME_FROM_ID という名前のストアドプロシージャを呼び出します。

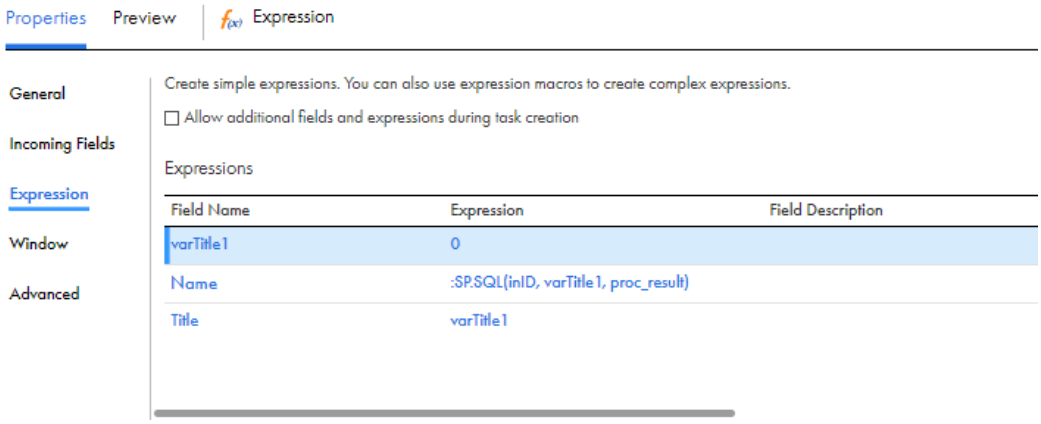
```
:SP.GET_NAME_FROM_ID(inID, PROC_RESULT)
```

inID はストアドプロシージャの入力フィールド、または式トランスフォーメーションの変数です。マッピングを実行すると、データ統合は、PROC_RESULT の値を式の出力フィールドに適用します。

ストアドプロシージャが複数の出力パラメータを返す場合は、それぞれの出力パラメータに対して式変数を作成する必要があります。例えば、ストアドプロシージャがタイトルも返す場合は、式トランスフォーメーションで varTitle1 という変数フィールドを作成し、そのフィールドを Title という出力フィールドの式として使用します。次のような式になります。

```
:SP.GET_NAME_FROM_ID(inID, varTitle1, PROC_RESULT)
```

次の図に、式トランスフォーメーションの設定方法を示します。



データ統合は、ストアドプロシージャで宣言された順序で出力パラメータを返します。この例では、データ統合は、ストアドプロシージャの最初の出力フィールドの値を varTitle1 に適用し、varTitle1 を式トランスフォーメーションの Title フィールドに渡します。ストアドプロシージャの 2 番目の出力フィールドの値を式の出力フィールドに適用します。

式フィールドと変数のデータ型は、ストアドプロシージャの入力/出力変数と戻り値のデータ型と一致する必要があります。

ストアドプロシージャをマッピング実行の前または後で呼び出す

ストアドプロシージャをマッピング実行の前または後で処理するように、接続されていない SQL トランスフォーメーションを設定できます。データ統合は、指定された時間にストアドプロシージャを呼び出します。ストアドプロシージャを:SP 式を使用して呼び出す必要はありません。ストアドプロシージャは、マッピングがソースからデータを受信する前または後、またはマッピングがターゲットにデータをロードする前または後に実行するように設定できます。

以下のストアドプロシージャタイプを設定できます。

- Source Pre Load。ストアドプロシージャは、マッピングがソースからデータを受信する前に実行されます。
- Source Post Load。ストアドプロシージャは、マッピングがソースからデータを受信した後に実行されます。
- Target Pre Load。ストアドプロシージャは、マッピングがターゲットにデータを送信する前に実行されます。
- Target Post Load。ストアドプロシージャは、マッピングがターゲットにデータを送信した後に実行されます。

[詳細] タブで、ストアドプロシージャタイプを設定し、ストアドプロシージャの呼び出しテキストを入力します。呼び出しテキストでは、ストアドプロシージャの名前に続けてかっこ内に適切な入力パラメータを指定します。入力パラメータがない場合は、かっこのみを含める必要があります。SQL 文の EXEC を含めたり、:SP キーワードを使用したりしないでください。

例えば、ストアドプロシージャ Drop_Table を呼び出すには、次の呼び出しテキストを入力します。

```
Drop_Table()
```

文字列の入力パラメータを渡す場合はそれを引用符で囲まずに入力します。文字列の中に空白が含まれている場合は、パラメータを二重引用符で囲みます。例えば、ストアドプロシージャ Drop_Table で入力パラメータとしてテーブル名が必要な場合は、次の呼び出しテキストを入力します。

```
Drop_Table(Customer_list)
```

接続されていない SQL トランスフォーメーションの例

生活費の増加に伴って従業員の給与を更新しているとします。手元には従業員の名前と ID を含む CSV ファイルがあります。それぞれの従業員の給与を増額し、その増額分を計算して、データを新しい CSV ファイルに書き込む必要があります。

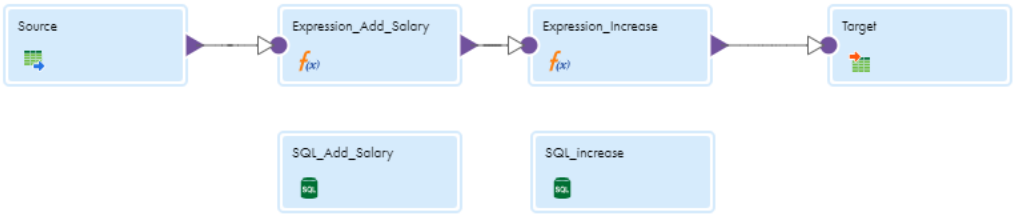
次のようなソースデータがあります。

```
EMP_ID, EMP_NAME
1001, John
1002, Alice
1003, Mary
1004, Mark
1005, Stephan
```

Oracle データベースには、従業員の給与をファイルに追加する ADD_SALARY というストアドプロシージャと、昇給を計算する SALARY_INCREASE という 2 番目のストアドプロシージャがあります。マッピングでは、接続

されていない 2 つの SQL トランスフォーメーションを使用してストアドプロシージャを呼び出し、2 つの式トランスフォーメーションを使用して SQL トランスフォーメーションを呼び出します。

次の図に、マッピングを示します。



次の方法でトランスフォーメーションを構成します。

ソーストランスフォーメーション

使用するソースデータをロードするように、ソーストランスフォーメーションを設定します。

SQL_Add_Salary **トランスフォーメーション**

ADD_SALARY ストアドプロシージャを呼び出すように最初の SQL トランスフォーメーションを設定します。

[SQL] タブで、ADD_SALARY ストアドプロシージャを含む接続を選択してから、ストアドプロシージャを選択します。

[接続されていないストアドプロシージャ] を選択します。

このストアドプロシージャには、従業員 ID の入力フィールドが 1 つあり、出力フィールドには現在の給与が返されます。

SQL_Increase **トランスフォーメーション**

SALARY_INCREASE ストアドプロシージャを呼び出すように 2 番目の SQL トランスフォーメーションを設定します。

[SQL] タブで、SALARY_INCREASE ストアドプロシージャを含む接続を選択してから、ストアドプロシージャを選択します。

[接続されていないストアドプロシージャ] を選択します。

このストアドプロシージャには、従業員名の入力フィールドが 1 つあり、出力フィールドには新しい給与が返されます。

Expression_Add_Salary **トランスフォーメーション**

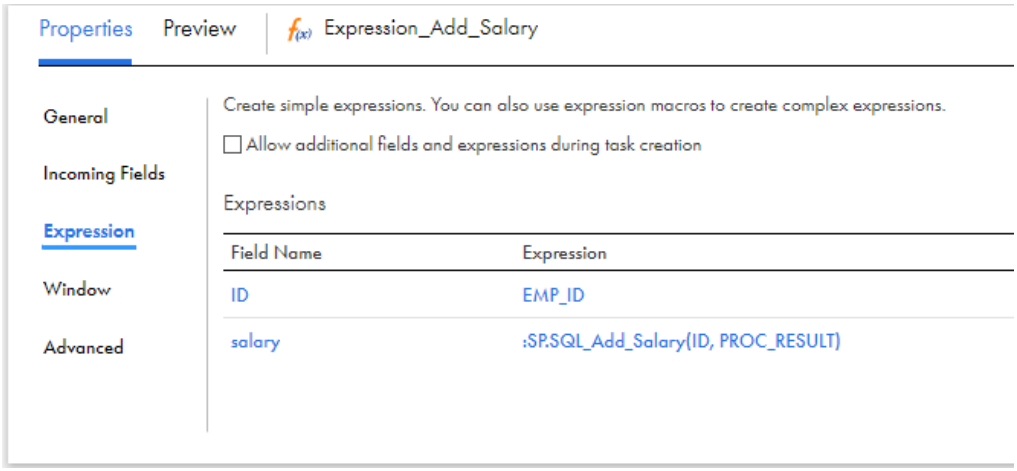
SQL_Add_Salary トランスフォーメーションを呼び出すように最初の式トランスフォーメーションを設定します。ストアドプロシージャの出力をキャプチャするために、入力パラメータの変数フィールドと出力フィールドを作成します。

[式] タブで、ID という名前の変数フィールドを追加し、その値をソースの［EMP_ID］フィールドとして設定します。最初の SQL トランスフォーメーションで ADD_SALARY ストアドプロシージャの戻り値を取得するための［salary］という出力フィールドを作成します。次の式を使用して ADD_SALARY ストアドプロシージャを呼び出すように［salary］フィールドを設定します。

```
:SP.SQL_Add_Salary(ID, PROC_RESULT)
```

この式は、ストアドプロシージャの入力パラメータとして変数フィールド ID を受け取り、給与値を［SALARY］出力フィールドに返します。

次の図に、式トランスフォーメーションの設定方法を示します。



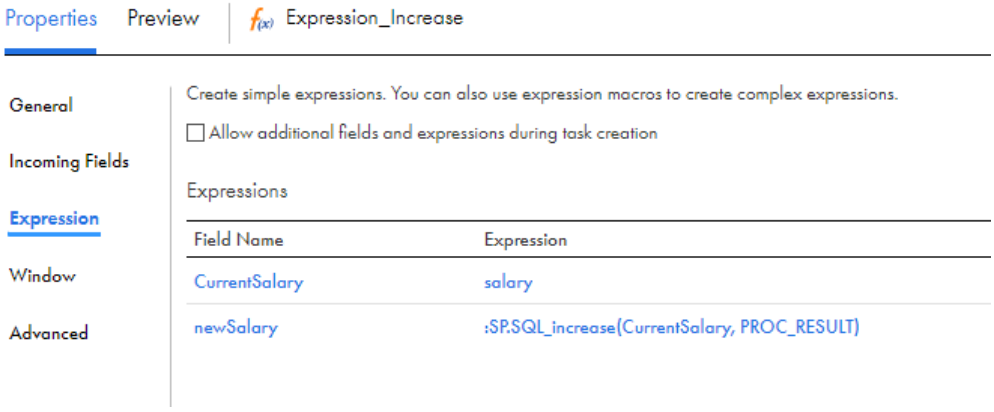
Expression_Increase **トランスフォーメーション**

SQL_increase トランスフォーメーションを呼び出すように 2 番目の式トランスフォーメーションを設定します。[CurrentSalary] という変数フィールドを追加し、その値を [入力給与] フィールドとして設定します。SALARY_INCREASE ストアドプロシージャの戻り値を取得するための [newSalary] という出力フィールドを追加します。次の式を使用して SALARY_INCREASE ストアドプロシージャを呼び出すように [newSalary] フィールドを設定します。

```
:SP.SQL_increase(CurrentSalary, PROC_RESULT)
```

この式は、変数フィールド [CurrentSalary] をストアドプロシージャの入力パラメータとして受け取り、新しい給与を [newSalary] 出力フィールドに返します。

次の図に、式トランスフォーメーションの設定方法を示します。



ターゲットトランスフォーメーション

実行時にターゲットファイルを作成するようにターゲットトランスフォーメーションを設定します。

マッピングを実行すると、次のような結果が得られます。

```
EMP_ID, EMP_NAME, salary, newSalary
1001, John, 400, 480
1002, Alice, 500, 600
1003, Mary, 400, 480
1004, Mark, 700, 840
1005, Stephan, 600, 720
```

クエリ処理

Microsoft SQL Server データベースまたは Oracle データベースに対して実行される保存済みクエリまたはユーザー入力クエリを処理するように、SQL トランスフォーメーションを設定できます。クエリを処理するように SQL トランスフォーメーションを設定する場合、アクティブなトランスフォーメーションを作成します。入力行ごとに複数の行をトランスフォーメーションで返すことができます。

クエリの入力時に、SQL を書式設定して、構文を検証することができます。または、文字列パラメータを作成して、マッピングタスクにクエリを定義できます。

以下のタイプの SQL クエリを作成できます。

静的 SQL **クエリ**

クエリ文は変更されませんが、クエリパラメータを使用してデータを変更できます。データ統合によって SQL クエリが一度準備され、そのクエリがすべての入力行に対して実行されます。

動的 SQL **クエリ**

クエリ文およびクエリデータを変更できます。データ統合は、入力行ごとに SQL クエリを準備します。

静的クエリを作成することによって、パフォーマンスを最適化できます。

静的 SQL クエリ

各入力行に対して同じクエリ文を実行する必要があるが、クエリ内のデータを入力行ごとに変更する場合、静的 SQL クエリを作成します。静的 SQL クエリを作成する場合、SQL エディタでパラメータのバインドを使用して、クエリデータのパラメータを定義します。

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、これらのパラメータをトランスフォーメーションの入力フィールドにバインドします。パラメータを入力フィールドにバインドする場合、クエリにフィールド名を指定します。フィールド名を疑問符（? ）で囲みます。クエリデータは、入力フィールドのデータの値に基づいて変更されます。

例えば、以下の静的クエリでは、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
```

例

以下の静的 SQL クエリは、SQL トランスフォーメーションの Employee_ID および Dept 入力フィールドにバインドするクエリパラメータを使用しています。

```
SELECT Name, Address FROM Employees WHERE Employee_Num = ?Employee_ID? and Dept = ?Dept?
```

ソースには以下の行があります。

Employee_ID	Dept
100	Products
123	HR
130	Accounting

データ統合は、行から次のクエリ文を生成します。

```
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’
```

複数のデータベース行の選択

SQL クエリに SELECT 文が含まれる場合、トランスフォーメーションによって、取得したデータベース行ごとに 1 行が返されます。SELECT 文のカラムごとに出力フィールドを設定する必要があります。出力フィールドは、SELECT 文のカラムと同じ順序である必要があります。

データベースカラムの出力フィールドを設定する場合、選択する各データベースカラムのデータ型を設定する必要があります。ネイティブデータ型をリストから選択します。ネイティブデータ型を選択すると、データ統合によってトランスフォーメーションデータ型が設定されます。

トランスフォーメーションのネイティブデータ型は、データベースカラムのデータ型と一致する必要があります。実行時に、データ統合によってデータベース内のカラムのデータ型がトランスフォーメーションのネイティブデータベースタイプと照合されます。データ型が一致しない場合は、データ統合によって行エラーが生成されます。

動的 SQL クエリ

動的 SQL クエリでは、入力行ごとに異なるクエリ文を実行できます。動的 SQL クエリを作成する場合、文字列の置換を使用してクエリ内に文字列変数を定義し、これらの変数をトランスフォーメーションの入力フィールドにリンクします。

クエリ文を変更するには、変更するクエリ部分の文字列変数をクエリ内に設定します。文字列変数を設定するには、クエリに入力フィールドを名前で指定し、名前をティルダ文字（~）で囲みます。クエリは、フィールドのデータの値に基づいて変更されます。

クエリ変数を含むトランスフォーメーション入力フィールドは、 文字列型データ型である必要があります。文字列の置換を使用して、クエリ文とクエリデータを変更できます。

動的 SQL クエリを作成すると、データ統合では、入力行ごとにクエリが準備されます。入力フィールドに次のタイプの動的クエリを渡すことができます。

完全なクエリ

SQL クエリ全体をソースデータのクエリ文で置換できます。

部分クエリ

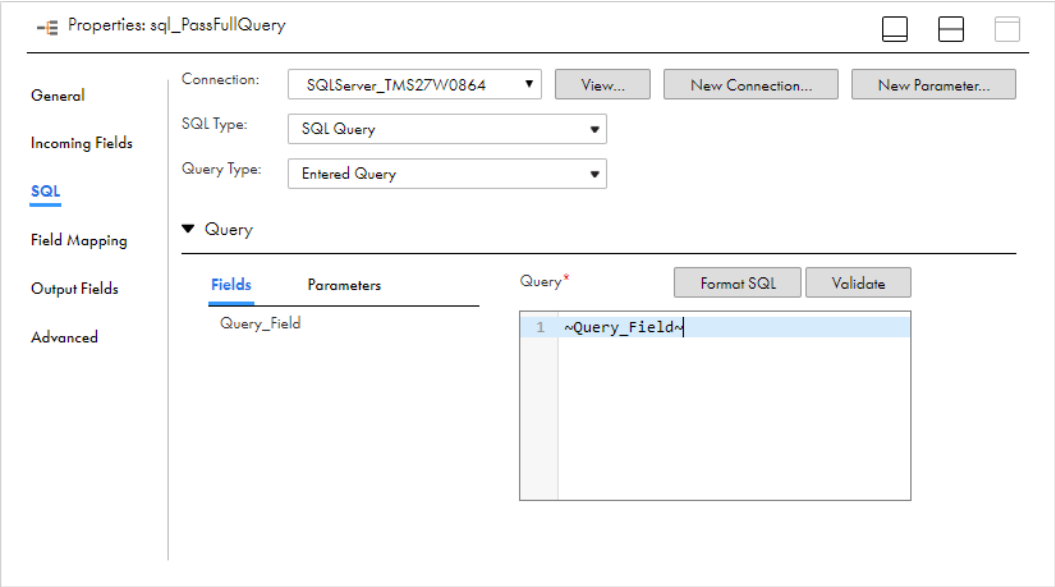
クエリ文の一部（テーブル名など）を置換できます。

完全なクエリの受け渡し

トランスフォーメーションの入力フィールドを介して完全な SQL クエリを渡すことができます。完全なクエリを渡すには、以下のように完全なクエリを表す 1 つの文字列変数で構成されたクエリを SQL エディタで作成します。`~Query_Field~`

完全なクエリを渡すには、出力フィールドに完全なクエリを渡すようにソースを設定します。次に、Query_Field 入力フィールドのクエリを受け取るように SQL トランスフォーメーションを設定します。

次の図は、トランスフォーメーションの設定を示しています。



データ統合によって、動的クエリ内の~Query_Port~変数は、ソースの SQL 文で置換されます。クエリーが準備され、処理のためにデータベースに送信されます。データベースによってクエリーが実行されます。SQL トランスフォーメーションによって、データベースエラーが SQLError 出力フィールドに返されます。

完全なクエリーを渡すときに、入力行ごとに複数のクエリー文を渡すことができます。たとえば、ソースに以下の行が含まれている場合があります。

```
DELETE FROM Person WHERE LastName = 'Jones'; INSERT INTO Person (LastName, Address) VALUES ('Smith', '38 Summit Drive')
DELETE FROM Person WHERE LastName = 'Jones'; INSERT INTO Person (LastName, Address) VALUES ('Smith', '38 Summit Drive')
DELETE FROM Person WHERE LastName = 'Russell';
```

任意のタイプのクエリーをソースデータで渡すことができます。クエリに SELECT 文を設定する場合、データベースから取得するデータベースカラムの出力フィールドを設定する必要があります。SELECT 文と他のタイプのクエリを混在させると、データベースカラムが取得されない場合、データベースカラムを表す出力フィールドには NULL 値が入ります。

文字列内のテーブル名の置換

部分クエリを使用して、テーブル名を置換することができます。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フィールドを設定します。クエリに入力フィールドを名前で指定し、名前をティルダ文字（~）で囲みます。

例えば、次の動的クエリには文字列変数~Table_Field~が含まれています。

```
SELECT Emp_ID, Address from ~Table_Field~ where Dept = 'HR'
```

ソースによって、以下の値が Table_Field カラムに渡されるとします。

Table_Field

Employees_USA

Employees_England

Employees_Australia

データ統合では、~Table_Field~変数が入力フィールド内のテーブル名で置換されます。

