pitney bowes

Spectrum™ Technology Platform

バージョン 12.0

Dataflow Designer ガイド

目次

1 - はじめに

このセクションの構成

クライアント ツールのインストール

Spectrum™ Technology Platform クライアント ツールは、 サーバーの管理や、データフローとプロセス フローの設計および実行に使用するアプリケーションです。クライアント ツールを インストールする前に、Spectrum™ Technology Platform サーバーをインストールする必要があります 。

インストールする前に、リリース ノートに目を通してください。リリース ノートには、互換性に関する重要な情報やリリースに固有のインストール上の注意事項が記載されています。

この手順では、以下のクライアント ツールのインストール方法について説明します。

- **Enterprise Designer** — データフローの作成、変更、実行に使用します。
- **Job Executor** — コマンド ラインまたはスクリプトからジョブを実行できるコマンド ライン ツールです。ジョブは、Enterprise Designer を使用して、Spectrum™ Technology Platform で作成および保存されたものである必要があります。
- **Process Flow Executor** — コマンド ラインまたはスクリプトからプロセス フローを実行することのできるコマンド ライン ツールです。プロセス フローは、Enterprise Designer を使用して、Spectrum™ Technology Platform で作成および保存されたものである必要があります。
- **Administration Utility** — 管理ユーティリティでは、いくつかの管理機能をコマンドラインから実行できます。この機能はスクリプトで利用できるため、特定の管理タスクを自動化できます。対話式の操作で機能を実行することもできます。

  注： Spectrum バージョン 11.0 から Management Console は、これまでのリリースのようなインストール可能なクライアントではなく、Web ベースのツールになりました。

クライアント ツールをインストールするには、次の操作を行います。

1. Web ブラウザを起動し、次の Spectrum™ Technology Platform の Welcome ページを開きます。

   http://<サーバー名>:<ポート>

   例えば、Spectrum™ Technology Platform が "myspectrumplatform" という名前のコンピュータにインストールされており、デフォルトの HTTP ポート 8080 を使用している場合は、次のアドレスに移動します。

   http://myspectrumplatform:8080

2. **[プラットフォーム クライアント ツール]** をクリックします。
3. インストールするクライアント ツールをダウンロードします。

Enterprise Designer の起動

Enterprise Designer は、データフローを作成するための Windows アプリケーションです。Enterprise Designer を起動するには

1. **[スタート] > [すべてのプログラム] > [Pitney Bowes] > [Spectrum™ Technology Platform] > [クライアント ツール] > [Enterprise Designer]** の順に選択します。
2. サーバー名または IP アドレスを入力するか、ドロップダウン リストから選択します。クラスタ内で Spectrum™ Technology Platform を使用している場合は、クラスタのロード バランサーの名前または IP アドレスを入力します。
3. ユーザ名とパスワードを入力します。
4. [ポート] フィールドに、Spectrum™ Technology Platform 通信に使用するように設定されているサーバーのネットワーク ポートを入力します。デフォルトのポート番号は 8080 です。
5. クライアントとサーバーの間で HTTPS 接続を使用する場合は、**[セキュアな接続を使用]** をクリックします。

   注 : セキュアな接続は、サーバー上で HTTPS 通信が設定されている場合にのみ使用できます。Windows 7 で Enterprise Designer を実行している場合、Enterprise Designer とサーバー間の通信のセキュリティ保護に使用される証明書のタイプによっては、**[サーバー名]** フィールドでの IP アドレスの使用が正常に機能しない可能性があります。IP アドレスが機能しない場合は、代わりにホスト名を使用します。

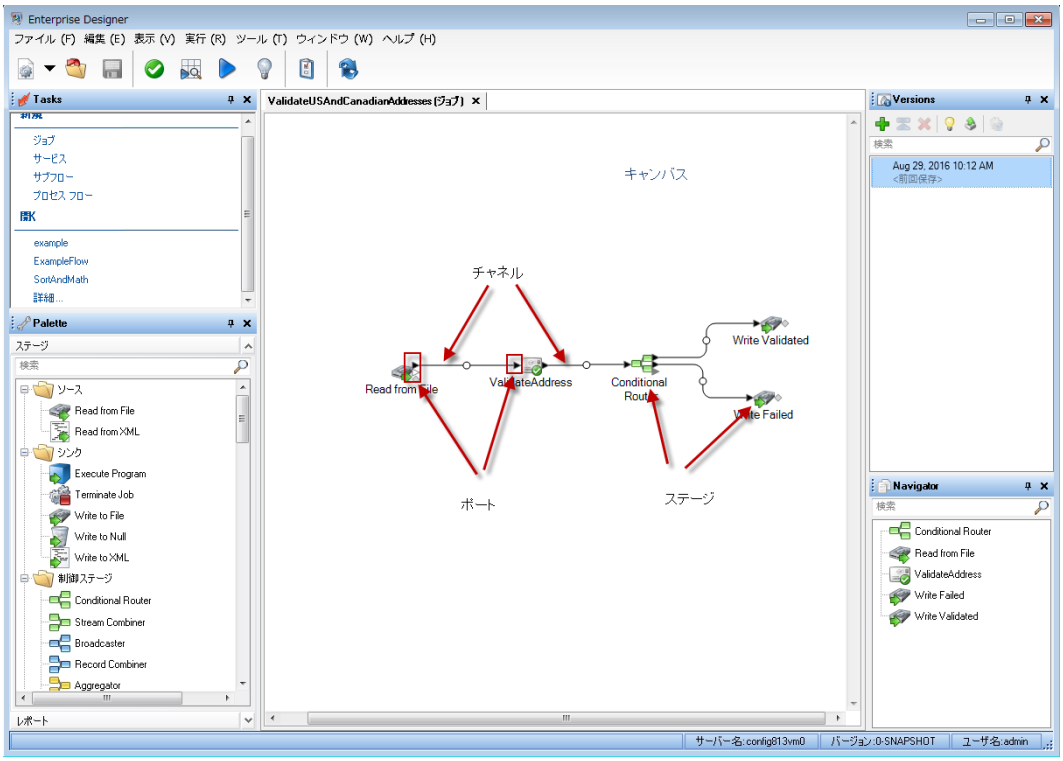
6. **[ログイン]** をクリックします。

Enterprise Designer の概要

Enterprise Designer は、データフローを作成するためのビジュアル ツールです。このクライアントを使用すると、以下の操作が行えます。

- ジョブ、サービス、サブフロー、およびプロセス フローを作成および変更する。
- データフローをテストして問題の有無を確認する。
- サービスをエクスポーズおよび隠す。
- レポートの生成

Enterprise Designer ウィンドウは、次のようなウィンドウです。



データフローを操作するには、いくつかの重要用語を理解する必要があります。

キャンバス キャンバスは主要なワークエリアです。上記の図に示すキャンバスには、ValidateUSAndCanadianAddresses という名前のデータフローが開かれています。これはジョブ データフローで、ファイルに対してデータの読み書き操作を行ってバッチ処理を実行するという意味です。この場合、データフローは 2 つのファイルに出力を書き込みます。

ステージ ステージ (キャンバス上のアイコンによって表される) は、レコードのソート、住所の検証、同様のレコードの照合など、特定のタイプのアクティビティを実行します。ステージを追加するには、(ウィンドウの左側にある) パレットからキャンバスにステージをドラッグします。

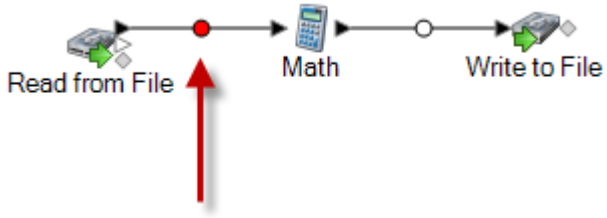
ステージにユーザの注意が必要である場合、アイコンに青い円が表示されます。

注意が必要なステージが存在する場合、データフローは正しく実行できません。したがって、ステージをダブルクリックして、必要な項目を設定する必要があります。必要な項目をすべて設定すると、青い円は消えます。

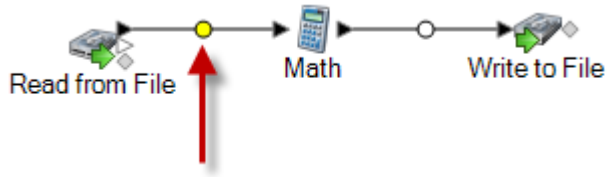
チャネル チャネルとは、2 つ以上のステージ間の接続のことです。ステージ間でのレコードの引き渡しにはチャネルが使用されます。上記の例では、Read from File ステージはチャネルによって ValidateAddress ステージに接続されているのを確認できます。レコードは Read from File のデータフローに読み込まれた後、このチャネルを介して ValidateAddress に送信されます。その後、ValidateAddress はチャネルを介して Conditional Router に接続されます。Conditional Router は、レコードを分析し、Dataflow Designer によって定義された条件に応じて、そのレコードをデータフローのさまざまなパスを介して送信します。Conditional Router には外側に向かう 2 つのチャネルがあり、1 つは Write Validated ステージに、もう 1 つは Write Failed ステージに接続されています。

さまざまな状態を反映して、チャネル中央のドットの色が変わる場合があります。

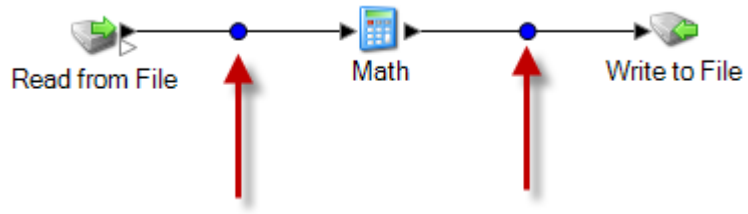
赤 エラーを表します。例えば、タイプ変換に失敗してフィールドが下流のステージで使用できないといった場合が挙げられます。



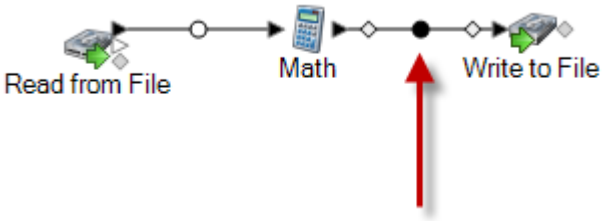
黄 下流のステージで必要なフィールドが削除されています。



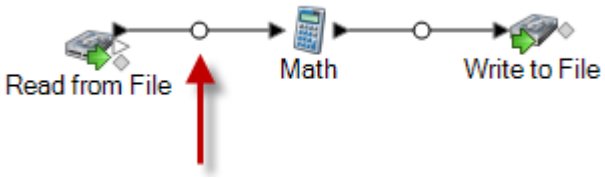
青 自動タイプ変換によって、フィールドが下流のステージで求められるデータ タイプに正しく変換されています。



黒 チャネル内でフィールド名が変更されています。



白 フィールドに対して何のアクションも実行されていません。



ポート ステージ アイコンを詳しく見ると、各ステージの側に小さな三角形またはダイヤモンド型のポートがあることに気付きます。ポートとは、ステージがチャネルに対してデータの読み書き操作を行うためのメカニズムです。データフローにデータを読み込むステージ ("ソース" と呼ばれる) は、常にデータフローの開始点になるので、出力ポートしかありません。データフローからデータを送信するステージ ("シンク" と呼ばれる) は、常にデータフローの終点になるので、入力ポートしかありません。その他すべてのステージには入力ポートと主力ポートの両方があります。また、ステージによっては、ステージの処理中にエラーを引き起こすレコードを出力するためのエラー ポートや、ステージの出力に関するレポートを生成するためのレポート ポートを備えているものもあります。

さらに、Enterprise Designer ウィンドウには以下の機能もあります。

フィーチャー	説明
タスク	新しいジョブ、サービス、サブフロー、またはプロセス フローをすばやく作成できます。また、最近開いたデータフローを開くこともできます。
サーバエクスプローラー	Spectrum™ Technology Platformサーバーに保存されたすべてのフローを表示します。サーバー エクスプローラが表示されない場合は、**[表示] > [サーバー エクスプローラ]** を選択します。フローを複数のフォルダに整理することができます。フォルダを作成するには、サーバー名を右クリックして、**[新しいフォルダ]** を選択します。フロー名は、どのフォルダに配置したかに関係なく、一意でなければなりません。配置するフォルダが異なっていても、同じ名前のフローを 2 つ作成することはできません。

フィーチャー	説明
パレット	ステージとレポートが含まれます。これらはすべて、データフローに追加できます。パレットで使用できるステージは、ライセンスを供与しているモジュールによって異なります。
キャンバス	ステージをドラッグし、チャネルに接続してデータフローを作成するワークエリア。一度に複数のデータフロー キャンバスを開くことができます。
バージョン	Enterprise Designer のバージョン機能を使用すると、データフローの改訂履歴を保持できます。データフローの以前のバージョンを表示したり、実行のために古いバージョンをエクスポーズしたりできます。また、データフローを以前のバージョンに戻す必要がある場合に備えて、変更履歴を保持することもできます。
ナビゲーター	フロー内のステージとレポートのリストが表示されます。**[Navigator (ナビゲータ)]** ウィンドウ内のアイテムを右クリックすることで、そのオプションを編集できます。

初めてのデータフロー (ジョブ)

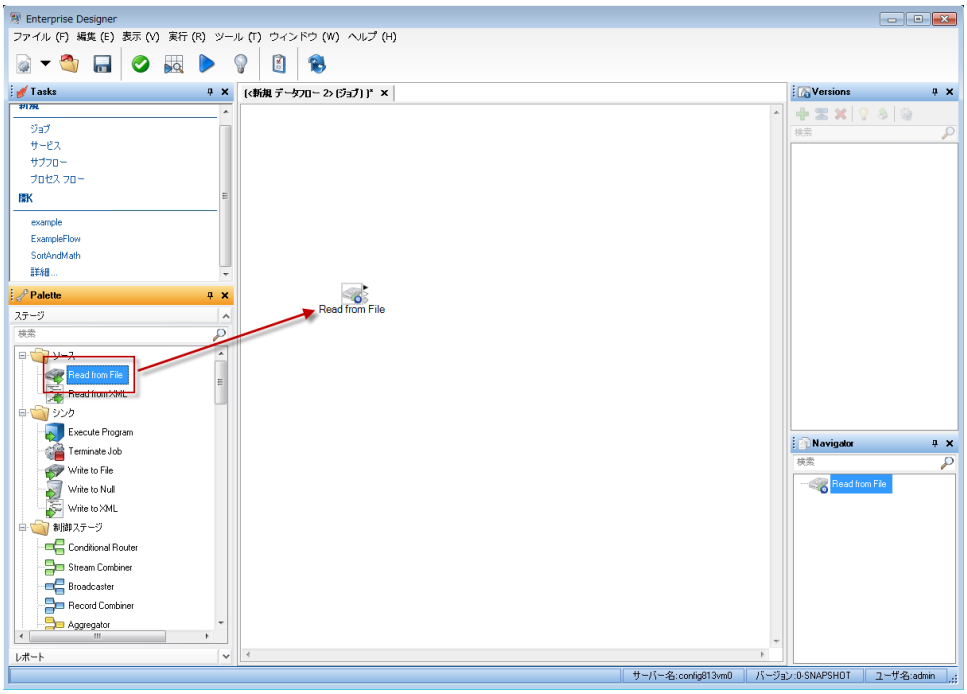
このトピックでは、ファイルからデータを読み込み、ソートし、ファイルに書き出すという簡単なデータフローを作成します。 このデータフローはファイルからデータを読み込み、出力をファイルに書き出すため、「ジョブ」と呼ばれます。ジョブは、バッチ処理を実行するデータフローです (データフローのもう 1 つの主要タイプは、API 、またはサーバーへの Web サービス呼び出しを介してインタラクティブ処理を実行する「サービス」です)。

1. 最初のステップは、データフローへの入力として使用する、何らかのサンプル データを作成することです。 テキスト エディタを使用して、次のようなファイルを作成します。

```
FirstName,LastName,Region,Amount
Alan,Smith,East,18.23
Jeannie,Wagner,North,45.43
Joe,Simmons,East,10.87
Pam,Hiznay,Central,98.78
```

2. 適当な場所にファイルを保存します。
3. **[スタート] > [すべてのプログラム] > [Pitney Bowes] > [Spectrum™ Technology Platform] > [クライアント ツール] > [Enterprise Designer]** の順に選択します。

4. **[ファイル] > [新規作成] > [データフロー] > [ジョブ]** の順に選択します。
5. これでデータフローの作成を開始する準備ができました。 最初のステップは、データフローへの入力を定義することです。 これを行うには、次の手順を実行します。
   a) Read from File ステージをキャンバス上にドラッグします。



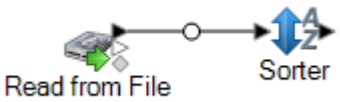
   b) キャンバス上の Read from File ステージをダブルクリックします。
   c) **[ファイル名]** フィールドで、「1（10ページ）」ステップで作成したファイルを指定します。
   d) **[レコード タイプ]** フィールドで、**[区切り記号付き]** を選択します。
   e) **[フィールド区切り文字]** フィールドで、**[カンマ (,)]** を選択します。
   f) **[最初の行はヘッダ レコード]** ボックスをオンにします。
   g) **[フィールド]** タブをクリックします。
   h) **[再生成]** をクリックし、**[はい]** をクリックします。

      Read from File ステージが入力ファイルのフィールド用に自動的に設定されます。

   i) **[検出タイプ]** をクリックします。 入力ファイルがスキャンされ、フィールドごとに適切なデータ タイプが決定されます。 **[Amount]** フィールドのタイプが string から double に変わったことに注意してください。
   j) Read from File ステージの設定が完了しました。 **[OK]** をクリックします。
6. 次に、レコードを地域でソートするステージを追加します。 これを行うには、次の手順を実行します。
   a) Sorter ステージをキャンバス上にドラッグします。

b) Read from File ステージの右側 (出力ポート) にある黒い三角形をクリックし、キャンバス上の Sorter ステージの左側にドラッグして、Read from File ステージと Sorter ステージを接続するチャネルを作成します。

データフローは次のようになっているはずです。



c) キャンバス上の Sorter ステージをダブルクリックします。
d) **[追加]** をクリックします。
e) **[フィールド名]** フィールドで、**[Region]** を選択します。
f) Sorter ステージの設定が完了しました。 **[OK]** をクリックします。

7. 最後に、データフローが出力を書き出す出力ファイルを定義します。 これを行うには、次の手順を実行します。
   a) Write to File ステージをキャンバス上にドラッグします。
   b) Sorter ステージの右側にある黒い三角形をクリックし、キャンバス上の Write to File ステージの左側にドラッグします。

   データフローは次のようになっているはずです。

   c) [Write to File] ステージをダブルクリックします。
   d) **[ファイル名]** フィールドで、出力ファイルを指定します。 任意のファイルを指定できます。
   e) **[フィールド区切り文字]** フィールドで、**[カンマ (,)]** を選択します。
   f) **[最初の行はヘッダ レコード]** ボックスをオンにします。
   g) **[フィールド]** タブをクリックします。
   h) **[クイック追加]** をクリックします。
   i) **[すべて選択]** をクリックし、**[OK]** をクリックします。
   j) **[上へ移動]** ボタンと **[下へ移動]** ボタンを使用して、次の順序になるようにフィールドを並べ替えます。

      FirstName
      LastName
      Region
      Amount

      これによって、出力ファイル内のレコードのフィールドが、入力ファイルと同じ順序になります。

k) Write to File ステージの設定が完了しました。 **[OK]** をクリックします。

8. Enterprise Designer で、**[ファイル] > [保存]** の順に選択します。
9. データフローの名前を指定して、**[OK]** をクリックします。
10. これでデータフローを実行する準備ができました。 **[実行] > [現在のフローを実行]** の順に選択します。
11. **[実行の詳細]** ウィンドウが表示され、ジョブのステータスが表示されます。 **[更新]** をクリックします。 ステータスが **[正常終了]** になったら、**[閉じる]** をクリックします。

Write to File ステージで指定した出力ファイルを開きます。 Sorter ステージで指定したとおりに、レコードが地域でソートされていることがわかります。

```
FirstName,LastName,Region,Amount
Pam,Hiznay,Central,98.78
Alan,Smith,East,18.23
Joe,Simmons,East,10.87
Jeannie,Wagner,North,45.43
```

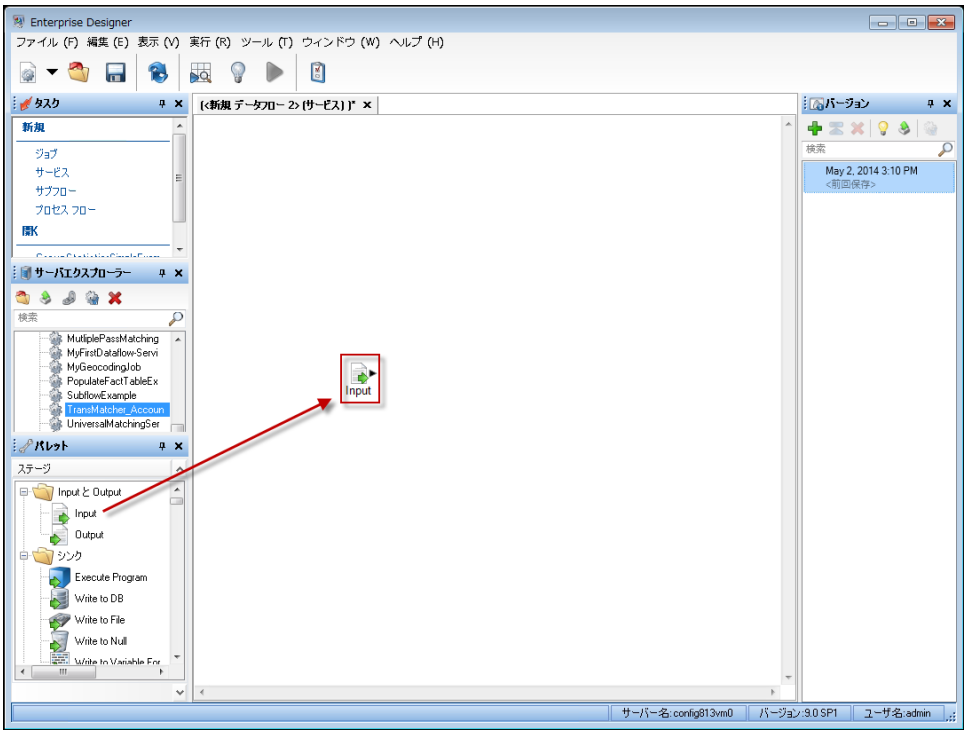
おめでとうございます。 初めてのジョブ データフローの設計と実行が完了しました。

初めてのデータフロー (サービス)

このトピックでは、API または Web サービス呼び出しからデータを受け取り、API または Web サービスを介して応答を返す簡単なデータフローを作成します。このデータフローはSpectrum™ Technology Platformサーバーにサービスとしてエクスポーズされるもので、「サービス」データフローです(データフローのもう 1 つの主要タイプは、バッチ処理を実行し、ファイルまたはデータベースからデータを読み込み、データを処理し、出力をファイルまたはデータベースに書き込む「ジョブ」です)。

1. **[スタート] > [すべてのプログラム] > [Pitney Bowes] > [Spectrum™ Technology Platform] > [クライアント ツール] > [Enterprise Designer]** の順に選択します。
2. **[ファイル] > [新規作成] > [データフロー] > [サービス]** の順に選択します。
3. これでデータフローの作成を開始する準備ができました。最初のステップは、データフローへの入力を定義することです。このデータフローは、FirstName と LastName の 2 つのフィールドを入力として受け取ります。

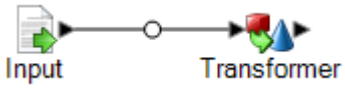
   a) Input ステージをパレットからキャンバスにドラッグします。



b) キャンバス上の Input ステージをダブルクリックします。
c) **[追加]** をクリックし、もう一度 **[追加]** をクリックします。
d) **[フィールド名]** フィールドで、「`FirstName`」と入力します。
e) **[OK]** をクリックし、もう一度 **[OK]** をクリックします。
f) **[追加]** をクリックし、もう一度 **[追加]** をクリックします。
g) **[フィールド名]** フィールドで、「`LastName`」と入力します。
h) **[OK]** をクリックし、もう一度 **[OK]** をクリックします。
i) データフロー入力の定義が完了しました。**[OK]** をクリックします。

4. 次に、[FirstName] フィールドと [LastName] フィールドのデータの大文字小文字をすべて大文字に変更するためのステージを追加します。
   a) Transformer ステージをパレットからキャンバスにドラッグします。
   b) Input ステージの右側 (出力ポート) にある黒い三角形をクリックし、キャンバス上の Transformer ステージの左側にドラッグして、Input ステージと Transformer ステージを接続するチャネルを作成します。

      データフローは次のようになっているはずです。

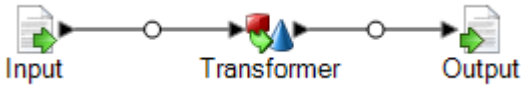


   c) [Transformer] ステージをダブルクリックします。
   d) **[追加]** をクリックします。

e) 左側のツリーの **[書式設定]** の下にある **[大文字小文字]** をクリックします。
f) **[フィールド]** フィールドで、**[FirstName]** を選択します。**[大文字]** が選択されている状態のままにします。
g) **[追加]** をクリックします。
h) **[フィールド]** フィールドで、**[LastName]** を選択します。**[大文字]** が選択されている状態のままにします。
i) **[追加]** をクリックします。
j) **[閉じる]** をクリックします。
k) [FirstName] フィールドと [LastName] フィールドの値を大文字に変更するように、Transformer ステージの設定が完了しました。**[OK]** をクリックします。

5. 最後に、データフローの出力を定義します。このデータフローは、出力として [FirstName] フィールドと [LastName] フィールドを返します。
   a) Output ステージをキャンバス上にドラッグします。
   b) Transformer ステージの右側にある黒い三角形をクリックし、キャンバス上の Output ステージの左側にドラッグします。

      データフローは次のようになっているはずです。

   c) キャンバス上の Output ステージをダブルクリックします。
   d) **[エクスポーズ]** ボックスをオンにします。[FirstName] と [LastName] の横のチェック ボックスがオンになっているはずです。
   e) **[OK]** をクリックします。
6. Enterprise Designer で、**[ファイル] > [保存]** の順に選択します。
7. データフローに「`MyFirstDataflow-Service`」という名前を付けて、**[OK]** をクリックします。
8. **[ファイル] > [エクスポーズ/アンエクスポーズして保存]** の順に選択します。これによってデータフローがエクスポーズされ、サーバー上でサービスとして利用できるようになります。
9. このサービスをテストするには、次の手順を実行します。
   a) Web ブラウザで次の URL に移動することにより、Management Console を開きます。

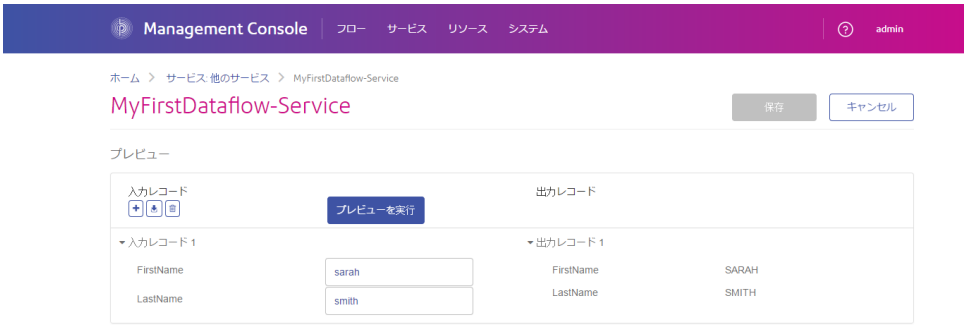
      http://*server*:*port*/managementconsole

      ここで *server* は、 Spectrum™ Technology Platform サーバーの名前または IP アドレスで、 *port* はSpectrum™ Technology Platformが使用する HTTP ポートです。デフォルトの HTTP ポートは 8080 です。

   b) **[サービス] > [他のサービス]** に移動します。

c) サービスのリストで、**MyFirstDataflow-Service** の横にあるチェック ボックスをオンにし、[編集] ボタン をクリックします。
d) [FirstName] フィールドにすべて小文字で名前を入力します。
e) [LastName] フィールドにすべて小文字で名前を入力します。
f) **[プレビューを実行]** をクリックします。

データフローの Transformer ステージで指定したとおり、名前のフィールドがすべて大文字になったことを確認できます。



おめでとうございます。初めてのサービス データフローの設計と実行が完了しました。このサービスはサーバー上で利用できるようになり、API または Web サービス呼び出しを介してアクセス可能です。このサービスの SOAP エンドポイントのリソース URL:

```
http://<ServerName>:<Port>/soap/MyFirstDataflow-Service
```

このサービスの REST エンドポイントのリソース URL:

```
http://<ServerName>:<Port>/rest/MyFirstDataflow-Service
```

データフロー テンプレート

データフロー テンプレートは、Spectrum™ Technology Platform とそのモジュールを使ってビジネス ニーズを満たす方法を具体的に示すものです。これらのテンプレートを参照することで、パーシング、正規化、名前や住所の妥当性検証、住所のジオコーディングなど、さまざまな必須条件を特定のモジュールを利用して満たす方法を理解できます。

データフロー テンプレートは、ライセンスしたモジュールごとに提供されます。例えば、Data Normalization モジュールのライセンスを取得した場合は、姓名の正規化 データフロー テンプレートを受け取ります。Universal Addressing モジュールのライセンスを取得すると、米国の住所とカナダの住所を検証する 2 つのデータフロー テンプレートを受け取ります。

テンプレートは、目的によって、サンプル データ付きのジョブである場合や、サンプル データのないサービスである場合があります。データフローは提供時の状態ですぐに使用することができ、ジョブとして提供されたデータフローを実行してその機能を確かめることができます。または、入力ファイルや出力ファイルを変更したり、サービスを独自のジョブに取り入れて入力ファイルや出力ファイルを追加したりして、データフローを独自に調整することもできます。

> **注：** これらのサンプルは、さまざまな Spectrum™ Technology Platform 機能を具体的に示すことを意図しています。特定のビジネス環境向けに完全なソリューションを提供することは意図していません。

テンプレートを使用したデータフローの作成

データフロー テンプレートは、ライセンスしたモジュールごとに提供されます。テンプレートを使用してデータフローを作成するには

- Enterprise Designer で、**[ファイル] > [新規作成] > [データフロー] > [テンプレートから作成]** を選択します。
- または、**[新規作成]** アイコンをクリックし、[新しいデータフローをテンプレートから作成] を選択します。

インストールされているモジュールで使用可能なテンプレートの一覧が表示されます。

データフローのインポートとエクスポート

インポート機能とエクスポート機能を使用して、他の Enterprise Designer ユーザとデータフローを交換できます。

> **注：** データフローの交換は、同じバージョンの Spectrum™ Technology Platform の間でのみ行うことができます。

- データフローをエクスポートするには、**[ファイル] > [エクスポート]** を選択します。バージョン機能を使用してデータフローのバージョンを保存している場合は、現在選択されているバージョンがエクスポートされるバージョンです。

  > **注：** サービスおよびジョブの名前の定義では、特殊文字を使用しないでください。使用するとエクスポートでエラーが発生する場合があります。

- プロセス フローをインポートするには、**[ファイル] > [インポート] > [プロセス フロー]** を選択します。
- データフローをインポートするには、**[ファイル] > [インポート] > [データフロー]** を選択します。データフローをインポートする前に、データフローのステージがシステムで使用できる状態になっている必要があります。インポートするデータフローに使用できないステージが含まれていると、エラーが発生します。
- サーバエクスプローラーを使用してデータフローをまとめている場合は、データフローを右クリックして **[エクスポート]** を選択しても、データフローをエクスポートできます。サーバエクスプローラーを使用してデータフローをインポートするには、データフローをインポートするサーバエクスプローラー内の場所を右クリックし、**[インポート]** を選択します。