```
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'
```

パッシブモードの設定

SQL トランスフォーメーションを作成する際に、SQL トランスフォーメーションがアクティブモードではなくパッシブモードで実行されるように設定できます。パッシブトランスフォーメーションでは、通過する行の数は変更されません。トランザクション境界と行タイプが維持されます。

SQL トランスフォーメーションがクエリを処理するように設定する場合は、トランスフォーメーションの作成時にパッシブモードを設定できます。トランスフォーメーションの詳細プロパティでパッシブモードを設定します。

トランスフォーメーションをパッシブモードに設定し、SELECT クエリが複数の行を返す場合、データ統合は最初の行とエラーを SQLError フィールドに返します。このエラーは、SQL トランスフォーメーションが複数行を生成したことを示します。

SQL クエリに複数の SQL 文がある場合、データ統合はすべての文を実行しますが、最初の SQL 文のデータのみを返します。SQL トランスフォーメーションでは 1 行が返されます。SQLError フィールドには、すべての SQL 文のエラーが含まれます。複数のエラーが発生した場合、SQLError フィールド内でエラーはセミコロン (;)で区切られます。

クエリで使用できる SQL 文

SQL トランスフォーメーションで、特定のデータ定義、データ操作、データ制御言語、およびトランザクション制御文を使用できます。

次の表に、SQL トランスフォーメーションの SQL クエリで使用できる文を示します。

文のタイプ	文	説明
データ定義	ALTER	データベースの構造を変更します。
データ定義	COMMENT	データディクショナリにコメントを追加します。
データ定義	CREATE	データベース、テーブル、またはインデックスを作成します。
データ定義	DROP	インデックス、テーブル、またはデータベースを削除します。
データ定義	RENAME	データベースオブジェクトの名前を変更します。
データ定義	TRUNCATE	テーブルからすべての行を削除します。
データ操作	CALL	PL/SQL または Java サブプログラムを呼び出します。
データ操作	DELETE	テーブルから行を削除します。
データ操作	EXPLAIN PLAN	データベースの Explain テーブルに文のアクセスプランを書き込みます。
データ操作	INSERT	行をテーブルに挿入します。
データ操作	LOCK TABLE	アプリケーションプロセスが同時にテーブルを使用または変更することを防止します。

文のタイプ	文	説明
データ操作	MERGE	ソースデータを使用してテーブルを更新します。
データ操作	SELECT	データベースからデータを取得します。
データ操作	UPDATE	テーブルの行の値を更新します。
データ制御言語	GRANT	データベースユーザに特権を付与します。
データ制御言語	REVOKE	データベースユーザのアクセス特権を削除します。
トランザクションコントロール	コミット	作業ユニットを保存し、その作業ユニットのデータベース変更を実行します。
トランザクション制御	ROLLBACK	最後の COMMIT 以降のデータベースへの変更を取り消します。

クエリ処理に関するルールおよびガイドライン

クエリを処理するように SQL トランスフォーメーションを設定する場合は、以下のルールおよびガイドラインを使用します。

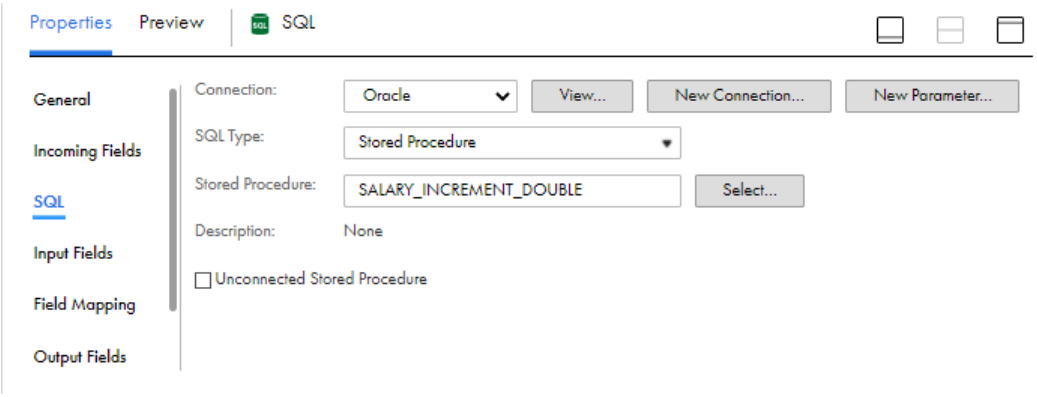
- 出力フィールドの数および順序は、クエリの SELECT 句内のフィールドの数および順序と一致する必要があります。
- トランスフォーメーション内の出力フィールドのネイティブデータ型は、データベース内の対応するカラムのデータ型と一致する必要があります。データ型が一致しない場合は、データ統合によって行エラーが生成されます。
- SQL クエリに INSERT、UPDATE、または DELETE 句が含まれる場合、トランスフォーメーションによってデータが SQLError フィールド、パススルーフィールド、および NumRowsAffected フィールド（有効な場合）に返されます。出力フィールドを追加した場合、出力フィールドは NULL データ値を受け取ります。
- SQL クエリに SELECT 文が含まれており、トランスフォーメーションにパススルーフィールドが設定されている場合、トランスフォーメーションはクエリからデータベースのデータが返されるかどうかにかかわらず、パススルーフィールドにデータを返します。SQL トランスフォーメーションは、NULL データを含む行を出力フィールドに返します。
- 出力フィールドの数が SELECT 句内のカラム数よりも多い場合、余分のフィールドは NULL 値を受け取ります。
- 出力フィールドの数が SELECT 句内のカラム数よりも少ない場合、データ統合によって行エラーが生成されます。
- クエリ内のパラメータのバインドの代わりに文字列の置換を使用できます。ただし、入力フィールドは文字列データ型である必要があります。

SQL トランスフォーメーションの設定

SQL トランスフォーメーションを設定するには、**［プロパティ］**パネルを使用します。トランザクションを設定する際に、トランスフォーメーションの名前と説明を指定し、フィールドとフィールドマッピングを設定し

て、トランスフォーメーションが処理する SQL のタイプを指定します。トランスフォーメーションの詳細プロパティも設定します。

次の図は、トアドプロシージャを処理するように設定された SQL トランスフォーメーションの［**プロパティ**］パネルを示します。



［**プロパティ**］パネルの次のタブを使用して、トランスフォーメーションを設定します。

全般

SQL トランスフォーメーション名とオプションの説明を設定します。

受信フィールド

トランスフォーメーションに含めるデータを決定するフィールドルールを定義します。

接続されていない SQL トランスフォーメーションには適用されません。

SQL

データベース接続と、トランスフォーメーションが処理する SQL のタイプ（ストアドプロシージャ、ストアド関数、またはクエリ）を定義します。

トランスフォーメーションがストアドプロシージャ、ストアド関数、または保存済みクエリを処理するように設定した場合は、このタブでストアドプロシージャ、ストアド関数、または保存済みクエリを選択します。

ストアドプロシージャを処理するようにトランスフォーメーションを設定する場合は、接続されていないトランスフォーメーションとしてトランスフォーメーションを実行できます。

トランスフォーメーションがユーザー入力クエリを処理するように設定すると、このタブに SQL エディタが表示されます。SQL エディタにクエリを入力します。

入力フィールド

ストアドプロシージャを処理するトランスフォーメーションの場合に、ストアドプロシージャの入力フィールドを表示します。

フィールドマッピング

ストアドプロシージャおよびストアド関数に対して、選択したストアドプロシージャまたはストアド関数の入力フィールドに受信フィールドをマップする方法を指定します。

クエリまたは接続されていない SQL トランスフォーメーションのフィールドマッピングは設定しません。

出力フィールド

ストアドプロシージャ、ストアド関数、および保存済みクエリの場合、SQL トランスフォーメーションの出力フィールドのプレビューが表示されます。ユーザー入力クエリの場合は、データベースから取得するカラムの出力フィールドを設定します。

クエリの場合、出力フィールドには、SQLError フィールド、オプションの NumRowsAffected フィールド、およびオプションのパススルーフィールドが含まれます。

詳細

トランスフォーメーションの詳細プロパティを定義します。詳細プロパティは、トランスフォーメーションが処理する SQL のタイプによって異なります。

注: フィールド名の競合は、アップストリームトランスフォーメーションで解決する必要があります。SQL トランスフォーメーションでフィールド名の競合解決ルールを使用することはできません。

SQL タイプの設定

SQL トランスフォーメーションの［**SQL**］タブで、ストアドプロシージャ、ストアド関数、保存済みクエリ、またはユーザー入力クエリを処理するように SQL トランスフォーメーションを設定できます。

トランスフォーメーションを設定する手順は、トランスフォーメーションが処理する SQL のタイプによって異なります。

ストアドプロシージャまたはストアド関数の選択

SQL トランスフォーメーションの［**SQL**］タブで、ストアドプロシージャまたはストアド関数を処理するように SQL トランスフォーメーションを設定できます。

1. SQL トランスフォーメーションの［**プロパティ**］パネルで［**SQL**］タブをクリックします。
2. データベースへの接続を選択します。

   接続を選択するか、パラメータを使用することができます。

   注: 接続をパラメータ化する場合は、ストアドプロシージャまたはストアド関数を選択した後にパラメータを作成します。
3. SQL タイプを［**ストアドプロシージャ**］または［**ストアド関数**］に設定します。
4. ［**選択**］をクリックしてデータベースからストアドプロシージャまたはストアド関数を選択するか、呼び出すストアドプロシージャまたはストアド関数の正確な名前を入力します。

   ストアドプロシージャ名とストアド関数名では大文字と小文字が区別されます。

   注: マッピングを開いているときに新しいストアドプロシージャをデータベースに追加すると、新しいストアドプロシージャは使用可能なストアドプロシージャのリストに表示されません。リストを更新するには、マッピングを閉じて再度開きます。
5. トランスフォーメーションがストアドプロシージャを処理し、未接続モードでトランスフォーメーションを実行する場合は、［**未接続のストアドプロシージャ**］を選択します。
6. 接続をパラメータ化する場合は、［**新しいパラメータ**］をクリックし、接続パラメータの詳細を入力します。

保存済みクエリの選択

SQL トランスフォーメーションの［**SQL**］タブで、保存済みクエリを処理するように SQL トランスフォーメーションを設定できます。

1. SQL トランスフォーメーションの［**プロパティ**］パネルで［**SQL**］タブをクリックします。
2. データベースへの接続を選択します。

   接続を選択するか、パラメータを使用することができます。
3. SQL タイプを［**SQL クエリ**］に設定します。

4. クエリタイプを［**保存済みクエリ**］に設定します。
5. ［**選択**］をクリックして、データ統合アセットのリストから保存済みクエリを選択します。

クエリの入力

SQL トランスフォーメーションの［**SQL**］タブで、ユーザー入力クエリを処理するように SQL トランスフォーメーションを設定できます。オプションとして、クエリをパラメータ化できます。クエリをパラメータ化する場合は、マッピングタスクに完全なクエリを入力します。

1. SQL トランスフォーメーションの［**プロパティ**］パネルで［**SQL**］タブをクリックします。
2. データベースへの接続を選択するか、パラメータを使用します。
3. SQL タイプを［**SQL クエリ**］に設定します。
4. クエリタイプを［**入力済みクエリ**］に設定します。
5. クエリをパラメータ化しない場合は、クエリエディタでクエリを入力します。

   受信フィールドが［**フィールド**］タブにリストされます。フィールドをクエリに追加するには、フィールドを選択して［**追加**］をクリックします。

   SQL を書式設定して、構文を検証することができます。

   注: 構文検証では、一般的な SQL 構文チェックを実行しますが、データベースに対して SQL を確認しません。検証では、SQL がデータベースに対して有効であっても、構文エラーを返すことがあります。この場合でも、マッピングを保存して実行できます。

   クエリの設定後に受信フィールドを更新する場合は、［**SQL**］タブで変更内容を更新します。
6. クエリをパラメータ化する場合は、次の手順を実行します。
   a. ［**パラメータ**］タブを開いて、新しい文字列パラメータを作成します。
   b. パラメータを選択して［**追加**］をクリックすることによって、パラメータをクエリエディタに追加します。

      パラメータをクエリエディタに追加すると、データ統合では、そのパラメータがドル記号文字($)で囲まれます。

      SQL の書式設定またはクエリの検証は行わないでください。

SQL トランスフォーメーションのフィールドマッピング

ストアドプロシージャまたはストアド関数で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を使用する方法を定義するには、SQL トランスフォーメーションでフィールドマッピングを設定します。トランスフォーメーションがクエリを処理する場合、フィールドマッピングは設定しません。

［**プロパティ**］パネルの［**フィールドマッピング**］タブでフィールドマッピングを設定します。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

SQL トランスフォーメーションにフィールドをマッピングする方法。次のいずれかのオプションを選択します。

- 手動。受信フィールドをストアドプロシージャまたはストアド関数の入力フィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。タスクで、すべてのフィールドマッピングを設定できます。

- 一部パラメータ化。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。
  フィールドマッピングパラメータの詳細については、「*マッピング*」を参照してください。

フィールドの表示

［受信フィールド］リストに表示されるフィールドを制御します。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示します。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似した名前のフィールドをマッピングできます。

［オートマップ］ > **［オートマップを元に戻す］**をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**［アクション］** > **［マップ解除］**をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

アクションメニュー

追加フィールドリンクオプション。次のオプションが用意されています。

- 選択項目をマップ。選択した受信フィールドを、選択したストアドプロシージャまたはストアド関数の入力フィールドにリンクします。
- 選択項目をマップ解除。選択したフィールドのリンクをクリアします。
- すべてクリア。すべてのフィールドマッピングをクリアします。

表示

フィールド名を**［ストアドプロシージャの入力フィールド］**リストに表示する方法を決定します。技術フィールド名またはラベルを使用します。

SQL トランスフォーメーションの出力フィールド

SQL トランスフォーメーションの出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。Mapping Designer には、出力フィールドについて、名前、型、精度、スケール、基点が表示されます。

［出力フィールド］タブ上の情報は、SQL タイプによって異なります。

ストアドプロシージャおよびストアド関数の出力フィールド

SQL トランスフォーメーションがストアドプロシージャまたはストアド関数を処理する場合は、出力フィールドにデータベースの出力パラメータが含まれます。トランスフォーメーション出力フィールドを編集することはできません。出力フィールドをデータフローから除外したり、出力フィールドの名前を変更したりしてから、

出力フィールドをダウンストリームトランスフォーメーションに渡す場合は、ダウンストリームトランスフォーメーション用にフィールドルールを設定します。

クエリの出力フィールド

SQL トランスフォーメーションがクエリを処理する場合は、出力フィールドに次のフィールドが含まれます。

取得したカラムフィールド

SQL クエリに SELECT 文が含まれる場合、トランスフォーメーションによって、取得したデータベース行ごとに 1 行が返されます。

ユーザー入力クエリの場合は、SELECT 文のカラムごとに出力フィールドを設定する必要があります。出力フィールドは、SELECT 文のカラムと同じ順序である必要があります。

保存済みクエリの場合は、データ統合によって出力フィールドが作成されます。

SQLError **フィールド**

データ統合は、接続エラーまたは構文エラーを検出した場合に行エラーを SQLError フィールドに返します。SQL エラーが発生しなかった場合は、NULL を SQLError フィールドに返します。

例えば、次の SQL クエリでは、Employees テーブルに Product_ID が含まれない場合、Oracle データベースから行エラーが生成されます。

```
SELECT Product_ID FROM Employees
```

データ統合は、1 行を生成します。SQLError フィールドには、次の 1 行のエラーテキストが含まれます。