2 - フローの設計

このセクションの構成

フローのタイプ

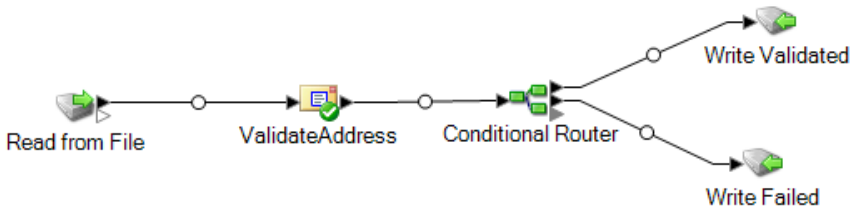
データフローは、何らかのソースからデータを受け取り、そのデータを処理し、何らかのデスティネーションに出力を書き込むという一連の操作です。データは、単純なソートから複雑なデータ品質および強化操作まで、任意の処理を行うことができます。データフローの概念は単純ですが、分岐したパス、複数の入力ソース、複数の出力デスティネーションを備えた非常に複雑なデータフローを設計することができます。

データフローには、ジョブ、サービス、サブフロー、およびプロセス フローの 4 種類があります。

ジョブ

ジョブは、バッチ処理を行うデータフローです。ジョブは、1 つ以上のファイルやデータベースからデータを読み込み、そのデータを処理し、1 つ以上のファイルやデータベースに結果を書き込みます。ジョブは、Enterprise Designer では手動で実行できます。また、Job Executor を使用してコマンド ラインから実行することもできます。

次のデータフローはジョブです。Read from File ステージを入力に使用し、2 つの Write to File ステージを出力として使用しています。

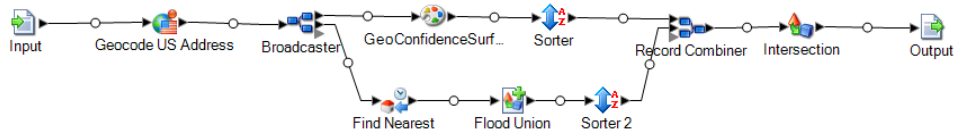


サービス

サービスは、Web サービスとして、またはSpectrum™ Technology PlatformAPI を使用して、アクセスできるデータフローです。レコードをサービスに渡し、レコードを処理する際に使用するオプションを任意で指定できます。サービスはデータを処理し、そのデータを返します。

一部のサービスは、モジュールをインストールしたときに使用可能になります。例えば、Universal Addressing モジュールをインストールすると、お使いのシステム上で ValidateAddress サービスが使用可能になります。Enterprise Designer でサービスを作成し、そのサービスをシステム上でユーザ定義サービスとしてエクスポーズしなければならない場合もあります。例えば、Location Intelligence モジュールのステージは、このモジュールのステージを使用するサービスをまず作成しなければ、サービスとして使用できません。

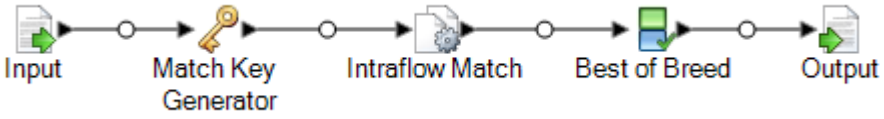
Enterprise Designer で独自のカスタム サービスを設計することもできます。例えば、以下のデータフローは、洪水が発生する危険性のある住所であるかどうかを調べます。



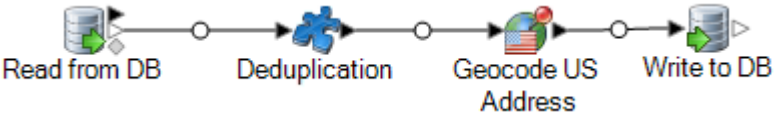
> **注：** サービス名、オプション名、フィールド名は、最終的に XML 要素となるため、XML 要素名として無効な文字を含んではいけません (例えば、空白は無効です)。この要件を満たさないサービスは、サービスとしては機能しますが、Web サービスとしてエクスポーズすることはできません。

サブフロー

サブフローは、他のデータフロー内で再利用可能なデータフローです。サブフローは、データフローに簡単に組み込むことができる、再利用可能なプロセスを作成する場合に便利です。例えば、各ステージで特定の設定を使用して重複除去を実行するサブフローを作成し、複数のデータフローで同じ重複除去プロセスを使用できるようにしたいことがあります。これを実現するには、次のようなサブフローを作成します。



その後、このサブフローをデータフロー内で使用することができます。例えば、ジオコーディングを実行するデータフロー内で重複除去サブフローを使用し、ジオコーディング操作の前にデータを重複除去することができます。



この例では、データはデータベース内から読み込まれ、重複除去サブフローに渡されます。このサブフローで、データは Match Key Generator、Intraflow Match、Best of Breed の順に処理され、最後にサブフローから親データフロー (この場合は Geocode US Address) の次のステージに送信されます。上記のように、サブフローは、データフロー内ではパズルのピース アイコンとして表されます。

Enterprise Designer の **[ユーザ定義ステージ]** フォルダに表示されるサブフローは、保存してエクスポーズできます。

プロセス フロー

プロセス フローでは、ジョブや外部アプリケーションなどの一連のアクティビティが実行されます。プロセス フロー内の各アクティビティは、1 つ前のアクティビティが終了した後に実行されます。プロセス フローは、複数のデータフローを連続して実行したり、外部プログラムを実行したりする場合に便利です。例えば、プロセス フローでは、名前を正規化し、住所を照合してから、郵便料金の割引が受けられるようにレコードを正しい順序にソートするための外部アプリケーションを呼び出すようなジョブを実行することができます。次の図はそのようなプロセス フローを示したものです。

この例では、ジョブ Standardize Names と Validate Addresses は Spectrum™ Technology Platform サーバーでエクスポーズされたジョブです。プログラムの実行は外部アプリケーションを起動し、成功アクティビティはプロセス フローの終了を示します。

フロー入力

データフローの入力を定義するには、"ソース" ステージを使用します。データフロー内の最初のステージ。ソースは、処理する入力データを定義します。

ジョブに対する入力

ジョブに対する入力データのソースはファイルまたはデータベースです。Spectrum™ Technology Platformでは、さまざまなファイル形式やデータベース タイプを入力ソースとして使用できます。使用できる入力データ ソースの種類は、どのモジュールについてライセンスを取得しているかによって異なります。Enterprise Data Integration モジュールでは、任意のモジュールのほとんどのデータ ソースを使用できます。

> **注：** ジョブを設計するときは、形式に誤りのある入力レコードが含まれる可能性を考慮してください。形式に誤りのあるレコードとは、Spectrum™ Technology Platformが提供するパーサー クラスでパースできないレコードです。形式に誤りのあるレコードの処理については、**形式に誤りのある入力レコードの管理**（23ページ）を参照してください。

サービスに対する入力

サービスに対する入力データは、入力ステージで定義されます。このステージでは、サービスが Web サービスや API 呼び出しから受け取るすべてのフィールドを定義します。

ジョブ入力の定義

ジョブの入力データは、ファイル、データベース、またはクラウド サービスから取得されます。どれが使用されるかは、ライセンスしたモジュールによって異なります。各モジュールは、複数のソースからの入力に対応しています。ソースを設定する手順は、そのタイプによって大きく異なります。お使いのモジュールに対応するソリューション ガイドについては、**support.pb.com/spectrum** を参照してください。

形式に誤りのある入力レコードの管理

形式に誤りのあるレコードとは、Spectrum™ Technology Platform がパースできないレコードのことです。Spectrum™ Technology Platform は、形式に誤りのあるレコードを検出すると、次の 1 つ以上の処理を行います。

- ジョブを停止する
- 処理を続行する
- 不正なレコードが一定の数だけ検出されるまで処理を続行する
- 不正なレコードを (オプションのシンク ステージによって) ログ ファイルに書き出して処理を続行する

> **注：** 形式に誤りのあるレコードに対する機能は、サーバー上にあるファイルを読み込み、ソートが設定されていないソースに対してのみ適用されます。ソースにリモート ファイルが設定されているか、ソート フィールドが設定されている場合は、形式に誤りのあるレコードが検出されると、形式に誤りのあるレコードに対する設定に関わらず、ジョブは終了します。

形式に誤りのあるレコードを管理するには

1. Enterprise Designer でジョブを開きます。
2. 形式に誤りのあるレコード シンクをデータフローに追加します。
   a) 入力ファイルおよびソース ステージを定義し、データフローにサービスおよびサブフローを追加することにより、ジョブを作成します。
   b) 以下のいずれかの方法を実行します。
      - データフローのソース ステージのオプションの出力ポートに、シンク ステージを接続します。オプションのポートとは、ソース ステージの黒い出力ポートのすぐ下にある白抜きの出力ポートのことです。このポート上にマウスを合わせると、"error_port" と記されたツール チップが表示されます。形式に誤りのあるレコードは、このシンクに書き出されます。

- データフローのソース ステージのオプションの出力ポートに何も接続しません。この場合、Spectrum™ Technology Platform は形式に誤りのあるレコードを無視します。

完成したデータフローは次のようになります。

ジョブを実行すると、[実行履歴] にはジョブの実行中に検出された形式に誤りのあるレコードの数を示す列が表示されます。

3. デフォルトでは、Spectrum™ Technology Platform は形式に誤りのあるレコードを検出するとジョブを終了します。このデフォルトの動作は、Management Console で変更できます。システムのデフォルトの動作にかかわらず、ジョブのデフォルトの動作を以下の手順でオーバーライドすることができます。
   a) Enterprise Designer でジョブを開きます。
   b) 開いているジョブ内で、**[編集] > [ジョブ オプション]** を選択します。
   c) **[形式に誤りのあるレコードでジョブを停止しない]** を選択するか、または **[形式に誤りのあるレコードを次の個数検出した時点でジョブを停止]** を選択して、ジョブを停止するまでに検出してもよい形式に誤りのあるレコードの数を入力します。

サービス入力の定義

Input ステージは、サービスまたはサブフローの入力フィールドを定義します。また、データ インスペクションで使用するテスト データを定義します。

[入力フィールド] タブ

データフローが入力として受け付けるフィールドが一覧表示されます。Input ステージがデータフローの別のステージに接続されている場合は、データフローのステージによって使用されるフィールドの一覧が表示されます。詳細については、**サービスまたはサブフローの入力フィールドの定義**（25ページ）を参照してください。

[インスペクション入力] タブ

データ インスペクション ツールで使用するテスト入力レコードを指定できます。データ インスペクションの詳細については、**データフローのインスペクション**（147ページ）を参照してください。

サービスまたはサブフローの入力フィールドの定義

サービスまたはサブフローの入力フィールドを定義するには、Input ステージを使用します。

> **注：** 入力フィールドで階層データを定義した場合は、データをインポートしたり、データを垂直に表示したりすることはできません。

1. Input ステージをキャンバス上にドラッグします。
2. Input ステージを、データフローの後続ステージに接続します。
3. Input ステージをダブルクリックします。
4. 入力として使用するフィールドを選択します。表示されるフィールドの一覧は、Input ステージが接続されているステージによって異なります。
5. フィールド リストに新しいフィールドを追加するには、**[追加]** をクリックします。**[カスタム フィールドの追加]** ウィンドウが表示されます。
6. **[追加]** を再度クリックします。
7. **[フィールド名]** フィールドに、このフィールドに対して使用する名前を入力します。
8. データ タイプを選択します。

   次のデータ タイプがサポートされています。

   bigdecimal 小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。

   boolean true と false の 2 つの値を持つ論理タイプ。

   bytearray バイトの配列 (リスト)。

   > **注：** bytearray は REST サービスの入力としてはサポートされていません。

   date 月、日、年を含むデータ タイプ。例: 2012-01-30、January 30, 2012。デフォルトの日付の形式は Management Console で指定できます。

   datetime 月、日、年、時、分、秒を含むデータ タイプ。例: 2012/01/30 6:15 PM。

   double 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

   float 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。

integer 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ 2^{31}-1 (2,147,483,647)。

list 厳密に言えば、リストはデータ タイプではありません。しかし、フィールドが階層データを含む場合、"リスト" フィールドとして扱われます。Spectrum™ Technology Platform では、リストは複数の値で構成されるデータのコレクションです。例えば、Names フィールドには名前の値のリストを含めることができます。これは、XML 構造では次のように表すことができます。

```
<Names>
    <Name>John Smith</Name>
    <Name>Ann Fowler</Name>
</Names>
```

XML のリスト データ タイプが複数の値で構成される単純データ タイプであるのに対し、Spectrum™ Technology Platform のリスト データ タイプは XML の複合データ タイプに似ているという点で、Spectrum™ Technology Platform のリスト データ タイプは XML スキーマのリスト データ タイプと異なることに注意してください。

long 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ 2^{63}-1 (9,223,372,036,854,775,807)。

string 文字シーケンス。

time 時刻を含むデータ タイプ。例: 21:15:59 or 9:15:59 PM。

また、必要に応じて、ユーザ定義の新しいデータ タイプを追加して、定義済みのデータ タイプの一覧に加えることができます。例えば、名前 (文字列) のリストや、AddressLine1 (文字列)、City (文字列)、StateProvince (文字列)、PostalCode (文字列) などを含む住所の新しいデータ タイプを定義できます。フィールドを作成した後は、[入力オプション] ダイアログにアクセスして [データ タイプ] 列のボタンを押すと、そのデータ タイプを表示できます。**[データ タイプ詳細]** ダイアログ ボックスが表示され、フィールドの構造が示されます。

9. **OK** を再度クリックします。
10. **[エクスポーズ]** 列のチェックボックスをオンにすると、フィールドがステージ操作で使用可能になります。このチェック ボックスをオフにして **[OK]** をクリックすると、フィールドがフィールド リストから削除されます。
11. **[データ タイプ名]** フィールドには、このサービスに対する SOAP および REST Web サービス リクエストにおいて、入力レコードに使用するデフォルトの要素名が表示されます。デフォルト値は `Row` です。入力レコードに対して別の要素名を使用する場合は、ここに入力します。

例えば、デフォルト値 `Row` を使用する場合、JSON Web サービス リクエストでは以下に示すように、`Row` を入力レコードの要素名として使用します。

```
{
  "Input":
  {
    "Row": [
      {
        "AddressLine1": "1825 Kramer Ln",
        "City": "Austin",
        "StateProvince": "TX"
      }
    ]

  }
}
```

[データ タイプ名] フィールドの値を "`Address`" に変更する場合は、JSON リクエストでは以下に示すように、`Address` を `Row` の代わりにレコードの要素名として使用する必要があります。

```
{
  "Input":
  {
    "Address": [
      {
        "AddressLine1": "1825 Kramer Ln",
        "City": "Austin",
        "StateProvince": "TX"
      }
    ]

  }
}
```

Web サービス データ タイプの定義

[データ タイプ名] フィールドにより、作成しているサービスの WSDL (SOAP) および WADL (REST) インターフェイスの制御が可能です。Rows 要素の名前は、サービス内のこのステージに付けた名前によって決まり、Row 要素の名前は、ここに入力したテキストによって決まります。

> **注：** WSDL の場合はリクエストとレスポンスの両方に影響しますが、WADL の場合はレスポンスのみに影響が生じます。

このステージに名前を付け、このフィールドにテキストを入力する前のコードの例を、次に示します。

```
<Rows>
     <Row>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
     </Row>
     <Row>
          <FirstName>Jane</FirstName>
    <LastName>Doe></LastName>
     </Row>
</Rows>
```

このステージに名前を付け、このフィールドにテキストを入力した後、コードは次のようになります。

```
<Names>
     <Name>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
     </Name>
     <Name>
          <FirstName>Jane</FirstName>
          <LastName>Doe></LastName>
     </Name>
</Names>
```

フィールド

フラット データと階層データ

Spectrum™ Technology Platform は、フラット データと階層データをサポートしています。一般に、データフローの入力および出力としては、フラット データと階層データのどちらも使うことができます。多くはありませんが、Enterprise Routing モジュールのいくつかのステージで、データを階層的な形式にする必要があります。

フラット データ

フラット データは、1 行に 1 つのレコードと、各レコードのフィールドで構成されます。フィールドは特殊文字で区切るか、行に対して定義された位置に配置します。例えば、カンマ区切りフィールドを持つフラット データを以下に示します。

```
Sam,43,United States
Jeff,32,Canada
Mary,61,Ireland
```

フラット データをデータフローに読み込むには、Read from File、Read from DB、Input のいずれかのステージを使います。フラット データをデータフローから書き出して出力するには、Write to File、Write to DB、Output のいずれかのステージを使います。

階層データ

階層データは、親/子の関連性を持つデータ要素から構成される木のような構造です。Spectrum™ Technology Platform では、階層データの読み書きは、XML および Variable Format File 形式で行えます。例えば、次に示すのは XML 形式の階層データです。

```
<customers>
 <customer>
  <name>Sam</name>
  <age>43</age>
  <country>United States</country>
 </customer>
 <customer>
  <name>Jeff</name>
  <age>32</age>
  <country>Canada</country>
 </customer>
 <customer>
  <name>Mary</name>
  <age>61</age>
  <country>Ireland</country>
 </customer>
</customers>
```

この例は、`<customer>` が 1 つのレコードを表し、各レコードが単純 XML 要素 (`<name>`、`<age>`、および `<country>`) で構成される構造を示しています。

データの変換

多くの場合、フラット データから階層データ、または階層データからフラット データに変換する必要があります。例えば、データ フロー入力が階層形式で、そのデータ フローをフラット データとして出力する場合などです。特定のステージ (特に Location Intelligence モジュールのステージ) 用にフラット入力データを階層データに変換してから、出力用としてデータをフラット データに再変換する必要がある場合などもあります。

フラット データを階層データに変換するには、次の方法があります。

- プロセス リスト ツール
- データフローの Aggregator ステージ

階層データをフラット データに変換するには、Splitter ステージを使います。

フラット データからリストへの変換

プロセス リストは、サービスまたはサブフロー内で、データをフラット化してリストに変換するためのツールです。これは Location Intelligence モジュールなどのように、リスト入力が必要なステージがデータフローに含まれる場合に役立ちます。

1. 既存のデータフローで、出力をリストに変換するステージを右クリックします。Input、Output 以外の任意のステージを使用できます。
2. **プロセス リスト** を選択します。ステージの背景に青い正方形が表示されます。
3. ステージをプロセス リストの中に、またはプロセス リストから外に移動するには、**Shift** キーを押しながらステージをドラッグします。

   注： データをプロセス リストで処理する必要のあるステージが複数ある場合は、サブフローを作成し、それをデータフローに組み込み、プロセス リスト機能をサブフロー全体に適用します。

4. プロセス リストの入力および出力フィールドは "ListField" という名前です。[フィールドの名前変更] 機能を使用して、Input ステージのフィールドを入力チャネルの "ListField" にマップし、"ListField" を Output ステージのフィールドにマップする必要があります。詳細については、フィールド名の変更（44ページ）を参照してください。
5. リストのデータを入力と同じ順序にする場合は、プロセス リスト ボックスを右クリックして、[オプション] を選択します。[ソート順序を維持する] ボックスを選択します。
6. 次のステージへのデータ入力がリストの形式になっていることを確認するには、データフローを検証またはインスペクションします。データのインスペクションの詳細については、データフローのインスペクション（147ページ）を参照してください。

データ タイプ

Spectrum™ Technology Platform では、さまざまな数値、文字列、および複合データ タイプがサポートされています。実行する処理の種類に応じて、これらのデータ タイプを使用できます。住所検証データフローの場合は、文字列データ タイプだけで済ますこともできます。算術計算を含むデータフローの場合は、数値または Boolean データ タイプを使用できます。空間処理を実行するデータフローの場合は、複合データ タイプを使用できます。これらを組み合わせたデータフローの場合は、さまざまなデータ タイプを使用できます。

Spectrum™ Technology Platform では、以下のデータ タイプがサポートされています。

bigdecimal 小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。

boolean true と false の 2 つの値を持つ論理タイプ。

bytearray バイトの配列 (リスト)。

> **注：** bytearray は REST サービスの入力としてはサポートされていません。

date 月、日、年を含むデータ タイプ。例: 2012-01-30、January 30, 2012。デフォルトの日付の形式は Management Console で指定できます。

datetime 月、日、年、時、分、秒を含むデータ タイプ。例: 2012/01/30 6:15 PM。

double 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

float 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。

integer 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ 2^{31}-1 (2,147,483,647)。

list 厳密に言えば、リストはデータ タイプではありません。しかし、フィールドが階層データを含む場合、"リスト" フィールドとして扱われます。Spectrum™ Technology Platform では、リストは複数の値で構成されるデータのコレクションです。例えば、Names フィールドには名前の値のリストを含めることができます。これは、XML 構造では次のように表すことができます。

```
<Names>
    <Name>John Smith</Name>
    <Name>Ann Fowler</Name>
</Names>
```

XML のリスト データ タイプが複数の値で構成される単純データ タイプであるのに対し、Spectrum™ Technology Platform のリスト データ タイプは XML の複合データ タイプに似ているという点で、Spectrum™ Technology Platform のリスト データ タイプは XML スキーマのリスト データ タイプと異なることに注意してください。

long 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ 2^{63}-1 (9,223,372,036,854,775,807)。

string 文字シーケンス。

time 時刻を含むデータ タイプ。例: 21:15:59 or 9:15:59 PM。

フィールドのデータ タイプの指定

次の場合にフィールドのデータ タイプを指定できます。

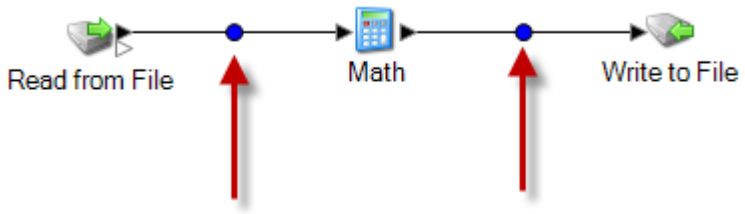
- **ソース ステージ:** データフローの開始時にデータ タイプを設定し、後のデータフローでデータ タイプを変換する必要がないようにできます。Read from DB の場合はデータ タイプは自動的に選択されて変更できないことに注意してください。
- **シンク ステージ:** データ タイプを指定すると、データフローによって返されるデータ フォーマットを制御できます。Write to DB の場合はデータ タイプは自動的に選択されて変更できないことに注意してください。
- **Transformer ステージ:** カスタム スクリプトを使用する場合、このステージでデータ タイプを指定できます。
- **Math ステージと Group Statistics ステージ:** これらのステージでは算術計算が実行されるので、特定の数値データ タイプの使用を選択すると、除算演算の精度など、計算の結果に影響する場合があります。ステージへの入力フィールドのデータ タイプとは異なるデータ タイプをフィールドに対して指定した場合、「**自動データ タイプ変換**（32ページ）」で説明するように、下流のチャネルでフィールドが指定のタイプに自動的に変換されます。

  注： 各ステージは異なるデータ タイプをサポートします。各ステージでサポートされるデータ タイプの詳細については、ステージのドキュメントを参照してください。

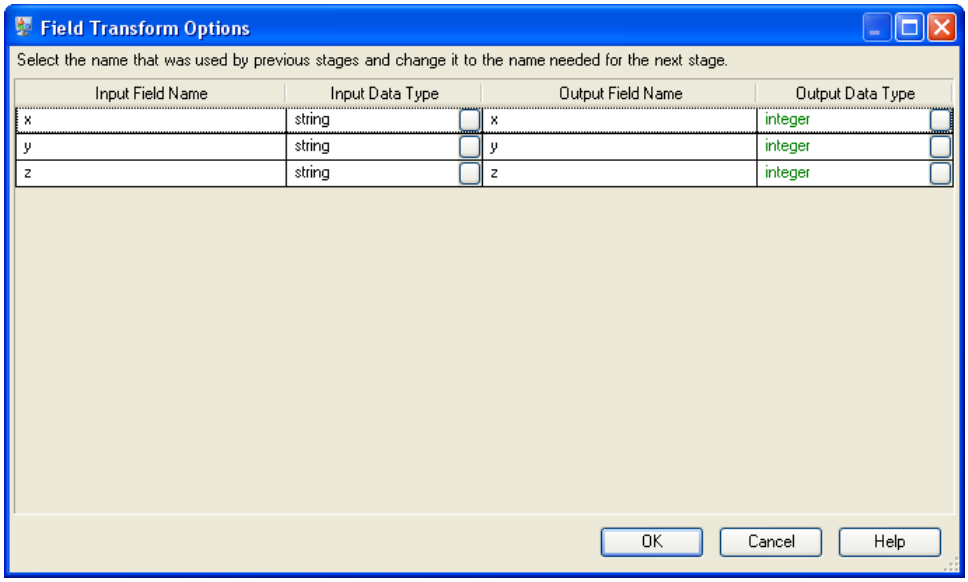
自動データ タイプ変換

ステージに提示されたデータのタイプが適切でないとき、Spectrum™ Technology Platform では、場合によってデータを自動的に適切なタイプに変換できます。例えば、Validate Address は文字列データだけを入力として受け付けます。PostalCode 入力フィールドが integer タイプの場合、Spectrum™ Technology Platform は自動的にフィールドを文字列に変換し、PostalCode フィールドを正しく処理できます。同様に、Math ステージでは、データは数値データ タイプである必要があります。入力データが文字列タイプの場合、Spectrum™ Technology Platform は Math ステージの [フィールド] タブで指定されているデータ タイプにデータを変換できます。

データ タイプの自動変換は、データフローのチャネルで行われます。チャネルでデータ タイプが正しく変換される場合、チャネルの中央に青い点が表示されます。

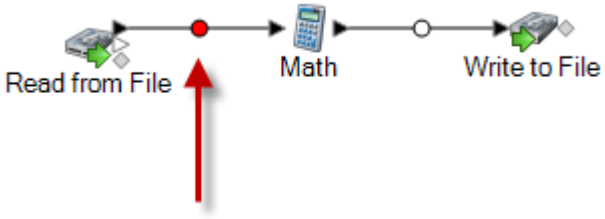


チャネルをダブルクリックすると、行われるデータ タイプ変換を確認できます。次の例の場合、文字列データが整数データに変換されています。



このダイアログ ボックスでは自動データ タイプ変換のデータ タイプを変更できないことに注意してください。出力データ タイプは、下流ステージでの設定によって決まります。

フィールドの値が有効でない場合や、フィールドが変換できない場合は、チャネルに赤い円が表示されます。



タイプ変換オプションを使用することで、タイプ変換に失敗した場合のデータフローの処理を指定できます。

データフローのデータ タイプ変換オプションの設定

データ タイプ変換は、データフローがフィールドを、ステージで必要なデータ タイプに自動的に変換する際に行われます。データ タイプ変換は、一部のステージ内でも行われます。例えば、Read from DB では、ソース データが数値データ タイプであっても、フィールドで文字列データ タイプを使用することができます。データは、データフローに読み込まれる際に文字列データ タイプに変換されます。

データ タイプ変換を制御するために使用できる設定が 2 つあります。1 つは、数値、日付、および時間データを書式設定して文字列に変換する方法を決定する設定です。例えば、文字列に変換される日付データを dd/mm/yyyy ではなく mm/dd/yyyy のフォーマットで表現したい場合があり

ます。もう 1 つは、システムがあるデータ タイプのフィールドを別のデータ タイプに変換できない場合にどうするかを制御する設定です。

システムに対するデフォルトのデータ タイプ変換の設定は、Management Console で指定されています。Enterprise Designer では、デフォルト フォーマットを個別のデータフローについてオーバーライドできます。

この手順は、データフローでデフォルトのデータ タイプ変換オプションをオーバーライドする方法を示しています。

> **注 :** サブフローは、属しているデータフローからタイプ変換設定を継承します。サブフローのタイプ変換設定を指定することはできません。

1. Enterprise Designer でデータフローを開きます。
2. **[編集] > [タイプ変換オプション]** を選択します。
3. **[次の値でシステム デフォルト オプションをオーバーライド]** ボックスをチェックします。
4. **[エラー ハンドリング]** フィールドで、フィールドの値をステージで必要とされるデータ タイプに自動変換できない場合の対処方法を指定します。

データフローの処理を失敗として扱う	フィールドを変換できない場合、データフローは失敗します。
レコードのエラー	フィールドを変換できない場合、レコードは失敗しますが、データフローの実行は続行されます。
フィールドをデフォルト値で初期化する	フィールドを変換できない場合、フィールドの値はここで指定する値に置き換えられます。一部のレコードに不正なデータが含まれることがわかっていて、不正なデータをデフォルト値に置き換える場合、このオプションが役に立ちます。各データ タイプの値を指定します。

5. 文字列に変換する日付および時間データで使用する形式を指定します。日付または時間を文字列に変換する場合、文字列にはここで指定する形式が使用されます。
   a) **[ロケール]** フィールドで、文字列に変換する日付で使用する形式を持つ国を選択します。**[日付]**、**[時間]**、および **[日付/時刻]** フィールドのデフォルト値は、ここでの選択によって決まります。また、月のスペルを表記するときの言語も、ここでの選択によって決まります。例えば、英語を指定した場合、1 年の最初の月は "January" ですが、フランス語を指定した場合は "Janvier" になります。
   b) **[日付]** フィールドで、日付データを文字列に変換するときに使用する形式を指定します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

      例えば、形式として **[M/D/YY]** を選択し、日付フィールドに 2012-3-2 が含まれている場合、日付データは文字列 `3/2/12`。

c) **[時間]** フィールドで、時間データを文字列に変換するときに使用する形式を指定します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

例えば、形式として **[h:mm a]** を選択し、時間フィールドに 23:00 が含まれている場合、時間データは文字列 `11:00 PM`。

d) **[日付/時刻]** フィールドで、DateTime データ タイプを含むフィールドを文字列に変換するときに使用する形式を選択します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

例えば、形式として **[M/d/yy h:mm a]** を選択し、DateTime フィールドに 2012-3-2 23:00 が含まれている場合、DateTime データは文字列 `3/2/12 11:00 PM`。

e) **[整数]** フィールドで、数値型 (Float および Double データ タイプ) で使用する書式設定を選択します。

例えば、形式 **[#,###]** を選択すると、数字 4324 は`4,324`。

> **注 :** このフィールドを空白にしておくと、数字には Spectrum™ Technology Platform 8.0 以降と同じ形式が使用されます。特に、桁区切り文字は使用されません。小数点記号としてドット (".") が使用されます。10^{-3} 未満または 10^{7} 以上の数字は指数表示されます。負の数には先頭にマイナス記号 ("-") が付きます。また、このフィールドを空白にしておくと、BigDecimal データ タイプを使用する数字には常に形式 #,###.000 が使用されます。

f) **[小数]** フィールドで、Decimal (Integer および Long データ タイプ) を含む数字で使用する書式設定を選択します。

例えば、形式 **[#,##0.0#]** を選択すると、数字 4324.25 は`4,324.25`。

> **注 :** このフィールドを空白にしておくと、数字には Spectrum™ Technology Platform 8.0 以降と同じ形式が使用されます。特に、桁区切り文字は使用されません。小数点記号としてドット (".") が使用されます。10^{-3} 未満または 10^{7} 以上の数字は指数表示されます。負の数には先頭にマイナス記号 ("-") が付きます。また、このフィールドを空白にしておくと、BigDecimal データ タイプを使用する数字には常に形式 #,###.000 が使用されます。

また、ニーズを満たす形式がない場合は、独自の日付、時間、および数値形式を指定することもできます。独自の日付または時間形式を指定するには、**日付および時間パターン**（36ページ） に記載される表記を使用して、その形式をフィールドに入力します。独自の数値形式を指定するには、**数字パターン**（39ページ） に記載される表記を使用して、形式をファイルに入力します。

6. **[Null の処理]** で、タイプ変換が必要なフィールドに NULL 値が含まれていた場合の処理を選択します。次のいずれかのオプションを選択すると、**[タイプ変換エラー]** で **[データフローの**

処理を失敗として扱う]と**[レコードの処理を失敗として扱う]**のどちらを選択したかに基づいて、NULL 値を含むデータフローまたはレコードがエラーを返します。

文字列型の Null のエラー	NULL 値を含む文字列フィールドでタイプ変換が必要な場合に、データフローまたはレコードでエラーを返します。
Boolean 型の Null のエラー	NULL 値を含む Boolean フィールドでタイプ変換が必要な場合に、データフローまたはレコードでエラーを返します。
数値型の Null のエラー	NULL 値を含む数値フィールドでタイプ変換が必要な場合に、データフローまたはレコードでエラーを返します。数値フィールドには、double、float、long、integer、および Big Decimal フィールドが含まれます。
日付型の Null のエラー	NULL 値を含む日付フィールドでタイプ変換が必要な場合に、データフローまたはレコードでエラーを返します。これには、date、time、および DateTime フィールドが含まれます。

日付および時間パターン

日付および時間データのデータ タイプ オプションを定義する場合、あらかじめ定義されているオプションでニーズを満たすことができないときは、独自の日付または時間パターンを作成できます。日付または時間パターンを作成するには、次の表に記載される表記法を使用します。例えば、次のパターンを使用します。

dd MMMM yyyy

これは、次のような日付を生成します。

14 December 2012

文字	説明	例
G	年代指示子	AD
yy	年数の 2 桁表記	96
yyyy	年数の 4 桁表記	1996
M	月の数字表記。	7
MM	月の数字表記。数字が 10 未満の場合は、2 桁の数字にするためにゼロが付加されます。	07

文字	説明	例
MMM	月の名前の短縮表記	Jul
MMMM	月の名前の完全表記	July
w	年の週数	27
ww	年の週数の 2 桁表記。週が 10 未満の場合はゼロが付加されます。	06
W	月の週数	2
D	年の日数	189
DDD	年の日数の 3 桁表記。数字が 3 桁未満の場合はゼロが付加されます。	006
d	月の日数	10
dd	月の日数の 2 桁表記。数字が 10 未満の場合はゼロが付加されます。	09
F	週の日数	2
E	曜日の短縮表記	Tue
EEEE	曜日の完全表記	Tuesday
a	AM/PM マーカー	PM
H	1 日の時間。最初の時間は 0 で表記し、最後の時間は 23 で表記	0
HH	1 日の時間の 2 桁表記。最初の時間は 0 で表記し、最後の時間は 23 で表記します。数字が 10 未満の場合はゼロが付加されます。	08
k	1 日の時間。最初の時間は 1 で表記し、最後の時間は 24 で表記	24

文字	説明	例
kk	1 日の時間の 2 桁表記。最初の時間は 1 で表記し、最後の時間は 24 で表記します。数字が 10 未満の場合はゼロが付加されます。	02
k	午前中 (AM) または午後 (PM) の時間。最初の時間は 0 で表記し、最後の時間は 11 で表記	0
KK	1 日の時間の 2 桁表記。最初の時間は 1 で表記し、最後の時間は 24 で表記します。数字が 10 未満の場合はゼロが付加されます。	02
h	午前中 (AM) または午後 (PM) の時間。最初の時間は 1 で表記し、最後の時間は 12 で表記します。	12
hh	午前中 (AM) または午後 (PM) の時間の 2 桁表記。最初の時間は 1 で表記し、最後の時間は 12 で表記します。数字が 10 未満の場合はゼロが付加されます。	09
m	1 時間の分数	30
mm	1 時間の分数の 2 桁表記。数字が 10 未満の場合はゼロが付加されます。	05
s	1 分の秒数	55
ss	1 分の秒数の 2 桁表記。数字が 10 未満の場合はゼロが付加されます。	02
S	1 秒のミリ秒数	978
SSS	1 秒のミリ秒数の 3 桁表記。数字が 3 桁未満の場合は、3 桁にするために 1 つまたは 2 つのゼロが付加されます。	978 078 008
z	時間帯の名前の省略形。時間帯に名前がない場合は GMT オフセットを使用	PST GMT-08:00

文字	説明	例
zzzz	時間帯の名前の完全表記。時間帯に名前がない場合はGMTオフセットを使用	太平洋標準時 GMT-08:00
Z	RFC 822 時間帯	-0800
X	ISO 8601 時間帯	-08Z
XX	分を付加した ISO 8601 時間帯	-0800Z
XXX	分、およびコロン区切り文字を時間と分の間に付加した ISO 8601 時間帯	-08:00Z

数字パターン

数値データのデータ タイプ オプションを定義する場合、あらかじめ定義されているオプションでニーズを満たすことができないときは、独自の数字パターンを作成できます。基本的な数字パターンは、次の要素で構成されます。

- 通貨記号などの接頭辞 (オプション)
- オプションのグループ文字を含む数字のパターン (例えば、桁区切り文字として使用するカンマ)
- 接尾辞 (オプション)

例えば、次のパターンを使用します。

$ ###,###.00

次のような形式の数字が生成されます (最初の 3 桁の数字の後に桁区切り文字を使用していることに注意)。

$232,998.60

負の数のパターン

デフォルトでは、負の数は、正の数と同じ形式で表記されますが、数字の先頭に負記号が追加されます。数字記号に使用される文字はロケールに基づきます。マイナス記号 "-" は、ほとんどのロケールで使用されます。例えば、次の数字パターンを指定するとします。

```
0.00
```

マイナス 10 は、ほとんどのロケールで次の形式で表記されます。

```
-10.00
```

ただし、負の数に使用する別の接頭辞または接尾辞を定義する場合は、2 つ目のパターンを指定し、そのパターンと 1 つ目のパターンをセミコロン (";") で区切ります。例:

```
0.00;(0.00)
```

このパターンでは、負の数は、次のように括弧で囲んで表記されます。

```
(10.00)
```

指数表記

数字を指数表示する場合は、文字 `E` の後に続けて、指数に含める最小桁数を指定します。例えば、次のパターンを使用するとします。

```
0.###E0
```

数字 `1234` は、次のような形式で表記されます。

```
1.234E3
```

つまり、これは 1.234 x 10^3 を表します。

次のことに注意してください。

- 指数文字の後に続く桁数は、指数の最小桁数を表します。最大桁数はありません。
- 負の指数は、このパターンの接頭辞や接尾辞ではなく、ローカライズされた負の記号を使用した形式で表記されます。
- 指数表記パターンにグループ区切り文字 (桁区切り文字など) を含めることはできません。

特殊な数字パターン文字

次の文字は、その文字自体では別の文字を表しますが、実際には結果の数字で再現されます。次の特殊文字を数字パターンの接頭辞または接尾辞のリテラル文字として使用する場合は、特殊文字を引用符で囲みます。

記号	説明

`0`	必要な位置にゼロを含むパターン内の桁を表します。例えば、数字 27 に次のパターンを適用するとします。
	`0000`
	次のように表されます。
	`0027`

記号	説明

#	桁を表しますが、ゼロは省略されます。例えば、数字 27 に次のパターンを適用するとします。
	`####`
	次のように表されます。
	`27`

.	選択したロケールで使用される小数点記号または通貨区切り文字。例えば、米国の場合、小数点記号にはドット (.) が使用されますが、フランスの場合、小数点記号にはカンマ (,) が使用されます。
-	選択したロケールで使用される負の記号。ほとんどのロケールでは、マイナス記号 (-) が使用されます。

,	選択されたロケールで使用されるグループ文字。選択されたロケールの適切な文字が使用されます。例えば、米国の場合、区切り文字にはカンマ (,) が使用されます。
	一般に、グループ区切り文字は千の位を区切るのに使用されますが、一部の国では一万の位を区切るのに使用されます。グループのサイズは、グループ区切り文字間の桁数を指定する定数です。例えば、3 は 100,000,000、4 は 1,0000,0000 になります。パターンに複数のグループ区切り文字を指定すると、最後のグループ区切り文字と数値の末尾との間隔が、使用される間隔です。例えば、次のパターンはすべて同じ結果を生成します。
	`#,##,###,####`
	`######,####`
	`##,####,####`

E	指数表記の仮数と指数を区切ります。パターン内で E を引用符で囲む必要はありません。**指数表記**（40ページ） を参照してください。
;	正のサブパターンと負のサブパターンを区切ります。**負の数のパターン**（39ページ） を参照してください。

記号	説明

%	数字を 100 で乗算し、その数字をパーセンテージで表示します。例えば、数字 .35 に次のパターンを適用するとします。
	`##%`
	次のような結果が生成されます。
	`35%`

¤	選択したロケールの通貨記号。double の場合、国際通貨記号が使用されます。パターンで表す場合、小数点記号ではなく、通貨区切り文字が使用されます。

'	接頭辞または接尾辞の特殊文字を引用符で囲むのに使用されます。例:
	`"'#'#"`
	123 は次のように表記されます。
	`"#123"`
	単一引用符自体を作成するには、2 つの単一引用符を続けて使用します。
	`"# o''clock"`

フィールドのデータ タイプの指定

Spectrum™ Technology Platform は、Management Console で指定されたタイプ変換オプションの設定、または Enterprise Designer で指定されたデータフローのタイプ変換オプションを使用して、フィールドのデータ タイプを必要に応じて自動的に変更します。ほとんどの場合、フィールドのデータ タイプを手動で変更する必要はありません。必要なデータ タイプの変換はすべて自動的に行われます。ただし、ステージで入力データを必要なデータ タイプに変換できない場合は、データ タイプを上流チャネルで手動で変更することが必要となることもあります。

手動で実行できるタイプ変換はわずかです。以下に、そのいくつかを示します。

- Polygon および MultiPolygon タイプは、ジオメトリ タイプに変換することも、ジオメトリ タイプから変換することもできます。
- Date、Time、および DateTime データ タイプは、String タイプに変換することも、String タイプから変換することもできます。

フィールドのデータ タイプを手動で変更するには、次の手順に従います。

1. Enterprise Designer で、フィールドのデータ タイプを変更するチャネルをダブルクリックします。チャネルとは、キャンバス上の 2 つのステージを接続する線です。
2. 変更するデータ タイプの横にある小さな正方形ボタンをクリックします。

注 : データ タイプの横に小さな正方形ボタンが表示されない場合、現在の状況ではデータ タイプを手動で変換できません。

3. Date、Time、および DateTime データ タイプの場合、次の操作を行います。

   注 : 選択されたデータ タイプに基づいて、適切なオプションのみが表示されます。

   a) **[ロケール]** フィールドで、文字列に変換する日付で使用する形式を持つ国を選択します。**[日付]**、**[時間]**、および **[日付/時刻]** フィールドのデフォルト値は、ここでの選択によって決まります。また、月のスペルを表記するときの言語も、ここでの選択によって決まります。例えば、英語を指定した場合、1 年の最初の月は "January" ですが、フランス語を指定した場合は "Janvier" になります。

   b) **[日付]** フィールドで、日付データを文字列に変換するときに使用する形式を指定します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

   例えば、形式として **[M/D/YY]** を選択し、日付フィールドに 2012-3-2 が含まれている場合、日付データは文字列 `3/2/12`。

   c) **[時間]** フィールドで、時間データを文字列に変換するときに使用する形式を指定します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

   例えば、形式として **[h:mm a]** を選択し、時間フィールドに 23:00 が含まれている場合、時間データは文字列 `11:00 PM`。

   d) **[日付/時刻]** フィールドで、DateTime データ タイプを含むフィールドを文字列に変換するときに使用する形式を選択します。選択したロケールで最も一般的に使用される形式の一覧が表示されます。

   例えば、形式として **[M/d/yy h:mm a]** を選択し、DateTime フィールドに 2012-3-2 23:00 が含まれている場合、DateTime データは文字列 `3/2/12 11:00 PM`。

   e) **[整数]** フィールドで、数値型 (Float および Double データ タイプ) で使用する書式設定を選択します。

   例えば、形式 **[#,###]** を選択すると、数字 4324 は`4,324`。

   注 : このフィールドを空白にしておくと、数字には Spectrum™ Technology Platform 8.0 以降と同じ形式が使用されます。特に、桁区切り文字は使用されません。小数点記号としてドット (".") が使用されます。10^{-3} 未満または 10^{7} 以上の数字は指数表示されます。負の数には先頭にマイナス記号 ("-") が付きます。また、このフィールドを空白にしておくと、BigDecimal データ タイプを使用する数字には常に形式 #,###.000 が使用されます。

   f) **[小数]** フィールドで、Decimal (Integer および Long データ タイプ) を含む数字で使用する書式設定を選択します。

例えば、形式 **[#,##0.0#]** を選択すると、数字 4324.25 は`4,324.25`。

注 : このフィールドを空白にしておくと、数字には Spectrum™ Technology Platform 8.0 以降と同じ形式が使用されます。特に、桁区切り文字は使用されません。小数点記号としてドット (".") が使用されます。10^{-3} 未満または 10^{7} 以上の数字は指数表示されます。負の数には先頭にマイナス記号 ("-") が付きます。また、このフィールドを空白にしておくと、BigDecimal データ タイプを使用する数字には常に形式 #,###.000 が使用されます。

4. **[OK]** をクリックします。

   データ タイプ名の色が緑に変わります。

5. **[OK]** を再度クリックして、変更を保存します。

フィールド名の変更

さまざまな状況で、データフロー内のフィールドの名前を変更しなければならないことがあります。例:

- あるステージの入力で必要とされるフィールドの名前が、前のステージの出力で使われているフィールドの名前と異なる場合
- 下流のステージで同じ名前のフィールドにデータが出力されるとき、フィールド内のデータを保持したい場合

注 : フィールド名を変更すると、以降のステージでは古い名前を使用できなくなります。

1. データフローで、2 つのステージの間のチャネルをダブルクリックします。**[Field Transform オプション]** ダイアログ ボックスが表示されます。
2. 必要に応じてフィールド名を変更します。

   例えば、後のステージでは "AddressLine3" というフィールド名が必要で、前のステージでは "FirmName" というフィールド名が使用されているものとします。この場合、[出力フィールド名] として AddressLine3 に対応する [入力フィールド名] のドロップダウン矢印をクリックして、"FirmName" を選択します。

   出力フィールド名の色が緑に変わります。

3. **[OK]** をクリックします。

予約済みフィールド名

以下のフィールド名はシステムによって使用されているため、データフローでは使用できません。

Status
Status.Code
Status.Description

制御ステージ

制御ステージは、フロー内での異なるパスへのデータ移動、レコードの分割またはグループ化、基本データ変換の実行、および算術演算に使用します。

Aggregator

Aggregator は、フラット データを階層データに変換します。Aggregator は、単一のソースから入力データを取得し、指定したフィールドに基づいてデータをグループ化することによりスキーマ (データの構造的階層) を作成し、そのグループをスキーマの中に作成します。

> **注：** データの中に、それによってデータをグループ化するフィールド (ID フィールドなど) がある場合は、Aggregator に引き渡す前にデータをソートする必要があります。データのソートは、データをデータフローに引き渡す前にソートしておくか、Enterprise Designer 内で入力ファイルをソートするか (ジョブまたはサブフローの場合、サービスには適用不可)、データフローに Sorter ステージを追加することによって行うことができます (ジョブ、サービス、サブフローの場合)。

グループ化方法

ツリーで **[グループ化]** を選択してから **[追加]** をクリックして、階層への集約のベースとして使用するフィールドを選択します。選択したフィールドに同じ値を持つレコードのデータは、単一の階層に集約されます。複数のフィールドを選択した場合は、レコードを 1 つの階層にグループ化するには、すべてのフィールドのデータが一致する必要があります。

例えば、アカウント番号でデータをグループ化する場合は、アカウント番号フィールドを選択します。アカウント番号フィールドに同じ値を持つすべての入力レコードのデータが、単一の階層レコードにグループ化されます。

注： フィールドの一覧から選択できるようにするには、ステージを Aggregator の入力ポートに接続する必要があります。

出力リスト

[出力リスト] で選択したフィールドによって、Aggregator が作成する各レコードに含めるフィールドが決まります。フィールドを追加するには、**[出力リスト]** を選択し、**[追加]** をクリックして、以下のオプションのいずれかを選択します。

既存フィールド	フィールドをデータフローから階層に追加する場合は、このオプションを選択します。
新しいデータ タイプ	子フィールドを追加できる親フィールドを作成する場合は、このオプションを選択します。
テンプレート	このオプションにより、Aggregator の出力ポートに接続されているステージのデータに基づくフィールドを追加できます。

フィールドに子フィールドを追加する場合は、**[リスト]** ボックスをオンにします。

[名前] テキスト ボックスにフィールドの名前を入力するか、自動的に表示された名前でよい場合はそのままにします。Aggregator ステージでは、フィールド名に無効な XML 文字を使用できないことに注意してください。使用できるのは、英数字、ピリオド (.)、アンダースコア (_)、およびハイフン (-) です。

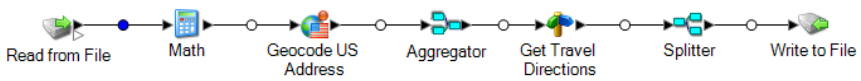
[追加] をクリックしてフィールドを追加します。別のフィールドを指定して階層の同じレベルに追加するか、**[閉じる]** をクリックします。

子フィールドを既存のフィールドに追加するには、親フィールドを選択し、**[追加]** をクリックします。

注： フィールド グループを変更するには、行をハイライト表示して、**[変更]** をクリックします。フィールド グループを削除するには、行をハイライト表示して、**[削除]** をクリックします。また、フィールドを選択してから **[上へ移動]** または **[下へ移動]** をクリックすることによって、フィールドの順序を変更できます。

Aggregator の例

Aggregator の機能が適用できる例としては、ストリートの住所のグループから道順を得る場合が挙げられます。始点と終点という 2 つの地点、またはルート内の複数の通過点に対し、この機能を適用できます。このような種類の機能を実行するデータフローの例を次に示します。

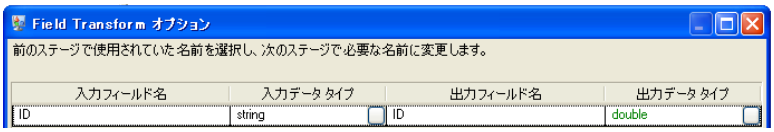


このデータフローは、次のように機能を実行します。

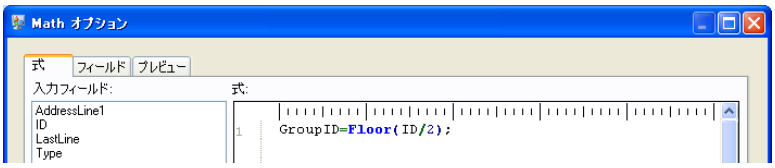
1. **Read from File** ステージでは、ストリートの住所がフラット ファイルに格納されています。このファイルには、次のようなフィールドがあります。

   - **ID**: ファイル内の住所を一意に識別するためのフィールドです。
   - **Type**: 住所が差出住所と宛先住所のどちらであるかを表します。
   - **AddressLine1**: ストリートの住所を格納します。
   - **LastLine**: 都市、州、および郵便番号などの情報を格納します。

2. Read from File ステージと Math ステージの間の **Field Transform** は、ID フィールドのフォーマットを string タイプから double タイプに変換します。Math ステージが、string タイプのデータを入力として受け取ることができないためです。



3. **Math** ステージは、データフローの下流で使用するグループ ID (GroupID) フィールドを計算する式を作成します。この例では、ID フィールドの値を 2 で割った数以下の最大の整数、つまり小数点以下を切り捨てた数を、グループ ID とします。つまり、ID が 3 の場合、式は 3/2 となり、解は 1.5 となります。1.5 の小数点以下を切り捨てて、グループ ID は 1 となります。ID が 2 の場合は、式は 2/2、解は 1 となり、小数点以下を切り捨てる必要はありません。したがって、ID が 2 の場合も 3 の場合もグループ ID は同じ 1 になります。

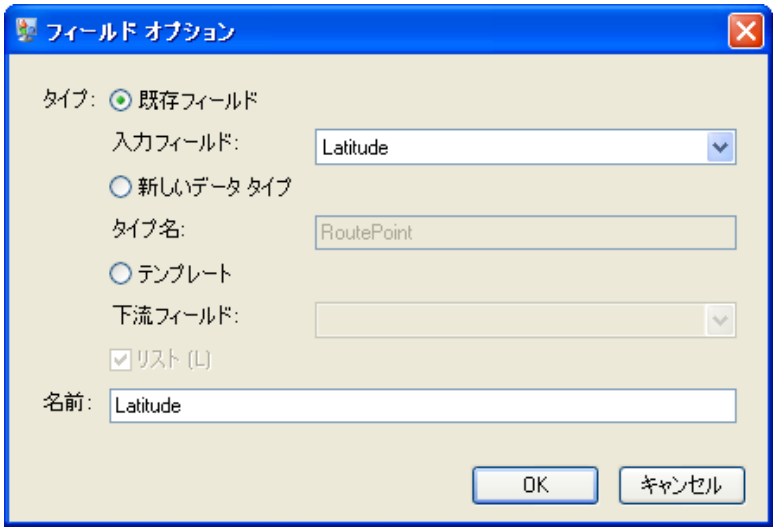


4. **Geocode US Address** は、各住所の緯度と経度を取得します。
5. **Aggregator** ステージは、データを GroupID フィールドによってグループ化し、出力リストに緯度と経度に基づくルート ポイント (RoutePoints) が含まれるようにします。

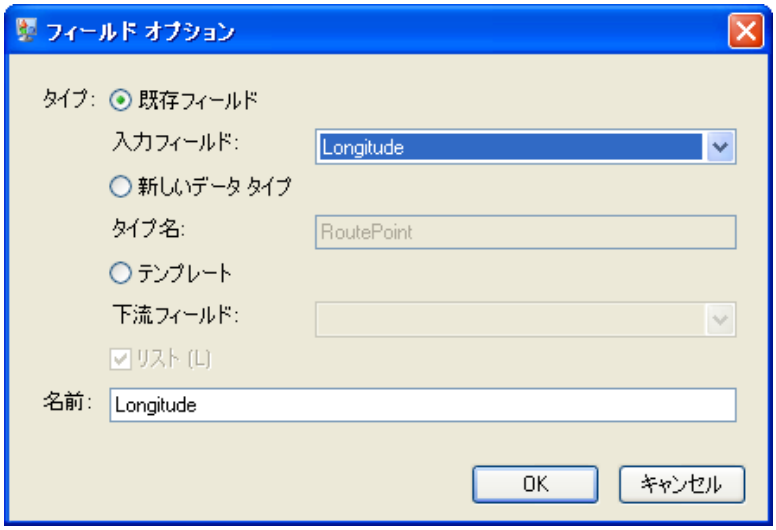
   以下に、このデータフロー用の Aggregator ステージを**手動**で設定する方法を示します。

   - [Aggregator] ステージをダブルクリックし、**[グループ化]** をダブルクリックします。
   - **[GroupID]** フィールドを選択し、**[OK]** をクリックします。このフィールドを使用することによって、データフローにおける次のステージのためのルート ポイントを含めることができるようになります。ルート ポイントは、データフローで道順を生成するために必須です。

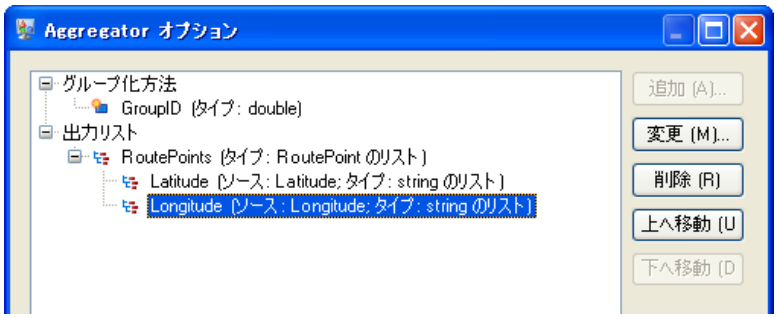
- **[出力リスト]** をダブルクリックします。**[フィールド オプション]** ダイアログ ボックスが表示されます。
- **[新しいデータ タイプ]** を選択します。**[タイプ名]** フィールドで、`RoutePoint` と入力します。**[名前]** フィールドで、`RoutePoints` と入力します。デフォルトで、これはリストであり、変更できません。したがって、チェック ボックスは無効になっています。
- **[OK]** をクリックします。
- **[RoutePoints]** をクリックし、**[追加]** をクリックします。**[フィールド オプション]** ダイアログ ボックスが再度表示されます。
- ルート ポイントは、緯度と経度で構成されているため、まず **[既存フィールド]** で、既存の入力フィールド "`Latitude`" を追加する必要があります。**[名前]** フィールドの値は自動的に入力されます。



"Longitude" に対して、このステップを繰り返します。



完成した Aggregator ステージは、次のように表示されます。



6. **Get Travel Directions** は、ポイント ID 0、2、4 からポイント ID 1、3、5 までの道順をそれぞれ提供します。
7. **Splitter** ステージでは、データを RouteDirections フィールドで分割し、出力リストに Get Travel Directions ステージから使用可能なフィールドをすべて取り込む必要があります。
8. **Write to File** ステージでは、道順が出力ファイルに書き出されます。

Broadcaster

Broadcaster は、レコードのストリームを取得して複数のストリームに分割するので、複数のレコードを複数のステージに送信して同時に処理できます。

Broadcaster には変更する設定はありません。

Conditional Router

Conditional Router ステージは、指定された条件に基づいて、レコードをデータフローの各種パスに送信します。このステージは、定義された条件に応じて、1 つ以上の出力ポートを使用できます。出力ポートには、1 から順に番号が付けられます (以下 "ポート" と表記)。

出力ポートはさまざまなステージに接続され、定義された条件に基づいてデータがどのステージに送信されるかが決まります。例えば、マッチしたあるレコード セットをポート 1 に送信し、マッチしなかった別のレコード セットをポート 2 に送信することができます。

式全体が真と評価された場合のみ、入力レコードは Conditional Router ステージの出力ポートに書き出されます。

Conditional Router の設定

1. **[制御ステージ]** の下で **[Conditional Router]** をクリックして、キャンバスまでドラッグし、データフロー内の適切な位置に配置します。
2. Router をキャンバス上の他のステージに接続します。

   注： ポート設定を定義する前に、これを必ず行う必要があります。これを行わなければ、ポートが編集可能になりません。

3. キャンバス上の **[Conditional Router]** ステージをダブルクリックします。
   [Conditional Router オプション] ウィンドウが表示されます。
4. **[ポート]** 行に対する **[条件/式]** 列の正方形ボタンをクリックします。
   [Expression Editor] ウィンドウが表示されます。
5. **[式タイプの選択]** セクションで、次のいずれかを選択します。
   - **Expression Builder で作成した式**: このオプションは、基本式を作成する場合に選択します。基本式には、グループや式を追加でき、グループや式は、さまざまな論理演算子を用いて組み合わせることもできます。詳細については、**Expression Builder の使用**（51ページ）を参照してください。
   - **カスタム式**: このオプションは、Groovy スクリプト言語を使用して式を記述する場合に選択します。詳細については、**カスタム式の記述**（54ページ）を参照してください。
   - **デフォルト式**: このオプションは、レコードをデフォルトでこのポートに送信する場合に選択します。他のポートのどの式ともマッチしないレコードは、このポートに送信されます。"デフォルト"式を持つ出力ポートを必ず用意して、どのポートにも一致しない行が生じることなく、すべての行が Router から書き出されるようにします。
6. **[OK]** をクリックします。**[Expression Editor]** ウィンドウが閉じます。
7. **[Conditional Router オプション]** ウィンドウで **[OK]** をクリックします。

Expression Builder の使用

Conditional Router ステージの **Expression Builder** で式を作成し、それが真と評価された場合のみ入力レコードをステージの出力ポートに送信することができます。

1. 各親グループは、子の式と子グループの適切な条件付き組み合わせで構成されます。
2. 各式は、左オペランド、右オペランド、論理演算子で構成されます。
3. 各グループでは、構成条件のすべてが成立する場合にグループ全体を真と評価するか、それともいずれかが成立すればグループ全体を真と評価するかを指定する必要があります。

Expression Builder を使用して式を作成するには

1. **[Expression Editor]** で、**[Expression Builder で作成した式]** オプションを選択します。

   デフォルトで Expression Builder のオプションは選択されており、**[Expression Builder]** セクションの左側にある式階層ツリーに親グループが表示されます。

2. 選択されているグループに子グループを追加するには、**[グループの追加]** をクリックします。

   新しく追加されたグループは、親グループの子として追加され、ツリー内においてデフォルトで選択状態になります。各グループ内で、子の式や子グループを追加できます。

3. 各グループに対し、**[式の結合方法]** ヘッダの下で **[すべて真]** または **[いずれかが真]** を選択します。

   - [すべて真]: グループのすべての子条件が真である場合のみ、グループは真と評価されます。
   - [いずれかが真]: 子条件が 1 つでも真であれば、グループは真と評価されます。

4. 選択されているグループに子の式を追加するには、**[式の追加]** をクリックします。

   新しく追加された式は、親グループの子として追加され、ツリー内においてデフォルトで選択状態になります。

   この子の式を定義するには

   a) **[フィールド]** ドロップダウンを使用して入力ファイルのいずれか1つの列を選択することにより、選択式の左オペランドを指定します。
   b) **[演算子]** フィールドから適切な演算子を選択することにより、選択式の2つのコンポーネントを接続する論理演算子を指定します。演算子は以下のとおりです。

表 1 : Expression Builder の演算子

演算子	説明
等しい	フィールドの値が指定された値またはフィールドとマッチするかどうかを確認します。

演算子	説明
が次の値と異なる	フィールドの値が指定された値またはフィールドとマッチしないかどうかを確認します。
が NULL である	フィールドが NULL 値かどうかを確認します。
が NULL でない	フィールドが NULL 値でないかどうかを確認します。
空	フィールドが NULL かどうか、または長さ 0 の文字列かどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
空でない	フィールドが NULL でないかどうか、または長さ 0 の文字列でないかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
が次の値より小さい	フィールドの数値が指定された値未満かどうかを確認します。この演算子は、数値データ タイプおよび数字を含む文字列フィールドに対してのみ適用できます。 **注：** この演算子は、Boolean データ タイプのフィールドには使用できません。
が次の値以下	フィールドの数値が指定された値以下かどうかを確認します。この演算子は、数値データ タイプおよび数字を含む文字列フィールドに対してのみ適用できます。 **注：** この演算子は、Boolean データ タイプのフィールドには使用できません。
が次の値より大きい	フィールドの数値が指定された値よりも大きいかどうかを確認します。この演算子は、数値データ タイプおよび数字を含む文字列フィールドに対してのみ適用できます。 **注：** この演算子は、Boolean データ タイプのフィールドには使用できません。