```
ORA-0094: “Product_ID”: invalid identifier Database driver error... Function Name: Execute SQL Stmt: SELECT Product_ID from Employees Oracle Fatal Error
```

クエリーに複数の文が含まれ、SQL エラー時で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した場合、SQL トランスフォーメーションでは、1 つのクエリー文に対してデータベースから行が返されますが、別のクエリー文に対してデータベースエラーが返されることがあります。データベースエラーは、SQL トランスフォーメーションによって別の行に返されます。

NumRowsAffected **フィールド**

NumRowsAffected 出力フィールドを有効にすると、各入力行に指定された INSERT、UPDATE、または DELETE クエリ文の影響を受けた行の数を返すことができます。データ統合は、クエリ内の各文に NumRowsAffected を返します。NumRowsAffected はデフォルトでは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 文が含まれない場合、各出力行の NumRowsAffected はゼロになります。

次の表に、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。

クエリー文	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 クエリー	ゼロの NumRowsAffected が含まれる 1 行。

クエリに複数の文が含まれる場合、データ統合は、文ごとに NumRowsAffected を返します。NumRowsAffected には、入力行に指定された INSERT、UPDATE、DELETE の各文の影響を受け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。

データ統合は、DELETE 文から 1 行を返します。NumRowsAffected は 1 になります。SELECT 文から 1 行返しますが、NumRowsAffected はゼロです。INSERT 文から 1 行返し、NumRowsAffected は 1 になります。

パススルーフィールド

受信フィールドをパススルーフィールドとして定義して、SQL トランスフォーメーション経由でデータを渡します。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、パススルーフィールドからデータを返します。

ソース行に SELECT 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フィールド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、各行にパススルーフィールドの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い場合、SQL トランスフォーメーションは、NULL 値を含むパススルーカラムデータを出力フィールドに返します。例えば、INSERT 文、UPDATE 文、および DELETE 文を含むクエリは行を返しません。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、SQLError メッセージ、および NULL 値を出力フィールドに返します。

パススルーフィールドを定義するには、パススルーフィールドの領域で **[追加]** をクリックし、SQL トランスフォーメーションのパススルーを行うフィールドを選択します。受信フィールドをパススルーフィールドとして設定した場合、データ統合によって、パススルーフィールド領域にサフィックス「_output」が付いたフィールドが追加されます。

フィールドをパススルーフィールドとして設定してからソースのフィールド名を変更した場合、データ統合ではパススルーフィールド名が更新されないため、データはフィールドを通過しません。SQL トランスフォーメーションで古いパススルーフィールドを削除し、更新された受信フィールドをパススルーフィールドとして設定します。

詳細プロパティ

[詳細] タブで SQL 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。詳細プロパティは、トランスフォーメーションがストアドプロシージャまたはストアド関数を処理するか、クエリを処理するかによって異なります。

ストアドプロシージャまたはストアド関数の詳細プロパティ

次の表に、トランスフォーメーションがストアドプロシージャまたはストアド関数を処理する場合の詳細プロパティを示します。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
サブ秒の精度	日時フィールドのサブ秒の精度。日時データの位取りを編集できるデータベースでは、精度を変更できます。 プッシュダウンの最適化を有効にした場合、データベースはサブ秒の精度設定に関係なく、日時の値全体を返します。 0 から 9 までの正の整数値を入力します。デフォルトは 6 マイクロ秒です。
ストアドプロシージャタイプ	接続されていないトランスフォーメーションの場合に、ストアドプロシージャが実行されるタイミングを決定します。 次のいずれかのオプションを選択します。 - ターゲットロード前。ターゲットがロードされる前に実行されます。 - ターゲットロード後。ターゲットがロードされた後に実行されます。 - ノーマル。行単位で実行されます。 - ソースロード前。マッピングがソースからデータを受信する前に実行されます。 - ソースロード後。マッピングがソースからデータを受信した後に実行されます。
Call Text	ストアドプロシージャタイプが [ターゲットロード前]、[ターゲットロード後]、[ソースロード前]、または [ソースロード後] である接続されていないトランスフォーメーションの場合に、ストアドプロシージャの呼び出しテキストを入力します。 呼び出しテキストは、ストアドプロシージャ名に続けて、かっこ内に入力パラメータを指定します。入力パラメータがない場合は、かっこのみを含める必要があります。SQL 文の EXEC を含めたり、:SP キーワードを使用したりしないでください。 [通常] ストアドプロシージャタイプには適用されません。

クエリの詳細プロパティ

次の表に、トランスフォーメーションが保存済みクエリまたはユーザー入力クエリを処理する場合の詳細プロパティを示します。

プロパティ	説明
動作	トランスフォーメーション動作。アクティブまたはパッシブ。 アクティブの場合、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。パッシブの場合、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 1 つの出力行を生成します。 デフォルトはアクティブです。
行内の SQL エラー時でも処理を継続する	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。 文内の SQL エラーを無視するには、このオプションを有効にします。データ統合によって、その行の残りの SQL 文の実行が続行されます。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が、SQLError フィールドに、失敗した SQL 文およびエラーメッセージが記載されます。 **ヒント**: データベースエラーをデバッグするには、このオプションを無効にします。このオプションを無効にしない場合、エラーと発生元のクエリー文を関連付けることができないことがあります。 デフォルトでは無効になっています。
自動コミット	各データベース接続に対する自動コミットを有効にします。 クエリ内の各 SQL 文によってトランザクションが定義されます。SQL 文が完了するか、次の文が実行されると、コミットが実行されます。 デフォルトでは無効になっています。
最大出力行数	SQL トランスフォーメーションで SELECT クエリから出力できる最大行数。 行数を無制限に設定するには、このプロパティをゼロに設定します。デフォルトは 600 です。
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。[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。
トランスフォーメーション範囲	データ統合が受信データに対してトランスフォーメーションロジックを適用する方法。 次のいずれかのオプションを選択します。 - 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。トランスフォーメーション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。データの行が同一トランザクション内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。 - 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、データ統合は受信トランザクションの境界を削除します。データの行がソース内のすべての行に依存している場合は、[すべての入力] を選択します。 デフォルトは [行] です。

第 32 章

構造パーサートランスフォーメーション

構造パーサートランスフォーメーションは、インテリジェント構造モデルに基づいて、入力データをユーザー定義の構造化形式に変換します。構造パーサートランスフォーメーションを使用して、ログファイル、クリックストリーム、XML ファイルまたは JSON ファイル、Word テーブル、およびその他の非構造化形式または半構造化形式などのデータを分析できます。

構造パーサートランスフォーメーションは次のタイプのソースに接続できます。

- ローカル入力ファイルを処理するためのフラットファイルに基づくソーストランスフォーメーション
- HDFS での入力ファイルのストリーミングやローカル入力ファイルの処理を行うための、Hadoop Files V2 接続に基づくソーストランスフォーメーション

構造パーサートランスフォーメーションの設定時に、このトランスフォーメーションをインテリジェント構造モデルに関連付けます。インテリジェント構造モデルとは、実行時にモデルが解析することが想定されるデータを表すためにインテリジェント構造検出によって生成されるアセットです。モデルは、構造パーサートランスフォーメーションを設定する前、または構造パーサートランスフォーメーションの設定時に作成できます。

インテリジェント構造検出は、入力データのサンプルまたは提供されたスキーマに基づいてインテリジェント構造モデルを生成します。モデルは次のような入力タイプから作成することができます。

- テキストファイル（CSV ファイルなどの区切りファイルやテキスト階層を含む複雑なファイルを含む）
- 機械生成されたファイル（Web ログやクリックストリームなど）
- JSON ファイル
- XML ファイル
- ORC ファイル
- Avro ファイル
- Parquet ファイル
- Microsoft Excel ファイル
- PDF フォームフィールド内のデータ
- Microsoft Word テーブル内のデータ
- XSD ファイル
- COBOL コピーブック

プレビュー通知: COBOL コピーブックに基づいたインテリジェント構造モデルの作成をプレビューできるようになりました。プレビュー機能は評価を目的としてサポートされていますが、保証対象外で、本番環境または本番環境にプッシュする予定の環境には対応していません。Informatica は、本番環境用の今後のリリースに

プレビュー機能を含める予定ですが、市場や技術的な状況の変化に応じて導入を行わ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。

インテリジェント構造検出によってインテリジェント構造モデルが生成された後に、そのモデルを調整して出力データの構造をカスタマイズできます。モデルのノードを編集し、ノードの結合、除外、フラット化、または縮小を行うことができます。

Hadoop ファイルソースからの入力の処理

Hadoop ファイルソースを構造パーサートランスフォーメーションに接続すると、ローカルファイル入力または HDFS 入力を処理できます。例えば、Hadoop 環境のストリーミングモードの場合、構造パーサートランスフォーメーションで Hadoop ファイルソースからの大きなデータ入力を処理できます。

Hortonworks Data Platform または Cloudera ディストリビューションへの Hadoop ファイル接続を作成し、構造パーサートランスフォーメーションに入力を提供できます。

Hadoop Files V2 コネクタは、1 つのファイルまたはディレクトリ全体を処理できます。ディレクトリを処理する場合、コネクタはファイルを再帰的に処理します。

構造パーサートランスフォーメーションで Hadoop ファイルソースからの出力を処理できるようにするには、構造パーサートランスフォーメーションでフィールドマッピング設定を定義します。**［受信フィールド］**パネルの**［データ］**および**［ファイルパス］**フィールドを**［構造パーサー入力フィールド］**パネルの**［データ］**および**［ファイルパス］**フィールドにマッピングします。フィールドのマッピングは自動または手動で実行できます。

フラットファイルソースからの入力の処理

フラットファイルソースを追加すると、構造パーサートランスフォーメーションで 1 つ以上のファイルからの入力を処理できます。

ソーストランスフォーメーションは、構造パーサートランスフォーメーションで処理する 1 つ以上のファイルのファイルパスまたはファイルパスのリストが含まれる参照ファイルを使用します。ソーストランスフォーメーションを設定する場合は参照ファイルを使用していることを確認してください。**［ソース］**タブの**［オブジェクト］**フィールドで、参照ファイルを選択します。

フラットファイルソースの設定

構造パーサートランスフォーメーションでフラットファイルソースを使用するには、ソーストランスフォーメーションに 1 つのソースフィールドがあることを確認します。

1. ソーストランスフォーメーションの**［ソース］**タブで、**［オブジェクト］**フィールドの近くにある**［形式オプション］**を選択します。
2. **［テキスト修飾子］**で**［なし］**を選択します。
3. **［フィールドラベル］**で**［自動生成］**を選択します。
4. **［OK］**をクリックします。

フラットファイルにアクセスするための構造パーサートランスフォーメーションの設定

構造パーサートランスフォーメーションが参照ファイルでファイルにアクセスできるようにするには、フィールドマッピングを設定します。

- 構造パーサートランスフォーメーションの**［フィールドマッピング］**タブで、**［受信フィールド］**パネルのフィールドを**［構造パーサー入力フィールド］**パネルの**［ファイルパス］**フィールドにマッピングします。

構造パーサーフィールドマッピング

構造パーサートランスフォーメーションのフィールドマッピングを設定し、構造パーサートランスフォーメーションのフィールドにマッピングするソーストランスフォーメーションのフィールドを定義します。**［プロパティ］**パネルの**［フィールドマッピング］**タブでフィールドマッピングを設定します。

構造パーサーの**［構造パーサー入力フィールド］**パネルには、次の［フィールドマッピング］フィールドが含まれます。

データ

ソーストランスフォーメーションの［データ］フィールドを構造パーサートランスフォーメーションの［データ］フィールドにマッピングするために使用されます。このフィールドは、Hadoop ファイルソース、および中間ストリームを使用する構造パーサートランスフォーメーションの場合にマッピングします。

ファイルパス

ソーストランスフォーメーションのファイルパスまたは参照ファイルを構造パーサートランスフォーメーションの［ファイルパス］フィールドにマッピングするために使用されます。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

構造パーサートランスフォーメーションの入力フィールドリストでのフィールド名の表示方法を決定します。技術フィールド名またはラベルを使用します。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをターゲットフィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。

注: 構造パーサートランスフォーメーションでは、パラメータ化されたフィールド名は使用できません。**［すべてパラメータを使用します］**または**［部分的にパラメータを使用します］**オプションは選択しないでください。

自動マップ

一致する名前または類似した名前のフィールドをリンクします。一致するフィールドをリンクした後、他のフィールドマッピングを手動で設定できます。

フィールドの表示

［受信フィールド］リストに表示されるフィールドを制御します。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示します。

Hadoop ファイルソースの場合

Hadoop ファイルソースの場合、**［受信フィールド］**パネルの**［データ］**および**［ファイルパス］**フィールドを**［構造パーサー入力フィールド］**パネルの同じフィールドにマッピングします。**［自動マップ］**オプションを使用すると、これらのフィールドを自動的にマッピングできます。

フラットファイルソースの場合

フラットファイルソースの場合、**［受信フィールド］**パネルのフィールドを**［構造パーサー入力フィールド］**パネルの**［ファイルパス］**フィールドにマッピングします。

中間ストリームを使用する構造パーサートランスフォーメーションの場合

構造パーサートランスフォーメーションをソーストランスフォーメーションの後ではなく中間ストリームで使用している場合は、受信データポートを**［構造パーサー入力フィールド］**パネルのフィールドにマッピングします。

出力フィールド

構造パーサートランスフォーメーションで使用するインテリジェント構造を選択すると、［プロパティ］パネルの**［出力フィールド］**タブにインテリジェント構造モデル出力フィールドが表示されます。

マッピングでは、出力フィールドは次のグループに分類されます。

- ［未定義］グループ。このグループには、インテリジェント構造で識別されなかったデータが含まれます。詳しい分析のためにこのデータをターゲットファイルに渡すことができます。
- ［出力］グループ。インテリジェント構造モデルが識別するデータを含む 1 つ以上のグループ。

パススルーフィールドを転送する 1 つまたは複数のグループを選択できます。パススルーフィールドは、トランスフォーメーションのフィールドマッピングでマッピングせず、トランスフォーメーションで選択した 1 つまたは複数のグループにそのまま転送されるフィールドです。後からマッピングでパススルーフィールドを使用する場合、例えば、タイムスタンプフィールドを次のダウンストリームトランスフォーメーションに渡す場合は、このオプションを使用します。構造パーサートランスフォーメーションは、パススルーフィールドを選択した各グループに渡します。

詳細モードのマッピングでは、出力フィールドは 1 つのグループに入れられます。パススルーフィールドを有効にするようにトランスフォーメーションを構成する場合、このグループにはパススルーフィールドが含まれます。

［出力フィールド］タブには、各出力グループの最初の 5 つのフィールド（各フィールドの名前、タイプ、精度、スケール、オリジンなど）が表示されます。グループの下部にあるリンクをクリックすると、グループ内のすべてのフィールドを表示できます。フィールドのオリジンは、インテリジェント構造モデルのそれぞれのノードのパスを示します。インテリジェント構造モデルのノードの名前に特殊文字が含まれている場合、Secure Agent はそれらをアンダースコア（_）文字に置き換え、**［出力フィールド］**タブには、修正後の名前が出力フィールド名として表示されます。**［出力フィールド］**タブには、名前が修正されたフィールドのオリジンは表示されません。

出力フィールドの精度を編集するには、その精度をクリックして、必要な精度を入力します。

注: インテリジェント構造検出では、10 進数フィールドに精度とスケール適用されません。

トランスフォーメーション出力フィールドを編集することはできません。データフローから出力フィールドを除外する場合、または出力フィールドをダウンストリームトランスフォーメーションまたはターゲットに渡す前に出力フィールド名を変更する場合は、ダウンストリームトランスフォーメーションのフィールドルールを設定します。

リレーショナル出力の出力グループ

リレーショナル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した場合、トランスフォーメーションには複数の出力グループが含まれることがあります。各出力グループには 1 つ以上のフィールドが含まれており、リレーショナルターゲットに書き込むか、以降の処理のためにダウンストリームトランスフォーメーションに渡すことができます。

詳細モードでは、構造パーサートランスフォーメーションはリレーショナル出力を生成できません。

JSON、JSON 行、および XML 出力の出力グループ

JSON、JSON 行、または XML 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した場合、トランスフォーメーションには 1 つの出力グループが含まれます。出力グループには、出力データを示す 1 つの文字列フィールドが含まれます。デフォルトの精度は 1,000,000 文字です。文字列フィールドのデータは、ダウンストリームトランスフォーメーションまたは適切なタイプのターゲットに渡すことができます。例えば、XML 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した場合、出力フィールドをフラットファイルターゲットに接続し、出力を XML ファイルに保存することができます。

Avro、Parquet、および ORC 出力の出力グループ

Avro、Parquet、または ORC 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した場合、トランスフォーメーションには 1 つの出力グループが含まれます。出力グループには、出力データを示す 1 つのバイナリフィールドが含まれます。デフォルトの精度は 1,000,000 文字です。バイナリフィールドのデータは、ダウンストリームトランスフォーメーションまたは適切なタイプのターゲットに渡すことができます。例えば、Parquet 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した場合、出力フィールドを Amazon S3 や Hadoop Files などのターゲットに接続できます。

詳細モードでは、構造パーサートランスフォーメーションは Avro、Parquet、または ORC 出力を生成できません。

詳細プロパティ

構造パーサートランスフォーメーションの詳細プロパティを設定できます。詳細プロパティによって、トランスフォーメーション範囲を制御します。

次のプロパティを設定することができます。

プロパティ	説明
トランスフォーメーション範囲	データ統合が入力データにトランスフォーメーションロジックを適用する方法を指定します。次のいずれかのオプションを選択します。 - 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。データの行が同一トランザクション内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。 - 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、データ統合は入力トランザクションの境界を削除します。データの行がソース内のすべての行に依存している場合は、[すべての入力] を選択します。 - 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。トランスフォーメーション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。

構造パーサートランスフォーメーションの設定

構造パーサートランスフォーメーションを設定する場合は、インテリジェント構造モデルをトランスフォーメーションに関連付けて、出力タイプを選択します。

インテリジェント構造モデルをトランスフォーメーションに関連付けるには、既存のモデルを選択するか、新しいモデルを作成します。新しいモデルを作成する場合は、次のいずれかのアクションを選択します。

- 新規設計。モデルは、インテリジェント構造モデルページで作成します。インテリジェント構造モデルを作成する方法については、「*コンポーネント*」を参照してください。
- サンプルからの自動生成。サンプルファイルを選択すると、インテリジェント構造検出によりモデルが作成され、選択した場所に保存されます。

モデルを使用する前に、インテリジェント構造モデルページでモデルを選択して表示することもできます。

構造パーサートランスフォーメーションを設定する場合は、データ出力タイプを選択します。このトランスフォーメーションでは、次の出力タイプを生成できます。

- リレーショナル
- JSON
- JSON 行
- XML
- Avro
- Parquet
- ORC

構造パーサートランスフォーメーションの設定

構造パーサートランスフォーメーションをマッピングに追加し、トランスフォーメーション設定を行います。

1. 構造パーサートランスフォーメーションをマッピングに追加し、全般設定を行います。
2. **［プロパティ］**パネルで、**［構造パーサー］**をクリックし、次のいずれかのオプションを選択してインテリジェント構造モデルをトランスフォーメーションと関連付けます。

オプション	説明
選択	既存のインテリジェント構造モデルを選択します。
［新規］ > ［新規を設計］	新しいモデルをインテリジェント構造モデルページで作成します。
［新規］ > ［サンプルファイルから自動生成］	サンプルファイルとモデルの場所を選択し、**［作成］**をクリックします。インテリジェント構造検出によってモデルが作成され、選択した場所に保存されます。

3. **［出力に名前を付けて保存］**リストから構造パーサートランスフォーメーションの出力タイプを選択します。
4. JSON 出力タイプを選択した場合は、**［空のタグを含める］**を選択して、入力に存在しないタグを含むすべてのモデルタグを実行時に出力へ追加できます。トランスフォーメーションによって、入力に存在しないタグが NULL 値の空のタグとして追加されます。
5. **［プロパティ］**パネルで、**［受信フィールド］**をクリックし、受信フィールドを設定します。
6. **［プロパティ］**パネルで、**［フィールドマッピング］**をクリックし、フィールドマッピングを設定します。

7. 必要に応じて、**[プロパティ]** パネルで **[詳細]** をクリックし、トランスフォーメーション範囲を設定します。
8. マップの前のトランスフォーメーションを構造パーサートランスフォーメーションにリンクします。
9. 構造パーサートランスフォーメーションをダウンストリームトランスフォーメーションにリンクし、出力グループを選択します。

インテリジェント構造モデルの選択

[プロパティ] パネルの **[構造パーサー]** タブで、構造パーサートランスフォーメーションに取り込むインテリジェント構造モデルを選択します。

1. 構造パーサートランスフォーメーションの **[プロパティ]** パネルで、**[構造パーサー]** タブをクリックします。
2. **[選択]** をクリックします。

   [インテリジェント構造の選択] ダイアログボックスが表示されます。
3. **[参照]** リストで、インテリジェント構造モデルを選択します。
4. インテリジェント構造モデルを検索するには、検索条件を選択して、**[検索]** フィールドに検索する文字を入力します。

   インテリジェント構造モデルは名前または説明で検索できます。インテリジェント構造モデルのソートには、名前、タイプ、説明、タグ、ステータス、または最終更新日時を使用できます。
5. 構造パーサートランスフォーメーションに取り込むインテリジェント構造モデルを選択して、**[選択]** をクリックします。

   インテリジェント構造モデルが **[プロパティ]** パネルに表示されます。

出力グループの選択

構造パーサートランスフォーメーションをダウンストリームトランスフォーメーションにリンクする場合、ダウンストリームトランスフォーメーションにマッピングする出力グループを 1 つ選択する必要があります。複数の出力グループをマッピングする場合は、出力グループごとに異なるダウンストリームトランスフォーメーションを作成します。

1. 構造パーサートランスフォーメーションをダウンストリームトランスフォーメーションにリンクします。

   [出力グループの選択] ダイアログボックスが表示されます。
2. インテリジェント構造によって識別されたデータをマッピングするには、出力グループのいずれかを選択します。代わりに、インテリジェント構造で識別されなかったデータをマッピングするには、**未定義**出力グループを選択します。
3. **[OK]** をクリックします。

構造パーサートランスフォーメーションのルールおよびガイドライン

構造パーサートランスフォーメーションを使用する場合は、次のルールおよびガイドラインを考慮してください。

- Mapping Designer で構造パーサートランスフォーメーションを含むマッピングを開き、トランスフォーメーションに関連付けられているインテリジェント構造モデルを変更した場合、Mapping Designer にメッセージが表示されます。メッセージ内に提示されるリンクをクリックして、マッピングに適合するようにモデルを更新します。
- すべての出力フィールドに文字列データ型が割り当てられていた古いバージョンのインテリジェント構造検出で作成されたインテリジェント構造モデルを更新すると、更新中に構造パーサートランスフォーメーションによってフィールドデータ型が変更される可能性があります。影響を受けるフィールドのいずれかにおいて、ダウンストリームトランスフォーメーションまたはターゲットに文字列が必要である場合は、データ型を文字列に戻すようにモデルを編集します。次に、Mapping Designer で提示されたリンクをクリックして、トランスフォーメーション内のモデルを再度更新します。
  インテリジェント構造モデルを編集する方法については、「*コンポーネント*」を参照してください。
- 次の条件がすべて当てはまる場合、構造パーサートランスフォーメーションは XSD 入力ファイル内の再帰的要素を解析できます。

  - 再帰的要素が繰り返し要素内にネストされていない。

  - 再帰的要素に常に同じ名前の要素が使用されている。

  - 構造パーサートランスフォーメーションが詳細モードのマッピングで使用されていない。

  トランスフォーメーションで再帰的要素を解析できない場合は、単一の出力グループにデータが出力されます。このトランスフォーメーションでは、2023 年 10 月リリース以降に作成されたマッピング、または 2023 年 10 月リリースにアップグレードした後にスキーマファイルを再アップロードした場合に、マッピング内の再帰的要素を解析できます。
- マップレットで構造パーサートランスフォーメーションを使用する場合は、ランタイムエラーを防ぐために、組み合わせたマップレットトランスフォーメーション名が 80 文字を超えないようにしてください。詳細については、「マップレットトランスフォーメーション名」 (ページ 316)を参照してください。

詳細モードのマッピングに関するルールとガイドライン

詳細モードのマッピングで構造パーサートランスフォーメーションを使用する場合は、次のルールおよびガイドラインを考慮してください。

- 詳細モードでは、構造パーサートランスフォーメーションは JSON オブジェクトを入力として処理できますが、JSON 配列は処理できません。
- 詳細モードのデフォルトでは、ソースデータのスキーマと、トランスフォーメーションに関連付けられたインテリジェント構造モデルのスキーマが正確に一致する必要があり、トランスフォーメーションはパススルーフィールドを処理できません。パススルーフィールドを処理するには、マッピングタスクの Spark セッションプロパティで次の Spark カスタムプロパティを設定します: `Spark.MSPEnableUnassigedData=true`
  このプロパティを設定すると、出力グループには *UnassignedData* という配列が含まれます。この配列には、インテリジェント構造によって識別されなかったデータが含まれています。

出力タイプのルールおよびガイドライン

構造パーサートランスフォーメーションの出力タイプを選択する場合は、次のルールおよびガイドラインを考慮してください。

- Parquet 出力タイプを使用するには、まず *HADOOP_HOME* および *hadoop.home.dir* 環境変数を設定します。

- AVRO 出力タイプを使用する場合、トランスフォーメーションでは同じ名前の入力フィールドの出力は生成されません。
- トランスフォーメーションは ORC 出力をローカルの日時に基づいて作成します。同じ入力を異なる場所または環境で処理すると、時間および時間形式が異なる出力が生成される可能性があります。
- 大規模な XML ファイルや JSON ファイルを作成するには、バイナリ出力タイプを使用することをお勧めします。

構造パーサートランスフォーメーションの例

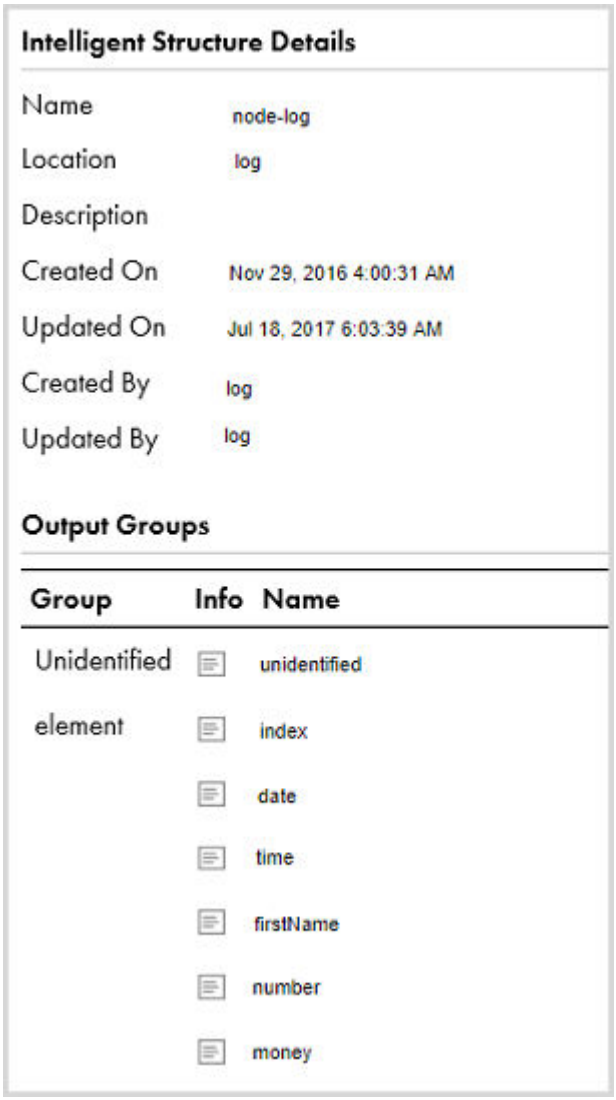
ログファイル内の構造化されていないデータを解析して、リレーショナル形式でターゲットファイルにデータを書き込む必要があるケースについて説明します。

構造化されていないデータを分析し、その構造を検出するインテリジェント構造モデルを設定する必要があります。

次の図に、解析するログファイルを示します。

```
[2015-06-19 08:32:55] [info]  [ 9936] Commons Daemon procrun (1.0.13.0 64-bit) started
[2015-06-19 08:32:55] [info]  [ 9936] Running 'VDS_NODE_65_inw00004001' Service...
[2015-06-19 08:32:55] [info]  [ 4156] Starting service...
[2015-06-19 08:32:56] [info]  [ 4156] Service started in 1159 ms.
[2015-06-19 16:03:42] [info]  [ 9704] Console SHUTDOWN event signaled
[2015-06-19 16:03:42] [info]  [ 9704] Stopping service...
[2015-06-19 16:03:48] [info]  [ 9704] Service stopped.
[2015-06-19 16:03:48] [info]  [ 9936] Run service finished.
[2015-06-19 16:03:48] [info]  [ 9936] Commons Daemon procrun finished
[2015-06-19 16:03:48] [error] [ 4156] Failed to set service status
```

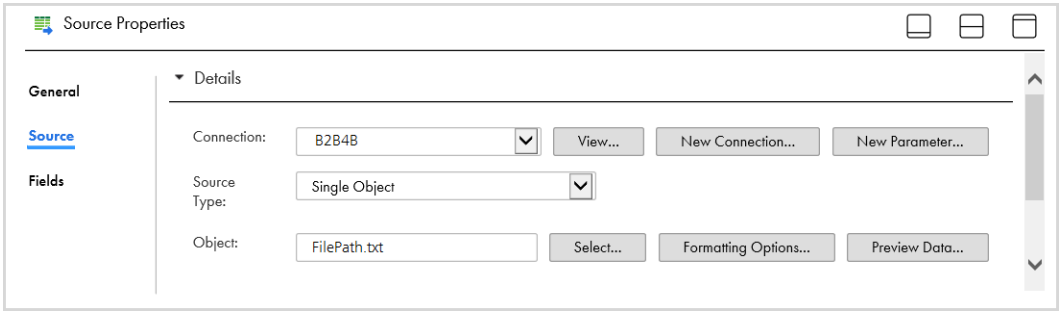
Informatica Cloud でインテリジェント構造モデルを作成します。次の図は、インテリジェント構造モデルの詳細を示しています。



ログファイルを解析するには、マッピングの構造パーサートランスフォーメーションを使用してログファイルのデータを変換します。

Mapping Designer で、解析するログファイルへのパスが含まれているフラットファイルをソースオブジェクトとして追加します。

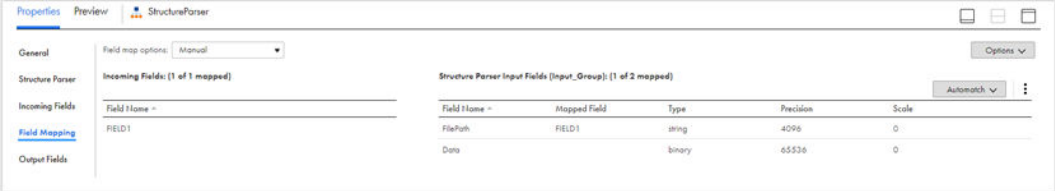
次の図に、選択したソースファイルのプロパティを示します。



構造パーサートランスフォーメーションを追加します。すでに設定したインテリジェント構造モデルを使用するように設定します。リレーショナル出力タイプを選択します。

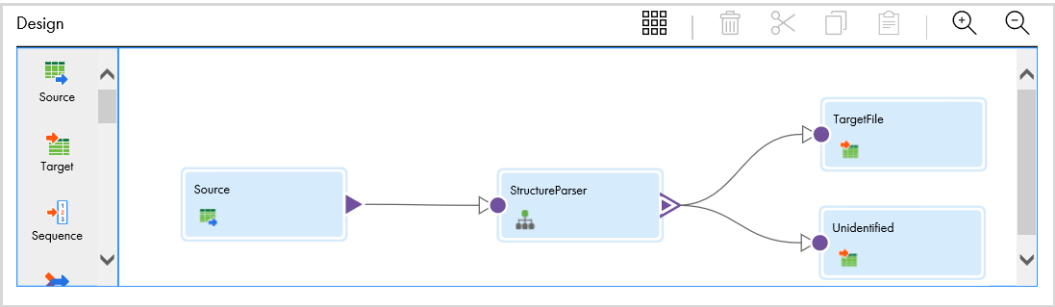
ソースオブジェクトを構造パーサートランスフォーメーションに接続します。受信データをトランスフォーメーションの各フィールドにマッピングするには、構造パーサートランスフォーメーションを選択します。**[フィールドマッピング]** タブで、**[パス]** を構造パーサー入力フィールドの **[ファイルパス]** にマッピングします。

次の図に、フィールドマッピングを示します。



TargetFile という名前が付けられた、処理対象の解析された出力グループのテキストファイルターゲットトランスフォーメーションを追加します。識別されなかったデータ用に、もう 1 つのテキストファイルターゲットトランスフォーメーション Unidentified を追加します。

次の図に、マッピングを示します。



マッピングを実行して、階層形式のデータを TargetFile トランスフォーメーションに書き込みます。マッピングにより、インテリジェント構造で識別されなかったデータが Unidentified トランスフォーメーションに送信されます。

次の画像は、TargetFile トランスフォーメーションからの解析済みデータの出力ファイルを示しています。