演算子	説明
が次の値以上	フィールドの数値が指定された値以上かどうかを確認します。この演算子は、数値データ タイプおよび数字を含む文字列フィールドに対してのみ適用できます。 **注：** この演算子は、Boolean データ タイプのフィールドには使用できません。
が次で始まる	フィールドが指定された文字で始まっているかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
が次で始まらない	フィールドが指定された文字で始まっていないかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
含む	フィールドに指定された文字列が含まれているかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
が次を含まない	フィールドに指定された文字列が含まれていないかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
が次で終わる	フィールドが指定された文字で終了しているかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。
が次で終わらない	フィールドが指定された文字で終了しているかどうかを確認します。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。

演算子	説明
正規表現と一致	フィールドを、対象テキストの文字列 (特定の文字、単語、文字のパターン等) を識別する正規表現と照合します。値フィールドには、有効な正規表現パターンが含まれる必要があります。 **注：** この演算子は、文字列データ タイプのフィールドにのみ使用できます。

c) [値] または [フィールド] を選択することにより、選択式の右オペランドを指定します。

- [値]: 選択式の左オペランドが、この値と比較されます。
- [フィールド]: 選択式の左オペランドが、同じ入力ファイルのこの列と比較されます。右オペランドの列をドロップダウンから選択します。

5. 任意のエンティティに兄弟となる式またはグループを追加するには、ツリー内でそのエンティティを選択して **[式の追加]** または **[グループの追加]** をそれぞれクリックします。
6. 子の式または子グループを、ある親グループから別の親グループに移すには、左側の条件ツリー内で目的の親グループのヘッダまでそれをドラッグします。
7. 上記の手順を繰り返して、子の式や子グループを必要なだけ追加し、所望の式条件を完成させます。
8. **[OK]** をクリックします。

[Conditional Router オプション] ウィンドウの **[条件/式]** 列に、定義した式条件が表示されます。レコードがステージの対応する出力ポートに書き出されるためには、この条件が真と評価される必要があります。

カスタム式の記述

Groovy スクリプト言語を使用して式を作成することにより、Conditional Router によるレコードのルーティング方法を制御する独自のカスタム式を記述することができます 。

Groovy スクリプトの使用

Groovy の詳細については、**groovy-lang.org** を参照してください。

Conditional Router ステージで使用される Groovy 式は、レコードをポートに書き出すべきかどうかを示す Boolean 値 (真または偽) になる必要があります。レコードは、式が真として評価された最初の出力ポートに送信されます。

例えば、バリデーションの確信レベルが 85 以上のレコードをあるステージに送信し、バリデーションの確信レベルが 85 未満のレコードを別のステージに送信する必要がある場合、スクリプトは次のようになります。

```
data['Confidence']>=85
```

他方のポートのスクリプトは次のようになります。

```
data['Confidence']<85
```

ルータは、Confidence フィールドの値を条件と照合して、レコードの送信先出力ポートを決定します。

フィールドに単一の値があるかどうかを確認する

次の式は、Status フィールドの値が 'F' の場合に真として評価されます。完全一致が要求されるため、フィールドの値が 'f' の場合は真として評価されません。

```
return data['Status'] == 'F';
```

フィールドに複数の値があるかどうかを確認する

次の式は、Status フィールドの値が 'F' または 'f' の場合に真として評価されます。

```
boolean returnValue = false;
if (data['Status'] == 'F' || data['Status'] == 'f')
{
 returnValue = true;
}
return returnValue;
```

フィールドの長さを評価する

次の式は、PostalCode フィールドの文字数が 6 文字以上の場合に真として評価されます。

```
return data['PostalCode'].length() > 5;
```

フィールド値に含まれる文字を確認する

次の式は、PostalCode フィールドにダッシュが含まれている場合に真として評価されます。

```
boolean returnValue = false;
if (data['PostalCode'].indexOf('-') != -1)
{
 returnValue = true;
}
return returnValue;
```

スクリプトのガイドライン

1. 列名は、一重引用符または二重引用符で囲む必要があります。

   例えば、次の構文は誤りです。列名 `PostalCode`が一重引用符または二重引用符で囲まれていないためです。

   ```
   return data[PostalCode];
   ```

2. 列名を指定する必要があります。

   例えば、次の構文は誤りです。列が指定されていないためです。

   ```
   return data[];
   ```

3. `return`文は、`boolean` 値を返す必要があります。

   例えば、次のスクリプトは誤りです。`row.set('PostalCode', '88989')`が`boolean` 値を返さないためです。これはただ、`PostalCode`フィールドに値 `88989` を設定しているだけです。

   ```
   return row.set('PostalCode', '88989');
   ```

4. フィールドの値の設定には等号 (=) を使用し、フィールドの値の確認には二重等号 (==) を使用します。

Group Statistics

Group Statistics ステージでは、分析対象のグループに分割された複数のデータ行に対し、統計操作を実行できます。グループが定義されていない場合、すべての行が 1 つのグループに含まれるものとして処理されます。

グループは、複数のデータ行にわたって同じ値を持つ 1 つまたは複数のフィールドによって定義されます。

例えば、次の表のデータは、地域、州、またはその両方でグループ化できます。

地域	State
東	MD
東	MD
東	CT
西	CA
西	CA

地域によってグループ化する場合は、東部および西部に分類されます。州によってグループ化する場合は、カリフォルニア州、コネチカット州、メリーランド州に分類されます。地域および州によってグループ化する場合は、東部/メリーランド州、東部/コネチカット州、西部/カリフォルニア州に分類されます。

入力

Group Statistics ステージは、任意のフィールドを入力として受け取ります。グループ化は、数値または文字列データに対して行うことができます。

オプション

表 2 : [操作] タブ

オプション	説明
入力フィールド	レコードのグループ化や計算に使えるデータフロー内のフィールドを表示します。
行	計算のカテゴリとして使いたい 1 つまたは複数のフィールドを指定します。例えば、手持ちのデータに [地域] フィールドが含まれていて、地域別の総人口を計算したいようなときは、[地域] フィールドでグループ化を行います。 フィールドを追加するには、**[入力フィールド]** リスト内で目的のフィールドを選択し、**[>>]** をクリックします。

オプション	説明
列	これはオプションです。ピボット テーブルを作成する場合、クロス集計用の列の値として取り込むフィールドを指定します。 フィールドを追加するには、**[入力フィールド]** リスト内で目的のフィールドを選択し、**[>>]** をクリックします。 例えば、地域と出荷日を含むデータがあり、各州の 1 日あたりの出荷件数を集計したい場合は、州フィールドを行、出荷日フィールドを列として指定します。
行と列は設定された順序でソート済み	入力データが既にソート済みであることを示します。 このチェックボックスがオンの場合、ステージにおいてデータはソートされることなく、指定された操作が入力データに対して直接実行されます。
操作	各グループに対して実行する計算を指定します。操作を追加するには、**[入力フィールド]** リスト内で操作に使いたいフィールドを選択し、**[>>]** をクリックします。 サポートされる Group Statistics 操作の詳細については、**操作**（59ページ）を参照してください。
タイプ	入力フィールドと出力フィールドについて、データ タイプを指定します。 **Integer** 正と負の自然数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ $2^{31}-1$ (2,147,483,647)。 **経度** 正と負の自然数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ $2^{63}-1$ (9,223,372,036,854,775,807)。 **Float** 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} (1.4E-45) ～ $(2-2^{23})\times2^{127}$ (3.4028235E38)。 **二重線** 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} (4.9E-324) ～ $(2-2^{-52})\times21023$ (1.7976931348623157E308)。 **注：** integer タイプおよび long タイプを使用する場合に、演算の入力値や計算値に小数データが含まれていると、データが失われることがあります。
計算されたレコード数を取得	選択された操作を実行するグループ内の実際のレコード数を返します。 この *Computational Count* 列には、操作を実行する列に `null` 値が含まれる入力レコードはカウントされません。

[フィールド] タブ

[フィールド] タブは、ピボット テーブルを作成するとき使います。詳細については、**ピボット テーブルの作成**（66ページ）を参照してください。

[出力] タブ

オプション	説明
グループごとに 1 行を返す	行グループごとに、グループ内のすべての行についての集計データを含む 1 行だけを返します。個別の行は削除されます。このオプションがオフの場合は、すべての行が返されます。どのデータも削除されません。 Percent Rank または ZScore 操作では、このオプションは使えません。
各グループの行数を返す	各グループの行数を返します。行数が設定されるデフォルトの出力フィールド名は GroupCount です。
各グループに対し、一意の ID を返す	各行グループのユニーク ID を返します。ID は 1 から始まり、追加のグループが見つかるたびに 1 ずつ増加します。デフォルトのフィールド名は、GroupID です。

操作

以下の計算が使用できます。

平均 グループごとに、指定されたフィールドの平均値を計算します。例えば、ある一群のレコードの特定のフィールドに値 10、12、1、600 が含まれているとき、そのグループについてのそのフィールドの平均値は (10+12+1+600)÷4 = 155.75 と計算されます。

最大値 グループごとに、指定されたフィールドの最大値を返します。例えば、ある一群のレコードの特定のフィールドに値 10、12、1、600 が含まれているとき、そのグループについてのそのフィールドの最大値は 600 となります。

最小値 グループごとに、指定されたフィールドの最小値を返します。例えば、ある一群のレコードの特定のフィールドに値 10、12、1、600 が含まれているとき、そのグループについてのそのフィールドの最小値は 1 となります。

Percent Rank グループ内のレコードごとに、指定されたフィールド内の値をグループ内の他のレコードと比較したときの百分位数ランクを計算します。百分位数ランクは、そのフィールドについてそれよりも小さな値を持つレコードがグループ内の何パーセントを占めるかを表します。

百分位数 (Percentile) グループごとに、特定のフィールドについて指定した百分位数 (0 - 100) を表す値を計算します。百分位数は、それよりもスコアの小さいレコードが全体の何パーセントを占めるかを表します。例えば、ある一群のレコードが値 22、26、74 を持つとき、第 60 百分位数を指定して百分位数計算を実行すると、値 35.6 が返されます。これは、そのフィールドの値が 35.6 であるレコードはグループ内で第 60 百分位数のレコードに属することを意味します。

標準偏差 グループごとに、指定されたフィールドの標準偏差を計算します。標準偏差は、グループ内の値のばらつき度合いを表します。標準偏差が小さいほど、より多くの値が平均値のまわりに集中し、値のばらつきが小さくなります。逆に、標準偏差が大きいほど、値のばらつきが大きくなります。標準偏差はデータと同じ単位で表現されます。標準偏差は、分散の平方根です。

合計 グループごとに、指定されたフィールドの値の合計を計算します。

分散 グループごとに、指定されたフィールドの分散を計算します。分散は、グループ内の値のばらつき度合いを表します。標準偏差を 2 乗したものです。

ZScore グループ内のレコードごとに、ZScore を返します。ZScore は、値がグループの平均値から標準偏差のいくつ分だけ離れているかを示します。

英字を最初 グループごとに、最初の辞書値を返します。長さまたは辞書の位置が同じフィールド値が複数存在する場合は、その値のうちの最初のものを返します。例えば、グループの中に Joel と Joey というフィールド値を持つレコードが存在する場合、l の方が y よりもアルファベット順で前にあるため、そのグループに対する [英字を最初] の値は Joel になります。

英字を最後 グループごとに、最後の辞書値を返します。長さまたは辞書の位置が同じフィールド値が複数存在する場合は、その値のうちの最後のものを返します。例えば、グループの中に Joel と Joey というフィールド値を持つレコードが存在する場合、y の方が l よりもアルファベット順で後にあるため、そのグループに対する [英字を最後] の値は Joey になります。

最長 グループごとに、最長の値を返します。例えば、グループの中に Joel と Jacob というフィールド値を持つレコードが存在する場合、Jacob は 5 文字、Joel は 4 文字であるため、そのグループに対する最長値は Jacob になります。

最短 グループごとに、最短の値を返します。例えば、グループの中に Joel と Jacob というフィールド値を持つレコードが存在する場合、Joel は 4 文字、Jacob は 5 文字であるため、そのグループに対する最短値は Joel になります。

最新 グループごとに、最新の日付/日時値を返します。例えば、グループの中に 15-12-2014 と 24-12-2014 というフィールド値を持つレコードが存在する場合、そのグループに対する最新の値は 24-12-2014 になります。

最古　グループごとに、最も古い日付/日時値を返します。例えば、グループの中に 15-12-2014 と 24-12-2014 というフィールド値を持つレコードが存在する場合、そのグループに対する最も古い値は 15-12-2014 になります。

出力列

フィールド名	説明 / 有効な値
<操作>Of<入力フィールド名>	計算の結果が入ります。Group Statistics では、操作ごとに 1 つのフィールドが作成され、操作とフィールドに基づいてフィールドの名前が付けられます。例えば、`Sum`操作を `Population` という名前のフィールドに対して実行した場合のデフォルトのフィールド名は `SumOfPopulation` となります。
<値>_<操作>	ピボットの結果が入ります。ここで、`<Value>`はピボット列内のいずれかの値、`<Operation>` はその列に対して実行される操作です。詳細については、ピボット テーブルの作成（66ページ）を参照してください。
GroupCount	グループ内のレコード数を示します。
GroupID	各グループに順番に割り当てられる一意の番号。最初のグループの GroupID は値 1、その次は値 2 などとなります。
ComputationalCount<操作>Of<入力フィールド名>	操作を実行するグループ内の実際のレコード数を示します。 例えば、`Average`操作を `Salary` 列に対して実行すると、`ComputationalCountAverageOfSalary` という列が生成されます。
Status	Group Statistics 計算の成功または失敗を報告します。 **NULL**　成功 **F**　失敗

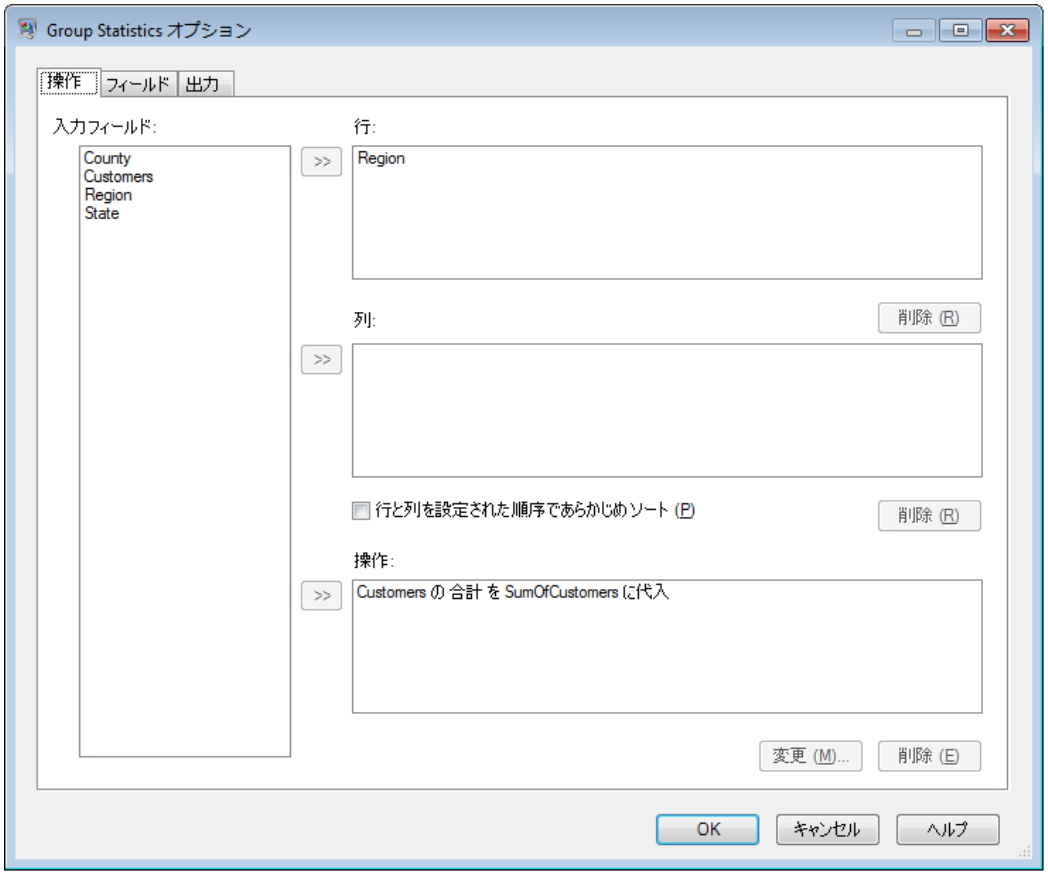
フィールド名	説明 / 有効な値
Status.Code	失敗の原因 (ある場合)。 ステータス コードには以下のものがあります。 **UnableToDoGroupStatistics** Group Statistics ステージで計算を実行できませんでした。 **百分位数の計算でエラーが発生しました** 指定された入力データで百分位数を計算できませんでした。
Status.Description	エラーの詳細な説明。 **入力フィールドの値をフィールドタイプに変換できませんでした。桁あふれの可能性があります!** 入力フィールドの値がデータ タイプの可能な値を超えています。double などの、より大きな値をサポートしているデータ タイプに変換してみてください。

Group Statistics の例

次の入力データは、郡内の顧客数を示しています。このデータは、郡の位置する米国の州 (MD、VA、CA、NV) と地域 (East または West) も示しています。最初の行はヘッダ レコードです。

```
Region|State|County|Customers
East|MD|Calvert|25
East|MD|Calvert|30
East|MD|Prince Georges|30
East|MD|Montgomery|20
East|MD|Baltimore|25
East|VA|Fairfax|45
East|VA|Clarke|35
West|CA|Alameda|74
West|CA|Los Angeles|26
West|NV|Washoe|22
```

各地域の顧客総数を計算したければ、**[操作]** タブの行として [地域] フィールドを定義します。操作については、[顧客] フィールドに対して合計操作を実行します。



次の結果が得られます。

```
Region|SumOfCustomers
    East|210.0
    West|122.0
```

> **注：** この例は、行だけでデータを集計する基本的な Group Statistics 操作を示しています。ピボット テーブルを作成すれば、**[操作]** タブでグループ化に使う列を指定して行と列の両方で集計することもできます。
>
> ピボット テーブルの作成の詳細については、ピボット テーブルの作成（66ページ）を参照してください。

ピボット テーブル

ピボット テーブルとは、データを視覚的に分析しやすくするために、データフロー内の列の値を集計したり入れ換えたりするものです。ピボットを使うと、入力の列を（クロスタブとして知られる）クロス集計の形式で整理して行、列、および集計値を生成できます。また、フィールドを

入力として使用できますが、表示することはできません。ピボットの使い方には、2次元のピボットと、1 次元の集計データのグループ化があります。

この例は、シャツの売上データを示しています。

表 3 : 入力データ

地域	性別	スタイル	出荷日	単位	価格	原価
東	男児	T シャツ	1/31/2010	12	11.04	10.42
東	男児	ゴルフ	6/31/2010	12	13.00	10.60
東	男児	ファンシー	2/25/2010	12	11.96	11.74
東	女児	T シャツ	1/31/2010	10	11.27	10.56
東	女児	ゴルフ	6/31/2010	10	12.12	11.95
東	女児	ファンシー	1/31/2010	10	13.74	13.33
西	男児	T シャツ	1/31/2010	11	11.44	10.94
西	男児	ゴルフ	2/25/2010	11	12.63	11.73
西	男児	ファンシー	2/25/2010	11	12.06	10.51
西	女児	T シャツ	2/25/2010	15	13.42	13.29
西	女児	ゴルフ	6/31/2010	15	11.48	10.67
北	男児	T シャツ	2/25/2010	17	16.04	10.42
北	男児	ファンシー	2/25/2010	12	11.56	12.42
北	女児	T シャツ	2/25/2010	16	12.32	18.42
北	男児	ゴルフ	1/31/2010	18	11.78	13.23

地域	性別	スタイル	出荷日	単位	価格	原価
北	女児	T シャツ	2/25/2010	12	18.45	11.64
北	女児	ゴルフ	2/25/2010	14	11.23	19.85
北	男児	ファンシー	1/31/2010	16	12.54	13.42
北	女児	T シャツ	2/25/2010	17	181.73	15.83
南	男児	ファンシー	1/31/2010	19	14.15	13.42
南	女児	T シャツ	2/25/2010	11	11.85	12.92
南	女児	ファンシー	1/31/2010	13	11.54	14.35
南	男児	T シャツ	2/25/2010	15	14.14	14.73
南	男児	ゴルフ	2/25/2010	16	17.83	17.83
南	女児	ファンシー	6/31/2010	11	18.24	12.35
南	女児	T シャツ	1/31/2010	20	19.94	12.95
南	男児	ゴルフ	2/25/2010	12	21.25	19.56

出荷日ごとに各地域で販売した数量を調べたいと思います。これを行うには、ピボットを使って次のテーブルを生成します。

表 4 : ピボット テーブル

地域	1/31/2010_ShipDate	2/25/2010_ShipDate	6/31/2010_ShipDate
東	32	12	22
北	34	88	
南	52	54	11

地域	1/31/2010_ShipDate	2/25/2010_ShipDate	6/31/2010_ShipDate
西	11	37	15

この例で、列は出荷日、行は地域、表示したいデータは数量です。ここでは合計操作による集計値で出荷数量の総数が表示されます。

ピボット テーブルの作成

ピボット テーブルとは、入力データに基づいてテーブルの行と列のカテゴリを作成することで、データを分析しやすいように集約するものです。詳細については、「ピボット テーブル（63ページ）」を参照してください。

Group Statistics ステージのオプションで、次の操作を行います。

1. **[操作]** タブで、**[入力フィールド]** からピボット テーブルで行ラベルとして使用するデータを含むフィールドを選択します。続いて、**[行]** フィールドの横にある **[>>]** ボタンをクリックします。
2. ピボット テーブルの列として使用するデータを含むフィールドを選択し、**[列]** フィールドの横にある **[>>]** ボタンをクリックします。

   ヒント： この段階で、インスペクションを実行して現在の選択の結果を確認します。選択した行と列によるクロス集計の結果を、可視化することができます。

3. 入力レコードのソートをスキップするには、**[行と列は設定された順序でソート済み]** をオンにします。

   このオプションがオンの場合、入力レコードはソートされることなくステージで処理されます。

   注： レコードが既にソートされている場合は、このオプションをオンにします。

4. 実行する操作を定義するには、**[操作]** フィールドの横にある **[>>]** ボタンをクリックします。

   [操作の追加] ウィンドウで、次の操作を行います。

   a) 実行する **[操作]** を選択します。
   b) **[入力フィールド]** セクションで、操作を実行する入力フィールドの **[名前]** と **[タイプ]** を選択します。
   c) **[出力フィールド]** セクションで、操作を実行した後に生成される出力フィールドの **[名前]** を入力し、**[タイプ]** を選択します。
   d) 操作を実行する入力レコードの実際の数を、独立した出力列として取得するには、**[計算されたレコード数を取得]** をオンにします。

`null` 値を含むレコードは、カウント `ComputationalCount<Operation>Of<InputFieldName>` に含まれません。

Computational Count がサポートされている関数:

- Average
- 分散
- ZScore
- 標準偏差
- 百分位数 (Percentile)
- Percent Rank
- 合計

その他の操作に対しては、このチェックボックスは無効のままになります。

5. ピボット テーブルの各列の出力フィールドを定義するには、ステージ オプションの **[フィールド]** タブをクリックします。

   ヒント： フィールドを正確に定義するために、この手順の前にインスペクション フローを 1 度実行して、データによって生成される列の名前を確認してください。

   a) **[追加]** をクリックします。
   [フィールドの追加] ウィンドウが表示されます。
   b) **[フィールドの追加]** ウィンドウのグリッド列は、**[操作]** タブで選択した **[列]** フィールドに基づきます。これらのグリッド列に、インスペクション フローを実行するときに列見出しとして表示される値を入力します。

   インポート機能を使用して [データ] 列のレコードを一発で設定することもできます。データを CSV またはテキスト ファイルからインポートする:

   1. **[インポート]** をクリックします。
   2. **[ファイル名]** フィールドでソース ファイルを参照して指定します。
   3. **[フィールド区切り文字とレコード区切り文字]** に値を入力します。
   4. [OK] をクリックします。

   ファイル内のすべてのレコードが [Column Data] テーブルに設定されます。

   注： ヘッダ行のないファイルをソース ファイルとして使用してください。

   例えば、**[操作]** タブの **[列]** で `ShipDat e` という入力フィールドを選択した場合は、**[フィールドの追加]** ウィンドウのグリッドには `ShipDate` というラベルの列が表示されます。このグリッド列に、データフローの入力データに存在する正確な `ShipDate` の値を、`2/25/2010` や `1/31/2010` のように入力します。

c) **[操作]** フィールドで、入力した各列フィールド値に対して出力フィールドを生成する、1つ以上の操作を選択します。選択した操作は、フィールド名だけに影響を与え、実際の計算は制御しないことに注意してください。

[フィールド] タブの **[操作]** フィールドに一覧表示される操作を変更するには、**[操作]** タブで **[操作]** フィールドの値を変更します。

重要： Computational Count の操作オプションである `ComputationalCount<Operation>Of<InputFieldName>` は、**[操作]** タブで **[操作]** を定義する際に **[計算されたレコード数を取得]** チェックボックスをオンにした場合にのみ表示されます。

d) **[追加]** をクリックします。

6. **[OK]** をクリックします。

上のグリッドに入力された各入力値を、選択された **[操作]** の各値にマッピングすることによって、出力フィールドが自動的に作成されます。グリッドに入力した入力列値と選択した操作のデカルト積が、最終的な出力列の自動生成に使用されます。

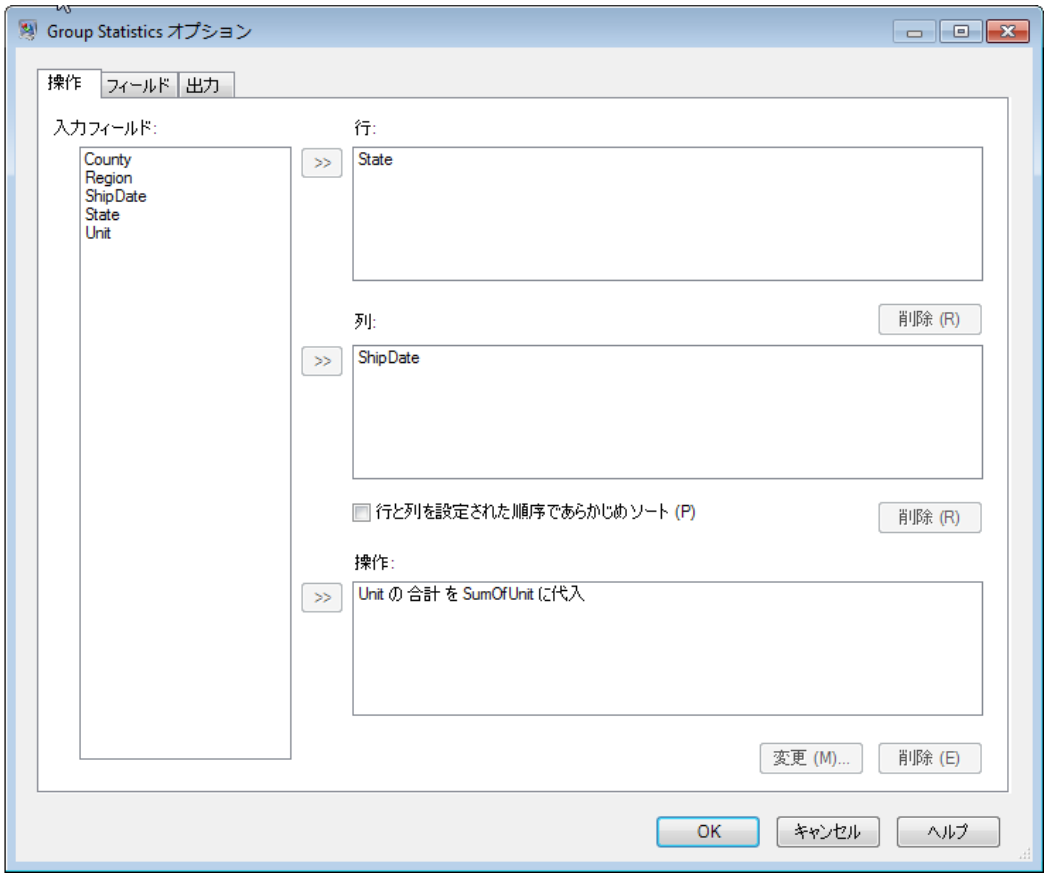
出力列の名前は、命名規則 `<Data>_<Operation>Of<InputFieldName>` に従います。ここで、`<Data>` は最初のフィールドで指定した値、`<Operation>` は **[操作]** フィールドで選択した操作、`<InputFieldName>` は `operation` を実行する入力列です。

ピボット テーブルの例

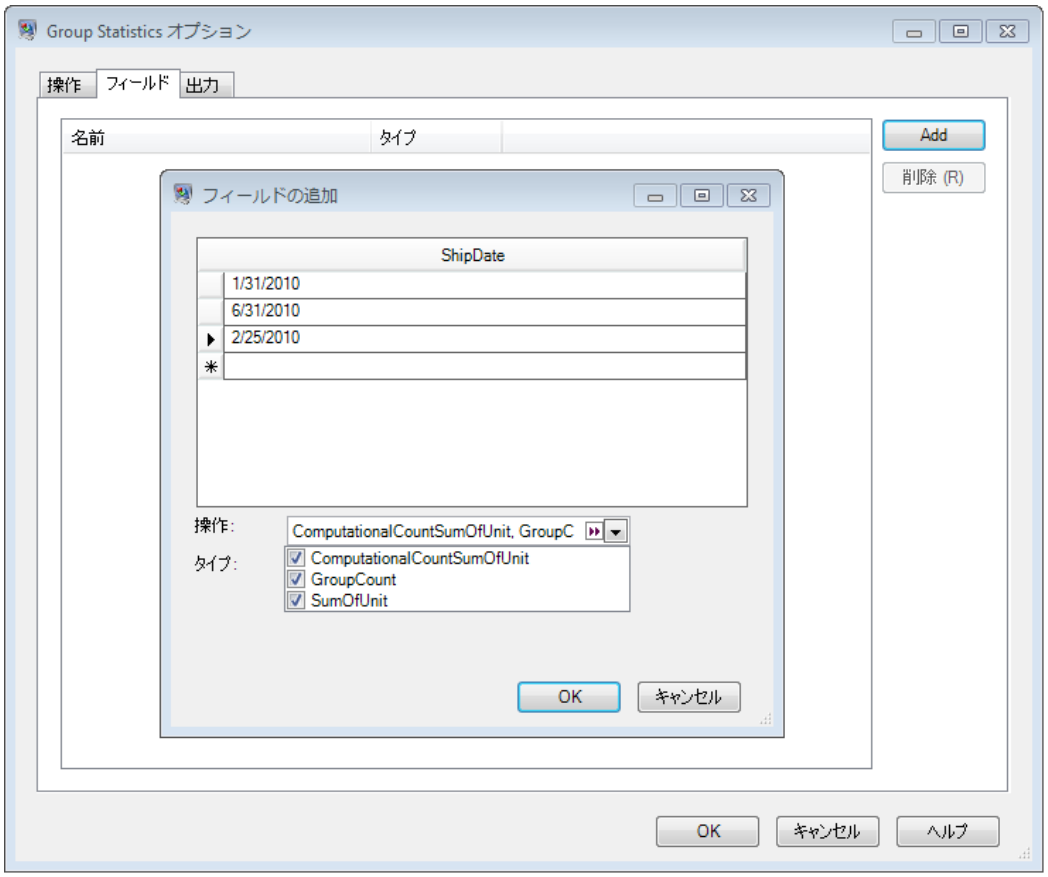
以下は、サービス業務部門からの出荷情報を示す入力データです。

```
Region,State,County,ShipDate,Unit
East,MD,Calvert,1/31/2010,
East,MD,Calvert,6/31/2010,212
East,MD,Calvert,1/31/2010,633
East,MD,Calvert,6/31/2010,234
East,MD,Prince Georges,2/25/2010,112
East,MD,Montgomery,1/31/2010,120
East,MD,Baltimore,6/31/2010,210
East,VA,Fairfax,1/31/2010,710
West,CA,SanJose,1/31/2010,191
West,CA,Alameda,2/25/2010,411
West,CA,Los Angeles,2/25/2010,
West,CA,Los Angeles,2/25/2010,215
West,CA,Los Angeles,6/31/2010,615
West,CA,Los Angeles,6/31/2010,727
```

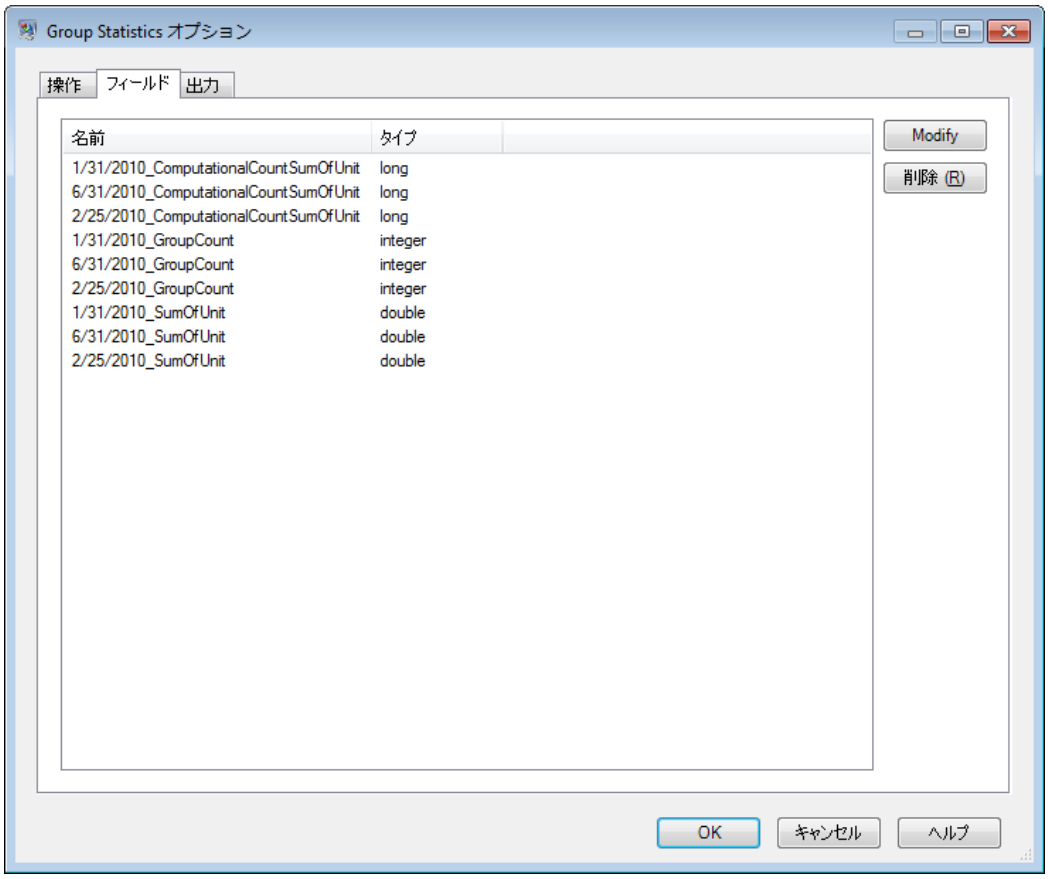
各出荷日に各州宛てに発送された出荷件数を調べるには、次の図のように Group Statistics ステージを設定することによってピボット テーブルを作成します:



ステージ オプションの **[フィールド]** タブで、データフローの入力データの `ShipDate` フィールドに出現する正確な日付をグリッドに追加し、各列値に対して表示する **[操作]** の値を選択します。



[フィールドの追加] ウィンドウの **[OK]** をクリックすると、作成された出力列が自動的に **[フィールド]** タブに一覧表示されます。これらの出力列は、正確な入力値と、**[フィールドの追加]** ウィンドウで選択した操作のデカルト積です。



出力

```
State,1/31/2010_GroupCount,1/31/2010_ComputationalCountSumOfUnit,
1/31/2010_SumOfUnit,2/25/2010_GroupCount,2/25/2010_ComputationalCountSumOfUnit,
2/25/2010_SumOfUnit,6/31/2010_GroupCount,6/31/2010_ComputationalCountSumOfUnit,
6/31/2010_SumOfUnit
VA,1,1,710,,,,,,
CA,1,1,191,3,2,626,2,2,1342
MD,3,2,753,1,1,112,3,3,656
```

Math

Math ステージは、単一のデータ行に対する算術演算を処理します。1 つ以上の式を使用する、様々な算術関数を実行できます。データは文字列として入力されますが、値は数値または Boolean (データに対して実行する演算の種類によります) である必要があります。

1. [制御ステージ] の下で [Math] ステージをクリックして、キャンバスまでドラッグし、データフロー内の配置したい位置にドロップします。

2. ステージをキャンバス上の他のステージに接続します。
3. [Math] ステージをダブルクリックします。**[Math オプション]** ダイアログ ボックスの [式] タブが表示されます。このタブには、[入力フィールド]、[Calculator]、[式] キャンバスが表示されます。または、[関数] タブをクリックして、Calculator の代わりに関数を使用することもできます。

[入力フィールド] コントロールには、入力ポートに存在する有効なフィールドの一覧が表示されます。フィールド名の構文は、非常に柔軟性の高いものですが、Groovy スクリプトの規則に基づく一部の制約があります。Groovy スクリプトの詳細な情報については、Web サイト **groovy-lang.org** を参照してください。

Calculator の使用

[Calculator] コントロールには、数値定数や演算子を式に挿入するためのボタンがあります。フィールド、定数、演算子、関数をダブルクリックすることにより、それらを式に挿入できます。

表 5 : Calculator の演算子

演算子	説明
Backspace	式の中で 1 文字分後退します。
pi	pi は円の直径に対する円周の割合 (円周率) を表す数学定数です。
e	オイラー数。自然対数の底を表す数学定数です。
/	除算
*	乗算
+	加算
-	減算
x^y	累乗 (例えば、x^2 は x の 2 乗)
Mod	モジュロ。演算の剰余です。

演算子	説明
;	セミコロン。式の末尾に使用します。
=	代入演算子
()	括弧。式の演算の優先順位を指定します。
。	小数点
if\else	条件文。条件が真の場合は処理を実行し、条件が偽の場合は別の処理を実行します。
if\else if\else	複数の条件文。条件が真の場合は処理を実行し、条件が偽の場合は別の処理を実行します。
==	算術関数の等号
!=	不等号
&&	論理積
||	論理和
>	次の値より大きい
>=	次の値以上
<	次の値より小さい
<=	次の値以下

関数と定数の使用

Math ステージには、式で使用できるいくつかの関数が用意されています。関数の一般的な形式は、関数(パラメータ)、関数(パラメータ,パラメータ)、関数(パラメータ,...) です。ここで、"パラメータ" は、数値定数、変数、または数式です。関数は別の数式と共に使用できます (例えば、x=Sin(y)*Cos(z))。

定数、変換、算術演算、および三角関数があります。次に、サポートされている各関数をカテゴリ別に示します。

表 6 : サポートされている関数

関数	説明
定数	
e	自然対数の底である数学定数です。
false	値 false を表す Boolean 定数です。
Infinity	無限大を表す数学定数です。
NaN	数値ではない値を表す数学定数です。
Pi	円の直径に対する円周の割合 (円周率) を表す数学定数です。
true	値 true を表す Boolean 定数です。
変換	
Abs (値)	パラメータを 1 つ指定します。 指定された値の絶対値を返します。
Ceil (値)	パラメータを 1 つ指定します。 端数を切り上げた値を返します (例えば、Ceil(5.5) は 6 を返します)。
DegToRad (値)	パラメータを 1 つ指定します。 指定された値を角度からラジアンに変換します。
Floor (値)	パラメータを 1 つ指定します。 端数を切り捨てた値を返します (例えば、Floor(5.5) は 5 を返します)。

関数	説明
RadToDeg (値)	パラメータを 1 つ指定します。 指定された値をラジアンから角度に変換します。
Round (値)	パラメータを 1 つ指定します。 四捨五入した値を返します。
Math	
Avg (値, 値,...)	パラメータを 1 つ以上指定します。 指定された値の平均値を返します。
Exp (値)	パラメータを 1 つ指定します。 オイラー数の累乗を返します。
Fac (値)	パラメータを 1 つ指定します。 指定された値の階乗を返します (例えば、Fac(6) は 6*5*4*3*2*1 を意味し、720 を返します)。
Ln (値)	パラメータを 1 つ指定します。 指定された値の自然対数 (底 e) を返します。
Log (値)	パラメータを 1 つ指定します。 指定された値の自然対数 (底 10) を返します。
Max (値, 値,...)	パラメータを 1 つ以上指定します。 引き渡された値の最大値を返します。
Min (値, 値,...)	パラメータを 1 つ以上指定します。 引き渡された値の最小値を返します。
Sqrt (値)	パラメータを 1 つ以上指定します。 引き渡された値の平方根を返します。

関数	説明
Sum (値)	パラメータを 1 つ指定します。 指定された値の合計を返します。
三角関数	
ArcCos (値)	パラメータを 1 つ指定します。 値の逆余弦を返します。
ArcSin (値)	パラメータを 1 つ指定します。 値の逆正弦を返します。
ArcTan (値)	パラメータを 1 つ指定します。 値の逆正接を返します。
Cos (値)	パラメータを 1 つ指定します。 値の余弦を返します。
Ln (値)	パラメータを 1 つ指定します。 指定された値の自然対数 (底 e) を返します。
Sin (値)	パラメータを 1 つ指定します。 値の正弦を返します。
Tan (値)	パラメータを 1 つ指定します。 値の正接を返します。

条件文の使用

条件文を使用すると、各種条件が真または偽のいずれであるかに応じて処理を実行できます。括弧 (と) を使用してグループ化することにより、さらに複雑な条件を指定できます。

表 7 : 条件

条件	説明
が次に等しい	式 = = 式
が等しくない	式 != 式
次の値より大きい	式 > 式
次の値以上	式 >= 式
次の値より小さい	式 < 式
次の値以下	式 <= 式
条件が偽	! 条件
And	条件 && 条件
Or	条件 || 条件

If 文

```
if(condition)
{
     actions to take if condition is true
}
```

中括弧は、"if" の後で複数の文を実行する場合のみ必要です。

If-Else If 文

```
if(condition)
{
     actions to take if condition is true
}
else if(condition)
{
     actions to take if condition is true
```

```
}
else if...
```

```
if(SideLength != NaN)
 {
 AreaOfPolygon=
 ((SideLength^2)*NumberOfSides)/
 (4*Tan(pi/NumberOfSides));
 }
 else if(Radius != NaN)
 {
 AreaOfPolygon=
 (Radius^2)*NumberOfSides*Sin((2*pi)/NumberOfSides)/2;
 }
```

else if 文は複数指定できます。中括弧は、"if-else- if-else" の後で複数の文を実行する場合にのみ必要です。

Else-If 文

```
if(condition)
 {
  actions to take if condition is true
 }
 else if(condition)
 {
  actions to take if condition is true
 }
 else if...
 else
 {
  actions to take if no conditions are met
 }
```

Expressions コンソールの使用

Expressions コンソールは、Math ステージで評価する数式を入力するために使用します。[入力フィールド]、[Calculator]、[関数] コントロールを使用して、このコンソールに値を挿入します。式は手動で入力することもできます。式は、定数、変数、または算術演算の形式で指定し、数値定数と変数で構成します。数値定数は、整数または小数で、符号を付けることができます。変数は入力行からのデータを表します。例えば、フィールド x、y、z が入力に定義されている場合、x、y、z を式で使用できます。変数は、実行時にフィールドの値で置き換えられます。

また、Math ステージでは、グループ化した式も使用できます。グループ化した式は括弧でくくられ、演算子の優先順位よりも優先されます。例えば、2*5^2 は 50 ですが、(2*5)^2 は 100 になります。

> **注：** 式の末尾には必ずセミコロンを付けます。

また、条件文を Expressions コンソールで使用すると、各種条件が真または偽のいずれであるかに応じて処理を実行できます。条件文の詳細については、条件文の使用（76ページ）を参照してください。

Math ステージでは主に、式の結果を変数に代入する代入式を扱います。このステージでは複数の代入演算がサポートされているため、以前の代入演算の結果を使用できます。

代入式の例

次の例では、x=10、z=1000 となります。

```
x=5+5
z=x*100
```

次の例では、ポリゴンの面積を一辺の長さと辺の数に基づいて計算します。

```
AreaOfPolygon=
      ((SideLength^2)*NumberOfSides)/
      (4*Tan(pi/NumberOfSides));
```

[フィールド] コントロールの使用

[フィールド] コントロールでは、入力および出力フィールドのタイプを変更できます。このコントロールでは、フィールドのタイプを変更できます。その場合は、[タイプ] 列のドロップダウン矢印をクリックし、一覧から次のいずれかのオプションを選択します。

boolean true と false の 2 つの値を持つ論理タイプ。Boolean 変数は、条件文で処理の流れを制御するために使用できます。次のコードは、Boolean 式の例を示したものです。

```
if(x && y)
      z=1;
  else if(x)
      z=2;
  else if(y)
      z=3;
  else
      z=4;
```

double 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

float 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。

integer	正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ $2^{31}-1$ (2,147,483,647)。
long	正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ $2^{63}-1$ (9,223,372,036,854,775,807)。

[プレビュー] コントロールの使用

[プレビュー] コントロールでは、数式をテストできます。[入力データ] エリアに、フィールドが一覧表示されます。式に引き渡す値を指定すると、その出力が [入力データ] の下の [結果] エリアに表示されます。

数値フィールドは 0 (double タイプの場合は 0.000) に初期化され、Boolean タイプ フィールドは False に初期化されます。double タイプおよび float タイプ フィールドでは、小数点以下の桁数が 4 桁に制限され、integer タイプおよび long タイプ フィールドでは小数点は使用しません。

Record Combiner

Record Combiner は、複数のストリームからの 2 つ以上のレコードを単一のレコードに結合します。Record Combiner は、1 つ以上のステージ入力ポートを使用できます。例えば、1 つのステージ入力 (ポート) から 1 つのレコードのグループを受信し、2 つめのステージ入力 (ポート 2) から別のレコードのグループを受信して、それらのレコードを単一のレコードに結合することができます。中間のステージを削除しても、ポート番号が連番になるように再度割り当てられることはありません。

> **注：** Record Combiner は、すべての入力ポートがレコードを受信するまで、レコードを出力しません。すべての入力ポートのレコードを結合してから、レコードを出力します。

入力ストリームに同じ名前のフィールドが存在する場合に保持する必要があるポートを指定できます。例えば、2 つのストリームからのレコードを結合する場合に、その両方のストリームに AccountNumber という名前のフィールドが含まれているとします。この場合、保持したいストリームに対応する Record Combiner 入力ポートを選択することで、どちらのストリームの AccountNumber フィールドを保持するかを指定できます。保持しないストリームに含まれる AccountNumber フィールドのデータは破棄されます。

Record Joiner

Record Joiner は、ストリームに含まれるフィールド間の関連性に基づいてさまざまなストリームからのレコードを結合する、SQL スタイルの `JOIN` 操作を実行します。Record Joiner を使用すると、複数のファイル、複数のデータベース、またはデータフローの上流にある任意のチャネル

からのレコードを結合できます。少なくとも 2 つの入力チャネルを Record Joiner に接続する必要があります。その後、JOIN 操作の結果が 1 つの出力チャネルに書き込まれます。オプションで、結合条件と一致しないレコードを別の出力チャネルに書き込むこともできます。

> **注：** Record Joiner を使用する前に、SQL JOIN 操作について十分に理解しておく必要があります。詳細については、**wikipedia.org/wiki/Join_(SQL)** を参照してください。

結合の定義

オプション	説明
左ポート	JOIN 操作の左テーブルとして使用するレコードのあるポート。それ以外の入力ポートはすべて、JOIN 操作の右テーブルとして使用されます。 **注：** "左" テーブルと "右" テーブルは SQL JOIN の概念です。Record Joiner を使用する前に、SQL JOIN 操作について十分に理解しておく必要があります。詳細については、**wikipedia.org/wiki/Join_(SQL)** を参照してください。
結合タイプ	実行する JOIN 操作のタイプ。次のいずれかです。 **左外部** 左ポートとその他のポートの間に一致がない場合でも、左ポートからすべてのレコードを返します。このオプションは、左ポートからすべてのレコードと、他のポートの一致するすべてのレコードを返します。 **全体** すべてのポートからすべてのレコードを返します。 **内部** 左ポートと別のポートの間で一致するレコードのみを返します。例えば、4 つの入力ソースがあり、ポート 1 を左ポートとします。この場合、内部結合は、ポート 1 とポート 2 の間、ポート 1 とポート 3 の間、およびポート 1 とポート 4 の間で、一致するフィールドを持つレコードをそれぞれ返します。
結合フィールド	レコードを結合するために別のポートからのフィールドに含まれるデータと一致する必要がある、左ポートからのフィールド。 **注：** 結合フィールドとして使用できるのは、データ タイプが String、Integer、Date、Datetime のフィールドのみです。

オプション	説明
左ポートからのデータをソートする	左ポートのレコードが、**[結合フィールド]** に指定されたフィールドによってソート済みかどうかを指定します。レコードがソート済みの場合、このボックスをオンにするとパフォーマンスが向上します。このボックスをオンにしていない場合、Record Joiner は、**[結合フィールド]** に指定されたフィールドに従ってレコードをソートした後で、結合操作を実行します。 複数の結合フィールドを指定した場合、**[結合フィールド]** に表示されるフィールドの順に従ってレコードをソートする必要があります。例えば、次の 2 つの結合フィールドがあるとします。 量 地域 この場合、最初に Amount フィールド、次に Region フィールドによってレコードをソートする必要があります。 **重要：** このオプションを選択してもレコードがソートされない場合は、Record Joiner から誤った結果が返ります。このオプションは、左ポートのレコードがソート済みであると確信できる場合にのみ選択してください。
結合の定義	左ポートからのレコードを他のいずれかのポートからのレコードと結合する必要があるかどうかを判断するために使用する結合条件を定義します。例: `port1.Name = port2.Name` これは、port1 からのレコードの Name フィールドの値が port2 からのレコードの Name フィールドの値と一致する場合に、その 2 つのレコードを結合することを指定します。 結合条件を変更するには、**[変更]** をクリックします。レコードを結合するために左ポートからの結合フィールドのデータとデータが一致する必要がある、右ポートからのフィールドを選択します。左ポート フィールドを変更する場合は、**[キャンセル]** をクリックし、**[結合フィールド]** フィールドでフィールドを変更します。右ポートのレコードが結合フィールドによってソートされている場合は、**[右ポートからのデータをソートする]** ボックスをオンにします。このボックスをオンにすると、パフォーマンスが向上します。 **重要：** **[右ポートからのデータをソートする]** を選択してもレコードがソートされない場合は、Record Joiner から誤った結果が返ります。このオプションは、右ポートのレコードがソート済みであると確信できる場合にのみ選択してください。

フィールドの解決

このタブでは、複数の入力ポートに同じフィールド名が存在する場合に、どのポートのデータを結合レコードで使用するかを指定します。例えば、データの 2 つのソースに対して結合を実行する場合に、各ソースに DateOfBirth という名前のフィールドが含まれているとします。この場合、どのポートのデータを結合後のレコードの DateOfBirth フィールドで使用するかを指定できます。

同じ名前のフィールドが存在しても、データが異なり、結合後のレコードに両方のフィールドのデータを保持する場合は、データを Record Joiner に送信する前に、いずれかのフィールドの名前を変更する必要があります。フィールド名の変更には Transformer ステージを使用できます。

結合しないレコードの処理

レコードを Record Joiner の出力に含めるには、そのレコードが結合条件を満たしているか、結合後のレコードと結合条件を満たさないレコードの両方を返す結合タイプを選択する必要があります。例えば、完全結合は、レコードが結合条件を満たしているかどうかにかかわらず、すべての入力ポートからすべてのレコードを返します。すべてのポートからすべてのレコードを返さない結合タイプ (左外部、内部結合など) の場合は、結合条件と一致するレコードのみが Record Joiner の出力に含まれます。

結合操作の結果に含まれないレコードをキャプチャするには、**not_joined** 出力ポートを使用します。このポートからの出力には、通常の出力ポートに含まれないレコードがすべて含まれます。**not_joined** 出力ポートは、ここに示すように、Record Joiner ステージの右側にある白の三角形です。

このポートから出力されるレコードには、**InputPortIndex** というフィールドが追加されます。このフィールドには、そのレコードの Record Joiner 入力ポートの番号が含まれます。この番号から、レコードのソースを特定できます。

注：

- このステージの最適なパフォーマンスを得るためには、レコードの 2 つの独立したストリームを結合して、統合された出力が生成されるようにします。
- 1 つのパスが Broadcaster または Conditional Router のどちらかを使用して最初に分岐した後で Record Joiner を使用して再び結合されている場合、このフローはハングする可能性があります。分岐と結合の間で複数のステージが使用されている場合は、Sorter をできるだけ Record Joiner の近くで使用します。

Sorter

Sorter は、指定されたフィールドを使用してレコードをソートします。例えば、レコードを名前、都市、またはデータフロー内のその他のフィールドでソートできます。

Sorter によるレコードのソート

Sorter ステージでは、指定したフィールドを使用してレコードをソートできます。

1. [制御ステージ] の下で、[Sorter] をキャンバスまでドラッグし、データフロー内の配置したい位置にドロップします。
2. [Sorter] をダブルクリックします。
3. **[追加]** をクリックします。
4. **[フィールド名]** 列の下矢印をクリックして、ソートするフィールドを選択します。

   注： 使用可能なフィールドのリストは、データフローの以前のステージで使用したフィールドに基づくものです。

5. **[順序]** 列で、昇順または降順のいずれでソートするかを選択します。
6. **[タイプ]** 列で、フィールドのデータ タイプを選択します。

   注： 入力データが文字列形式でない場合は、**[タイプ]** 列は無効になります。

bigdecimal	小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。
double	正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。
float	正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。
integer	正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ 2^{31}-1 (2,147,483,647)。
long	正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ 2^{63}-1 (9,223,372,036,854,775,807)。
string	文字シーケンス。

7. ソートの前に、値の前後にある空白スペースを削除するには、**[トリム]**列のボックスをチェックします。トリム オプションを使用しても、フィールドの値は変更されません。このオプションは、ソートのためにのみ、値をトリムします。入力データが文字列形式でない場合は、**[トリム]** 列は無効になります。

8. **[NULL を次の値とみなす]** 列で、ソート済みリストにおける null 値の配置として **[最大値]** または **[最小値]** を選択します。配置は、**[順序]** フィールドと **[NULL を次の値とみなす]** フィールドで選択したオプションの組み合わせによって、以下の表のようになります。

順序	NULL を次の値とみなす:	ソート済みリストにおける null 値の配置
昇順	最大値	リストの末尾
昇順	最小値	リストの先頭
降順	最大値	リストの先頭
降順	最小値	リストの末尾

9. 操作を繰り返して、ソートするフィールドをすべて追加します。
10. **[上へ]** または **[下へ]** をクリックして、ソートの順序を必要に応じて変更します。これを使用すると、最初にあるフィールドによってソートし、その結果をさらに別のフィールドでソートできます。
11. 管理者によって定義されているデフォルトのソート パフォーマンス オプションをオーバーライドする場合は、**[詳細設定]** をクリックし、**[ソート パフォーマンス オプションをオーバーライド]** ボックスをチェックして、次のオプションを指定します。

メモリ内レコードの上限値
ソートでメモリ内に保持できるデータ行の最大数を指定します。この上限を越えると、ディスクにページングします。デフォルトでは、10,000 レコード未満のソートはメモリ内で行われ、10,000 レコードを越えるソートはディスク ソートとして実行されます。上限値は 100,000 レコードです。通常、メモリ内ソートはディスク ソートよりはるかに速いため、大部分のソートがメモリ内ソートとなり、大規模セットのみがディスクに書き出されるよう、この値を十分大きく設定してください。

> **注：** 複数のジョブを同時並行で実行する環境では、**[メモリ内レコードの上限値]** の設定を増やすと、メモリ不足になる可能性が高くなります。

一時ファイルの最大数
ソート プロセスで使用できる一時ファイルの最大数を指定します。使用する一時ファイルの数を増やすと、パフォーマンスが向上する可能性があります。ただし、最適なファイル数は Spectrum™ Technology Platform を実行しているサーバーの構成に大きく依存します。さまざまな設定を試して、使用する一時ファイル数の増減がパフォーマンスに与える影響を確認する必要が

あります。必要になる可能性がある一時ファイルの適切な数を計算するには、次の式を使用します。

```
(NumberOfRecords × 2) ÷ InMemoryRecordLimit =
NumberOfTempFiles
```

一時ファイルの最大数は 1,000 を超える値にはできないことに注意してください。

圧縮を有効にする 一時ファイルをディスクに書き込むときに圧縮します。

注： 最適なソート パフォーマンスの設定は、サーバーのハードウェア構成によって異なります。それでも一般には、次の式で妥当なソート パフォーマンスが得られます。

```
(InMemoryRecordLimit × MaxNumberOfTempFiles ÷ 2) >=
TotalNumberOfRecords
```

12. **[OK]** をクリックします。

    注： 必要に応じて、ソート条件を削除するには、行をハイライト表示し、**[削除]** をクリックします。

Splitter

Splitter では階層データがフラット データに変換されます。Splitter には 1 つの入力ポートと、Splitter から次のステージへとデータを送信する 1 つの出力ポートがあります。Splitter の機能を使用する一例として、情報の一覧をファイルで受け取って、その情報の各項目を個々のデータ行として抽出することがあります。具体的には、例えば、緯度/経度ポイントで表す一定の距離内に配置された複数のランドマークを入力として引き渡し、それらのランドマークを Splitter によって個々のデータ行に分割できます。

Splitter ステージの使用

1. [制御ステージ] の下で **[Splitter]** をクリックして、キャンバスまでドラッグし、データフロー内の配置したい位置にドロップして、入力および出力ステージに接続します。
2. [Splitter] をダブルクリックします。**[Splitter オプション]** ダイアログ ボックスが表示されます。
3. **[分割位置]** ドロップダウンをクリックして、このステージで使用できる別のリスト タイプを表示します。Splitter で作成するリスト タイプをクリックします。**[Splitter オプション]** ダイ

アログ ボックスは、選択内容に応じて調整され、そのリスト タイプで使用できるフィールドが表示されます。

また、**[分割位置]** ドロップダウンの横にある省略記号 (**...**) ボタンをクリックすることもできます。**[フィールド スキーマ]** ダイアログ ボックスが表示され、Splitter に入力されるデータのスキーマが示されます。リスト タイプは太字で示され、その後に続いて、各タイプの個々のリストが表示されます。また、これらのフィールドの形式も表示されます (string、double 等)。Splitter で作成するリスト タイプをクリックし、**[OK]** をクリックします。**[Splitter オプション]** ダイアログ ボックスは、選択内容に応じて調整され、そのリスト タイプで使用できるフィールドが表示されます。

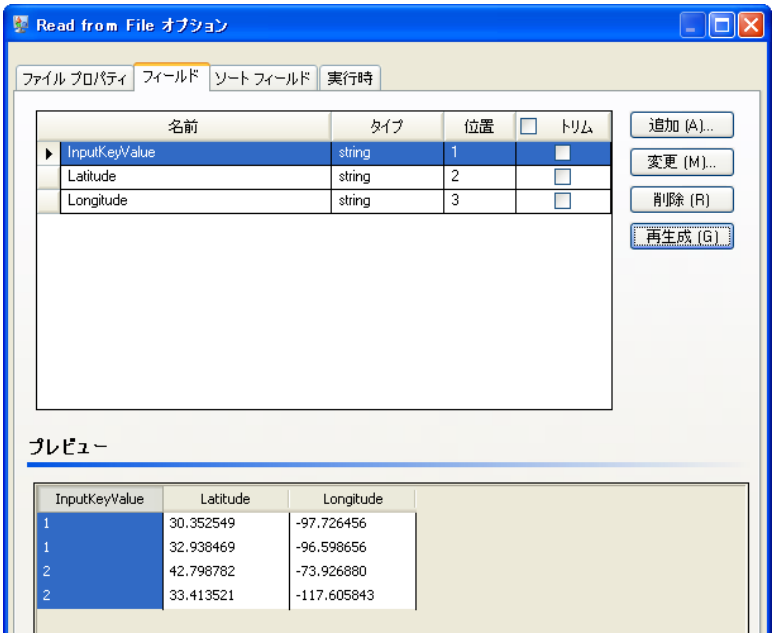
4. 抽出された分割リストの元のレコードを返すには、**[ヘッダ レコードを出力]** をクリックします。
5. レコードに分割リストがない場合のみ、元のレコードを返すには、**[入力リストが空の場合のみ]** をクリックします。
6. Splitter で出力に含めるフィールドを選択します。そのためには、該当するフィールドの **[含める]** ボックスをクリックします。
7. **[OK]** をクリックします。

Splitter の例

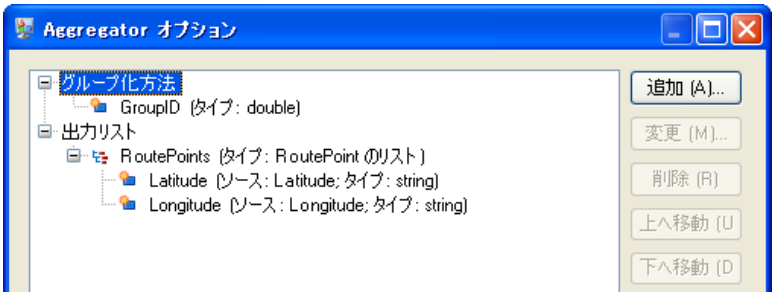
次の例では、ドライブの道順を含むルーティング ステージから出力を受け取って、各道順 (またはリスト項目) をデータ行に分割できます。データフローは次のようになります。

このデータフローは、次のように機能を実行します。

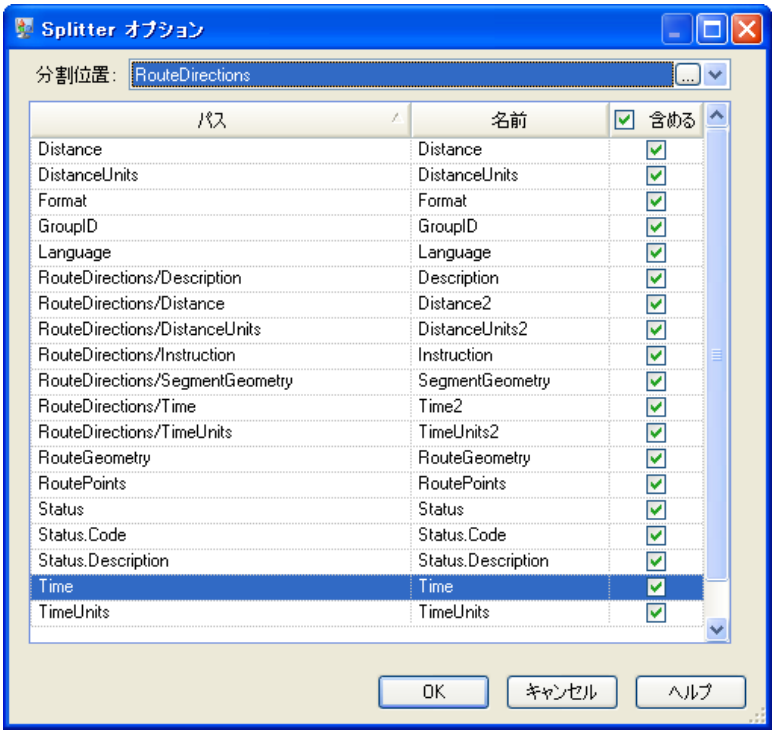
1. **Read from File** ステージには、個々のポイントの識別を容易にする緯度、経度、および入力キー値が含まれます。