```
"date","time","token","number","value"
"2015-06-19","08:32:55","info","9936","Commons Daemon procrun (1.0.13.0 64-bit)
started"
"2015-06-19","08:32:55","info","9936","Running 'VDS_NODE_65_inw00004001' Service..."
"2015-06-19","08:32:55","info","4156","Starting service..."
"2015-06-19","08:32:56","info","4156","Service started in 1159 ms."
"2015-06-19","16:03:42","info","9704","Console SHUTDOWN event signaled"
"2015-06-19","16:03:42","info","9704","Stopping service..."
"2015-06-19","16:03:48","info","9704","Service stopped."
"2015-06-19","16:03:48","info","9936","Run service finished."
"2015-06-19","16:03:48","info","9936","Commons Daemon procrun finished"
"2015-06-19","16:03:48","error","4156","Failed to set service status"
```

データをさらに解析する必要がある場合は、前のパーサーで出力を解析する構造パーサートランスフォーメーションの中間ストリームを追加で含めることができます。

第 33 章

トランザクション制御トランスフォーメーション

トランザクション制御トランスフォーメーションは、マッピングの実行時に行のセットをコミットまたはロールバックするアクティブなトランスフォーメーションです。トランザクション制御トランスフォーメーションを使用して、リレーショナル、XML、Amazon Redshift、REST V2 などのトランザクションターゲットに対してトランザクションをコミットまたはロールバックします。また、マッピングでこのトランスフォーメーションを使用して、データ統合が新しいトランザクションを開始するたびにデータを異なるフラットファイルに書き込むこともできます。

大量のデータを処理する場合は、トランザクション制御トランスフォーメーションを使用することをお勧めします。トランザクション制御トランスフォーメーションを使用すると、一定の間隔でデータをコミットできるため、データ損失を防ぐことができます。例えば、注文タイプでソートされているテーブル内の何千ものレコードを処理するマッピングを実行するとします。マッピングで異なる注文タイプが処理されるたびにデータをコミットすることができます。

トランザクション制御トランスフォーメーションにおけるトランザクションは、コミット行またはロールバック行によって関連付けられた行または行のセットです。トランザクションは、従業員 ID や注文入力日などの共通キーでソートされている行のグループをベースにすることができます。各トランザクションの行数は異なる場合があります。

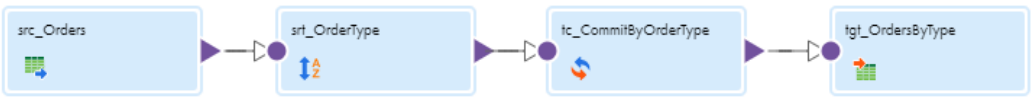
トランザクションを定義するには、トランスフォーメーションでトランザクション制御条件を指定します。条件が満たされているかどうかに基づいて、行をコミットするか、行をロールバックするか、またはトランザクション境界を変更せずにデータの処理を続行するように選択できます。

マッピングタスクを実行すると、データ統合により、トランスフォーメーションに入力される行ごとにトランザクション制御条件が評価されます。コミット行が評価されると、トランザクション内のすべての行がターゲットにコミットされます。データ統合は、ロールバック行を評価すると、トランザクションのすべての行をターゲットからロールバックします。

実行時に作成されるフラットファイルターゲットがマッピングに含まれる場合は、データ統合が新しいトランザクションを開始するたびに出力ファイルを生成できます。ターゲットフラットファイル名は動的に指定できます。

例

トランザクション制御を使って、注文タイプに基づいて注文情報を書き出すとします。特定のタイプのすべての注文が別のターゲットファイルに書き込まれるようにします。このためには、次のマッピングを作成します。



マッピングには次のトランスフォーメーションが含まれます。

ソーストランスフォーメーション

ソース ORDERS テーブルへの接続を設定します。

ソータートランスフォーメーション

ソート条件を設定して、ソースデータを ORDER_TYPE でソートします。

トランザクション制御トランスフォーメーション

統合サービスが新しい注文入力日を検出したときにデータをコミットするように、以下のトランザクション制御条件を作成します。

プロパティ	値
トランザクション制御条件	フィールド値が変わった場合
フィールド	ORDER_TYPE（文字列）
実行するアクション	次より前にコミット
Else	トランザクションを続行

ターゲットトランスフォーメーション

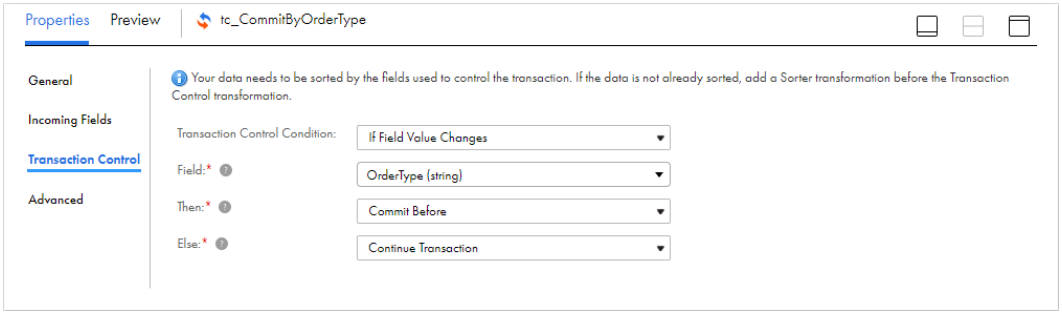
実行時に新しいファイルターゲットを作成し、動的ファイル名を指定します。ターゲット名に次の式を使用して、注文タイプごとに異なるターゲットファイルを作成します。

```
'Orders_ '||ORDER_TYPE||'.csv'
```

トランザクション制御条件

トランザクション制御条件は［**トランザクション制御**］タブで設定します。トランザクション制御条件は各行をテストして、行をコミットするタイミング、行をロールバックするタイミング、またはトランザクションの変更なしで続行するタイミングを特定します。

次の画像は、［**トランザクション制御**］タブを示しています。



注: トランザクション制御条件を指定する前に、入力データがソートされていることを確認します。入力データは、トランザクション制御条件で使用するフィールドでソートする必要があります。

次のいずれかのタイプの条件を使用して行データをテストできます。

フィールド値が変わった場合

フィールド値が変更されたかどうかをテストする場合は、［フィールド値が変わった場合］条件を使用します。例えば、ORDER_DATE が変更された場合、現在の行をターゲットに書き込む前にトランザクションをコミットします。そうでない場合、データの処理を続行します。

テストするフィールドを［**フィールド**］リストから選択します。

詳細

式を使用して行データをテストする場合は、詳細条件を使用します。例えば、NEW_FILE_FLAG='Y' AND DEPT_ID>1000 の場合、トランザクションをコミットしますが、現在の行は次のトランザクションに含めます。そうでない場合、データの処理を続行します。

式は条件の［**If**］パートで設定します。式では、フィールド、パラメータ、組み込み関数、およびユーザー定義関数を使用できます。

パラメータを使用します

式パラメータを使用して条件を表現できます。マッピングタスクの実行時またはパラメータファイルへの値の入力時にパラメータ値を入力します。

テスト結果に基づく条件の［**実行するアクション**］パートと［**Else**］パートでは、次のアクションを指定できます。

アクション	説明
トランザクションを続行	データ統合は、この行にはトランザクション変更を行いません。
次より前にコミット	データ統合はトランザクションをコミットし、新しいトランザクションを開始し、現在の行をターゲットに書き出します。現在の行は、新しいトランザクション内にあります。
次より後にコミット	データ統合は、現在の行をターゲットに書き出し、トランザクションをコミットし、新しいトランザクションを開始します。現在の行は、コミットされたトランザクション内にあります。
次より前にロールバック	データ統合は、現在のトランザクションをロールバックし、新しいトランザクションを開始し、現在の行をターゲットに書き出します。現在の行は、新しいトランザクション内にあります。
次より後にロールバック	データ統合は、現在の行をターゲットに書き出し、トランザクションをロールバックし、新しいトランザクションを開始します。現在の行は、ロールバックされたトランザクション内にあります。

条件の［**実行するアクション**］パートと［**Else**］パートには、異なるアクションを含める必要があります。

マッピングでのトランザクション制御トランスフォーメーションの使用

トランザクション制御トランスフォーメーションはトランザクションジェネレータです。トランザクション制御トランスフォーメーションは、マッピングでトランザクション境界を定義および再定義します。また、先行

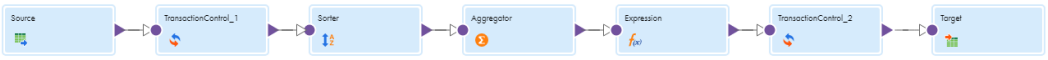
するアクティブソースまたはトランザクションジェネレータから渡されるトランザクション境界を削除したり、新しいトランザクション境界を後続に渡したりします。

Transaction Control トランスフォーメーションは、マッピング内の後続のトランスフォーメーションおよびターゲットに対して有効な場合と無効な場合があります。受け取ったトランザクション境界を削除するトランスフォーメーションを Transaction Control トランスフォーメーションの後に配置した場合、Transaction Control トランスフォーメーションは後続のトランスフォーメーションまたはターゲットに対して無効になります。トランザクション境界を削除するものには、以下のアクティブソースおよびトランスフォーメーションが含まれます。

- トランスフォーメーション範囲がすべての入力レベルの Aggregator トランスフォーメーション
- トランスフォーメーション範囲がすべての入力レベルの Java トランスフォーメーション
- トランスフォーメーション範囲がすべての入力レベルの Joiner トランスフォーメーション
- トランスフォーメーション範囲がすべての入力レベルの Rank トランスフォーメーション
- トランスフォーメーション範囲がすべての入力レベルの Sorter トランスフォーメーション
- 保存済みクエリまたはユーザー入力クエリを使用する、トランスフォーメーション範囲がすべての入力レベルの SQL トランスフォーメーション
- トランザクション制御トランスフォーメーション
- 複数のアップストリームトランザクション制御点に接続されている、複数の入力グループを持つ任意のトランスフォーメーション

トランザクション制御トランスフォーメーションは、ターゲットに対して無効であっても、後続のトランスフォーメーションに対しては有効である場合があります。トランスフォーメーション範囲が「トランザクション」レベルの後続のトランスフォーメーションでは、先行するトランザクション制御トランスフォーメーションによって定義されたトランザクション境界を使用することができます。

以下の図に示した有効なマッピングでは、トランザクション制御トランスフォーメーションはソータートランスフォーメーションに対しては有効ですが、ターゲットに対しては無効です。



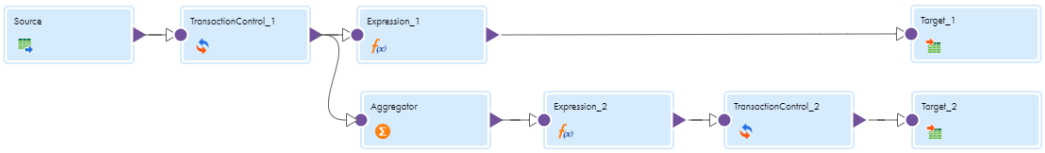
この例では、TransactionControl_1 はターゲットに対しては無効ですが、ソータートランスフォーメーションに対しては有効です。ソータートランスフォーメーションのトランスフォーメーション範囲は「トランザクション」です。TransactionControl_1 で定義されたトランザクション境界が使用されます。アグリゲータトランスフォーメーションのトランスフォーメーション範囲は「すべての入力」です。TransactionControl_1 で定義されたトランザクション境界が削除されます。トランザクション制御トランスフォーメーション TransactionControl_2 は、ターゲットに対する有効なトランザクション制御トランスフォーメーションです。

複数のターゲットを持つトランザクション制御マッピングの例

複数のターゲットを持つマッピングでは、トランザクション制御トランスフォーメーションは、ターゲットによって有効な場合と無効な場合があります。各ターゲットが有効な Transaction Control トランスフォーメーションに接続されている場合、そのマッピングは有効です。マッピング内のターゲットのうち 1 つでも有効な

Transaction Control トランスフォーメーションに接続されていないものがある場合、そのマッピングは無効です。

以下の図に、有効と無効の両方のトランザクション制御トランスフォーメーションを含む、有効なマッピングを示します。

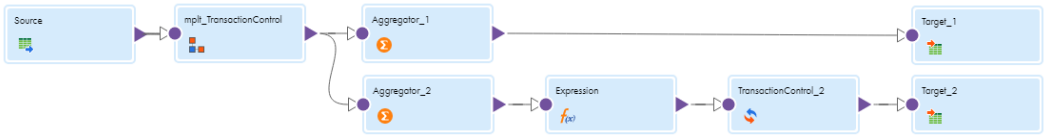


データ統合は TransactionControl_1 を処理し、トランザクション制御式を評価して、トランザクション境界を作成します。マッピングには、TransactionControl_1 と Target_1 の間にトランザクション境界を削除するトランスフォーメーションは含まれず、TransactionControl_1 は Target_1 に対して有効になります。データ統合は、Target_1 に対して TransactionControl_1 が定義したトランザクション境界を使用します。

しかし、マッピングには TransactionControl_1 と Target_2 の間にトランザクション境界を削除するトランスフォーメーションが含まれているため、TransactionControl_1 は Target_2 に対しては無効になります。データ統合でトランスフォーメーション範囲を［すべての入力］に設定してアグリゲータトランスフォーメーションを処理すると、TransactionControl_1 で定義されたトランザクション境界が削除され、オープントランザクション内のすべての行が出力されます。次にデータ統合は、TransactionControl_2 を評価し、トランザクション境界を作成して、そのトランザクション境界を Target_2 に対して使用します。

TransactionControl_1 でロールバックが発生した場合、データ統合は Target_1 からの行だけをロールバックします。Target_2 からの行はロールバックしません。

以下の図に、有効と無効の両方のトランザクション制御トランスフォーメーションを含む、無効なマッピングを示します。



マップレット mplt_TransactionControl には、トランザクション制御トランスフォーメーションが含まれています。これは Target_1 および Target_2 に対して無効です。Aggregator_1 のトランスフォーメーション範囲は「すべての入力」です。これは Target_1 に対するアクティブソースです。Aggregator_2 のトランスフォーメーション範囲は「すべての入力」です。これは Target_2 に対するアクティブソースです。TransactionControl_2 は Target_2 に対して有効です。

Target_1 が有効なトランザクション制御トランスフォーメーションに接続していないため、このマッピングは無効です。

マッピングでのトランザクション制御トランスフォーメーションの使用に関するガイドライン

マッピングでトランザクション制御トランスフォーメーションを使用する場合は、以下のガイドラインを参照してください。

- 入力データは、トランザクション条件で使用するフィールドでソートする必要があります。トランザクション制御トランスフォーメーションのソータートランスフォーメーションアップストリームを配置するか、ソート済みデータソースを使用します。

- 頻繁にコミットを実行するようにトランザクション制御条件を設定すると、パフォーマンスに影響が及ぶ可能性があります。
- マッピングに XML ターゲットが含まれ、コミット時に追加または新しいドキュメントを作成するように選択した場合、入力グループは同じトランザクション制御点からデータを受け取る必要があります。
- バッチまたはトランザクションリアルタイム処理をサポートしないターゲットに接続されているトランザクション制御トランスフォーメーションは、このようなターゲットに対しては無効です。
- 各ターゲットインスタンスは必ずトランザクション制御トランスフォーメーションに接続する必要があります。
- 複数のターゲットを同じトランザクション制御トランスフォーメーションに接続することができます。
- 1 つのターゲットに接続できる有効なトランザクション制御トランスフォーメーションは 1 つだけです。
- シーケンスジェネレータトランスフォーメーションで始まるパイプラインブランチには、トランザクション制御トランスフォーメーションを配置できません。
- 動的ルックアップトランスフォーメーションとトランザクション制御トランスフォーメーションを同じマッピングで使う場合、ロールバックされたトランザクションはターゲットデータと同期しない可能性があります。
- マッピング内のすべてのターゲットを有効なトランザクション制御トランスフォーメーションに接続する必要があります。あるいは、どのターゲットも接続しない必要があります。

詳細プロパティ

トランザクション制御トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。

第 34 章

共有体トランスフォーメーション

共有体トランスフォーメーションは、複数のパイプラインから 1 つのパイプラインにデータを統合するために使用するアクティブなトランスフォーメーションです。

データ統合パターンでは、通常、2 つ以上のデータソースをすべての行の共有体を含む 1 つのストリームに組み合わせます。多くの場合、各データソースは同じ構造ではないため、データストリームを自由に結合することはできません。共有体トランスフォーメーションでは、類似するストリームのメタデータを作成し、データソースを 1 つのターゲットに組み合わせることができます。

共有体トランスフォーメーションは UNION ALL SQL 文と同様に、複数のソースのデータを統合します。例えば、共有体トランスフォーメーションを使用して、ADP の従業員情報を Workday 従業員オブジェクトのデータと統合できます。

共有体トランスフォーメーションでデータソースを統合する場合、特定のフィールドを追加、変更、または削除できます。

実行時に、マッピングタスクは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プッシュします。マッピングの実行時に、フィールドマッピングに基づいてデータを 1 つの出力グループに統合します。

ジョイナトランスフォーメーションとの比較

共有体トランスフォーメーションでは、複数のソースのデータをマージできますが、ジョイナトランスフォーメーションのように結合条件に基づいたデータの結合や重複行の削除は行われません。

次の表に、複数ソースのデータもマージする共有体トランスフォーメーションとジョイナトランスフォーメーションの重要な相違点を示します。マッピング設計では次の相違点を考慮してください。

要件	共有体トランスフォーメーション	ジョイナトランスフォーメーション
重複行を削除する	いいえ。共有体トランスフォーメーションからルーター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ダウンストリームを使用して、重複を削除できます。	はい
結合条件に基づいてレコードを結合する	いいえ。共有体トランスフォーメーションは、複数のソースから垂直にデータを結合する SQL の UNION ALL 文と同じです。	はい。ジョイナトランスフォーメーションでは、ノーマル結合、右外部結合、左外部結合、完全外部結合がサポートされています。

要件	共有体トランスフォーメーション	ジョイナトランスフォーメーション
複数の入力グループを含める	はい。複数の入力グループと 1 つの出力グループを定義できます。	はい。2 つの入力グループ（マスタと詳細）を定義できます。
異種ソースを含める	○	はい
異なるデータ型をマージする	すべてのソースカラムが類似するデータ型である必要があります。各ソースのカラム数は同じである必要があります。	結合するソースの少なくとも 1 つのカラムのデータ型が同じである必要があります。
トランザクションを生成する	いいえ	○

共有体トランスフォーメーションの使用の計画

マッピングで共有体トランスフォーメーションを使用する場合は、以下のガイドラインを考慮してください。

- マッピングに共有体トランスフォーメーションを追加する前に、すべてのソーストランスフォーメーションを追加し、使用する他のアップストリームトランスフォーメーションを含めます。
- シーケンスジェネレータとソーストランスフォーメーションの両方を共有体トランスフォーメーションの 1 つの入力グループに接続する場合、共有体トランスフォーメーションからのシーケンスジェネレータトランスフォーメーションアップストリームを使用できます。

入力グループ

デフォルトでは、共有体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。3 つ以上のソースからデータをマージする場合は、追加のソースごとに入力グループを追加します。各グループの各アップストリームトランスフォーメーションに異なるフィールドルールを設定できます。

入力グループには次の特性があります。

- 共有体トランスフォーメーションは、入力グループに接続された最初のソースのフィールドに基づいて出力フィールドを初期化します。
- 各入力グループは、異なるフィールドマッピングモードまたはパラメータを使用できます。
- フィールドマッピングをパラメータ化したり、各入力グループに対してフィールドマッピングを定義したりできます。

入力グループを追加するには、Mapping Designer で、アップストリームトランスフォーメーションを共有体トランスフォーメーションの「新規グループ」グループに接続します。また、共有体トランスフォーメーションの **[受信フィールド]** タブで入力グループを追加することもできます。

入力グループは名前を変更できます。また、残りの入力グループが 2 つ以上ある限りは、入力グループを削除することもできます。入力グループの名前変更と削除は **[受信フィールド]** タブで行います。

出力フィールド

アップストリームトランスフォーメーションを共有体トランスフォーメーションに接続する場合、出力フィールドを初期化します。最初の出力フィールドは、グループ Input1 の入力フィールドの完全なコピーです。

出力フィールドを定義する場合、次の点に注意してください。

- 出力フィールドを初期化した後に、入力グループの接続や切断によって出力フィールドを変更することはできません。
- いずれかの共有体トランスフォーメーション入力グループを接続する前に出力フィールドを追加する場合は、手動で追加できます。
- 出力フィールドを追加する際には、フィールド名、データ型、精度、スケール、説明（オプション）を定義します。説明には最大 4000 文字を含めることができます。
- 共有体トランスフォーメーションを、どのフィールドも渡さないアップストリームトランスフォーメーションに接続する場合、出力フィールドは初期化されません。
- 実行時に、マッピングはフィールドマッピングに含まれない出力フィールドに NULL 値を渡します。

フィールドマッピング

共有体トランスフォーメーションは、複数のソースパイプラインからデータを統合できます。ソースには、同じ一連のフィールドやいくつかの一致するフィールドを含めたり、パラメータ化されたフィールドマッピングを使用したりできます。

共有体トランスフォーメーションでフィールドマッピングを操作する場合、次の点に注意してください。

- フィールドの名前、タイプ、精度、およびスケールが同じ入力グループを使用する必要があります。
- 一部の出力フィールドは編集、削除、または手動で追加できます。
- フィールドマッピングの一部として、入力グループを選択し、その入力グループのパラメータを指定します。
- すべての入力グループでフィールドのパラメータを使用できます。
- フィールドマッピングをパラメータ化するか、各入力グループに対してフィールド名でマッピングできます。実行時に、タスクは入力グループのフィールドの完全なコピーを出力フィールドとして追加します。

データ統合で同じ名前のフィールドを自動的にリンクして、フィールドのマッピングを手動で行う場合は、**[手動]** オプションを選択してから **[自動マップ]** をクリックします。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合は同じ名前のフィールドを照合します。
- スマートマップ。データ統合は類似する名前のフィールドを照合します。例えば、受信フィールドが Cust_Name でターゲットフィールドが Customer_Name である場合、データ統合は Cust_Name フィールドと Customer_Name フィールドを自動的にリンクします。

[正確なフィールド名] と [スマートマップ] を同じフィールドマッピングで使用できます。例えば、[正確なフィールド名] を使用して同じ名前のフィールドを照合してから、[スマートマップ] を使用して類似した名前のフィールドをマッピングできます。

[オートマップ] > **[オートマップを元に戻す]** をクリックして、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。単一フィールドのマッピングを解除するには、マップ解除するフィールドを選択して、**[アクション]** > **[マップ解除]** をクリックします。

データ統合は新しくマッピングされたフィールドを強調表示します。例えば、［正確なフィールド名］を使用すると、データ統合はマッピングされたフィールドを強調表示します。［スマートマップ］を使用すると、データ統合はスマートマップを使用してマッピングされたフィールドのみを強調表示します。

詳細プロパティ

共有体トランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルやトランスフォーメーションが省略可能か、必須かなどの設定を制御します。

以下のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーメッセージおよびステータスメッセージの詳細レベル。［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。
オプション	トランスフォーメーションがオプションかどうかを決定します。トランスフォーメーションがオプションで、受信フィールドがない場合、マッピングタスクの実行は可能で、データはデータフローの別のブランチへと進みます。トランスフォーメーションが必須で、受信フィールドがない場合、タスクは失敗します。 例えば、ソース接続のパラメータを設定します。データフローのあるブランチで、日付/時刻データのみがトランスフォーメーションに入力されるようにフィールドルールを使用してトランスフォーメーションを追加し、トランスフォーメーションがオプションになるように指定します。マッピングタスクを設定するときに、日付/時刻データのないソースを選択します。マッピングタスクはオプションのトランスフォーメーションのあるブランチを無視し、データフローはマッピングの別のブランチを経由して続行されます。

共有体トランスフォーメーション例

2 つのフラットファイルソースに従業員に関する人口統計データが含まれており、そのデータをマージするとします。

.txt ファイルで次のデータを受け取ります。