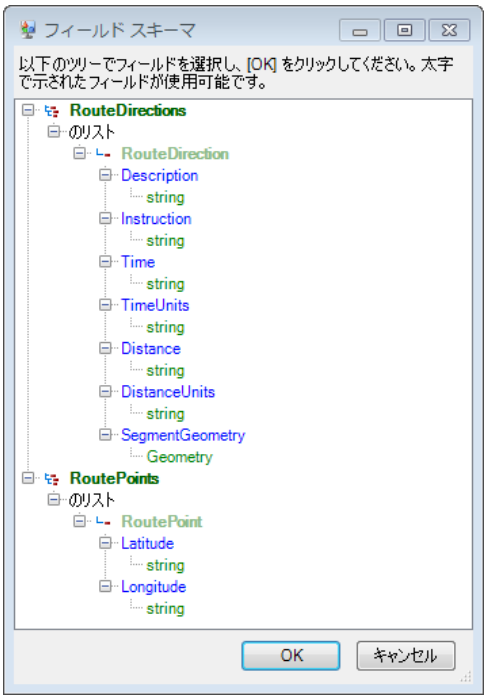
2. Aggregator ステージでは、Read from File ステージからのデータに基づいてスキーマ (データの構造的階層) を作成し、緯度と経度のグループをルート ポイントの一覧として識別します。このステップは、次のステップを適切に実行するために必要なステップです。



3. Location Intelligence モジュールの **Get Travel Directions** ステージでは、ステップ 2 からのルート ポイントを使用して、ある場所から別の場所への道順を作成します。
4. **Splitter** ステージでは、データを RouteDirections フィールドで分割し、出力リストに Get Travel Directions ステージから使用可能なフィールドをすべて取り込む必要があります。



スキーマは、次のように構成されます。RouteDirections と RoutePoints は、このジョブで使用できるリスト タイプです。



5. **Write to File** ステージでは、出力がファイルに書き出されます。

Stream Combiner

Stream Combiner は、複数のステージからのレコードで構成される 2 つ以上のストリームを結合します。Stream Combiner には 1 つ以上のステージ入力ポートがあります。例えば、1 つのステージから 1 つのレコードのグループを受信し、2 つめのステージから別のレコードのグループを受信して、それらのレコードを単一のストリームに結合することができます。

Stream Combiner に設定はありません。

Transformer

Transformer ステージは、フィールドの値と書式を変更します。入力フィールドと出力フィールドの名前が同じであれば、フィールドに対して複数のトランスフォームを実行できます。

一般トランスフォーム

フィールド作成

既存のフィールドの値または定数値を使用して、フィールドの値を置換するか、新しいフィールドを作成します。例えば、City という名前のフィールドがあり、City フィールドの値に "City of" という語句を追加する必要があるとします。その場合に、次のようなテンプレートを作成します。

```
City of ${City}
```

[To field (対象フィールド)] フィールドで、City フィールドを選択します。これにより、City フィールドの既存の値が、テンプレートを使用して構成された値と置換されます。例えば、City フィールドの値が Chicago の場合、新しい値は City of Chicago になります。

有効なテンプレートを作るには、いくつかの文字の前にバックスラッシュ ("\") を付ける必要があります。例えば、一重引用符の前には、\' のようにバックスラッシュを付ける必要があります。バックスラッシュでエスケープする必要がある記号のリストについては、**groovy-lang.org/syntax.html** を参照してください。

コピー

フィールドの値を別のフィールドにコピーします。

カスタム

Groovy 言語を使用して独自のトランスフォームを定義できます。詳細については、「**Custom Transform の作成**（95ページ）」を参照してください。

Location Intelligence モジュールのユーザの場合、Custom Transform は空間データセットにアクセスできます。『*Spectrum Spatial ガイド*』(**support.pb.com)** の「ステージ」セクションを参照してください。

名前の変更 フィールドの名前を変更します。データフローの既存のフィールド名の一覧から選択することも、別の名前を入力することもできます。

ステータス [ステータス] フィールドの値を [成功] または [失敗] に変更します。[失敗] に設定されている場合は、オプションの [説明] と [コード] も設定できます。

トランスフォームの書式設定

大文字小文字 文字を大文字または小文字に変更します。

マスク フィールドに対して文字を適用、またはフィールドから文字を削除します。詳細については、「**Mask Transform の使用**（100ページ）」を参照してください。

パディング フィールドの左側または右側に文字を追加します。

文字列関係のトランスフォーム

前後の空白削除 フィールドの先頭と末尾にある空白を削除します。また、一連の空白 (複数の連続する空白等) を単一の空白文字に置き換えます。

部分文字列の削除 文字列をフィールドからすべて削除します。例えば、"CA" を StateProvince フィールドから削除します。

部分文字列 一連の連続する文字をフィールド間でコピーします。

トリム フィールドの左側、右側、または両側から、指定された文字を削除します。このトランスフォームでは大文字と小文字が区別されます。

切り捨て 指定された個数の文字をフィールドの左側と右側から削除します。

リスト変換

この機能を利用すると、"input from read from XML" のような、リストに作用する canned (缶詰) 変換を作成できます。

リスト変換を定義する手順は次のとおりです。

1. リスト変換操作を選択します。右側のツリー ビューに入力フィールドが表示されます。
2. 操作を適用するツリー内の有効なフィールドを選択します。入力フィールドのツリー ビューの下に操作のプロパティが表示されます。
3. 操作のプロパティを指定し、[追加] をクリックします。変換が親ウィンドウ ([Transformer オプション] ウィンドウ) 内のリストに追加されます。

フィールドを作成 ユーザの選択したリスト タイプ フィールドの下にフィールドを作成できます。例えば、Football というリストに 2 つのクラブ Knitters と Lambs があり、ユーザが新しいクラブ Irons を追加すると、リストに 3 つのクラブができます。

ソート 選択されたフィールド内の値をソートします。複雑なリストの場合にはソートに使うキー要素を指定する必要がありますが、単純なリストではリスト内の要素がその

ままソートされます。ユーザはソート順序 (昇順または降順) を選択できます。Football の例で、リストに 3 つのクラブがあるとき、名前に基づいてクラブをソートするには [クラブ] の下の [名前] フィールドを選択する必要があります。ソート順序が昇順なら現在のクラブのエントリは Irons、Knitters、Lambs とリストされ、降順ならこの逆の順序でリストされます。プレイヤーのリストをソートしたければ、[プレイヤー] フィールドを選択する必要があります。リストに設定されているソート順序でソートされます。

合計 選択されたフィールドのすべての値を合計します。結果はユーザの指定したフィールドに保存されます。例えば、各フットボール クラブの合計得点を表示するには、[トーナメント] の下の [得点] フィールドを選択し、結果を出力するフィールドの名前を指定します。

コピー 選択されたフィールドからユーザの指定したフィールドに対してコピー操作を実行します。コピーするフィールドを選択すると、そのフィールドとその下のすべてのフィールド (もしあれば) が、指定した新しいフィールドにコピーされます。この操作は階層の同じレベルに対して実行されます。

名前の変更 選択されたフィールドに対して名前変更操作を実行して、ユーザの指定した新しい名前に変更します。

次の XML コードをリスト変換機能のリファレンスとしてご利用ください。

```
<?xml version="1.0"?>
<sports_details>
  <sports name="football">
 <clubs>
   <club name="Knitters">
    <player>Samuel</player>
    <player>Messi</player>
    <player>kaka</player>
    <player>Alan</player>
    <coach>Stuart</coach>
    <Tournament name="Football League">
   <result>won</result>
   <points>4</points>
    </Tournament>
    <Tournament name="UEFA">
   <result>draw</result>
   <points>2</points>
    </Tournament>
   </club>
   <club name="Lambs">
    <player>Ronaldo</player>
    <player>Neymar</player>
    <player>Zlatan</player>
    <player>Mesut</player>
    <coach>Ivan</coach>
```

```
    <Tournament name="Airtel League">
   <result>draw</result>
   <points>2</points>
    </Tournament>
    <Tournament name="Champions League">
   <result>lost</result>
   <points>0</points>
    </Tournament>
   </club>
   <club name="Irons">
    <player>Scott</player>
    <player>Paul</player>
    <player>John</player>
    <player>Andrew</player>
    <coach>Jeff</coach>
    <Tournament name="CAF">
   <result>won</result>
   <points>4</points>
    </Tournament>
    <Tournament name="Copa America">
   <result>won</result>
   <points>4</points>
    </Tournament>
   </club>
</clubs>
 </sports>
 <sports name="badminton">
<clubs>
   <club name="Shuttlers">
    <player>Saina</player>
    <player>Viktor</player>
    <player>Chen</player>
    <player>Srikanth</player>
    <coach>Jan</coach>
    <Tournament name="Olympic Games">
   <result>won</result>
   <points>4</points>
    </Tournament>
    <Tournament name="Commonwealth Games">
   <result>won</result>
   <points>4</points>
    </Tournament>
   </club>
   <club name="Choppers">
    <player>Wang</player>
    <player>Sindhu</player>
    <player>Carolina</player>
    <player>Li Xuerui</player>
    <coach>Ratchanok</coach>
    <Tournament name="World Junior">
   <result>draw</result>
   <points>2</points>
    </Tournament>
```

```
    <Tournament name="Uber Cup">
   <result>draw</result>
   <points>2</points>
    </Tournament>
   </club>
   <club name="Lobbers">
    <player>Nozomi</player>
    <player>Chou</player>
    <player>Marc</player>
    <player>Lin</player>
    <coach>Kevin</coach>
    <Tournament name="World Senior">
   <result>won</result>
   <points>4</points>
    </Tournament>
    <Tournament name="Thomas Cup">
   <result>won</result>
   <points>4</points>
    </Tournament>
   </club>
 </clubs>
  </sports>
</sports_details>
```

トランスフォームの順序の変更

複数のトランスフォームを特定の出力フィールドで実行する場合は、トランスフォームの実行順序を定義できます。

> **注:** 1つのフィールドを2つの異なる出力フィールドにマップ (例えば、ValidateAddress.City を Output.City1 と Output.City2 にマップ) して、各フィールドにトランスフォームを追加する場合は、2 番目のフィールドのトランスフォームを最初に実行する必要があります。したがって、2番目のフィールドのトランスフォーム (Output.City2) を最初に実行するように、トランスフォームの実行順序を変更する必要があります。

1. [Transformer] ステージをダブルクリックします。[Transformer オプション] ダイアログ ボックスが表示されます。
2. トランスフォームを選択し、[上へ移動] および [下へ移動] ボタンを使用してトランスフォームの順序を変更します。一番上のトランスフォームが最初に実行されます。

   > **注:** 従属トランスフォームをプライマリ トランスフォームの上に移動することはできません (プライマリ トランスフォームは、従属トランスフォームが依存するトランスフォームです)。

3. **[OK]** をクリックします。

Custom Transform の作成

Transformer ステージには、一般的な各種データ変換を実行するトランスフォームがあらかじめ定義されています。これらのトランスフォームがニーズを満たさない場合は、Groovy を使用して Custom Transform スクリプトを記述できます。この手順では、Groovy を使用して、基本的な Custom Transform を作成する方法について説明します。Groovy に関する詳細なドキュメントは、**groovy-lang.org** でご覧いただけます。

1. Enterprise Designer で、データフローに Transformer ステージを追加します。
2. [Transformer] ステージをダブルクリックします。
3. **[追加]** をクリックします。
4. **[全般]** の下の **[カスタム]** をクリックします。
5. **[Custom Transform 名]** フィールドに、作成するトランスフォームの名前を入力します。名前は一意でなければなりません。
6. **[スクリプト エディタ]** をクリックします。

   このエディターは、コード自動入力、パレット リスト関数およびフィールドなど、トランスフォームの開発を容易にするさまざまな機能を備えています。

タスク	手順
関数を追加するには	**[関数]** ウィンドウで、追加する関数をダブルクリックします。 **注 :** エディターに表示される関数は、Custom Transform スクリプトの記述を容易にするものです。これらの関数は、本来なら Groovy コードを数行書かないと実行できない処理を実行します。これらの関数は、標準的な Groovy 関数ではありません。

データフローフィールドから値を取得するには	**[入力フィールド]** ウィンドウで、目的の入力フィールドをダブルクリックします。スクリプトに次の形式で追加されます。
	
	`data['FieldName']`
	
	例えば、CurrentBalance フィールドから値を取得する場合は、次の式が追加されます。
	
	`data['CurrentBalance']`

タスク	手順

データフローフィールドの値を設定するには	スクリプト エディタに次のコードを入力します。
	`data['FieldName']=NewValue`
	例えば、Day フィールドに、PurchaseDate フィールドに含まれる曜日を設定する場合:
	`data['Day']=dayOfWeek(data['PurchaseDate'])`
	この例では、`dayOfWeek()`関数を使用して、PurchaseDate フィールドの日付の値から曜日を取得し、その結果を Day フィールドに書き込んでいます。
	ヒント： [出力フィールド] ウィンドウで出力フィールドの名前をダブルクリックすると、スクリプトにフィールド参照を追加できます。

数値データタイプを使用して新しいフィールドを作成するには	スクリプト エディタに次のコードを入力します。
	`data['FieldName'] = new constructor;`
	ここで、*constructor* は以下のいずれかです。
	java.lang.Double(*number*)
	Double データ タイプのフィールドを作成します。
	java.lang.Float(*number*)
	Float データ タイプのフィールドを作成します。
	java.lang.Integer(*number*)
	Integer データ タイプのフィールドを作成します。整数を指定することによって、新しい integer フィールドを作成することもできます。例えば、次の例では、値 23 を持つ integer フィールドが作成されます。
	`data['MyNewField'] = 23;`
	java.lang.Long(*number*)
	Long データ タイプのフィールドを作成します。
	例えば、Double データ タイプで値 23.10 を持つ、"Transactions" という新しいフィールドを作成するには、次のように指定します。
	`data['Transactions'] = new com.java.lang.Double(23.10);`

タスク	手順

date または time データ タイプを使用して新しいフィールドを作成するには	スクリプト エディタに次のコードを入力します。
	`data['FieldName'] = new constructor;`
	ここで、*constructor* は以下のいずれかです。
	com.pb.spectrum.api.datetime.Date(*year*,*month*,*day*)
	date データ タイプのフィールドを作成します。例えば、December 23, 2013 は、次のようになります。
	`2013,12,23`
	com.pb.spectrum.api.datetime.Time(*hour*,*minute*,*second*)
	time データ タイプのフィールドを作成します。例えば、4:15 PM は、次のようになります。
	`16,15,0`
	。
	com.pb.spectrum.api.datetime.DateTime(*year*,*month*,*day*,*hour*,*minute*,*second*)
	DateTime データ タイプのフィールドを作成します。例えば、4:15 PM on December 23, 2013 は、次のようになります。
	`2013,12,23,16,15,0`
	例えば、Date データ タイプで値 December 23, 2013 を持つ、"TransactionDate" という新しいフィールドを作成するには、次のように指定します。
	`data['TransactionDate'] = new com.pb.spectrum.api.datetime.Date(2013,12,23);`

Boolean データ タイプの新しいフィールドを作成するには	スクリプト エディタに次のコードを入力します。
	`data['FieldName'] = true or false;`
	例えば、IsValidated というフィールドを作成して false に設定するには、次のように指定します。
	`data['IsValidated'] = false;`

タスク	手順

新しいリスト フィールドを作成するには	`factory.create()` メソッドを使用してレコードに新しいフィールドを作成し、次に leftShift 演算子 << を使用して新しいレコードをリスト フィールドに追加します。
	`NewListField = []`
	
	`NewRecord = factory.create()`
	`NewRecord['NewField1'] = "Value"`
	`NewRecord['NewField12'] = "Value"`
	`...`
	`NewListField << NewRecord`
	
	`NewRecord = factory.create()`
	`NewRecord['NewField1'] = "Value"`
	`NewRecord['NewField12'] = "Value"`
	`...`
	`NewListField << NewRecord`
	`data['ListOfRecords'] = NewListField`
	
	次の例では、"addresses" という新しいリスト フィールドを作成し、2 つの "address" レコードを含めています。
	`addresses = []`
	`address = factory.create()`
	`address['AddressLine1'] = "123 Main St"`
	`address['PostalCode'] = "12345"`
	`addresses << address`
	
	`address = factory.create()`
	`address['AddressLine1'] = "PO Box 350"`
	`address['PostalCode'] = "02134"`
	`addresses << address`
	`data['Addresses'] = addresses`
	
	また、レコードのリストではなく、個々のフィールドのリストを含んだ新しいリスト フィールドを作成することもできます。次の例では、"PhoneNumbers " という新しいリスト フィールドを作成し、自宅と職場の電話番号を含めています。
	`phoneNumbers = []`
	`phoneNumbers << data['HomePhone']`
	`phoneNumbers << data['WorkPhone']`
	`data['PhoneNumbers'] = phoneNumbers`

タスク	手順

複数のフィールドを連結するには	+記号を使用します。例えば、次の式は、FirstNameフィールドとLastNameフィールドを1つの値に連結し、その値をFullNameフィールドに格納します。
	`String fullname = data['FirstName'] + ' ' + data['LastName'];`
	`data['FullName']=fullname;`
	次の例には、2つの入力フィールド(AddressLine1とAddressLine2)があります。これらのフィールドは連結されて、出力フィールドAddressに書き込まれます。
	`address1 = data['AddressLine1'];`
	`address2 = data['AddressLine2'];`
	`data['Address']=address1+ ',' + address2;`

フィールドをパースするには	分割文字を特定した後、substringを使用してフィールドをパースします。次の例では、PostalCodeフィールドが6文字以上の場合は、フィールドを5桁のZIPコードと+4番号に分割し、それぞれの番号を出力レコードの各フィールドに書き込みます。
	`if (data['PostalCode'].length() > 5)`
	`{`
	`  String postalCode = data['PostalCode'];`
	`  int separatorPosition = postalCode.indexOf('-');`
	`  String zip = postalCode.substring(0, separatorPosition);`
	`  String plusFour = postalCode.substring(`
	`  separatorPosition + 1,`
	`  postalCode.length();`
	`  data['Zip']=zip;`
	`  data['PlusFour']=plusFour;`
	`}`

条件付き処理を実行するには	ifまたはswitch文を使用します。これらは、条件付き処理で最も一般的に使用される構造です。詳細については、**groovy-lang.org**を参照してください。
	この例では、AddressCityフィールドを最初の住所行に設定し、都市がAustinの場合は都市名も設定します。
	`city = data['City'];`
	`address1 = data['AddressLine1']`
	`if(city.equals('Austin'))`
	`data['AddressCity']=address1 +',' + city;`

タスク	手順
ループ処理を実行するには	`for`ループを使用します。これは、唯一必要なループ構造です。ループ処理または構文の詳細については、**groovy-lang.org** を参照してください。

データを拡張するには	定数を定義し、連結文字+を使用します。例えば、次のスクリプトは、"Incorporated"という単語を FirmName フィールドに追加します。
	`firmname = data['FirmName'];`
	`constant = 'Incorporated';`
	`if(firmname.length() > 0)`
	`data['FirmName']=firmname + ' ' + constant;`

実行時に指定されたオプションにアクセスするには	データフローで実行時オプションが有効になっている場合、次の構文を使用して、実行時にデータフローに引き渡される設定にアクセスできます。
	`options.get("optionName")`
	例えば、`casing`という名前のオプションにアクセスするには、Custom Transform スクリプトに次の式を含めます。
	`options.get("casing")`

7. スクリプトの入力が完了したら、ウィンドウの "X" ボタンをクリックしてエディターを閉じます。
8. **[入力フィールド]** フィールドで、トランスフォームを適用するフィールドを選択します。
9. **[出力フィールド]** フィールドで、トランスフォームからの出力を書き込むフィールドを指定します。必要に応じて、新しいフィールドを定義できます。その場合は、**[出力フィールド]** フィールドの右にある **[追加]** ボタンをクリックします。
10. 作業が完了したら、ウィンドウ下部の **[追加]** ボタンをクリックします。
11. **[OK]** をクリックします。

Mask Transform の使用

Transformer ステージを使用して、フィールドに Mask Transform を適用できます。Mask Transform では、指定されたパターンを使用して、フィールドに文字を適用したり、フィールドから文字を

削除したりできます。例えば、Mask Transform を使用すると、8003685806 などの数字の文字列を (800) 368 5806。

1. Enterprise Designer で、Transformer ステージをキャンバスにドラッグして目的の位置に接続します。
2. [Transformer] ステージをダブルクリックします。
3. **[追加]** をクリックします。
4. **[書式設定]** を展開し、**[マスク]** を選択します。
5. 使用するマスクのタイプを選択します。

   適用 フィールドに文字を追加して、文字列を新しいパターンにまとめます。

   削除 文字列から文字のパターンを取り出します。

6. **[マスク文字列]** フィールドで、文字の追加または削除で使用するパターンを指定します。

   マスク文字列を指定するときには、リテラル文字とマスク文字の2種類の文字を使用します。

   リテラル文字は、文字列内の実際の文字を表します。削除マスクを使用するときは、入力文字は、リテラル文字と正確に一致している必要があります。正確に一致していれば、リテラル文字は入力から削除されます。同様に、適用マスクでは、リテラル文字は、マスク定義によって指定される入力内の位置に追加されます。

   マスク文字列では、マスク文字というもう1つの種類の文字も使用できます。マスク文字は、入力文字列の特定の位置にある文字のタイプを指定します。例えば、入力の 1 文字目が数字の場合、最初のマスク文字は # にする必要があります。入力内の要素のうち、このマスク文字と一致する要素のみが出力に残ります。

   次の表に、**[マスク文字列]** フィールドで使用できるマスク文字を示します。

表 8 : マスク文字

文字	定義
#	任意の数字。
'	エスケープ文字。特殊な書式文字をエスケープします。
U	任意の文字。すべての小文字を大文字にマップします。

文字	定義
L	任意の文字。すべての大文字を小文字にマップします。
A	任意の文字または数字。
?	任意の文字。
*	任意。
H	16 進文字 (0-9、a-f、または A-F)。

7. **[追加]** をクリックします。
8. **[OK]** をクリックします。

Mask Transform の例

これは、文字列に書式設定を適用する適用マスクです。です。この例で "(" および ")" と <スペース> はリテラルなので出力に追加されます。# はマスク文字なので、数字はすべて残ります。

入力: `8003685806`
マスク文字列: `(###) ### ####`
出力: `(800) 368 5806`

次の例は、ZIP Code 内のダッシュを取り除く削除マスクです。

入力: `60510-1135`
マスク文字列: `*****-****`
出力: `605101135`

Unique ID Generator

この Unique ID Generator ステージでは、特定のレコードを識別するユニーク キーを作成します。ユニーク ID は、トランザクションで名前および住所データをすべて送信するとは限らないが、同

じレコード/連絡先に帰属させる必要があるデータ ウェアハウスの取り組みではきわめて重要です。ユニーク ID は、個人、世帯、企業、および敷地レベルで実装されることがあります。Unique ID Generator は、ユニーク ID を作成する各種アルゴリズムを備えています。

ユニーク ID は、順番数字または日付/時間スタンプのいずれかに基づきます。また、オプションで、各種アルゴリズムを使用して、ID に付加するデータを生成することもできるので、より一意である可能性の高い ID を作成できます。順番数字または日付/時間スタンプ ID は必須で、生成された ID から削除できません。

Unique ID Generator は、いずれかのキー生成アルゴリズムを使用して非ユニーク キーを生成するのに使用できます。非ユニーク モードでは、マッチングに使用するキーを作成できます。これは、データ ウェアハウスで、ディメンションにキーを追加済みで、新しいレコードが既存のレコードと一致するかどうかを確認するために新しいレコード用のキーを生成する場合に便利です。

次の例では、入力の各レコードに出力で順番レコード ID を割り当てます。

レコード	レコード ID
John Smith	0
Mary Smith	1
Jane Doe	2
John Doe	3

Unique ID ステージによって、ユニーク ID を含む RecordID という名前のフィールドが作成されます。RecordID フィールドの名前は、必要に応じて変更できます。

ユニーク ID の定義

デフォルトで、Unique ID Generator ステージでは順番 ID を作成し、1 番目のレコードの ID は 0、2 番目のレコードの ID は 1、3 番目のレコードの ID は 2 になります。ユニーク ID の生成方法を変更する場合は、次の手順に従います。

1. Unique ID Generator ステージで、**[ルール]** タブの **[変更]** をクリックします。
2. ユニーク ID の生成方法を選択します。

オプション	説明
順番数字タグ開始位置	指定した数字を先頭の数字として、各レコードに増分値を割り当てます。例えば、先頭の数字として 0 を指定すると、1 番目のレコードの ID は 0、2 番目のレコードの ID は 1 になります。

オプション	説明
順番数字タグをデータベース フィールドの値で開始する	

オプション	説明
	データベース フィールドから読み込んだ最大値を先頭の数字として、各レコードに増分値を割り当てます。この数字に 1 を加算した数字が 1 番目のレコードに割り当てられます。例えば、データベース フィールドから読み込まれる数字が 30 の場合、1 番目のレコードの ID は 31、2 番目のレコードの ID は 32 になります。
接続	使用するデータベース接続を選択します。使用できるデータベース接続は、Management Console の Connection Manager に定義されている接続によって異なります。新しいデータベースを作成、あるいは既存の接続を変更または削除する必要がある場合は、**[管理]** をクリックします。 データベース接続を追加または変更する場合は、次のフィールドに必要な値を指定します。 **接続名** 接続の名前を入力します。任意の名前にすることができます。 **データベース ドライバ** 適切なデータベース タイプを選択します。 **接続オプション** データベースへの接続に使用するホスト、ポート、インスタンス、ユーザ名、およびパスワードを指定します。
テーブル/ビュー	クエリの実行対象とするデータベース内のテーブルまたはビューを指定します。
[データベース] フィールド	リストから列を選択して、ユニーク キーを生成します。 以下のデータタイプが、ユニーク ID の生成でサポートされています。 **long** 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ $2^{63}-1$ (9,223,372,036,854,775,807)。 **integer** 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ $2^{31}-1$ (2,147,483,647)。 **bigdecimal** 小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。 **double** 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times 2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

オプション	説明	
	float	正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。
日付/時間スタンプ	順番数字を割り当てるのではなく、日付/時間スタンプに基づいてユニーク キーを作成します。	
UUID	各レコードに対して、世界で一意 (universally unique) の 32 桁の識別キーを作成します。キーの桁は、ハイフンによって 5 つのグループに区切られて、8-4-4-4-12 という形式の合計 36 文字 (32 個の英数字と 4 つのハイフン) で表示されます。例: 123e4567-e89b-12d3-a456-432255330000	
無効	このオプションは、アルゴリズムを使用して非ユニーク キーを生成する場合にのみ選択します。	

3. **[OK]** をクリックします。

アルゴリズムを使用したユニーク ID の拡張

Unique ID Generator は、各レコードに順番数字を割り当てるか、各レコードに日付/時間スタンプを生成して、各レコードにユニーク ID を生成します。オプションで、順番または日付/時間ユニーク ID に追加情報を付加するアルゴリズムも使用できます。したがって、より複雑なユニーク ID や、真に一意である可能性の高いユニーク ID を作成できます。

1. Unique ID Generator ステージで、**[追加]** をクリックします。
2. **[アルゴリズム]** フィールドで、ID の追加情報の生成に使用するアルゴリズムを選択します。次のいずれかです。

   Consonant (子音) 指定されたフィールドを、子音を削除して返します。

   Double Metaphone 文字の発音表記に基づくコードを返します。Double Metaphone は Metaphone アルゴリズムの改良版で、さまざまな言語に多数存在する不規則性を考慮しています。

   Koeln ドイツ語で発音される名前に、音声によってインデックスを付けます。同じ発音を持つ名前を同じ表現にエンコードできるので、綴りに小さな相違があっても、マッチさせることができます。結果は常に一連の数字です。特殊文字および空白は無視されます。このオプションは、Soundex の制限に対応するために作成されました。

MD5 128 ビットのハッシュ値を生成するメッセージ ダイジェスト アルゴリズム。このアルゴリズムは、データの一貫性の確認によく使用されます。

Metaphone 選択したフィールドを Metaphone コード化したキーを返します。Metaphone は、英語の発音を使用して単語をコード化するアルゴリズムです。

Metaphone (スペイン語) 選択したフィールドをスペイン語用に Metaphone コード化したキーを返します。この Metaphone アルゴリズムは、スペイン語の発音を使用して単語をコード化します。

Metaphone 3 Metaphone アルゴリズムおよび Double Metaphone アルゴリズムを、より正確な子音および内部母音の設定で改良したもので、単語または名前の一致性を高く、または低くして、音声ベースで語を検索できるようにします。Metaphone 3 では、音声エンコーディングの精度が 98% に向上しています。このオプションは、Soundex の制限に対応するために作成されました。

Nysiis 近似の発音と正確な綴りをマッチさせ、同じように発音される単語にインデックスを付ける、音声コード アルゴリズム。New York State Identification and Intelligence System の一部です。例えば、住民のデータベースで誰かの情報を探しているとします。その人物の名前は "John Smith" のように聞こえますが、実際の綴りは "Jon Smyth" です。"John Smith" の完全一致を探す検索を実行した場合、返される結果はありません。しかし、NYSIIS アルゴリズムを使用してデータベースにインデックスを作成し、再度 NYSIIS アルゴリズムを使用して検索した場合は、正しいマッチが返されます。なぜなら、"John Smith" と "Jon Smyth" は、このアルゴリズムによってどちらも "JAN SNATH" というインデックスが付けられているからです。

Phonix 100 を越える変換ルールを適用することによって、名前文字列を単一の文字またはいくつかの文字のシーケンスに前処理します。これらのルールのうち 19 個は文字がその文字列の先頭にある場合にのみ適用され、12 個はその文字列の中間にある場合にのみ適用され、28 個は文字列の終わりにある場合にのみ適用されます。変換された名前文字列は、開始文字とそれに続く 3 桁 (ゼロおよび重複する数字を削除) で構成されるコードにエンコードされます。このオプションは、Soundex の制限に対応するために作成されました。このオプションは複雑なため、Soundex より遅くなります。

Soundex 選択したフィールドの Soundex コードを返します。Soundex は、単語の英語の発音に基づいて、固定長のコードを生成します。

部分文字列 選択されているフィールドの指定部分を返します。

3. **[フィールド名]** フィールドで、アルゴリズムを適用するフィールドを選択します。例えば、Soundex アルゴリズムを選択し、City という名前のフィールドを選択すると、City フィールドのデータに Soundex アルゴリズムを適用して ID が生成されます。

4. 部分文字列アルゴリズムを選択した場合、部分文字列で使用するフィールドの部分を指定します。
   a) **[開始位置]** フィールドで、部分文字列を開始するフィールド内の位置を指定します。
   b) **[長さ]** フィールドで、部分文字列に含める開始位置からの文字数を選択します。