```
employee_ID,first_name,last_name,location,email,phone
1211,John,Davis,Redwood City,jdavis@infa2.com,555-555-4444
0233,Miles,Simone,Barcelona,msimone@infa2.com,555-555-6666
1045,Billie,Coltrane,Philadelphia,bcoltrane@infa2.com,555-555-7777
0987,Django,Holiday,Paris,dholiday@infa3.com,444-444-4444
1199,Nina,Reinhardt,New York,nreinhardt@infa3.com,444-555-5555
```

2 番目のファイルには次のデータが含まれています。

```
ID,first,last,dept,e-mail,phone
0456,Joni,Smith,Marketing,j_smith@infa4.com,333-333-3333
1325,David,Mitchell,R&D,dmitchell@infa4.com,222-222-2222
1101,David,Harry,R&D,dharry@infa5.com,777-777-7777
0623,Debbie,Byrne,HR,dbyrne@infa5.com,888-888-8888
0777,Patti,Bowie,Sales,pbowie@infa5.com,999-999-9999
```

次のカラムを使用して MySQL でこれらのレコードを 1 つのデータセットにマージします。

- id

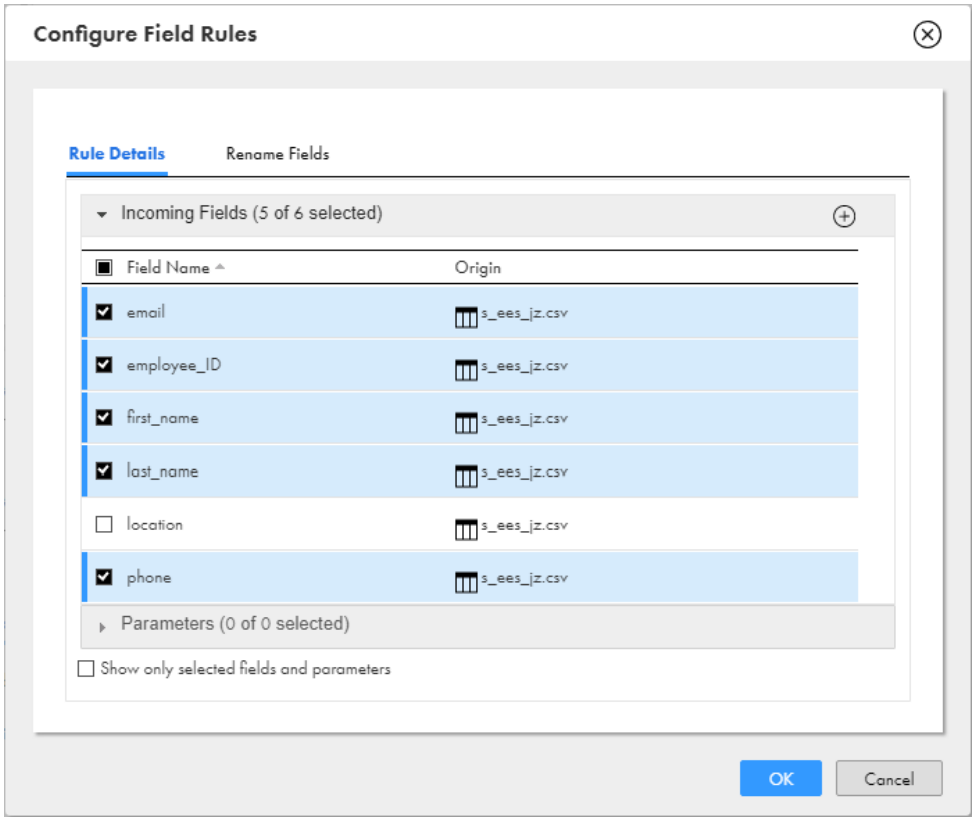
- last
- first
- email
- phone

注: 共有体トランスフォーメーションでマージするデータのデータ型、精度、およびスケールは同じである必要があります。

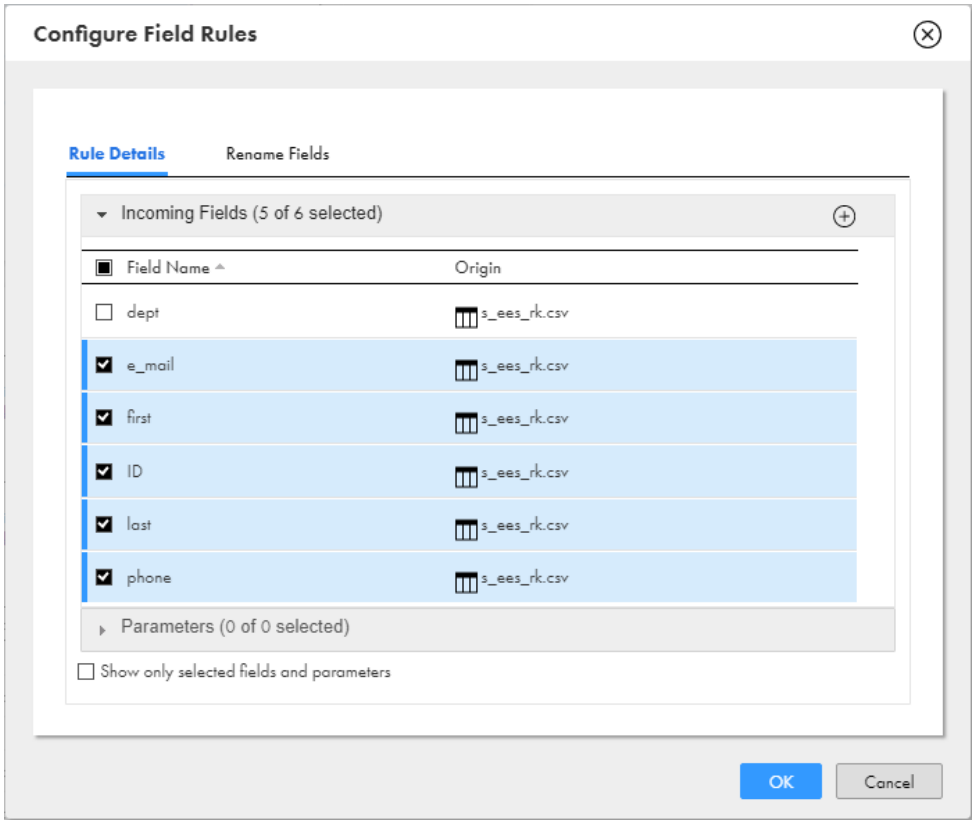
共有体トランスフォーメーションでファイルをマージするには、次の手順を実行します。

1. Secure Agent がアクセスできる場所にソースファイルがあることを確認します。
2. .csv ファイルにアクセスするための接続を定義します。
3. Mapping Designer でマッピングを作成します。
4. 2 つのソーストランスフォーメーションをマッピングに追加し、.csv ファイルのデータに接続します。
5. 共有体トランスフォーメーションを追加し、ソーストランスフォーメーションを接続します。
6. 共有体トランスフォーメーションのプロパティで、入力グループごとに次の手順を実行します。
   a. ［フィールドルール］セクションで、設定するグループをクリックします。
   b. （任意）受信フィールドで、出力でマージするフィールドを選択します。

   次の図は、最初の入力グループで選択したフィールドを示しています。

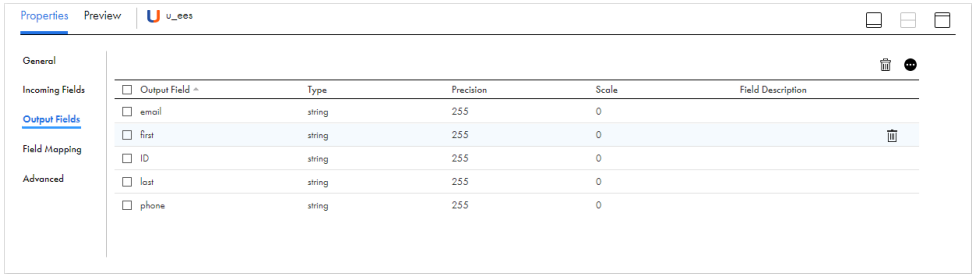


次の図は、2 番目の入力グループで選択したフィールドを示しています。



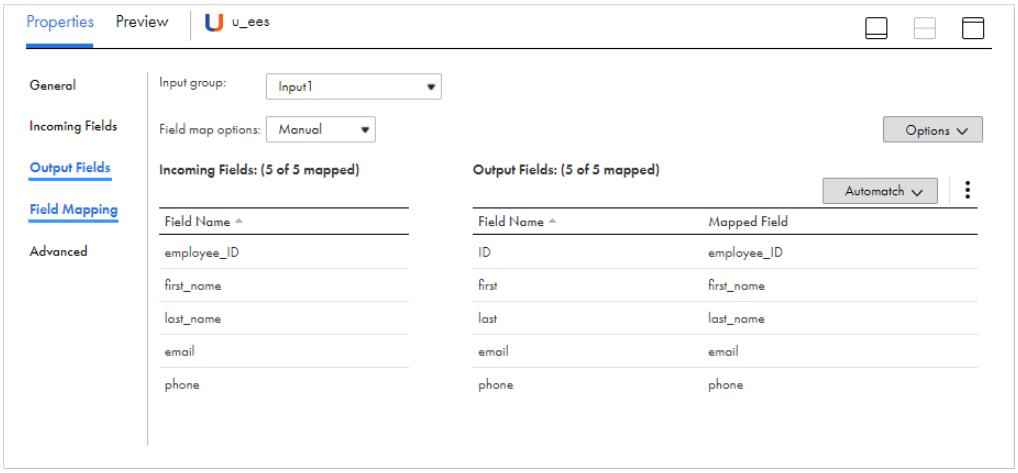
フィールドを除外するルールを指定していない場合、出力フィールドにマッピングしていないフィールドは実行時にタスクで無視されます。

c. ターゲットのフィールド名に合わせて、共有体トランスフォーメーションの出力フィールド名を編集します。

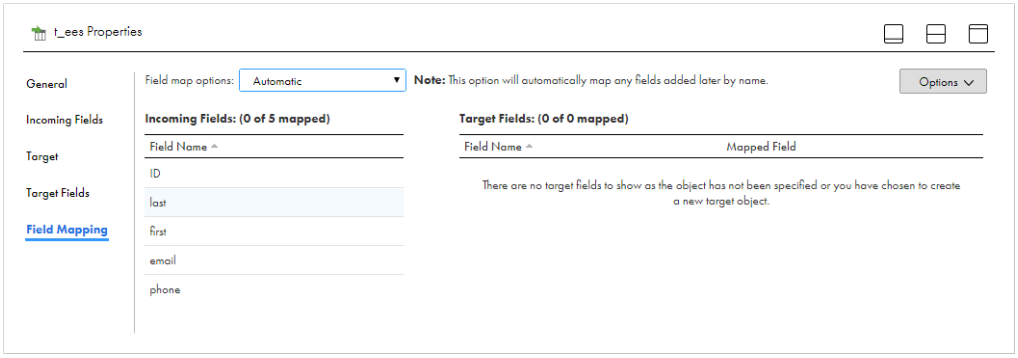


注: フィールドの選択、メタデータの変更、他のフィールドの追加、またはフィールドタイプの変換（integer から number など）を行うこともできます。

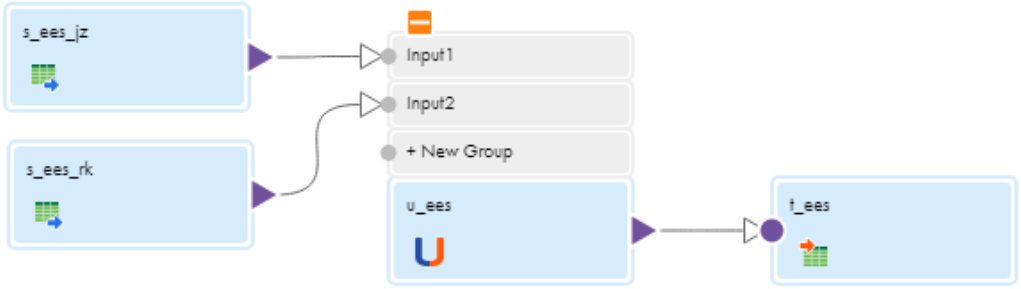
7. 共有体トランスフォーメーションのフィールドマッピングで、各入力グループのフィールドが適切にマッピングされていることを確認します。



8. ターゲットトランスフォーメーションをマッピングに追加します。
9. 共有体トランスフォーメーションをターゲットトランスフォーメーションに接続します。
10. ターゲットトランスフォーメーションのフィールドマッピングで、自動フィールドマッピングを選択します。



完了すると、次の画像のようなマッピングが表示されます。



第 35 章

Velocity トランスフォーメーション

データをフラット化せずに階層入力をある形式から別の形式に変換するには、マッピングで Velocity トランスフォーメーションを使用します。このトランスフォーメーションでは、JSON または XML データを、JSON、XML、またはテキスト出力（プレーンテキスト、電子メール、HTML など）に変換できます。

Velocity トランスフォーメーションを使用して階層データを変換し、ダウンストリーム API 呼び出しに準拠することや、キャンペーン管理システムや機械学習アルゴリズムなどのダウンストリームアプリケーションで使用することができます。また、Velocity トランスフォーメーションを使用して、データベース内の JSON BLOB などの行データを処理することもできます。

データを変換するために、Velocity トランスフォーメーションは指定された Apache Velocity スクリプトを使用します。スクリプトには、Velocity テンプレート言語（VTL）文、データ統合の組み込み関数、およびデータ統合の接続されていないルックアップ式を含めることができます。

次のいずれかの方法で、データを Velocity トランスフォーメーションに渡すことができます。

データをトランスフォーメーションに直接渡す。

入力フィールドを介して、データを Velocity トランスフォーメーションに直接渡すことができます。データは、JSON BLOB または文字列か、XML BLOB または文字列である必要があります。

変換するデータを含むファイルの名前と場所を渡す。

Velocity トランスフォーメーションで、XML ファイルや JSON ファイルなどのフラットファイルに保存されているデータを変換する場合は、入力フィールドを介してファイルの名前と場所をトランスフォーメーションに渡します。データは、ファイルパスとファイル名を含む文字列である必要があります。ファイルは、Secure Agent マシンからアクセスできる区切りフラットファイルである必要があります。

トランスフォーメーションは、変換されたデータを 1 つの文字列フィールドに返します。

Velocity トランスフォーメーションを使用するには、適切なライセンスが必要です。

Velocity トランスフォーメーションの入力形式

[入力形式] タブで受信データの形式を設定します。入力フィールドおよび入力タイプ、変換するデータの形式、テンプレート内のデータを参照する変数名などの、入力形式のプロパティを設定します。変換するデータがバイナリデータの場合やファイル内にある場合は、コードページも選択します。

次の表に、ファイル入力形式のプロパティを示します。

プロパティ	説明
入力フィールド	Velocity トランスフォーメーションの入力を含むアップストリームトランスフォーメーションのフィールド。選択するフィールドには、変換するデータか、データを含むフラットファイルのファイルパスと名前が含まれている必要があります。 トランスフォーメーションへの入力は文字データまたはバイナリデータである必要があるため、このフィールドには、アップストリームトランスフォーメーションの文字列フィールド、テキストフィールド、およびバイナリフィールドのみが表示されます。
入力タイプ	選択した入力フィールドのデータのタイプ。次のいずれかの入力タイプを選択します。 - バッファ。入力フィールドに、JSON BLOB や XML 文字列などの変換するデータが含まれている場合は、このオプションを選択します。 - ファイル。入力フィールドに、変換するデータを含むフラットファイルへのパスが含まれている場合は、このオプションを選択します。
入力形式	処理するデータの形式タイプ（JSON または XML）。

テンプレートの変数名	処理するデータを参照するためにテンプレートで定義する変数。先頭のドル記号やその他の特殊文字は含めません。
	例えば、次のテンプレートでは、処理するデータを参照するために変数名「root」が使用されています。
	`<xml>`
	`#foreach ($child in $root.getRootElement().getChildren() )`
	`  <$child.getChild("name").getText()>`
	`  <id>$child.getChild("id").getText()</id>`
	`  <size>$child.getChild("size").getText()</size>`
	`  </$child.getChild("name").getText()>`
	`#end`
	`</xml>`

次の表に、ファイル入力形式の詳細プロパティを示します。

プロパティ	説明
コードページ	変換するデータのコードページ。このフィールドは、入力タイプがファイルの場合、または入力フィールドにバイナリデータが含まれている場合に有効になります。 変換するデータのコードページが Secure Agent マシンのコードページと異なる場合は、コードページを選択します。それ以外の場合は、**[デフォルト]** を選択します。

ファイルソースのソース設定

ファイルからデータを読み取るように Velocity トランスフォーメーションを設定する場合は、単一の区切りフラットファイルからデータを読み取るようにソーストランスフォーメーションを設定し、形式オプションを設定します。フラットファイルでは、1 行にファイルパスが含まれている必要があります。

例えば、次のテキストは、Products.xml という名前の XML ファイルからデータを読み取るソースファイルの内容を示しています。

```
C:\IICS_XML_SourceFiles\Products.xml
```

ソーストランスフォーメーションの形式オプションを設定するには、次の手順を実行します。

1. **[ソース]** タブで、ファイルパスを含むファイルをソースオブジェクトとして選択し、**[形式オプション]** をクリックします。
2. **[形式オプション]** ダイアログボックスで、フラットファイルタイプが **[区切り]** になっていることを確認し、次のプロパティを設定します。

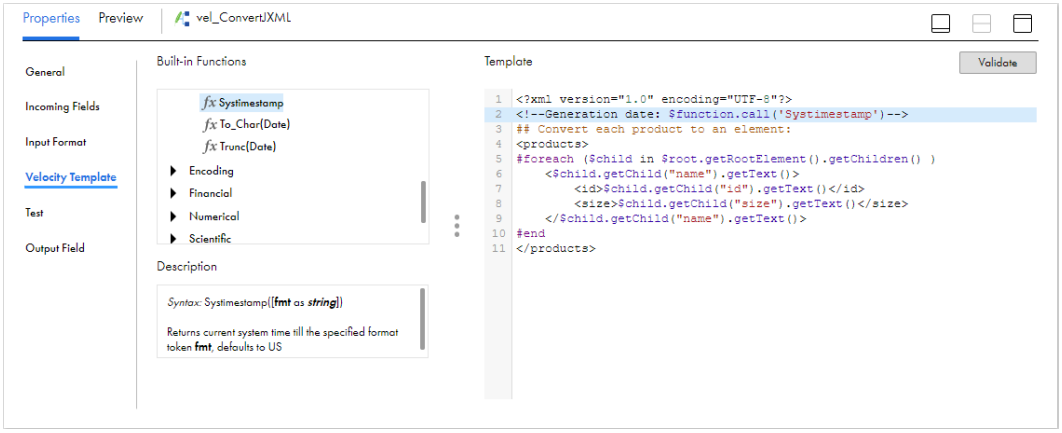
プロパティ	値
引用符	なし
フィールドラベル	自動生成
最初のデータ行	1

3. **[OK]** をクリックします。

Velocity テンプレート

Velocity テンプレートは **[Velocity]** タブで作成します。テンプレートを作成するには、テンプレートエディタにテンプレートを入力するか貼り付けます。その後、**[検証]** をクリックして構文を検証します。

次の図はテンプレートエディタを示しています。



テンプレートには、VTL 文、データ統合の組み込み関数、およびデータ統合の接続されていないルックアップ式を含めることができます。Velocity テンプレート言語の詳細については、Apache Velocity のドキュメントを参照してください。

組み込み関数を呼び出すには、次の構文を使用します。

```
$function.call('<function name>',<argument>,<argument>,...)
```

例えば、次のテンプレートコードは、XML コメントでシステムタイムスタンプを返します。

```
<!--Generation date: $function.call('Systimestamp')-->
```

次の関数呼び出しは、プロパティ$vendor.name を引数として INITCAP 関数に渡して、ベンダ名の各単語の最初の文字を大文字にします。

```
$function.call('InitCap',$vendor.name)
```

ヒント: **[組み込み関数]** リストで関数を選択して、関数呼び出し構文をテンプレートのカーソル位置にコピーすることができます。

接続されていないルックアップを呼び出すには、次の構文を使用します。

```
$function.call(':LKP.<Lookup transformation name>)',<argument>,<argument>,...)
```

例えば、次の式は、プロパティ$item.id および$item.category を引数として lkp_ItemPrices という名前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に渡します。

```
$function.call(':LKP.lkp_ItemPrices',$item.id,$item.category)
```

構文を検証するには、**[検証]** をクリックします。データ統合がテンプレートをチェックし、構文エラーを表示します。テンプレートの構文が無効な場合、マッピングは保存できますが、無効です。

検証では、テンプレート出力のテストやランタイムエラーの検出は行われません。**[テスト]** タブで、テンプレート出力をテストできます。

テンプレートのテスト

[テスト] タブで、テンプレートをテストできます。**[テスト]** タブにはサンプルの入力と出力のフィールドが含まれているため、テンプレートを作成するときにテンプレートをテストできます。

次の図は、**[テスト]** タブを示しています。

作成したテンプレートをテストするには、ランタイム環境を選択し、受信データの一部を **[サンプル入力]** フィールドに入力するか貼り付けてから、**[テスト]** をクリックします。Hosted Agent を除いて、データ統合サーバーが有効になっているランタイム環境を選択できます。

サンプル入力には最大 1000000 文字を含めることができます。最適な結果を得るために、サンプル入力が変換するデータの代表的なサンプルであることを確認してください。

[サンプル出力] フィールドにサンプル出力が表示されます。テンプレートをテストする場合、データ統合は組み込み関数式またはルックアップ式の呼び出しを処理しません。サンプル出力では、これらの式はテンプレートに入力されているとおりに表示されます。テンプレートで有効な出力が生成されない場合は、**[サンプル出力]** フィールドにエラーメッセージが表示されます。

Velocity トランスフォーメーションの出力

Velocity トランスフォーメーションは、変換されたデータを VELOCITYOUT という名前の文字列フィールドに返します。出力フィールドの名前とタイプは変更できませんが、精度は変更できます。**[出力フィールド]** タブで出力フィールドの精度を変更します。

精度は、返される文字列に含めることができる最大文字数です。データ統合は、返された文字列を精度の値で切り詰めます。デフォルトでは、出力フィールドの精度は 1000000 文字です。出力フィールドの精度を変更するには、**[精度]** フィールドに別の値を入力します。

ファイルターゲットのターゲット設定

Velocity トランスフォーメーションによって変換されたデータをターゲットファイルに書き込むことができます。実行時に作成する XML ファイルまたは JSON ファイルがターゲットファイルの場合は、出力ファイルのヘッダーとテキスト修飾子をなしに指定することを推奨します。

出力ファイルのヘッダーをなしに指定するには、ターゲットトランスフォーメーションの **[ターゲット]** タブを開きます。詳細プロパティで、ヘッダーオプションを **[ヘッダーなし]** に設定します。

テキスト修飾子をなしに指定するには、**[ターゲット]** タブで、**[オブジェクト]** フィールドの横にある **[形式オプション]** をクリックします。**[形式オプション]** ダイアログボックスで、テキスト修飾子を **[なし]** に設定します。

Velocity トランスフォーメーションのパーサー

Velocity トランスフォーメーションは、さまざまなパーサーを使用して JSON データと XML データを解析します。

Velocity トランスフォーメーションは、処理するデータのタイプに基づいて、次のパーサーを使用します。

JSON **データ**

JSON データを解析するために、Velocity トランスフォーメーションは Java パッケージ org.json を使用します。トランスフォーメーションは、処理するデータを `JSONObject(java.lang.String source)` コンストラクタに渡して、Java で JSONObject オブジェクトを作成します。JSONObject コンストラクタの詳細については、javadoc.io を参照してください。

XML **データ**

XML データを解析するために、Velocity トランスフォーメーションは Java パッケージ org.jdom2.input を使用します。トランスフォーメーションは、このパッケージで提供されている SAX パーサーを使用して、Java で SAXBuilder オブジェクトを作成します。SAX パーサーの詳細については、JDOM v2.0.6 API の仕様を参照してください。

例

次の例では、Velocity トランスフォーメーションを使用して、XML データの形式を別の形式に変換したり、JSON データを変換して拡張したりする方法を示します。

XML 変換の例

XML ファイル内に製品データがあり、別の形式の XML に変換する必要があります。

変換する XML ファイルの products.xml には、次の形式のデータが含まれています。

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
    <product>
        <id>1</id>
        <name>milk</name>
        <size>16 oz</size>
    </product>
    <product>
        <id>2</id>
        <name>water</name>
        <size> 8 oz</size>
    </product>
</document>
```