      例えば、LastName という名前のフィールドに次のデータが含まれているとします。

      ```
      Augustine
      ```

      開始位置を 3、終了位置を 6 に指定すると、次の部分文字列が作成されます。

      ```
      gustin
      ```

5. **[ノイズ文字の削除]** ボックスをチェックすると、アルゴリズムを適用する前に、英数字以外の文字 (ハイフン、空白スペース、その他の特殊文字など) がフィールドからすべて削除されます。
6. Consonant (子音) および部分文字列アルゴリズムの場合、**[ソート入力]** ボックスをチェックすると、アルゴリズムを適用する前に、フィールドのデータをソートできます。その後、フィールド内の文字または語をアルファベット順にソートできます。
7. **[OK]** をクリックして、設定を保存します。
8. 他のアルゴリズムを追加して、より複雑な ID を作成する場合は、これらの操作を必要に応じて繰り返します。

注： ユニーク キーの定義は常に異なる色で表示され、削除できません。

非ユニーク ID の定義

Unique ID Generator は、いずれかのキー生成アルゴリズムを使用して非ユニーク キーを生成するのに使用できます。非ユニーク モードでは、マッチングに使用するキーを作成できます。これは、データ ウェアハウスで、ディメンションにキーを追加済みで、新しいレコードが既存のレコードと一致するかどうかを確認するために新しいレコード用のキーを生成する場合に便利です。

1. Unique ID Generator ステージで、**[ルール]** タブの **[変更]** をクリックします。
2. **[無効]** を選択します。

   これで、ID 生成ルールのユニーク ID 部分が無効になります。このオプションを無効にすると、次の手順で選択するアルゴリズムのみが ID の作成に使用されます。つまり、ID の生成に使用するフィールドのデータが同じレコードには、すべて同じ ID が使用されます。ID は後でマッチングに使用できます。

3. **[OK]** をクリックします。
4. 警告が表示されたら、**[はい]** をクリックします。
5. Unique ID Generator ステージで、**[追加]** をクリックします。

6. **[アルゴリズム]** フィールドで、ID の追加情報の生成に使用するアルゴリズムを選択します。次のいずれかです。

Consonant (子音)	指定されたフィールドを、子音を削除して返します。
Double Metaphone	文字の発音表記に基づくコードを返します。Double Metaphone は Metaphone アルゴリズムの改良版で、さまざまな言語に多数存在する不規則性を考慮しています。
Koeln	ドイツ語で発音される名前に、音声によってインデックスを付けます。同じ発音を持つ名前を同じ表現にエンコードできるので、綴りに小さな相違があっても、マッチさせることができます。結果は常に一連の数字です。特殊文字および空白は無視されます。このオプションは、Soundex の制限に対応するために作成されました。
MD5	128 ビットのハッシュ値を生成するメッセージ ダイジェスト アルゴリズム。このアルゴリズムは、データの一貫性の確認によく使用されます。
Metaphone	選択したフィールドを Metaphone コード化したキーを返します。Metaphone は、英語の発音を使用して単語をコード化するアルゴリズムです。
Metaphone (スペイン語)	選択したフィールドをスペイン語用に Metaphone コード化したキーを返します。この Metaphone アルゴリズムは、スペイン語の発音を使用して単語をコード化します。
Metaphone 3	Metaphone アルゴリズムおよび Double Metaphone アルゴリズムを、より正確な子音および内部母音の設定で改良したもので、単語または名前の一致性を高く、または低くして、音声ベースで語を検索できるようにします。Metaphone 3 では、音声エンコーディングの精度が 98% に向上しています。このオプションは、Soundex の制限に対応するために作成されました。
Nysiis	近似の発音と正確な綴りをマッチさせ、同じように発音される単語にインデックスを付ける、音声コード アルゴリズム。New York State Identification and Intelligence System の一部です。例えば、住民のデータベースで誰かの情報を探しているとします。その人物の名前は "John Smith" のように聞こえますが、実際の綴りは "Jon Smyth" です。"John Smith" の完全一致を探す検索を実行した場合、返される結果はありません。しかし、NYSIIS アルゴリズムを使用してデータベースにインデックスを作成し、再度 NYSIIS アルゴリズムを使用して検索した場合は、正しいマッチが返されます。なぜなら、"John Smith" と "Jon Smyth" は、このアルゴリズムによってどちらも "JAN SNATH" というインデックスが付けられているからです。
Phonix	100 を越える変換ルールを適用することによって、名前文字列を単一の文字またはいくつかの文字のシーケンスに前処理します。これらのルールのうち 19 個は文字がその文字列の先頭にある場合にのみ適用され、12 個はその文字列

の中間にある場合にのみ適用され、28 個は文字列の終わりにある場合にのみ適用されます。変換された名前文字列は、開始文字とそれに続く 3 桁 (ゼロおよび重複する数字を削除) で構成されるコードにエンコードされます。このオプションは、Soundexの制限に対応するために作成されました。このオプションは複雑なため、Soundex より遅くなります。

Soundex 選択したフィールドの Soundex コードを返します。Soundex は、単語の英語の発音に基づいて、固定長のコードを生成します。

部分文字列 選択されているフィールドの指定部分を返します。

7. **[フィールド名]** フィールドで、アルゴリズムを適用するフィールドを選択します。例えば、Soundex アルゴリズムを選択し、City という名前のフィールドを選択すると、City フィールドのデータに Soundex アルゴリズムを適用して ID が生成されます。
8. 部分文字列アルゴリズムを選択した場合、部分文字列で使用するフィールドの部分を指定します。
   a) **[開始位置]** フィールドで、部分文字列を開始するフィールド内の位置を指定します。
   b) **[長さ]** フィールドで、部分文字列に含める開始位置からの文字数を選択します。

   例えば、LastName という名前のフィールドに次のデータが含まれているとします。

   ```
   Augustine
   ```

   開始位置を 3、終了位置を 6 に指定すると、次の部分文字列が作成されます。

   ```
   gustin
   ```

9. **[ノイズ文字の削除]** ボックスをチェックすると、アルゴリズムを適用する前に、英数字以外の文字 (ハイフン、空白スペース、その他の特殊文字など) がフィールドからすべて削除されます。
10. Consonant (子音) および部分文字列アルゴリズムの場合、**[ソート入力]** ボックスをチェックすると、アルゴリズムを適用する前に、フィールドのデータをソートできます。その後、フィールド内の文字または語をアルファベット順にソートできます。
11. **[OK]** をクリックして、設定を保存します。
12. 他のアルゴリズムを追加して、より複雑な ID を作成する場合は、これらの操作を必要に応じて繰り返します。

注： ユニーク キーの定義は常に異なる色で表示され、削除できません。

フロー出力

データフローの出力を定義するには、"シンク" ステージを使用します。シンクは、データフローの最終ステージです。ここでは、データフローの出力に対して行う操作を定義します。シンクでは、最後に他のアクションや別のデータフローを実行することもでき、例えば、適当なプログラムを実行することができます。

ジョブの出力

ジョブの出力は、ファイルまたはデータベースに書き込むことができます。Spectrum™ Technology Platformでは、データの出力先としてさまざまなファイル形式やデータベース タイプを指定できます。出力先として指定できるシンクの種類は、どのモジュールについてライセンスを取得しているかによって異なります。お使いのモジュールに対応するソリューション ガイドについては、**support.pb.com/spectrum** を参照してください。

サービスの出力

サービスからのデータ出力は、出力ステージで定義されます。このステージでは、サービスからWeb サービス リクエストや API 呼び出しへの応答として返されるすべてのフィールドを定義します。

サービス出力の定義

Output ステージでは、サービスまたはサブフローが返す出力フィールドを定義します。次の手順に従って、サービス出力を定義します。

1. キャンバス上の出力アイコンをダブルクリックします。**[出力オプション]** ダイアログ ボックスが表示されます。**[出力オプション]** ダイアログ ボックスを初めて開く場合は、入力に定義されているフィールドのリストが表示されます。
2. フィールド リストに新しいフィールドを追加するには、**[追加]** をクリックします。**[カスタム フィールドの追加]** ダイアログ ボックスが表示されます。また、カスタム フィールドを変更または削除することもできます。
3. **[追加]** を再度クリックします。
4. フィールド名をテキスト ボックスに入力します。
5. **[データ タイプ]** を選択し、**[OK]** を押します。次のデータ タイプがサポートされています。

   bigdecimal 小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用

してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。

boolean true と false の 2 つの値を持つ論理タイプ。

bytearray バイトの配列 (リスト)。

> **注：** bytearray は REST サービスの入力としてはサポートされていません。

date 月、日、年を含むデータ タイプ。例: 2012-01-30、January 30, 2012。デフォルトの日付の形式は Management Console で指定できます。

datetime 月、日、年、時、分、秒を含むデータ タイプ。例: 2012/01/30 6:15 PM。

double 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

float 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。

integer 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ 2^{31}-1 (2,147,483,647)。

list 厳密に言えば、リストはデータ タイプではありません。しかし、フィールドが階層データを含む場合、"リスト" フィールドとして扱われます。Spectrum™ Technology Platform では、リストは複数の値で構成されるデータのコレクションです。例えば、Names フィールドには名前の値のリストを含めることができます。これは、XML 構造では次のように表すことができます。

```
<Names>
    <Name>John Smith</Name>
    <Name>Ann Fowler</Name>
</Names>
```

XML のリスト データ タイプが複数の値で構成される単純データ タイプであるのに対し、Spectrum™ Technology Platform のリスト データ タイプは XML の複合データ タイプに似ているという点で、Spectrum™ Technology Platform のリスト データ タイプは XML スキーマのリスト データ タイプと異なることに注意してください。

long 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ 2^{63}-1 (9,223,372,036,854,775,807)。

string 文字シーケンス。

time 時刻を含むデータ タイプ。例: 21:15:59 or 9:15:59 PM。

また、必要に応じて、ユーザ定義の新しいデータ タイプを追加して、定義済みのデータ タイプの一覧に加えることができます。例えば、名前 (文字列) のリストや、AddressLine1 (文字列)、City (文字列)、StateProvince (文字列)、PostalCode (文字列) などを含む住所の新しいデータ タイプを定義できます。フィールドを作成した後は、[入力オプション] ダイアログにアクセスして[データ タイプ]列のボタンを押すと、そのデータ タイプを表示できます。**[データ タイプ詳細]** ダイアログ ボックスが表示され、フィールドの構造が示されます。

6. **OK** を再度クリックします。
7. **[エクスポーズ]** の横にあるチェック ボックスをクリックして、フィールド リストに含まれるフィールドすべてのチェック ボックスを選択します。フィールド リストからフィールドを選択して、フィールドをステージ操作用としてデータフローにエクスポーズします。チェック ボックスを再度クリックして、リストに含まれるフィールドすべてのチェック ボックスをクリアします。フィールド リストに含まれる 1 つ以上のフィールドのチェック ボックスをクリアし、**[OK]** をクリックすると、フィールドがフィールド リストから削除されます。

   注 : 入力フィールドで階層データを定義した場合は、データをインポートしたり、データを垂直に表示したりすることはできません。

8. **[OK]** をクリックして、キャンバスに戻ります。

Web サービス データ タイプの定義

[データ タイプ名] フィールドにより、作成しているサービスの WSDL (SOAP) および WADL (REST) インターフェイスの制御が可能です。Rows 要素の名前は、サービス内のこのステージに付けた名前によって決まり、Row 要素の名前は、ここに入力したテキストによって決まります。

注 : WSDL の場合はリクエストとレスポンスの両方に影響しますが、WADL の場合はレスポンスのみに影響が生じます。

このステージに名前を付け、このフィールドにテキストを入力する前のコードの例を、次に示します。

```
<Rows>
     <Row>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
     </Row>
     <Row>
          <FirstName>Jane</FirstName>
    <LastName>Doe></LastName>
     </Row>
</Rows>
```

このステージに名前を付け、このフィールドにテキストを入力した後、コードは次のようになります。

```
<Names>
     <Name>
          <FirstName>John</FirstName>
          <LastName>Doe</LastName>
     </Name>
     <Name>
          <FirstName>Jane</FirstName>
          <LastName>Doe></LastName>
     </Name>
</Names>
```

外部プログラムの実行

Execute Program ステージは、レコードを受け取ると、プログラムやコマンド ライン コマンド等の実行可能ファイルを呼び出します。Execute Program ステージをデータフローで使用するには

オプション

オプション	説明
コマンドライン	実行可能ファイルの名前と引数 (適用される場合)。引数には、データフロー内で使用可能なデータを指定できます。そのデータにアクセスするには、 [...] (参照) ボタンをクリックします。現在のジョブ ID、現在のジョブ名、または現在のユーザ名の 3 つのコンテキストから選択できます。使用可能なフィールドから選択することもできます。例えば、 JobStatus や JobComment を選択できます。
タイムアウト	指定時間内にコマンドが応答しない場合に、実行をキャンセルするかどうかを指定します。次のいずれかです。 **タイムアウトなし** コマンドが応答に失敗しても実行をキャンセルしません。 **タイムアウト (ミリ秒)** 指定されたミリ秒内にコマンドが応答しない場合は、実行試行をキャンセルします。

オプション	説明
環境変数	これはオプションです。コマンドを実行するときに使用する環境変数を指定します。環境変数を追加するには、**[追加]** をクリックします。 **[キー]** フィールドに適切なキーワードを入力します。例えば、"JAVA_HOME" と入力します。 **[値]** フィールドに適切な値を入力します。例えば、`C:\Java\jre7`と入力します。または、[...] (参照) ボタンをクリックすることにより、[フィールド リスト] ダイアログ ボックスからフィールドを選択することもできます。現在のジョブ ID、現在のジョブ名、または現在のユーザ名の 3 つのコンテキストから選択できます。使用可能なフィールドから選択することもできます。例えば、 JobStatus や JobComment を選択できます。

条件に基づくジョブの終了

レコード内で特定の条件が見つかった場合にジョブを終了するには、Terminate Job ステージと Conditional Router を組み合わせて使用します。レコードが Terminate Job に送信されると、ジョブは終了します。

> **注：** Terminate Job は、サービスやサブフローでは使用できません。

Terminate Job を使用するには、データフローに Conditional Router と Terminate Job ステージを追加します。次に、ステージを接続し、Conditional Router を設定します。コンディショナル ルータの設定には、ジョブの終了条件を指定してください。条件を満たすレコードが見つかると、そのレコードは Terminate Job に渡されてジョブが終了し、"ジョブはステージ <ステージ ラベル> で終了しました" というメッセージが表示されます。完了したデータフローは次のようになります。

レコードの破棄

Write to Null ステージはレコードを破棄します。レコードはカウントされますが、破棄されます。データフローの終了後に残さないレコードがある場合は、このステージを使用します。

埋め込まれたデータフロー

埋め込まれたデータフローを使うと、複数のステージをグループ化して、Enterprise Designer キャンバスに一度に表示されるステージの数を減らせます。グループ化されたステージは、1 つのステージとして表示されます。埋め込まれたデータフローは、以下の用途に使用できます。

- ステージをグループ化して複雑なデータフローのレイアウトを簡素化し、1 つのステージとしてキャンバス上に表示します。
- 繰り返し機能を使ってグループ内のレコードを処理します。
- 埋め込みデータフロー内のステージオプションを設定するのにフィールド内の値を使用します。

データフローには、埋め込まれたデータフローをいくつでも追加できます。また、埋め込まれたデータフローに、埋め込まれたデータフローを含めることもできます。

埋め込まれたデータフローとサブフローの相違点

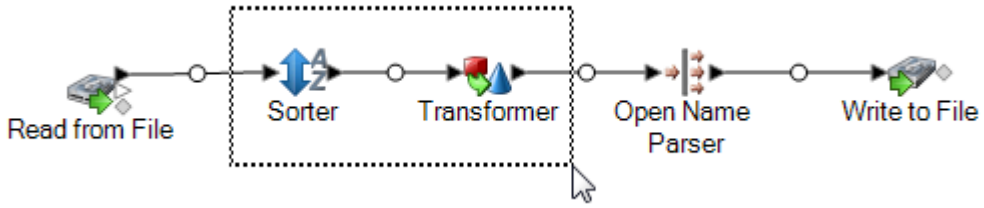
埋め込まれたデータフローとサブフローには、2 つの大きな違いがあります。まず、繰り返し処理は、埋め込まれたデータフローでしか使用できません。繰り返し処理を使うと、処理用にレコードを集約したり、フィールド値に基づいてステージのオプションを設定したりするために、レコードのグループを処理できます。埋め込まれたデータフローとサブフローのもう 1 つの相違点は、埋め込まれたデータフローが複数のデータフローで使用できないことです。データフローの一部を複数のデータフローで再利用する場合は、埋め込まれたデータフローの代わりにサブフローを作成します。埋め込まれたデータフローを他のデータフローで再利用することにした場合、埋め込まれたデータフローをサブフローに変換できます。ただし、埋め込まれたデータフローをサブフローに変換すると、繰り返しオプションが削除されます。

埋め込まれたデータフローにステージをグループ化する

埋め込まれたデータフローは、複数のステージを 1 つのステージにグループ化して、複雑なデータフローのレイアウトを簡素化し、フィールド値を使って処理オプションを設定できます。

1. データフローでは、埋め込まれたデータフローに変換するステージを追加します。
2. 埋め込まれたデータフローに変換するステージを選択するには、クリックとドラッグでボックスを描き、目的のステージを囲みます。

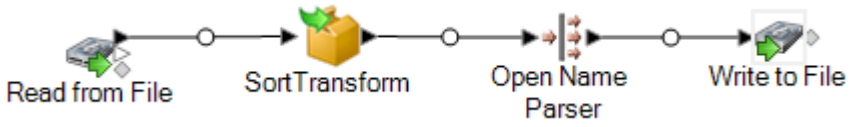
   例えば、次の例では、Sorter ステージと Transformer ステージを埋め込まれたデータフローに変換するために選択しています。



   注： Report ステージは、埋め込まれたデータフローに追加できません。

3. 選択したステージの 1 つを右クリックし、**[埋め込まれたデータフローにグループ化]** を選択します。
4. 埋め込まれたデータフローの名前を入力します。この名前は、キャンバス上に表示される埋め込まれたデータフローのラベルに使われます。
5. **[OK]** をクリックします。

以上の操作で、選択したステージは、埋め込まれたデータフローにグループ化されます。次の例では、SortTansform と名付けた、埋め込まれたデータフローを使用します。

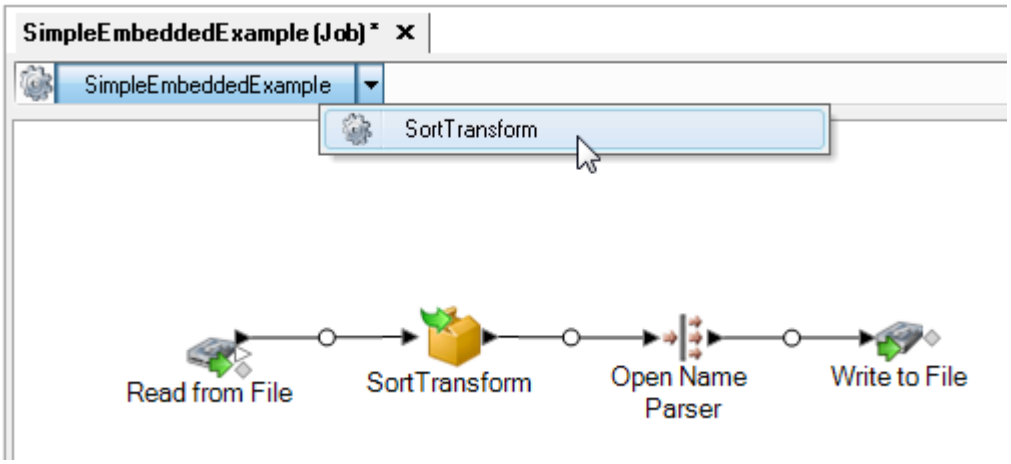


埋め込まれたデータフローの編集

埋め込まれたデータフローは、複数のステージを 1 つのステージにグループ化して、複雑なデータフローのレイアウトを簡素化し、フィールド値を使って処理オプションを設定できます。

1. アイコンを右クリックし、**[この埋め込まれたデータフローを編集]** を選択します。

   ヒント： あるいは、データフローの最上部にあるブレッドクラム リンクを使って、埋め込まれたデータフローを開くことができます。例えば、次の例は、SortTransform という名前の埋め込まれたデータフローを開きます。



埋め込まれたデータフローを開くと、Input ステージと Output ステージが表示されます。これらのステージは、埋め込まれたデータフローへの親データフローからの入力と、埋め込まれたデータフローから親データフローへの出力を表します。

2. 埋め込まれたデータフローを必要に応じて変更します。

   注: データフロー インスペクションは、埋め込まれたデータフローで使用できません。

埋め込まれたデータフローに加えた変更は、次回の親データフローの保存時に保存されます。

埋め込まれたデータフローで繰り返しを使用

繰り返し設定を使って、埋め込まれたデータフローが入力レコードを処理する方法を指定できます。デフォルトで、埋め込まれたデータフローは、データフローで他のステージがそうするように各レコードを個別に処理します。しかし、繰り返しを使うと、複数のレコードを一括して処理できます。この機能は、入力データ全体ではなくレコードのグループに基づいて比較や演算を実行する場合などに便利です。また、繰り返しを使って、各レコードのデータに基づくステージ オプションを設定できます。

繰り返しには、レコード単位の繰り返しとグループ単位の繰り返しの 2 つがあります。レコード単位の繰り返しでは、埋め込まれたデータフローで一度に 1 つのレコードが処理され、その結果が、埋め込まれたデータフローの次にあるステージに送られます。レコード単位の繰り返しが便利なのは、レコードごとにフィールド値を使ってステージ オプションを設定したい場合です。

グループ単位の繰り返しでは、レコードがキー フィールドでグループ化され、埋め込まれたデータフローが各グループを処理します。グループ内のすべてのレコードが 1 回の繰り返しで処理され、埋め込まれたサブフローの次にあるステージにグループが書き込まれます。グループ単位の繰り返しを使って、関連性があるレコードのグループに処理を実行できるほか、レコードのグループの処理中にステージ オプションも設定できます。例えば、顧客 ID でレコードをグループ化す

れば、顧客別にレコードを分析できます。特定の顧客が最もひいきにする店舗などがわかるでしょう。

繰り返しを使う際は、パフォーマンスへの影響を考慮してください。繰り返しが新たに開始されるたびに、埋め込まれたデータフローを初期化する処理によってオーバーヘッドが生じます。埋め込まれたデータフローが、他の埋め込まれたデータフローに含まれている場合には、このオーバーヘッドが特に大きくなります。例えば、埋め込まれたデータフローが 1,000 回繰り返され、そこに含まれる埋め込まれたデータフローが 1,000 回繰り返されるとしたら、繰り返しの回数は合計で 100 万回に達します。レコード単位の繰り返しではレコードごとに新しい繰り返しが発生するので、パフォーマンスへの影響はさらに大きくなります。

1. 埋め込まれたデータフローにステージを追加し、その中で繰り返しを使ってください。

   注： 繰り返しを使用する埋め込まれたデータフローに追加できるものには、以下の制限があります。

   - Stream Combiner ステージは、繰り返しを使用する埋め込まれたデータフローの最初のステージとして使うことができません。
   - 埋め込まれたデータフローには、クライアント側にあるファイルへ書き込みを行うシンクを追加できません。埋め込まれたデータフロー内のシンクは、Spectrum™ Technology Platform サーバーまたはファイル サーバー上のファイルに書き込む必要があります。

2. 埋め込まれたデータフローのアイコンをダブルクリックします。
3. **[繰り返しを有効にする]** チェック ボックスをオンにします。
4. 1 つ以上の入力チャネルが埋め込まれたデータフローに接続されている場合、**[ポート]** フィールドを使って、レコードを繰り返しの実行に使うポートを選択します。

   例えば、2 つの入力ポート、A と B があり、キー フィールドが変化するたびに繰り返しを選択するとします。ポート B を繰り返しに使うために選択した場合、埋め込まれたデータフローは、ポート B からのレコードでキー フィールドが変化するたびに繰り返しを新たに開始します。もう一方のポート A からのすべてのレコードは、埋め込まれたデータフローに読み込まれ、キャッシュされ、繰り返しのたびに使用されます。

5. 実行する繰り返しのタイプを選択します。

   キー フィールドが変化するたびに繰り返し
   このタイプの繰り返しでは、埋め込まれたデータフローは、1 つ以上のフィールドに同じ値を持つレコードのグループを処理します。埋め込まれたデータフローがレコードのグループの処理を終えると、埋め込まれたデータフローはリセットされ、別のレコードのグループが処理されます。このタイプの繰り返しを使って作成する埋め込まれたデータフローでは、レコードのグループを処理した後でレコード グループごとに個別に出力します。

ヒント： このタイプの繰り返しを選択する場合は、埋め込まれたデータフローの直前に Sorter ステージを配置し、キー フィールドを基準にレコードをソートしておくとパフォーマンスが向上します。

レコードごとに繰り返し このタイプの繰り返しでは、埋め込まれたデータフローは、一度に 1 つのレコードを処理します。1 つのレコードについて、埋め込まれたデータフローの処理が完了すると、結果が出力に送られ、次のレコードが処理されます。レコードごとに繰り返す埋め込まれたデータフローは、各レコードを新規のデータフロー実行として扱います。

6. **[キー フィールドが変化するたびに繰り返し]** を選択した場合は、**[値を比較するとき大文字と小文字を区別しない]** チェック ボックスをオンにすると、レコードのグループを決定するキー値の評価時の大文字と小文字の区別を無くすことができます。
7. 1 つ以上のキー フィールドを指定します。
   a) **[追加]** をクリックします。
   b) キー フィールドとして使うフィールドを選択します。
   c) 埋め込まれたデータフローでフィールドの値をステージ オプションの設定に使う場合は、設定するオプションの名前を指定します。
   d) **[OK]** をクリックします。
   e) 必要に応じて、さらにキー フィールドを追加します。

   キー フィールドが複数あり、**[キー フィールドが変化するたびに繰り返し]** オプションを選択した場合は、すべてのキー フィールドの値が同じレコードがグループ化されます。

埋め込まれたデータフローをグループ解除

埋め込まれたデータフローのグループ化を解除すると、埋め込まれたデータフローからステージが取り出され、埋め込まれたデータフローがあった位置で親データフロー内に配置されます。埋め込まれたデータフローに繰り返し設定があった場合は、削除されます。

埋め込まれたデータフローのステージをグループ解除するには、埋め込まれたデータフローを右クリックし、**[この埋め込まれたデータフローをグループ解除]** を選択します。

埋め込まれたデータフローをサブフローに変換

埋め込まれたデータフローを別のデータフローに再利用するには、埋め込まれたデータフローをサブフローに変換する必要があります。こうするのは、埋め込まれたデータフローをそのままで

は他のデータフローで使用できないためです。埋め込まれたデータフローは、サブフローに変換した後は他のサブフローと同様に使用できます。

> **注：** 埋め込まれたデータフローで繰り返しが有効になっている場合は、サブフローへの変換で繰り返し設定が削除されます。サブフローでは、繰り返しがサポートされません。

1. Enterprise Designer で、サブフローに変換したい埋め込まれたデータフローが含まれるデータフローを開きます。
2. 埋め込まれたデータフローを右クリックし、**[ステージをサブフローに変換]** を選択します。
3. 新しいサブフローの名前を入力し、**[OK]** をクリックします。

埋め込まれたデータフローがサブフローに変換され、Enterprise Designer パレットの [ユーザ定義ステージ] フォルダで使用可能になります。

レポート

Spectrum™ Technology Platform は、ジョブに対するレポート機能を提供します。一部のモジュールに付属する標準レポートを使用するか、独自のレポートを設計することができます。データフローにレポートが含まれる場合、データフロー全体が実行され、完了後にデータフロー内のレポート ステージが実行されて、レポートは選択されたフォーマット (PDF など) で保存されます。

ジョブへの標準レポートの追加

標準レポートとは、Spectrum™ Technology Platform モジュールに含まれる事前設定されたレポートです。例えば、Location Intelligence モジュールには、Point In Polygon Summary Report が含まれます。このレポートには、一致したポリゴンの数やジョブに使用されたデータベースといった情報を含む、Point In Polygon の計算結果の概要が示されます。

ジョブに標準レポートを追加する手順は以下のとおりです。

1. Enterprise Designer において、ウィンドウ左側の [パレット] の下で **[レポート]** をクリックします。
   使用可能なレポートの一覧が表示されます。
2. 追加したいレポートをキャンバス上にドラッグします。レポート アイコンを何かに接続する必要はありません。
3. レポートをダブルクリックします。
4. このレポートの作成に使用するステージを選択します。

5. **[パラメータ]** タブをクリックします。
6. **[デフォルトのレポート オプションを使用]** チェック ボックスをオフにし、PDF 以外のフォーマット (html や txt など) を指定する場合は、適切な出力フォーマットを選択します。

ジョブに対するレポート オプションの設定

レポートには、処理されたレコード数やジョブに使用された設定など、ジョブに関するサマリ情報が提供されます。レポート オプションは、出力フォーマットやアーカイブ オプションなど、ジョブによって生成されたレポートの処理方法を指定するものです。レポート オプションのデフォルト値は Management Console で指定されますが、Enterprise Designer においてジョブに対するデフォルト オプションをオーバーライドできます。

ジョブに対するレポート オプションを指定する手順は以下のとおりです。

1. Enterprise Designer でジョブを開き、**[編集] > [ジョブ オプション]** を選択します。
2. **[レポート]** タブをクリックします。
3. **[グローバル レポート オプションを使用]** チェック ボックスをオフにします。
4. レポートの保存に使用する形式を選択します。レポートは HTML、PDF、またはテキストとして保存できます。
5. レポートの保存場所を選択します。

   レポートをジョブ履歴に保存 レポートをジョブ履歴の一部としてサーバーに保存します。こうすると、Management Console および Enterprise Designer ユーザはレポートを実行履歴で参照できるようになって便利です。

   レポートをファイルに保存 指定した場所にあるファイルにレポートを保存します。これは、Spectrum™ Technology Platform ユーザではない人とレポートを共有したい場合に便利です。また、レポートのアーカイブを別の場所に作成したい場合にも便利です。この方法で保存されたレポートを表示する場合は、そのレポートの形式を開くことができる任意のツール (PDF レポートの場合は PDF ビューア、HTML レポートの場合は Web ブラウザ) を使用できます。

6. **レポートをファイルに保存** を選択したは、次のフィールドに入力します。

   レポートの場所 レポートの保存先フォルダです。

   レポート名に追加 ファイル名に含める変数の情報を指定します。以下から 1 つ以上を選択できます。

      ジョブ ID ジョブの実行に対して割り当てられる一意の ID です。システムで初めて実行したジョブの ID は 1 になります。2 回目

にジョブを実行したときには、それが前回と同じジョブでも異なるジョブでも、ジョブ ID は 2 になります (3 回目以降も同様)。

ステージ レポートにデータを提供したステージの名前であり、Enterprise Designer 内のレポート ステージで指定されているものです。

Date レポートが作成された年月日です

既存のレポートを上書きする 以前のレポートを、同じファイル名を持つ新しいレポートで置き換えます。新しいレポートと同じ名前の既存レポートがあるのにこのオプションを選択していない場合、ジョブは正常に完了しますが、新しいレポートは保存されません。その場合は、レポートが保存されなかったことを示すコメントが実行履歴に表示されます。

7. **[OK]** をクリックします。

ジョブを実行すると、そのジョブに関連するレポートがあるかどうかを示す列が [実行履歴] に表示されます。空のアイコンはレポートが存在しないことを表し、ドキュメントを 1 つ含むアイコンはレポートが 1 つ存在することを表し、ドキュメントを複数含むアイコンは複数のレポートが存在することを表します。[ジョブの詳細] で、レポートを表示、保存、印刷することができます。