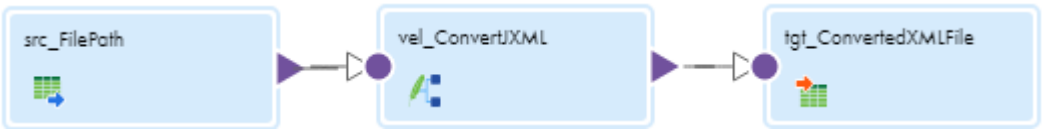
XML を変換して、各製品が独自の要素内に含まれるようにする必要があります（<milk>...</milk>など）。また、XML ファイル内のコメントにタイムスタンプを追加する必要があります。

ファイルを変換するには、まず、変換する XML ファイルのパスを含むソースファイルを作成して、マッピングが XML ファイルからデータを読み取れるようにします。その後、マッピングを作成します。

次の行を含む「filepath.txt」という名前のテキストファイルを作成します。

```
C:\XMLSources\products.xml
```

ソースファイルを作成したら、マッピングを作成します。次の図に、マッピングを示します。



次の方法でトランスフォーメーションを構成します。

ソーストランスフォーメーション

ファイルパスを含むファイルからデータを読み取るようにソーストランスフォーメーションを設定し、ソース形式オプションを設定します。

[ソース] タブで、ソースファイルにアクセスできる接続を選択し、ソースタイプを **[単一オブジェクト]** に設定して、ソースオブジェクトとして filepath.txt を選択します。

[形式オプション] をクリックして、次のプロパティを設定します。

プロパティ	値
フラットファイルタイプ	区切り
引用符	なし

プロパティ	値
フィールドラベル	自動生成
最初のデータ行	1

Velocity **トランスフォーメーション**

Velocity トランスフォーメーションの入力を設定し、テンプレートを作成します。

［入力形式］タブで、次のプロパティを設定します。

プロパティ	値
入力フィールド	ソーストランスフォーメーションからの受信文字列フィールド。
入力タイプ	ファイル
入力形式	XML
テンプレートの変数名	ルート
コードページ	UTF-8

［Velocity テンプレート］タブで、テンプレートエディタに次のテンプレートを入力し、構文を検証します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generation date: $function.call('Systimestamp')-->
## Convert each product to an element:
<products>
#foreach ($child in $root.getRootElement().getChildren() )
    <$child.getChild("name").getText()>
        <id>$child.getChild("id").getText()</id>
        <size>$child.getChild("size").getText()</size>
    </$child.getChild("name").getText()>
#end
</products>
```

ターゲットトランスフォーメーション

実行時に作成されるフラットファイルターゲットにデータを書き込むように、ターゲットトランスフォーメーションを設定します。ターゲットファイルにヘッダーとテキスト修飾子文字が含まれないようにするために、形式オプションを設定します。

［ターゲット］タブで、接続を選択し、ターゲットタイプを **［単一オブジェクト］**に設定します。**［オブジェクト］**フィールドの横にある **［選択］**をクリックします。**［実行時に新規作成］**を選択して、ファイル名に「products_converted.xml」と入力します。

［形式オプション］をクリックして、テキスト修飾子を **［なし］**に設定します。

詳細プロパティで、ヘッダーオプションを **［ヘッダーなし］**に設定します。

マッピングを実行すると、ターゲットファイルの products_converted.xml には次の XML が含まれます。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generation date: 06/17/2020 17:45:41.341526-->
<products>
    <milk>
        <id>1</id>
        <size>16 oz</size>
    </milk>
```

```
    <water>
        <id>2</id>
        <size> 8 oz</size>
    </water>
</products>
```

JSON 変換の例

データベースの JSON BLOB に保存されているオンラインレビューサービスからのベンダデータがあります。レコードをフィルタし、データを別の形式に変換して、JSON ファイルに書き込む必要があります。また、追加の属性でデータを拡張する必要があります。

データベースには、次の形式のデータを含む JSON BLOB が含まれています。

```
{"items":[
{"business_id":"1SWheh84yJXfytovILXOAQ","name":"Rancho Golf Club","address":"1200 E Camino Acequia
Drive","postal_code":"85016","stars":3.0,"review_count":5,"is_open":0,"attributes":
{"GoodForKids":"False"},"categories":"Golf, Active Life","hours":null},
{"business_id":"QXAEGFB4oINsVuTFxEYKFQ","name":"Garden Blessing Chinese Restaurant","address":"25 Eglinton
Avenue W","postal_code":"L5R 3E7","stars":2.5,"review_count":128,"is_open":1,"attributes":
{"RestaurantsReservations":"True","GoodForMeal":"{'dessert': False, 'latenight': False, 'lunch': True,
'dinner': True, 'brunch': False, 'breakfast': False}","BusinessParking":"{'garage': False, 'street': False,
'validated': False, 'lot': True, 'valet':
False}","Caters":"True","NoiseLevel":"u'loud'","RestaurantsTableService":"True","RestaurantsTakeOut":"True","Re
staurantsPriceRange2":"2","OutdoorSeating":"False","BikeParking":"False","Ambience":"{'romantic': False,
'intimate': False, 'classy': False, 'hipster': False, 'divey': False, 'touristy': False, 'trendy': False,
'upscale': False, 'casual':
True}","HasTV":"False","WiFi":"u'no'","GoodForKids":"True","Alcohol":"u'full_bar'","RestaurantsAttire":"u'casua
l'","RestaurantsGoodForGroups":"True","RestaurantsDelivery":"False"},"categories":"Specialty Food,
Restaurants, Dim Sum, Imported Food, Food, Chinese, Ethnic Food, Seafood","hours":
{"Monday":"9:0-0:0","Tuesday":"9:0-0:0","Wednesday":"9:0-0:0","Thursday":"9:0-0:0","Friday":"9:0-1:0","Saturday
":"9:0-1:0","Sunday":"9:0-0:0"}},
{"business_id":"gnKjwL_1w79qoiV3IC_xQQ","name":"Fujiyama Japanese Cuisine","address":"111 Johnston Rd, Ste
15","postal_code":"28210","stars":4.0,"review_count":170,"is_open":1,"attributes":
{"GoodForKids":"True","NoiseLevel":"u'average'","RestaurantsDelivery":"False","GoodForMeal":"{'dessert':
False, 'latenight': False, 'lunch': True, 'dinner': True, 'brunch': False, 'breakfast':
False}","Alcohol":"u'beer_and_wine'","Caters":"False","WiFi":"u'no'","RestaurantsTakeOut":"True","BusinessAccep
tsCreditCards":"True","Ambience":"{'romantic': False, 'intimate': False, 'touristy': False, 'hipster': False,
'divey': False, 'classy': False, 'trendy': False, 'upscale': False, 'casual':
True}","BusinessParking":"{'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet':
False}","RestaurantsTableService":"True","RestaurantsGoodForGroups":"True","OutdoorSeating":"False","HasTV":"Tr
ue","BikeParking":"True","RestaurantsReservations":"True","RestaurantsPriceRange2":"2","RestaurantsAttire":"'ca
sual'"},"categories":"Sushi Bars, Restaurants, Japanese","hours":
{"Monday":"17:30-21:30","Wednesday":"17:30-21:30","Thursday":"17:30-21:30","Friday":"17:30-22:0","Saturday":"17
:30-22:0","Sunday":"17:30-21:0"}}
...
```

レコードをフィルタして、星が 3 つ以上のレストランのみが含まれるようにする必要があります。また、郵便番号に基づいて各レコードに市区町村を追加する必要があります。

郵便番号と対応する市区町村は、次の形式でフラットファイル内にあります。

```
postal_code|city
15090|Wexford, PA
15102|Bethel Park, PA
15206|Pittsburgh, PA
15317|Canonsburg, PA
28012|Belmont, NC
28027|Concord, NC
...
```

ターゲットファイルに次の形式で JSON データを含める必要があります。

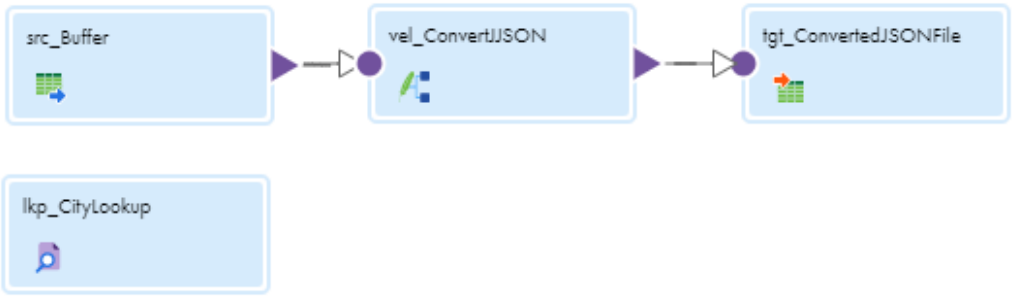
```
{ "Date": "<date>",
"vendors":[

    {
    "name": "<restaurant name>",
    "location": "<city, state/province>",
```

```
    "desc": "<description>",
    "stars": <number of stars>
    }
,
...
```

データを変換するには、JSON データを変換する Velocity トランスフォーメーションと、郵便番号に基づいて市区町村を返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使用してマッピングを作成します。

次の図に、マッピングを示します。



次の方法でトランスフォーメーションを構成します。

ソーストランスフォーメーション

JSON BLOB を含むデータベーステーブルからデータを読み取るように、ソーストランスフォーメーションを設定します。

Velocity **トランスフォーメーション**

Velocity トランスフォーメーションの入力を設定し、テンプレートを作成します。

[入力形式] タブで、次のプロパティを設定します。

プロパティ	値
入力フィールド	JSON BLOB を含むソーストランスフォーメーションからの受信文字列フィールド。
入力タイプ	バッファ
入力形式	JSON
テンプレートの変数名	inputRoot
コードページ	デフォルト

[Velocity テンプレート] タブで、テンプレートエディタに次のテンプレートを入力し、構文を検証します。

```
#set($comma ="")
{ "Date": "$function.call('Systimestamp')",
"vendors":[
#foreach($vend in $inputRoot.items)
    #if($vend.categories.toString().contains("Restaurants") && ($vend.stars > 3))$comma
    {
    "name": "$vend.name",
    "location": "$function.call(':lkp.lkp_CityLookup', $vend.postal_code)",
    "desc": "$vend.categories",
    "stars": $vend.stars
```

```
    }
    #set($comma =",")
    #end
#end
]
}
```

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

郵便番号に基づいて市区町村を返すように、ルックアップトランスフォーメーションを設定します。ルックアップトランスフォーメーションは、Velocity トランスフォーメーションのテンプレートから呼び出すことができるように、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である必要があります。

［全般］タブで、**［接続されていないルックアップ］**を選択します。

［受信フィールド］タブで、in_postal_code という名前の郵便番号用の受信フィールドを作成します。

［ルックアップオブジェクト］タブで、郵便番号と市区町村を含むテキストファイルを選択します。次に、**［形式オプション］**をクリックして、次のプロパティを設定します。

プロパティ	値
フラットファイルタイプ	区切り
区切り文字	その他: |
引用符	なし
フィールドラベル	行 1 からインポート
最初のデータ行	2

［ルックアップ条件］タブで、次のルックアップ条件を設定します。

```
postal_code = in_postal_code
```

［戻りフィールド］タブで、戻りフィールドとして city を選択します。

ターゲットトランスフォーメーション

実行時に作成されるフラットファイルターゲットにデータを書き込むように、ターゲットトランスフォーメーションを設定します。ターゲットファイルにヘッダーとテキスト修飾子文字が含まれないようにするために、形式オプションを設定します。

［ターゲット］タブで、接続を選択し、ターゲットタイプを **［単一オブジェクト］**に設定します。**［オブジェクト］**フィールドの横にある **［選択］**をクリックします。**［実行時に新規作成］**を選択して、ファイル名に「vendors.json」と入力します。

［形式オプション］をクリックして、テキスト修飾子を **［なし］**に設定します。

詳細プロパティで、ヘッダーオプションを **［ヘッダーなし］**に設定します。

マッピングを実行すると、ターゲットファイルの vendors.json には次のデータが含まれます。

```
{ "Date": "07/08/2020 12:33:44.647037",
"vendors":[

    {
    "name": "Fujiyama Japanese Cuisine",
    "location": "Charlotte, NC",
    "desc": "Sushi Bars, Restaurants, Japanese",
    "stars": 4.0
    }
,
    {
```

```
    "name": "D'Amico's Pizzeria",
    "location": "Mentor-on-the-Lake, OH",
    "desc": "Italian, Restaurants, Pizza, Chicken Wings",
    "stars": 4.0
    }
,
    {
    "name": "Villa Borghese",
    "location": "Las Vegas, NV",
    "desc": "Restaurants, Italian",
    "stars": 4.0
    }
,
    {
    "name": "3-Rivers Diner",
    "location": "Pittsburgh, PA",
    "desc": "Sandwiches, Salad, Restaurants, Burgers, Comfort Food",
    "stars": 4.0
    }
,
...
```

第 36 章

ベリファイヤトランスフォーメーション

ベリファイヤトランスフォーメーションによって、Data Quality で作成したベリファイヤアセットがマッピングに追加されます。

ベリファイヤアセットは、入力および出力アドレスデータのテンプレートを定義します。これは、ベリファイヤトランスフォーメーションの入力および出力フィールドに接続されます。ソースデータまたはアップストリームトランスフォーメーションのフィールドを、ベリファイヤトランスフォーメーションの対応する入力ポートに接続します。ベリファイヤトランスフォーメーションの出力ポートを、マッピングのダウンストリームトランスフォーメーションに、またはマッピングターゲットに接続します。

ベリファイヤトランスフォーメーションは、入力アドレスデータに次の操作を実行します。

- トランスフォーメーションが、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と比較します。
- 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。
- 出力アドレスの書き出しは、ベリファイヤアセットで指定された形式で行われます。Data Quality でベリファイヤアセットを定義して、ビジネスニーズに適し、メールキャリアが必要とする構造に準拠したアドレスレコードを作成します。
- また、各アドレスの配達ステータスや、アドレスにエラーやあいまいさが含まれる場合にはその性質についてレポートできます。
- あいまいな住所や不完全な住所がある場合は、提案を提供できます。

ベリファイヤトランスフォーメーションが読み書きできるアドレスステータス情報などのアドレス情報のタイプの詳細については、「*Data Quality*」オンラインヘルプのベリファイヤアセットのドキュメントを参照してください。

ベリファイヤトランスフォーメーションは、別の所で作成したアドレス検証ロジックをマッピングに追加できるという点において、マップレットトランスフォーメーションと類似しています。マップレットと同様に、ベリファイヤは再利用可能なアセットです。ベリファイヤトランスフォーメーションでは、受信フィールドと発信フィールドが示されます。ここでは、ベリファイヤに含まれるアドレスデータは表示されず、ベリファイヤの編集もできません。ベリファイヤを編集するには、ベリファイヤを Data Quality で開きます。

アドレス参照データ

ソースデータに含まれるアドレスの品質を検証するために、ベリファイヤトランスフォーメーションは、アドレスを 1 つ以上の参照データファイルのデータと比較します。

ベリファイヤ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、Secure Agent は入力データを評価し、必要な参照データファイルをダウンロードします。各参照データファイルは、1 つの国に固有です。Secure Agent は、入力アドレスデータで指定された国それぞれについて 1 つ以上のファイルをダウンロードします。

ファイルをダウンロードするためにアクションを実行する必要はありません。最新の参照データファイルがすでにシステム上にある場合、Secure Agent はそれらを再度ダウンロードしません。

参照データファイルは、それぞれにライセンスが必要です。Informatica からライセンスを購入します。ライセンスキー情報は、マッピングを実行する Secure Agent のシステム設定プロパティとして入力します。Administrator サービスの Secure Agent プロパティを見つけます。

参照データのプロパティの詳細については、Data Quality オンラインヘルプでベリファイヤのガイドを参照してください。

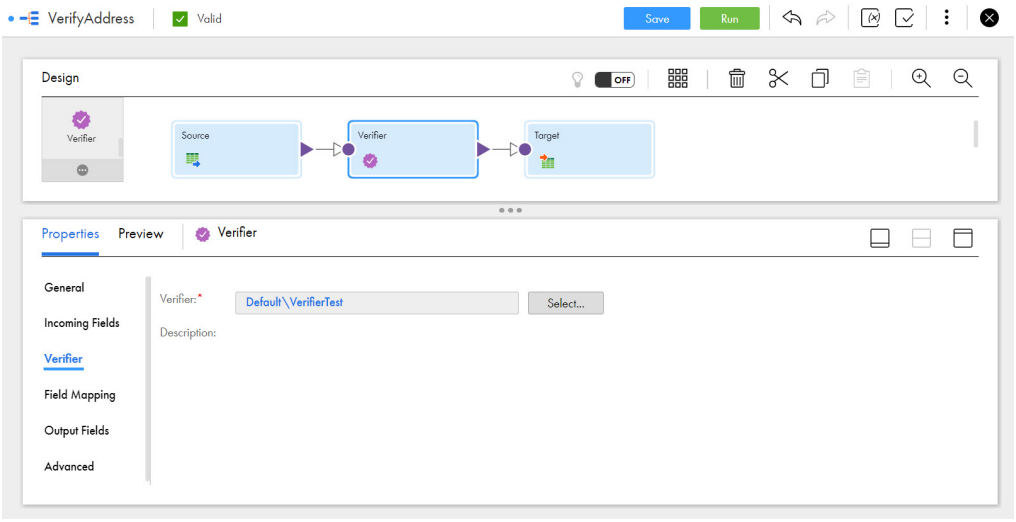
ベリファイヤトランスフォーメーションの設定

マッピングのベリファイヤトランスフォーメーションを設定する場合は、最初に、マッピングに含めるアドレスデータが含まれているベリファイヤアセットを選択します。次に、受信フィールドおよびフィールドマッピングを設定します。その後、出力フィールドを確認します。

トランスフォーメーションを設定するには、以下のタスクを実行します。

1. ベリファイヤトランスフォーメーションをソーストランスフォーメーションまたはその他のアップストリームオブジェクトに接続します。
2. ベリファイヤのタブで、トランスフォーメーションに含めるベリファイヤを選択します。

   次の図は、ベリファイヤを選択するために使用するオプションを示しています。



3. **[受信フィールド]** タブで、受信フィールドを設定します。

   デフォルトでは、トランスフォーメーションは、マッピング内の接続されたアップストリームオブジェクトからすべての受信フィールドを継承します。フィールドルールを定義して、受信フィールドを制限したり、名前を変更したりすることができます。
4. **[フィールドマッピング]** タブで、受信フィールドからのデータをベリファイアアセットのターゲットフィールドに接続するオプションを設定します。

ベリファイヤ入力は、データ内の入力フィールドの名前をすでに反映している可能性があります。この場合、**[自動マップ]** オプションを使用してフィールドの接続を行うことができます。

アップストリームオブジェクトフィールドに接続する方法については、「ベリファイヤトランスフォーメーションのフィールドマッピング」 (ページ 439)を参照してください。

5. **[出力フィールド]** タブのベリファイヤ出力フィールドを確認します。
6. **[全般]** タブでは、オプションとして、ベリファイヤトランスフォーメーションの名前を変更したり、説明を追加したりすることもできます。**[詳細]** タブで、トランスフォーメーションのトレースレベルを更新することもできます。デフォルトのトレースレベルはノーマルです。

注: アセットをトランスフォーメーションに追加した後に Data Quality でアセットを更新する場合は、トランスフォーメーションのアセットバージョンを最新バージョンと同期する必要がある場合があります。データ品質アセットの同期の詳細については、「「データ品質アセットの同期」 (ページ 102)」を参照してください。

ベリファイヤトランスフォーメーションのフィールドマッピング

データがアップストリームトランスフォーメーションかベリファイヤトランスフォーメーションに移行する方法を定義するように、フィールドマッピングを設定します。フィールドマッピングを ［フィールドマッピング］ タブ（**[プロパティ]** パネル）で設定します。

次のフィールドマッピングオプションを設定できます。

フィールドマップオプション

トランスフォーメーションにフィールドをマッピングする方法。

次のいずれかのオプションを選択します。

- 手動。受信フィールドをトランスフォーメーションの入力フィールドに手動でリンクします。自動的にマッピングされたフィールドのリンクを削除します。
- 自動。同名のフィールドを自動的にリンクします。リンクするフィールドがすべて同じ名前である場合に使用します。このオプションが指定されたフィールドを手動でリンクすることはできません。
- 全部パラメータ化。パラメータを使用してフィールドマッピングを表現します。タスクで、すべてのフィールドマッピングを設定できます。
  トランスフォーメーションのベリファイヤがパラメータ化されている場合、またはマッピングのいずれかのアップストリームトランスフォーメーションがパラメータ化されている場合は、［すべてパラメータを使用します］ オプションを選択します。
- 部分的にパラメータを使用します。実施するマッピング内のリンクを設定し、パラメータを使用して他のフィールドをマッピングタスクでマッピングできるようにします。または、パラメータを使用してマッピング内のリンクを設定し、すべてのフィールドとリンクをタスクに表示し設定できるようにします。

パラメータ

フィールドマッピングに使用するパラメータを選択するか、新しいパラメータを作成します。［すべてパラメータを使用します］ または ［部分的にパラメータを使用します］ をフィールドマップオプションに選択すると、このオプションが表示されます。パラメータは *フィールドマッピング*型である必要があります。

1 つのマッピング内の複数のベリファイヤトランスフォーメーションに同じフィールドマッピングパラメータを使用しないでください。

オプション

［受信フィールド］リストと**［ターゲットフィールド］**リストでフィールドが表示される方法を制御します。

以下のオプションを設定します。

- 表示されるフィールド。すべてのフィールド、マッピングされていないフィールド、またはマッピングされたフィールドを表示できます。
- フィールド名。技術フィールド名またはラベルを使用できます。

自動マップ

フィールドを一致する名前とリンクします。一致するフィールドをリンクし、その他のフィールドマッピングについては手動で設定できます。［手動］または［部分的にパラメータを使用します］フィールドマップオプションを選択すると、［自動マップ］オプションが表示されます。

次の方法でフィールドをマッピングできます。

- 正確なフィールド名。データ統合で同じ名前のフィールドを照合します。
- スマートマップ。データ統合で、類似する名前のフィールドを照合します。例えば、受信フィールドが Post Code、ターゲットフィールドが Postal Code の場合、データ統合は、Post Code フィールドを Postal Code フィールドに自動的にリンクします。

［正確なフィールド名］と［スマートマップ］を同じフィールドマッピングで使用できます。例えば、［正確なフィールド名］を使用して同じ名前のフィールドを照合してから、［スマートマップ］を使用して類似する名前のフィールドをマッピングできます。

［自動マップ］ > **［自動マップを取り消す］**をクリックすると、自動マッピングされたすべてのフィールドマッピングを元に戻すことができます。

単一フィールドをマップ解除するには、マップ解除するフィールドを選択して、フィールドのコンテキストメニューで**［アクション］** > **［マップ解除］**をクリックします。選択した 1 つ以上のフィールドのマッピングを解除するには、［ターゲットフィールド］コンテキストメニューで**［選択項目をマップ解除］**をクリックします。

トランスフォーメーションからすべてのフィールドマッピングをクリアするには、［ターゲットフィールド］コンテキストメニューで**［マッピングのクリア］**をクリックします。

入力マッピングおよび出力マッピングの理解

トランスフォーメーションに追加するベリファイヤアセットは、アドレステンプレートと、事前設定された入力および出力フィールドのセットです。アセット上の各入力および出力フィールドは、異なる種類のアドレス情報を表します。

ベリファイヤアセットにマッピングする、トランスフォーメーションの入力フィールド上のデータは、アセットの入力に想定される情報のタイプを反映している必要があります。フィールドが対応しないと、マッピングによる入力フィールドの評価の正確性が低下する可能性があります。

アセット入力には、アドレスデータの 1 つの要素が想定される場合や、1 つのフィールドに編成された複数の要素が想定される場合があります。同様に、アセット出力では、アドレスデータの 1 つの要素が書き込まれる場合や、1 つのフィールドに複数の要素が書き込まれる場合があります。必要な場合は、アセットデザイナと連携して入力の意味を識別します。

ベリファイヤトランスフォーメーションの出力フィールド

ベリファイヤトランスフォーメーションは、ベリファイヤアセットに各出力の出力フィールドを表示します。出力フィールドは、**［プロパティ］**パネルの**［出力フィールド］**タブに表示されます。

タブには、各出力フィールドの名前、タイプ、精度、およびスケールが表示されます。出力フィールドの名前は、アセット上の出力フィールドの名前です。

詳細プロパティ

ベリファイヤトランスフォーメーションに対して詳細プロパティを設定できます。詳細プロパティは、セッションログメッセージのトレースレベルを制御します。

次のプロパティを設定することができます。

プロパティ	説明
トレースレベル	データ統合がセッションログに書き込むエラーおよびステータスメッセージの詳細レベル。［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。

第 37 章

Web サービストランスフォーメーション

Web サービス要求を作成して、Web サービス応答をターゲットにマッピングするには、Mapping Designer で Web サービストランスフォーメーションを使用します。

Web サービスではアプリケーションを統合し、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 サービス記述言語（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、タスク実行の要求があります。

Web サービス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、データの変換、配信を行います。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。マッピングタスクは、ドキュメント/リテラルのエンコードを使用する SOAP メッセージを処理します。Web サービストランスフォーメーションでは、RPC/Encoded または Document/Encoded の WSDL ファイルがサポートされません。

例えば、Web サービストランスフォーメーションで SOAP 要求を Web サービスに送信し、getCityWeatherByZIP という Web サービス操作を実行するとします。Web サービストランスフォーメーションは、要求で郵便番号を渡します。Web サービスは天気予報の情報を取得し、その情報を SOAP 応答で返します。

SOAP 要求メッセージと応答メッセージには、XML スキーマに従ったデータなど、階層データを含めることができます。例えば、Web サービスクライアントが要求を送信し、顧客の注文を販売データベースに追加するとします。Web サービスは、次の階層を応答で返します。

```
Response
    Order
        Order_ID
        Order_Date
        Customer_ID
        Product
            Product_ID
            Qty
            Status
```

応答には、注文の各製品に関する情報など、注文に関する情報が含まれます。この応答は階層型です。注文要素内で、製品要素にさらに要素が含まれているためです。

Web サービストランスフォーメーションを使用するには、適切なライセンスが必要です。

Web サービス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. Web サービスコンシューマ接続を作成し、WSDL URL およびエンドポイント URL を使用します。
2. ビジネスサービスを定義します。ビジネスサービスは、操作が設定されている Web サービスです。
3. Mapping Designer のマッピングで Web サービストランスフォーメーションを設定します。

Web サービスコンシューマ接続の作成

Web サービス記述言語（WSDL）およびエンドポイント URL を使用して、Web サービスに接続します。Web サービスのセキュリティを有効にすることもできます。

1. 管理者で、**[接続]** を選択し、**[新しい接続]** をクリックします。
2. 接続の詳細を入力します。**[タイプ]** には、**[WSConsumer]** を選択します。

   WSConsumer 接続が接続リストに含まれない場合は、管理者で **[アドオンコネクタ]** ページを開いて、組織のコネクタをインストールします。

   注: Web サービスではなく、WSConsumer を選択してください。Web サービス接続は、マップレットまたは PowerCenter タスクに使用します。
3. 接続プロパティを入力します。
   a. Web サービス接続のランタイム環境を選択します。
   b. **[認証]** では、次のいずれかのオプションを選択します。
      - ユーザー名とパスワードを入力して Web サービスに認証するには、**[ユーザー名トークン]** を選択します。セキュリティトークンでパスワードから派生するダイジェストを送信して、パスワードを暗号化することを選択できます。
      - WSDL URL とエンドポイント URL を入力して Web サービスにアクセスするには、**[その他の認証]** を選択します。Web サービスがプライベート Web サービスである場合は、HTTP ユーザー名とパスワードを入力します。
4. 認証接続プロパティを入力します。表示されるプロパティは、選択した認証によって決まります。

   次の表に、設定するプロパティを示します。

プロパティ	説明
WSDL URL	Web サービスによって指定される URL。 HTTP Basic 認証の WSDL URL を使用している場合は、WSDL を Secure Agent が実行されているマシンにダウンロードします。WSDL URL の場合は、ファイルパスを入力します。例えば、次のディレクトリパスを入力できます。`C:\WSDL\Human_Resources.wsdl`
エンドポイント URL	Web サービスのエンドポイント URL。WSDL ファイルは、この URL を位置要素の中で指定します。
ユーザー名	ユーザー名トークン認証に適用。Web サービスに認証するためのユーザー名。
パスワード	ユーザー名トークン認証に適用。Web サービスの認証のためのパスワード。
パスワードの暗号化	ユーザー名トークン認証に適用。PasswordDigest プロパティを有効にして、ナンスとタイムスタンプをパスワードに組み合わせます。マッピングタスクでは、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。 このオプションを選択しない場合は、PasswordText プロパティが有効になり、マッピングタスクによって WS-Security SOAP ヘッダーのパスワードは変更されません。

プロパティ	説明
HTTP ユーザー名	Web サービスへのアクセスに使用するユーザー名。
HTTP パスワード	Web サービスにアクセスするためのパスワード。

5. 接続を保存します。

ビジネスサービスを定義する

ビジネスサービスは、操作が設定されている Web サービスです。Mapping Designer でビジネスサービスを定義し、操作を Web サービストランスフォーメーションに追加します。

1. **［新規］** > **［コンポーネント］** > **［ビジネスサービス］** をクリックし、**［作成］** をクリックします。
2. ビジネスサービスの名前を入力し、保存する場所を選択します。
3. 使用する接続を選択するか、新しい接続を作成します。
4. 必要に応じて、タスクを実行するたびに接続メタデータを更新する場合は動的更新を有効にします。
5. Web サービスから使用する操作を選択します。
6. 必要に応じて、要求と応答の choice 要素と派生型要素を指定するように操作を設定します。

   操作コンポーネントに choice 要素または complexType 要素が含まれていて、抽象属性が true である場合は、操作マッピングを設定するときに 1 つ以上の要素または派生型を選択する必要があります。

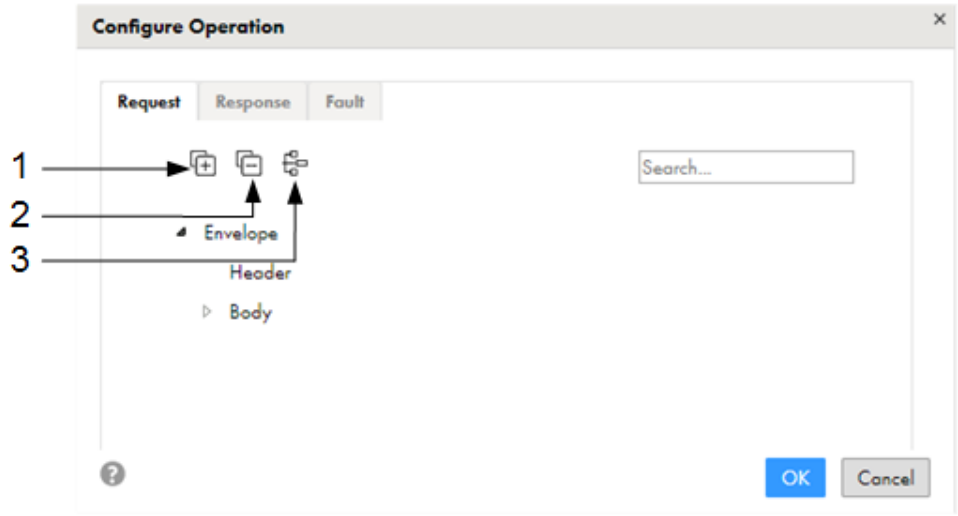
   必要に応じて、抽象属性が false である complexType 要素の場合は、complexType 要素に派生型を選択することもできます。

   a. 設定する操作の **［設定］** をクリックします。

   b. **［操作の設定］** ウィンドウで、**［要求］**、**［応答］**、**［障害］** のいずれかのタブをクリックし、設定する必要があるノードに移動します。

      注: WSDL で anyAttribute 要素を使用している場合、要求または応答に要素は表示されません。

上部のアイコンをクリックして、設定する必要があるノードに移動できます。



1. すべてのノードを展開。
2. すべてのノードを縮小。
3. 設定する必要があるノードのみを表示。このアイコンを選択してノードが展開されない場合、choice 要素または派生型を選択する必要はありません。

c. choice 要素または派生型を選択します。

関連するタブをクリックして、操作の要求と応答を設定してください。

d. 設定した操作を保存します。

7. 必要に応じて、さらに操作を追加し、操作を設定します。
8. ビジネスサービスを保存します。

設定していない必須 choice 要素と派生型がある場合は、ビジネスサービスを保存できません。

Web サービストランスフォーメーションの設定

ビジネスサービスを定義した後、Web サービスのデータへのアクセスやデータの変換を行うように、マッピングで Web サービストランスフォーメーションを設定できます。要件や利用可能なオプションは、Web サービストランスフォーメーションの接続先が 1 つのソースオブジェクトか複数のソースオブジェクトかで変わります。

トランスフォーメーションの設定

Web サービストランスフォーメーションを設定する場合、ソースオブジェクトを接続し、トランスフォーメーションのプロパティを設定して、受信フィールドを Web サービスで要求されるフィールドにマッピングし、応答を出力フィールドにマッピングして 1 つ以上の正常グループを作成します。フォルトグループはデータ統合によって自動的に作成されますが、出力フィールドにマッピングするかどうかを選択することはできます。

1. マッピングを作成し、作業対象のソースオブジェクトを追加します。
2. Web サービストランスフォーメーションをキャンバスに追加します。
3. ソースを Web サービストランスフォーメーションに接続します。
4. **[Web サービス]** タブでビジネスサービスおよびビジネス操作を選択します。

5. **［要求マッピング］**および**［応答マッピング］**タブで、ソースフィールドと Web サービス要求の間のフィールドマッピングを作成します。

   マッピングプロセスの図については、「Web サービストランスフォーメーションの例」 (ページ 451)を参照してください。

6. **［出力フィールド］**タブで、正常グループ、フォルトグループ、およびフィールドの詳細を確認します。必要に応じて、フィールドメタデータを編集できます。正常グループには、Web サービスの SOAP 応答が含まれます。フォルトグループには SOAP フォルトが含まれ、フォルトコード、文字列、およびフォルトの発生原因となったオブジェクト名が含まれます。
7. 詳細プロパティを定義します。
8. マッピングを保存して実行します。

マッピングプロセスの詳細については、次のセクションを参照してください。

詳細プロパティ

次の表に、Web サービストランスフォーメーションの**［詳細］**タブで使用できるプロパティを示します。

プロパティ	説明
キャッシュサイズ	Web サービスの要求と応答に使用可能なメモリ。Web サービスの要求または応答に多くの行またはカラムが含まれる場合は、キャッシュサイズを拡大するとよいでしょう。デフォルトは 100 KB です。
入力フラッシュを許可	マッピングタスクでは、グループのすべてのデータがあるとき、XML が作成されます。これを有効にすると、マッピングタスクでは、ルート値のすべてのデータを受信した後で、XML はフラッシュされます。これを有効にしない場合、マッピングタスクでは、すべてのグループのデータを受信した後で、XML がメモリに保存されて XML が作成されます。 **注**: 複数のソースオブジェクトに接続している場合は、入力フラッシュを許可するオプションを選択できません。
トランザクションコミット制御	トランスフォーメーションを通過する一連の行に基づいて、トランザクションのコミットまたはロールバックを制御します。マッピングタスクでトランザクションの変更を行にコミットするか、トランザクションの変更をロールバックするか、トランザクションを変更しないかを決める条件を指定するには、IIF 関数を入力します。データの量が多く、その処理方法を制御する場合は、トランザクションコミットコントロールを使用します。 **注**: 複数のソースオブジェクトに接続している場合は、トランザクションコミット制御を設定できません。

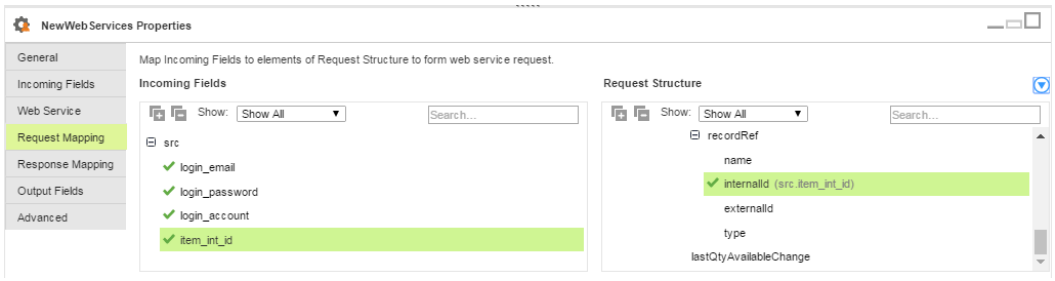
受信フィールドのマッピング

要求マッピングを定義する場合、複数のソースオブジェクト間のリレーションを設定できます。応答マッピングと出力フィールド間のリレーションを個別に設定します。

受信フィールドをマッピングする場合、次のガイドラインに注意してください。

- 式を受信フィールドに適用する必要がある場合は、Web サービストランスフォーメーションのアップストリームにある式トランスフォーメーションを使用します。
- 必要なすべての情報を Web サービス要求に確実に含めるには、入力派生型フィールドを要求構造のフィールドにマッピングします。

次の図に示すように、受信フィールドを要求マッピングにマップできます。



各受信フィールドを、マッピングする要求構造のノードにドラッグします。

複数のソースオブジェクトの使用

複数のソースがある場合、次の要件に注意してください。

- プライマリキーおよび外部キーとして指定するソースフィールドでは、Bigint データ型または String データ型を使用する必要があります。必要に応じて、ソーストランスフォーメーションのメタデータを編集できます。

  注: Bigint データ型をソースに使用できない場合は、Web サービストランスフォーメーションのアップストリームにある式トランスフォーメーションでデータを変換できます。
- ソースデータが、親オブジェクトのプライマリキー、および子オブジェクトの外部キーとプライマリキーでソートされていることを確認します。
- いずれかのフィールドまたはフィールドグループを繰り返し要素にマッピングします。受信フィールドで、各繰り返し要素がマッピングされている場所を表示できます。
- 各子オブジェクトの最低 1 つのフィールドを要求構造にマッピングします。
- 親オブジェクトのフィールドは、子オブジェクトのフィールドよりも高い階層の要求構造のフィールドにマッピングする必要があります。
- 子オブジェクトにプライマリキーと外部キーを選択します。

  - **[受信フィールド]** タブで、親オブジェクトとして指定するソースオブジェクトを選択します。

  - ツリーで受信フィールドを右クリックし、プライマリキーと外部キーを指定します。

  - 外部キーに親オブジェクトを指定します。
- 親オブジェクトに外部キーを選択しないでください。

発信フィールドのマッピング

[応答マッピング] タブで、使用する出力フィールドに応答構造をマッピングします。出力フィールドの形式には、[リレーショナル] または [非正規化] を選択できます。

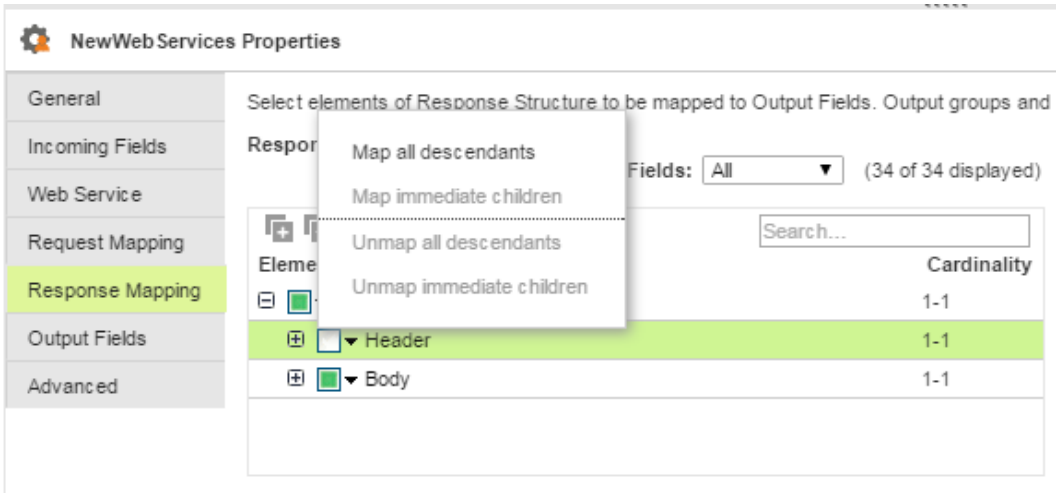
[リレーショナル] を選択すると、トランスフォーメーションによって次の出力グループが生成されます。

- 親要素の 1 つの出力グループ。
- カーディナリティが 1 より大きい要素ごとに 1 つの出力グループ。
- FaultGroup（使用している接続タイプでサポートされている場合）。

[非正規化] を選択すると、トランスフォーメーションによって次の出力グループが生成されます。

- 親要素の出力グループ。非正規化された出力では、親グループの要素値が子要素ごとに繰り返されます。
- FaultGroup（使用している接続タイプでサポートされている場合）。

応答を出力フィールドにマッピングする応答のノード内のノードを右クリックします。次の図に示すように、すべての子孫をマッピングするか、直下の子のみをマッピングするかを選択できます。



トランザクションコミット制御

トランザクションコミット制御では、トランスフォーメーションを通過する一連の行に基づいて、トランザクションのコミットとロールバックを制御できます。

マッピングタスクでトランザクションの変更を行にコミットするか、トランザクションの変更をロールバックするか、トランザクションを変更しないかを決める条件を指定するには、IIF 関数を入力します。マッピングタスクは、式の戻り値に基づいてコミットまたはロールバックを発行すると、新しいトランザクションを開始します。

注: 複数のソースオブジェクトに接続している場合は、トランザクションコミット制御を設定できません。

式には次の構文を使用します。

```
IIF (condition, value1, value2)
```

トランザクションコントロール式を作成するときは、次の組み込み変数を使用します。

- TC_CONTINUE_TRANSACTION.マッピングタスクでは、この行のトランザクション変更が実行されません。これは式のデフォルト値です。
- TC_COMMIT_BEFORE.マッピングタスクはトランザクションをコミットし、新しいトランザクションを開始し、カレント行をターゲットに書き出します。カレント行は、新しいトランザクション内にあります。
- TC_COMMIT_AFTER。マッピングタスクは、カレント行をターゲットに書き出し、トランザクションをコミットし、新しいトランザクションを開始します。カレント行は、コミットされたトランザクション内にあります。
- TC_ROLLBACK_BEFORE。マッピングタスクは、現在のトランザクションをロールバックし、新しいトランザクションを開始し、カレント行をターゲットに書き出します。カレント行は、新しいトランザクション内にあります。
- TC_ROLLBACK_AFTER.マッピングタスクは、カレント行をターゲットに書き出し、トランザクションをロールバックし、新しいトランザクションを開始します。カレント行は、ロールバックされたトランザクション内にあります。

トランザクションコントロール式がコミット、ロールバック、または継続以外の値に評価された場合、マッピングは無効になります。

例

トランザクションコミット制御を使って、注文入力日に基づいて注文情報を書き出す場合を想定します。特定日に受けたすべての注文は、同じトランザクションでターゲットにコミットするとします。

New_Date というフィールドを作成し、カレント行の注文日を前の行の注文日と比較して、このフィールドに情報を入力します。注文の日付が異なる場合、New_Date は 1 になります。

この場合は、次の式を使用して、マッピングタスクで新しい注文日が発生したときにデータをコミットします。

```
IIF(New_Date = 1, TC_COMMIT_BEFORE, TC_CONTINUE_TRANSACTION)
```

受信フィールドと要求フィールドの表示

受信フィールドと要求フィールドを確認するときは、ページに表示されるフィールドの数を減らすことができます。

次の方法により、フィールドの確認や検索を行うことができます。

ノードの展開と縮小

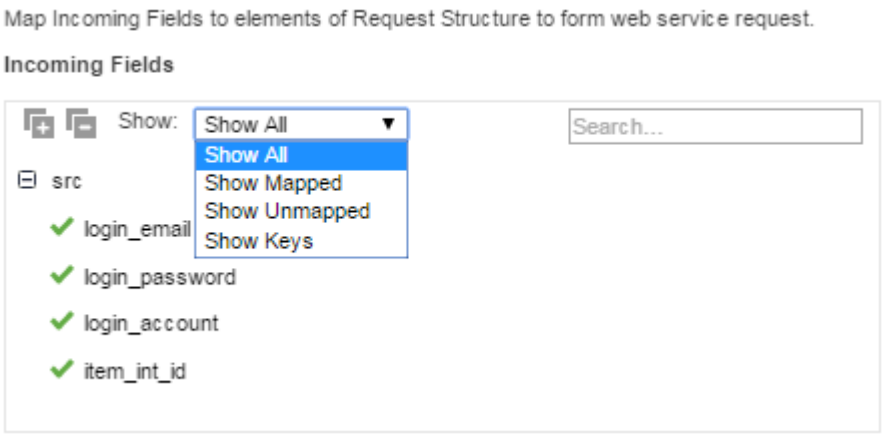
ツリーの親ノードと子ノードを表示するには、すべてのノードを展開します。親ノードのみを表示するには、ノードを縮小します。［要求マッピング］または［応答マッピング］タブの **［すべてのノードを展開］** および **［ノードを縮小］** アイコンを使用します。

フィールドの検索

特定のフィールドを検索するには、**［検索］** テキストボックスにフィールド名を入力します。

フィールドのフィルタリング

受信フィールドをフィルタリングして、すべてのフィールド、キー、マッピングされたフィールド、またはマッピングされていないフィールドを表示できます。他のツリービューにも同じようなオプションがあります。

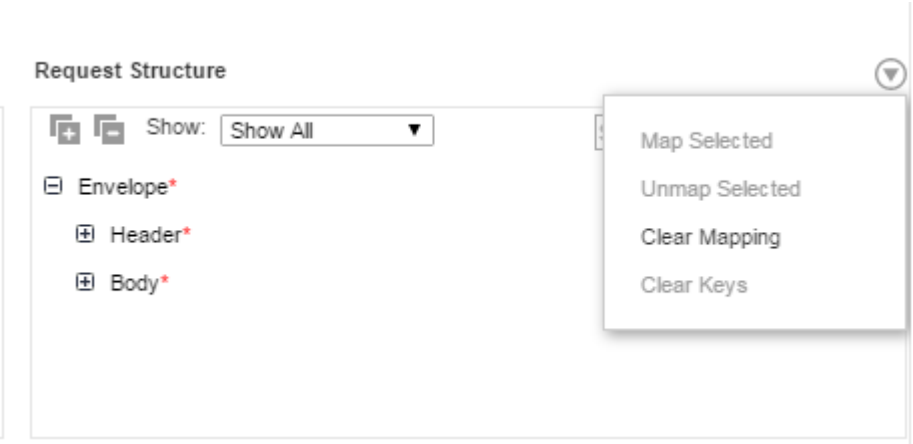


フィールドおよびフィールドマッピングの詳細の表示

階層の各ノードで、フィールドおよびフィールドマッピングの詳細を表示できます。ノードを右クリックしてプロパティを表示します。

フィールドマッピングおよびキーのクリア

要求構造ツリーで、マッピングのクリア、キーのクリア、または選択したフィールドのマッピングとマッピング解除を行うことができます。



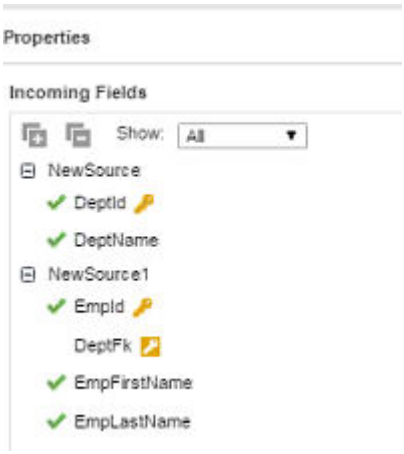
パススルーフィールド

パススルーフィールドは応答フィールドであり、現在のトランスフォーメーションでは必要ありませんが、今後のマッピングで使用する可能性があります。

例えば、Web サービストランスフォーメーションでは、ソースオブジェクトのプライマリキーしか必要とならないことがあります。しかし、ダウンストリームトランスフォーメーションでは、ソースオブジェクトのその他のフィールドを使用する可能性があります。

受信フィールドと応答フィールドは、Web サービストランスフォーメーションを通過して、正常グループとフォルトグループに入ります。ただし、マッピングに複数のソースが含まれるときは、プライマリソースのフィールドのみが通過します。最初に Web サービストランスフォーメーションに接続されたソースがプライマリソ

ースになります。例えば、次の図では、NewSource がプライマリソースであり、NewSource1 はセカンダリソースです。



NewSource1 はセカンダリソースなので、このソースのフィールドは通過せず、正常グループとフォルトグループに入りません。

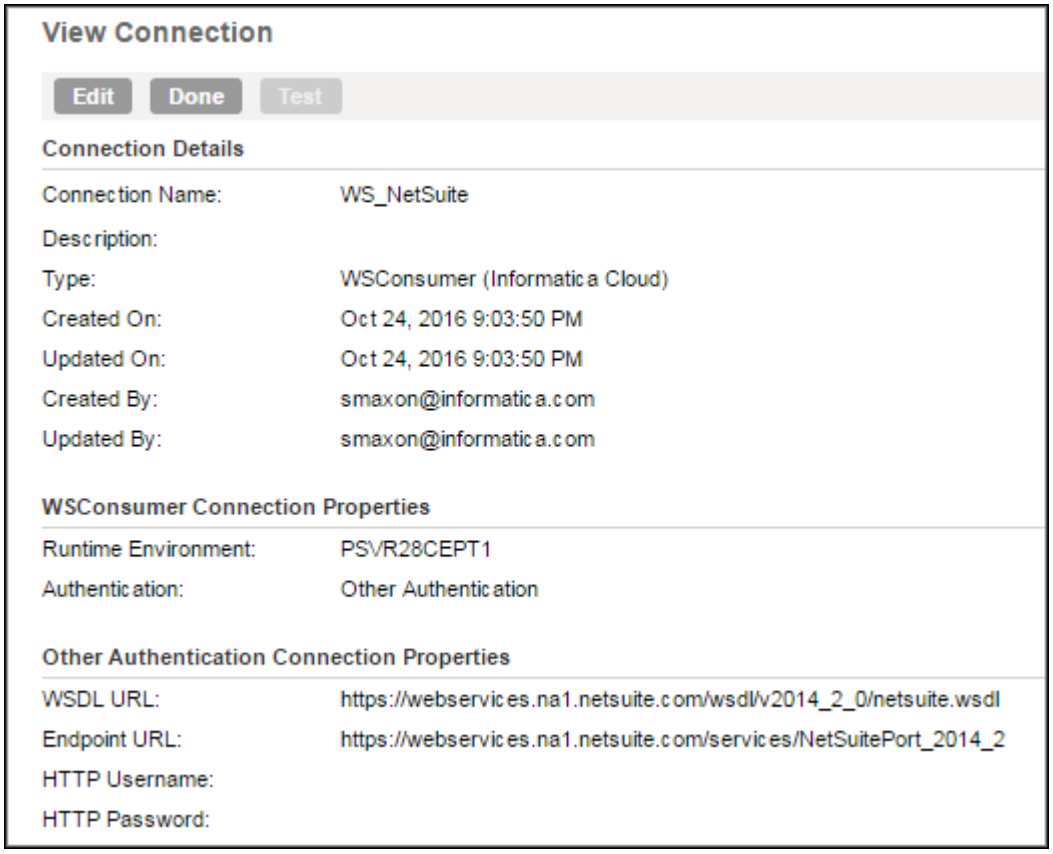
正常グループまたはフォルトグループでフィールドのソースを判断するには、**[受信フィールド]** タブを選択します。**[オリジン]** カラムには、各フィールドのソースが表示されます。

Web サービストランスフォーメーションの例

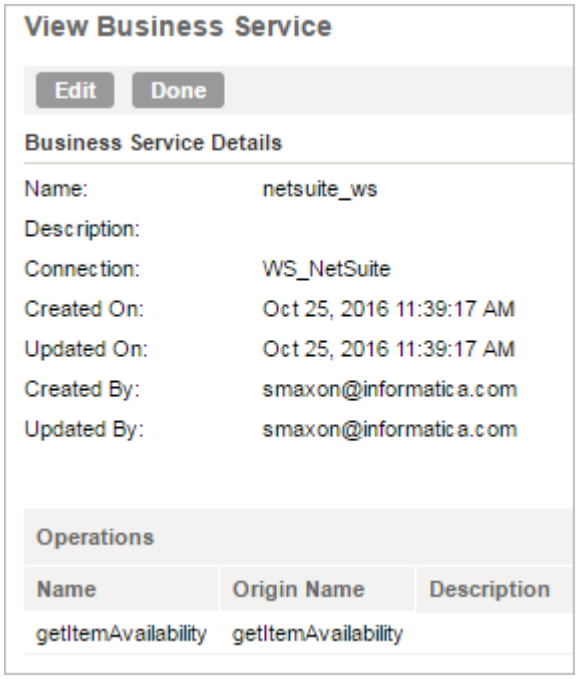
次の例は、Web サービストランスフォーメーションを使用して NetSuite とやり取りし、製品可用性に関する詳細を取得する方法を示しています。

この例を確認することで、NetSuite 操作である getItemAvailability を使用して SOAP 要求および応答を構造化するように Web サービストランスフォーメーションを設定する方法について学習できます。Web サービスクライアントは、要求で項目 ID を渡します。Web サービスは、最新の項目可用性に関する情報を SOAP 応答で返します。

最初に、Web サービスに接続するため、WSConsumer 接続を作成します。要求と応答の Web サービスメッセージを記述する WSDL を追加し、必要なセキュリティを設定します。この例では、次の NetSuite 接続を使用します。

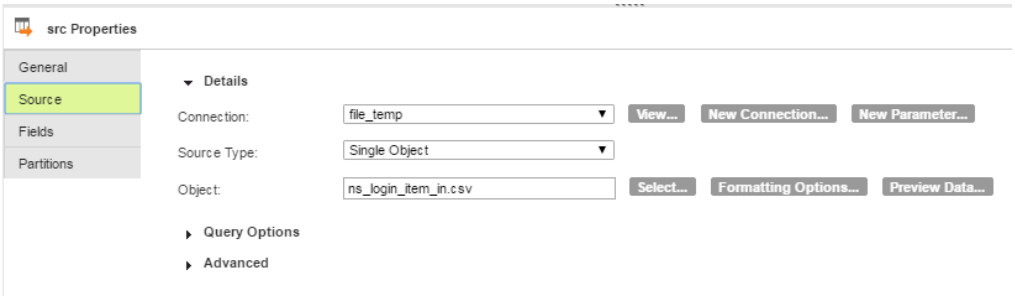


2 番目に、接続を使用してビジネスサービスを定義します。このケースでは、接続を使用してビジネスサービスを定義し、getItemAvailability 操作にアクセスします。



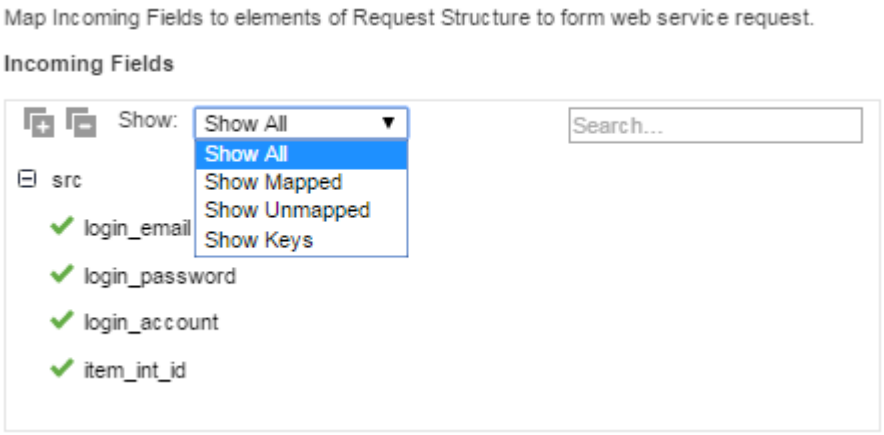
3 番目に、Web サービストランスフォーメーションを使用するマッピングを作成します。マッピングの例には、次の設定オプションが含まれます。

1. ソースは、ログインの詳細や項目 ID が含まれる 4 つのフィールドを格納する単純な.csv ファイルです。

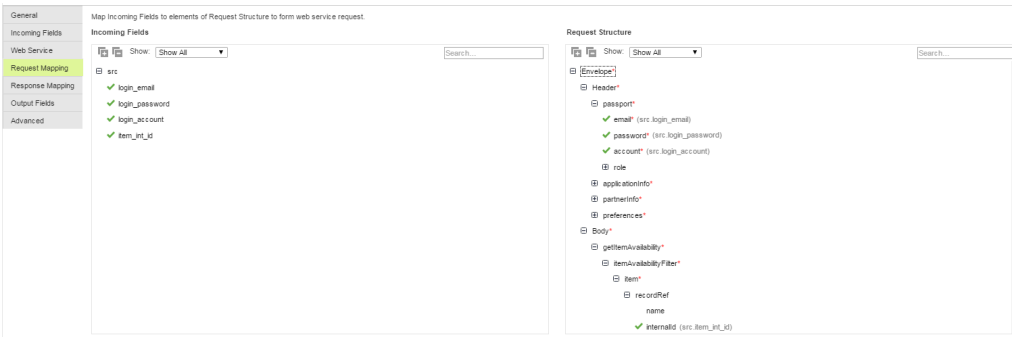


2. ［Web サービス］タブで、以前に定義したビジネスサービスと操作を選択します。

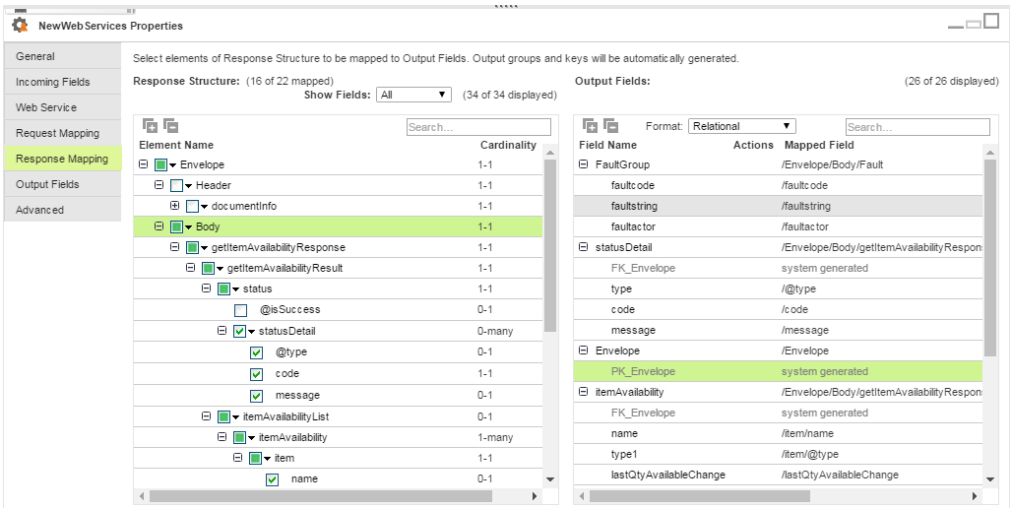
3. Web サービストランスフォーメーションを追加したら、ソースを接続し、4 つの受信フィールドを表示できます。受信フィールドのビューをフィルタリングして、すべてのフィールド、マッピングされたフィールド、マッピングされていないフィールド、またはキーを表示できます。



4. ［要求マッピング］では、受信フィールドが SOAP メッセージにマッピングされています。次の画像のように、Envelope には資格情報、Body には項目 ID が含まれます。受信フィールドを要求構造の項目にドラッグし、マッピングを作成します。



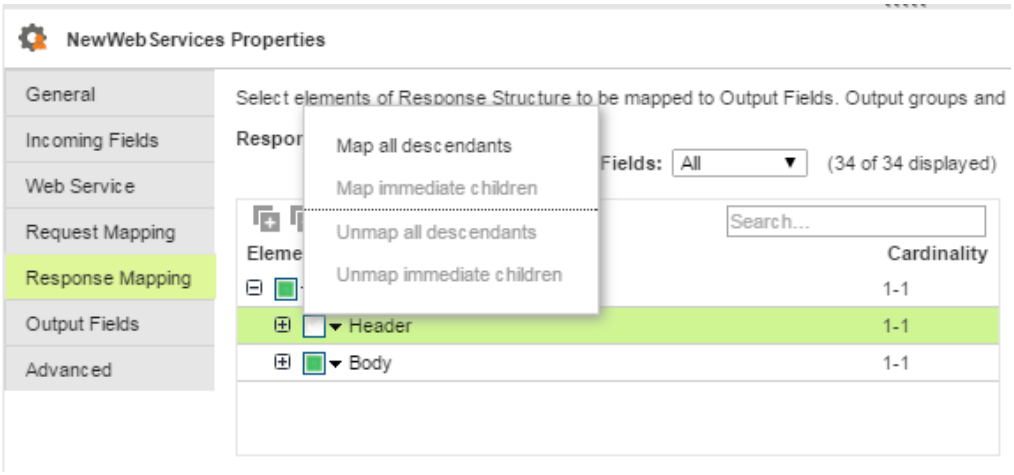
5. ［応答マッピング］タブで、使用する出力フィールドに SOAP メッセージをマッピングします。出力フィールドの形式に［リレーショナル］または［非正規化］を選択できます。この例では、［リレーショナル］形式を使用します。



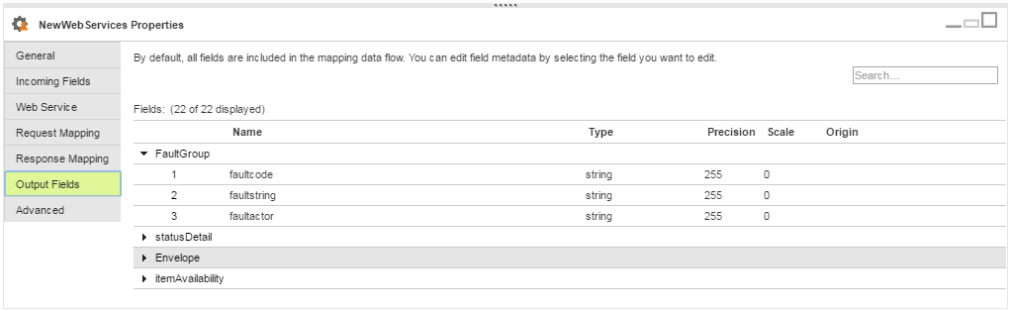
次の出力フィールドグループがシステムによって自動的に生成されます。

- FaultGroup（使用している接続タイプでサポートされている場合）。
- ドキュメント情報が含まれる Envelope（ヘッダーをマッピングする場合）。
- カーディナリティが 1 を超える、マッピングした各要素の 1 つのグループ。

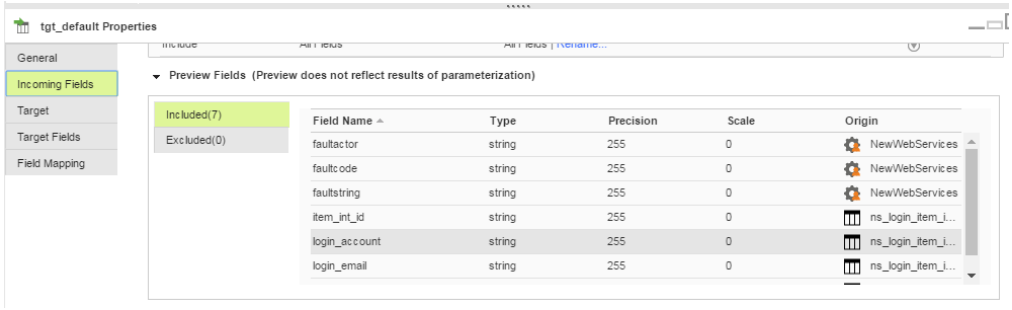
上の例では、3 つのグループがあります。次の画像のように、応答で階層のノードを選択し、マッピング（マッピング解除）するオプションを選択することで、オブジェクトやフィールドのマッピングまたはマッピング解除を行うことができます。下矢印をクリックして、特定のノードのマッピングオプションを表示します。

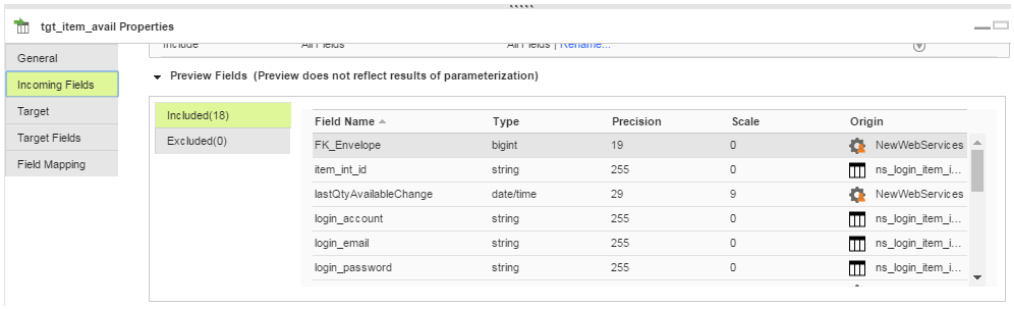


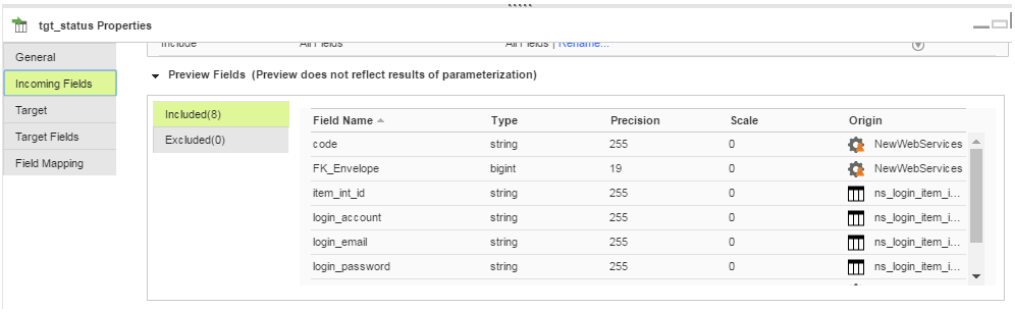
6. ［出力フィールド］タブで、各グループを表示できます。必要に応じて、フィールドのタイプ、精度、およびスケールを編集できます。



7. マッピングには、3 つのターゲット（ステータス、可用性、FaultGroup データのターゲット）が含まれます。







マッピングが実行されると、項目可用性とステータスコードが返されます。正常に実行できなかった場合、フォールト情報が含まれるレコードがデフォルトターゲットに作成されます。

マルチバイト階層データ

マッピングに階層データを処理するトランスフォーメーションが含まれていて、データにマルチバイト文字が使用されている場合は、UTF-8 を使用するようにシステムを設定する必要があります。

Windows では、Windows システムプロパティの INFA_CODEPAGENAME=UTF-8 環境変数を作成します。

Linux では、LC_LOCALE 変数を UTF-8 に設定します。

索引

L

M

N

O

P

S

の

は

ひ

ふ

へ

ま

め

ら

り

る