> **注：** レポートを削除するには、キャンバス上のレポート アイコンを右クリックし、**[削除]** を選択します。

レポートの表示

レポートを表示するには、ジョブを実行してから、次のいずれかの操作を行います。

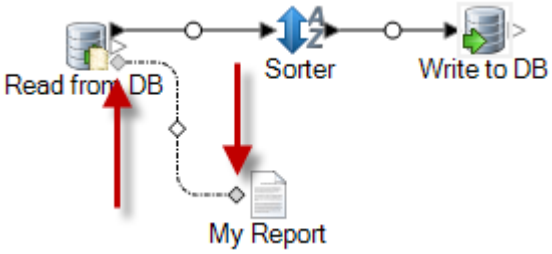
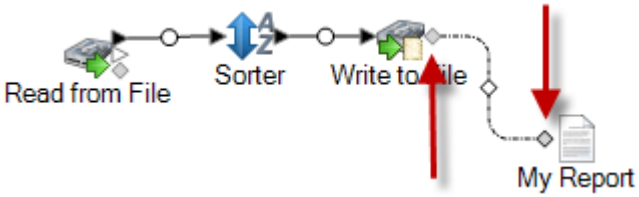
- ジョブを実行すると、Enterprise Designer で **[実行の詳細]** ウィンドウが表示されます。表示するレポートを選択します。
- Management Console の [実行] ノードで、**[履歴]** をクリックし、レポートを表示するジョブを選択して **[詳細]** をクリックします。

カスタム レポートの使用

Spectrum™ Technology Platform モジュールには、基本レポートに役立つレポートが付属しています。しかし、標準レポートでは対応できないレポート要件がある場合は、独自のカスタム レポートを作成してデータフローに含めることができます。

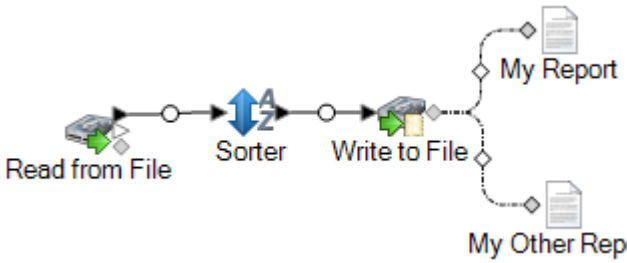
1. 任意のレポート設計ツールを使用して、レポート テンプレートを作成します。設計ツールは、JasperReports フォーマット (.jrxml) でレポートをエクスポートできるものでなければなりません。
2. .jrxml ファイルを、`server\app\import` サーバー上の Spectrum™ Technology Platform フォルダにコピーします。

   数秒でレポート テンプレートがシステムにインポートされ、Enterprise Designer で使用可能になります。
3. Enterprise Designer において、カスタム レポートを追加するジョブを開きます。
4. ウィンドウ左側の [パレット] の下で **[レポート]** をクリックします。
5. カスタム レポートをキャンバスにドラッグします。
6. 次のいずれかを行うことにより、レポートのデータ ソースを指定します。

オプション	説明
データフローの入力に関するレポートを作成する場合	以下に示すようにグレーのひし形のレポート ポートを使用して、レポートを作成したいソース ステージにレポートを接続します。  レポートは、データフローの入力データに基づくものとなり、データフロー内で生じる処理はまったく反映されません。
データフローの出力に関するレポートを作成する場合	以下に示すようにグレーのひし形のレポート ポートを使用して、レポートを作成したいシンク ステージにレポートを接続します。  レポートは、データフローの出力データに基づくものとなり、データに対するデータフローの影響が反映されます。
レポート テンプレートに埋め込まれたクエリを使用する場合	レポート テンプレートにおいて、JRXML ファイルの `<queryString>` 要素の中に SQL クエリが埋め込まれている場合、レポート アイコン

オプション	説明
	をダブルクリックして、**[組み込みクエリの使用]** ボックスをオンにしてから、クエリに使用するデータベース接続を選択します。 **注：** データベース接続を定義する必要がある場合は、Management Console を開き、**[リソース]**、**[接続]** の順に選択します。

以下に示すように、複数のレポートを 1 つのソースまたはシンクに接続できます。



7. レポートにユーザ定義パラメータが含まれる場合は、
   a) キャンバス上のレポート アイコンをダブルクリックします。
   b) **[パラメータ]** タブで、レポートのユーザ定義パラメータに使用する値を指定します。
8. オプション:必要な場合は、チャネルを右クリックし、ソースまたはシンクからのフィールドをレポート内のフィールドにマップします。

パフォーマンスに関する検討事項

パフォーマンス最適化のための設計ガイドライン

パフォーマンスが最適化されるようデータフローを慎重に設計することは、Spectrum™ Technology Platform のパフォーマンス向上のために実行できる最も重要な作業です。これらのガイドラインでは、データフローのパフォーマンス最適化に使用できる手法について説明します。

ステージ数の最小化

Spectrum™ Technology Platform は、並列処理によって高パフォーマンスを実現します。1 つのフロー内の各ステージは、それぞれのスレッドで非同期に実行します。ただし、ある種のデータフローを実行するときプロセッサがオーバースレッドになる可能性があります。これは、"実際の仕事" を行う以上の時間がスレッドの管理に費やされることを意味します。130 の個別ステージを持つデータフローの場合、1 つまたは 2 つのプロセッサを搭載した小規模なサーバーでのパフォーマンスが非常に悪いことが確認されています。

したがって、パフォーマンスに優れたデータフローを設計するためにまず検討すべきは、必要なだけのステージを使用し、それ以上は使用しないということです。必要以上のステージを使用した例としては、次のものがあります。

- 1 つで十分にもかかわらず複数のconditional routerを使用する
- 複数のトランスフォームを 1 つのステージに結合する代わりに複数の Transformer ステージを定義する

幸いなことに、通常はこのようなデータフローを設計し直して、冗長または不要なステージを削除し、パフォーマンスを向上させることができます。

フローが複雑な場合は、埋め込みタイプのフローまたはサブフローを使用してキャンバス上が乱雑にならないように整理することを検討してください。それによって、フローが見やすくなり、操作しやすくなります。埋め込みタイプのフローを使用しても実行時のパフォーマンス上のメリットはありませんが、Enterprise Designer でのフローの操作が容易になります。サブフローを使用して複雑なフローを簡素化すれば、フロー編集時の Enterprise Designer のパフォーマンスが向上します。

レコード長の短縮

同時並行で実行されるステージ間でデータが渡されるため、入力レコードの長さについても検討する必要があります。一般的に、長いレコードの入力では、短いレコードの入力に比べ、処理に時間がかかります。これは、読み込み、書き出し、ソートするデータが多いためです。複数のソート操作を行うデータフローは、レコード長を短くすることで特に恩恵を受けます。非常に長いレコードを処理する場合は、Spectrum™ Technology Platform ジョブを実行する前に入力から不要なフィールドを削除し、その後、生成された出力ファイルにフィールドを戻すことで、処理速度を向上させることができます。

ソートの適切な使用

ソート操作の最小化についても検討する必要があります。ソートは他の操作よりも時間がかかることが多いため、入力レコードの数およびサイズが増加すると問題となる場合があります。しかし、多くの Spectrum™ Technology Platform ステージが、ソートされた入力データを必要としたり、優先したりします。例えば、Universal Addressing モジュールと Enterprise Geocoding モジュールは、入力が国および郵便番号によってソートされている場合にパフォーマンスが最適化されます。Intraflow Match や Interflow Match などのステージでは、入力が [グループ化方法] フィー

ルドによってソートされている必要があります。外部ソート アプリケーションを使用して、入力データを事前にソートできる場合もあります。この場合、Spectrum™ Technology Platform データフロー内でのソートより速く処理できます。

ステージの実行時パフォーマンス オプション

実行時パフォーマンス オプションは、データフロー内の個々のステージの実行方法を制御し、データフローのパフォーマンスを高める設定を提供します。使用可能な設定は、Spectrum™ Technology Platform 環境の設定方法によって異なります。

- **[ローカル]** オプションは、ステージをローカル Spectrum™ Technology Platform サーバーで実行し、そのステージで 1 つの実行時インスタンスを使用するデフォルト設定です。実行時インスタンスの設定は増やすことができます。したがって、並行処理を利用してパフォーマンスを高めることができます。
- **[分散]** オプションは通常、ロード バランサーと複数の Spectrum™ Technology Platform サーバーがインストールされるクラスタ環境で使用されます。
- **[リモート]** オプションは、複数の Spectrum™ Technology Platform サーバーで構成されているが、分散処理が行えるように設定されていない環境で使用できます。このオプションを使用すると、ステージの処理を別のサーバーで実行できます。

データベースのプール サイズと実行時インスタンス数

ほとんどの Spectrum™ Technology Platform 環境において、バッチ ジョブであるか、Web サービスまたは API 要求に応答するサービスであるかの違いはあれ、複数のフローが同時に実行しています。同時処理を最適化するために、データベースのプール サイズ設定が使用できます。これによって、Spectrum のデータベースが処理する同時要求数と、同時に実行するフロー ステージのインスタンス数を制御する実行時インスタンス数が制限されます。最適なパフォーマンスを得るには、これら 2 つの設定を同時にチューニングする必要があります。

データベースのプール サイズ

Spectrum データベースには、住所の検証で使用する郵便データや、住所のジオコードで使用するジオコーディング データなど、特定のステージで使用するリファレンス データが含まれます。これらのデータベースは、それを使用するデータフロー ステージやサービスから同時に行われる複数の要求を受け付け、データフローの要求やサービスの要求のパフォーマンスを高めるように設定することが可能です。データベースのプール サイズは、Spectrum のデータベースが処理する同時要求の最大数を設定します。デフォルトでは、Spectrum データベースのプール サイズは 4 で、データベースは 4 つの要求を同時に処理できることを意味します。

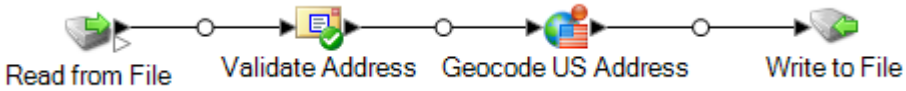
最適なプール サイズはモジュールによって異なります。一般的には、サーバーが搭載する CPU の数の半分から 2 倍のプール サイズを設定すると、最適な結果が得られます。ほとんどのモジュー

ルに最適なプール サイズは CPU 数と同数です。例えば、サーバーが 4 つの CPU を搭載している場合は、プール サイズを 2 (CPU 数の半分) ～ 8 (CPU 数の 2 倍) の間で試すことができ、多くの場合、最適なサイズは 4 (CPU 数と同数) です。

プール サイズを変更するときは、データベースにアクセスするステージ用としてデータフローに指定されている実行時インスタンスの数を考慮する必要もあります。例えば、1 つの実行時インスタンスを使用するように設定された Geocode US Address ステージを持つデータフローがあるとします。この場合、米国ジオコーディング データベースのプール サイズを 4 に設定しても、パフォーマンスは向上しません。実行時インスタンスが 1 つしかないので、データベースへの要求は一度に 1 つになります。ただし、Geocode US Address の実行時インスタンスの数を 4 つに増やすと、パフォーマンスが向上します。データベースに同時にアクセスする Geocode US Address のインスタンスが 4 つあるので、プール全体を使用できます。

実行時インスタンス

データフローの各ステージはそれぞれのスレッドで非同期に動作し、他のステージから独立しています。このため、データフロー内の複数ステージが並列処理され、1 つのステージに対して複数の実行時インスタンスを利用することができます。これは、データの処理時間が異なるステージで構成されるデータフローで役に立ちます。その結果、スレッド間での作業の分配のバランスが悪くなる可能性があります。例えば、次の 2 つのステージで構成されるデータフローを考えてみましょう。



ステージの設定によって、Validate Address ステージが Geocode US Address ステージより速くレコードを処理することがあります。その場合、データフロー実行のある時点で、Validate Address はすべてのレコードを処理していますが、Geocode US Address にはまだ未処理のレコードがあります。このデータフローのパフォーマンスを向上させるには、最も速度の遅いステージ (この場合は Geocode US Address) のパフォーマンスを向上させる必要があります。その方法の 1 つは、ステージの実行時インスタンスを複数指定することです。例えば、実行時インスタンス数を 2 に設定すると、そのステージのインスタンスが 2 つになります。各インスタンスはそれぞれのスレッド内で動作し、レコードの処理に使用することができます。

一般的なルールとして、実行時インスタンス数は、少なくともリモート コンポーネントのインスタンス数と等しくなければなりません。リモート コンポーネントの詳細については、『*管理ガイド*』を参照してください。複数の実行時インスタンスを指定するとパフォーマンスが向上しますが、この値を高くしすぎると、システムのリソースに負荷がかかり、パフォーマンスが低下する可能性があります。

> **注：** 複数の実行時インスタンスを使用してパフォーマンスが向上するのは、複数のレコードを使用するジョブまたはサービス要求を実行する場合のみです。

チューニング手順

データベースのプール サイズと実行時インスタンス数に適切な値を設定するには、さまざまな設定を試して、リソースに過度に負荷をかけたりパフォーマンスを低下させたりすることなく、使用可能なサーバー リソースを最大限に活用できる設定を見つける必要があります。

> **注 :** データベースのプール サイズを調整する前にデータフローのプール サイズを最適化する必要があります。データフローのプール サイズの最適化については、「**データフローのプール サイズ**」を参照してください。

1. まず、さまざまな設定を試す際に使用するサンプル データを見つけます。実行時間の測定や一貫性の確認ができるように、サンプル データセットは十分に大きい必要があります。またサンプル データは、実際に処理するデータを代表するものでなければなりません。例えば、ジオコーディングのパフォーマンス テストを行う場合は、ジオコードする予定のすべての国に対して同数のレコードがテスト データに含まれるようにします。
2. 郵便データベースやジオコーディング データベースなど、データベース リソースの使用が必要なサービスまたはデータフローをテストする場合は、データベースの最新版がインストールされていることを確認してください。
3. サンプル データを用意し、最新のデータベース リソースをインストールしたら、ファイルからデータを読み込み、最適化したいステージでそれを処理し、ファイルに書き出す、簡単なデータフローを作成します。例えば、Validate Address のパフォーマンス設定をテストする場合は、Read from File、Validate Address、Write to File で構成されるデータフローを作成します。
4. データベース リソースのプール サイズを 1 に設定します。
   a. Management Console を開きます。
   b. **[リソース] > [Spectrum データベース]**に移動します。
   c. 最適化するデータベース リソースを選択し、変更ボタン をクリックします。
   d. **[プール サイズ]** フィールドに、1 と入力します。
   e. **[OK]** をクリックします。
5. ステージの実行時インスタンスを 1 に設定します。
   a. Enterprise Designer でデータフローを開きます。
   b. 複数の実行時インスタンスを使用するように設定するステージをダブルクリックします。
   c. **[実行時]** をクリックします。

      > **注 :** すべてのステージで複数の実行時インスタンスを使用できるわけではありません。ステージのウィンドウの下部に **[実行時]** ボタンがない場合、そのステージでは複数の実行時インスタンスを使用できません。

   d. **[ローカル]** を選択して 1 を指定します。

e. **[OK]** をクリックして **[実行時パフォーマンス]** ウィンドウを閉じます。さらに **[OK]** をクリックしてステージを閉じます。

6. データフローを複数回実行し、次の各項目の平均値を記録することによって、ベースラインパフォーマンスを算出します。

   - 経過時間
   - CPU 使用率
   - メモリ使用率

   ヒント： JMX コンソールを使用してパフォーマンスをモニタリングできます。詳細については、「**JMX コンソールによるパフォーマンスのモニタリング**」を参照してください。

7. ジョブの複数インスタンスの同時実行をサポートする必要がある場合は、それを実行します。それぞれの場合に対して、経過時間、CPU 使用率、メモリ使用率を記録します。

   ヒント： ファイル モニターを使用して、ジョブの複数インスタンスを同時に実行できます。詳細については、「**コントロール ファイルによるフローのトリガー**（169ページ）」を参照してください。

8. データベース リソースのプール サイズと、ステージ実行時インスタンスの設定値を増加させます。
9. サーバーを再起動します。
10. データフローを再度実行し、経過時間、CPU 使用率、メモリ使用率を記録します。
11. パフォーマンスが低下し始めるまで、データベース リソースのプール サイズとステージ実行時インスタンスの設定値を増加していきます。
12. ジオコーディング パフォーマンスをテストする場合は、単一国と複数国の入力を使用してこの手順を繰り返します。

分散処理

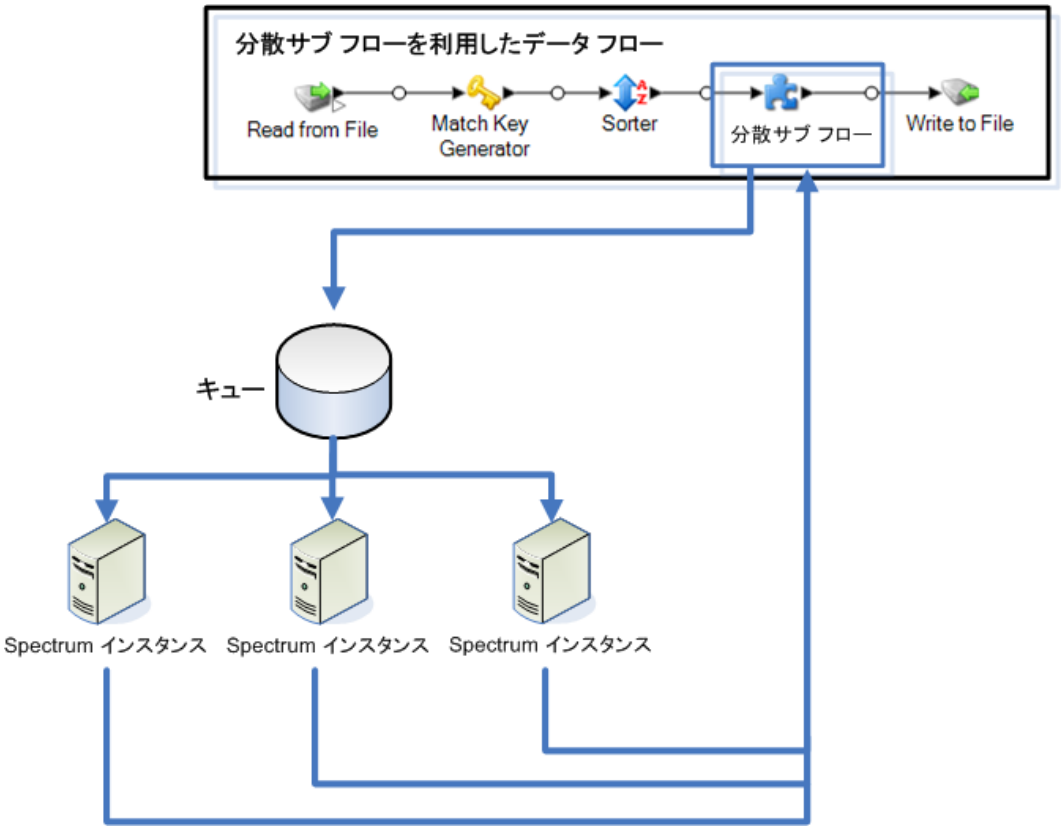
非常に複雑なジョブが存在する場合や、かなり大規模なデータ セット (何百万というレコードを含むものなど) を処理する場合は、1 台以上の物理サーバー上で Spectrum™ Technology Platform サーバーの複数のインスタンスにデータフローの処理を分散させることでデータフローのパフォーマンスを向上できる可能性があります。

最も拡張性の高い分散環境を構築する方法は、Spectrum™ Technology Platform をクラスタ内にインストールすることです。クラスタのインストールと設定手順については、『*インストール ガイド*』を参照してください。

> **注：** 単一の Spectrum™ Technology Platform サーバー上で分散処理を使用することも可能ですが、以下では、クラスタ内の分散処理の使用について説明します。単一のサーバーを使用している場合、分散サブフロー処理はマイクロバッチに分割され、クラスタではなく 1 台のサーバーによって処理されます。

クラスタ環境のセットアップが完了したら、複数のサーバーに分散させるデータフローの各パーツに対するサブフローを作成することで、分散処理をデータフロー内に組み込むことができます。サブフローに対していくつかの設定オプションを指定した後は、Spectrum™ Technology Platform で自動的に処理の分散が管理されます。

次の図は、分散処理を表したものです。



レコードがサブフロー内に読み込まれると、そのデータが各バッチにグループ化されます。その後、これらのバッチはクラスタに書き込まれ、バッチを処理するクラスタ内のノードに自動的に配布されます。この処理をマイクロフローと呼びます。サブフローは、複数のマイクロフローを同時に処理して、データフローのパフォーマンスを潜在的に向上させるように設定できます。分散インスタンスは、マイクロフローの処理を終了すると、その出力を親のデータフローに送り返します。

Spectrum™ Technology Platform のノード数が多いほど、同時に処理できるマイクロフロー数が増加するので、要求されるパフォーマンスを取得するために必要に応じて環境のスケーリングが可能です。

セットアップの完了後、クラスタ環境は容易に管理できます。クラスタ内のすべてのノードが各自の設定の同期を自動的に行っており、これは Management Console を利用して適用する設定や、Enterprise Designer で設計するデータフローが自動的にすべてのインスタンスで利用できるようになることを意味するからです。

分散処理用のデータフローの設計

分散処理は、データフローの各パーツを受け取り、それらのパーツの処理を Spectrum™ Technology Platform サーバー群のクラスタに分散させます。例えば、データフローでジオコーディングを実行する場合は、パフォーマンス向上のために、ジオコーディング処理をクラスタ内の複数の Spectrum™ Technology Platform ノード間で分散させることができます。

1. データフローのどのステージを分散させるのかを決定したうえで、分散化するステージを含むサブフローを作成します。

   次のステージは、分散処理で使用されるサブフローでは使用しないでください。

   - Sorter
   - Unique ID Generator
   - Record Joiner
   - Interflow Match

   次のステージは、分散処理用のサブフロー内で一緒に使用する必要があります。

   - マッチング ステージ (Intraflow Match、Transactional Match) および統合ステージ (Filter、Best of Breed、Duplicate Synchronization)
   - Aggregator および Splitter

   サブフロー内には別のサブフロー (ネストされたサブフロー) を含めないでください。

   分散処理に使用されるサブフロー内でマッチング操作を実行する場合は、次の点に注意してください。

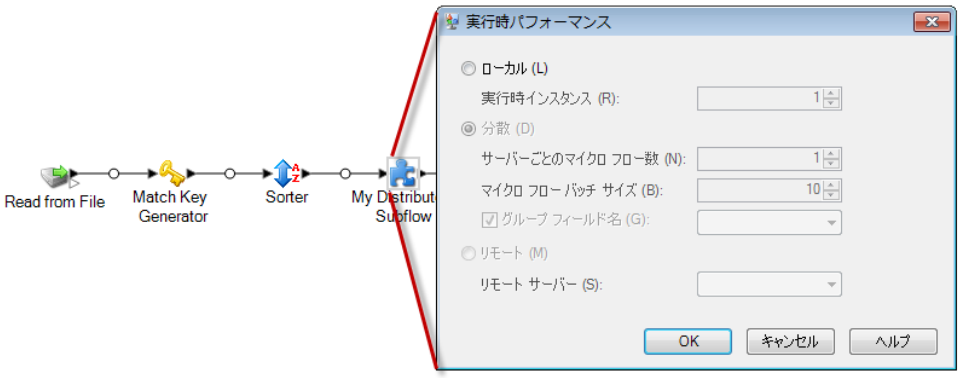
   - ソートは、サブフロー内ではなくジョブ内で行う必要があります。ソートは、ステージ内ではオフにし、ジョブ レベルで配置する必要があります。
   - マッチ分析は、分散サブフローではサポートされていません。
   - コレクション番号は、マイクロフロー バッチ グループ内で再利用されます。

   サブフローで Write Exception ステージを使用すると、予期せぬ結果が生じることがあります。代わりに、ジョブ レベルでこのステージをデータフローに追加してください。

2. 分散化するデータフローの一部としてサブフローを作成した後は、そのサブフローを親データフローに追加し、上流および下流ステージに接続します。分散処理で使用されるサブフローは、入力ポートを 1 つだけ持つことができます。
3. サブフローを右クリックして、**[オプション]** を選択します。
4. **[分散]** を選択します。
5. 各サーバーに送信するマイクロフローの数を入力します。
6. 各マイクロフロー バッチに含めるレコードの数を入力します。
7. オプション: (オプション) **[グループ フィールド名]** にチェックを入れ、マイクロフロー バッチをグループ化するフィールドの名前を選択します。

グループ フィールドを指定した場合は、グループが複数のバッチに分割されないので、バッチ サイズが **[マイクロ フロー バッチ サイズ]** フィールドで指定した値より大きくなる可能性があります。例えば、バッチ サイズとして 100 を指定したのに同じグループ内に 108 のレコードがある場合、バッチには 108 レコードが含まれます。同様に、バッチ サイズとして 100 を指定し、同じ ID を持つ 28 レコードの新しいグループがレコード 80 から始まる場合、そのバッチには 108 レコードが含まれます。

以下に、My Distributed Subflow という名前のサブフローが分散モードで実行されるように設定されたデータフローの例を示します。



リモート サーバー上でのサービスの実行

システム管理者が Management Console でリモート サーバーを有効にしている場合、データフロー内のステージの処理をリモート サーバー上で実行できます。リモート サーバーを使用すると、データフローの処理を複数の Spectrum™ Technology Platform サーバーに拡散してパフォーマンスを高めることができます。

システム管理者は、既に一部のステージをリモート サーバー上で実行するように指定している可能性があります。ステージが既にリモート サーバーにルーティングされている場合は、Enterprise Designer のキャンバス上のステージ アイコンの左上隅に赤の星印が表示されます。

次の手順では、データフローのステージをリモートで処理するように設定する方法について説明します。

1. Enterprise Designer でデータフローを開きます。
2. リモート サーバーにルーティングするステージをダブルクリックします。
3. **[実行時]** をクリックします。

   [実行時パフォーマンス] ダイアログが表示されます。
4. **[リモート]** をクリックし、そのステージの処理をルーティングするリモート サーバーを選択します。
5. **[OK]** をクリックします。

リモート サーバーのエラーのトラブルシューティング

ここでは、リモート サーバーの使用中に発生する可能性があるエラーについて説明します。

モジュールのライセンスがない

リモート サーバーは、モジュールと実行モード (バッチまたはリアルタイム) の両方に対して、ライセンスが必要です。リモート サーバー上のライセンスは、ローカル サーバー上のライセンスとは異なる可能性があります。Management Console を使用してリモート サーバーにログインし、正しいライセンスがインストールされていることを確認してください。ライセンス情報を表示するには、管理者権限を持つアカウントでログインする必要があります。

リモート サーバーは使用できません

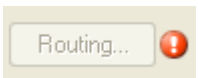
リモート サーバーが実行されていない場合、またはリモート サーバーが何かの理由で到達不能な場合、Enterprise Designer と Management Console ではリモート サービスを使用できません。画面下部のステータス バーに黄色のハザード アイコンが表示されます。

このアイコンをクリックすると、使用できないリモート サーバーについて説明するエラー メッセージが表示されます。

また、Enterprise Designer では、リモート ステージを使用するすべてのステージが、今後ステージを使用できないことを示すアイコンに置き換えられます。

ルーティングが変化した

ローカルとリモートの両方にインストールされていて、リモート サーバーにルーティングされるサービスを、削除または展開解除した後、Management Console でそのサービスをクリックすると、そのサービスの [オプション] タブのルーティング ボタンの横に、ルーティング変更インジケータ (点滅する感嘆符) が表示されます。このインジケータは、そのサービスのルーティングが変化したことを意味します。



ステージの最適化

マッチングの最適化

通常、マッチングはあらゆるデータ品質実装の中で最も時間のかかる操作の 1 つであり、できる限り効率的にマッチングを実行することが重要になります。マッチング結果とパフォーマンスは、常にバランスが保たれています。ファイル内の各レコードをそれ以外のすべてのレコードと比較する場合、すべてのマッチを確実に特定できます。しかし、データ量が増大すると、このアプローチは持続できなくなります。例えば、100 万レコードからなる入力ファイルがある場合、各レコードをそれ以外のすべてのレコードとマッチングすると、各マッチ ルールを評価するのにほぼ 1 兆回の比較が必要になります。

ファイル内の大部分のレコードがマッチしない場合、この問題を解決する一般的なアプローチは、マッチ キーを定義して、同じマッチ キーを持つレコードとのみ比較することです。適切なマッチ キーを定義することが、マッチング エンジンのパフォーマンスに影響を与える最も重要な変数です。適切なマッチ キーを定義するには、マッチング エンジンがレコードをどのように処理するか、および使用可能なオプションについて理解する必要があります。

デフォルトのマッチング方法では、マッチ キュー内のレコードをすべて比較して、最大数のマッチを特定します。そのため、この方法はしばしば最も時間のかかるマッチング方法になります。デフォルトのマッチング方法では、マッチ キュー内の先頭のレコードがサスペクト レコードになります。次のレコードが比較され、マッチした場合は重複として書き出されます。マッチしない場合はサスペクトとして追加され、次のレコードが 2 つのアクティブなサスペクトと比較されます。次のマッチ キューを考えてみましょう。

ユニーク ID	マッチ キー
1	123A
2	123A
3	123A
4	123A
5	123A
6	123A

ユニーク ID	マッチ キー
7	123A
8	123A
9	123A
10	123A

まず、レコード 2 がレコード 1 と比較されます。マッチしない場合、レコード 2 はサスペクトとして追加されます。次に、レコード 3 がレコード 1 および 2 と比較されます。以降も同様です。マッチング レコードが存在しない場合、比較の総回数は 45 回になります。マッチするレコードがある場合、比較回数はそれより少なくります。サイズ N のマッチ キューの場合、比較の最大回数は N×(N-1)÷2 です。キュー サイズが小さい場合は目立ちませんが、キュー サイズが増大すると、影響が大きくなります。例えば、キュー サイズ 100 の場合の比較回数は 4,450 回で、キュー サイズが 500 の場合の比較回数は 124,750 回です。

適切なマッチ キーの定義

適切なマッチ キーを定義するには、以下の点を考慮してください。

- 覚えておくべき最も重要な点は、大部分のレコードはマッチしないということです。したがって、マッチする可能性のあるレコードのみを比較したいと考えます。
- 同じマッチ キーを持つレコードのみを比較します。
- パフォーマンスは重要な検討事項です。
  - マッチ キーにより、マッチ キューのサイズが決定されます。
  - 一定のレコード数では、マッチ キュー サイズが倍になると、実行時間も倍になります。
  - "厳格" なマッチ キーによって、パフォーマンスが向上します。"厳格" なマッチ キーは限定的なマッチ キーで、多数のフィールドからの多数の文字で構成されます。
  - "あいまい" なマッチ キーによって、マッチ数が増加する場合があります。"あいまい" なマッチ キーはそれほど限定的ではないマッチ キーで、少数のフィールドからの少数の文字で構成されます。

パフォーマンスとマッチ結果のバランスを見出す

パフォーマンスと結果のバランスをうまく保つには、マッチ ルールとデータの密度について検討してください。

- マッチ ルールに関して、以下の点を考慮してください。
  - 完全一致を必要とするフィールドをマッチ キーに含めることができます。

  - マッチ ルールに適したキーを作成してください。例えば、発音表記に関するマッチ ルールでは、発音表記に関するマッチ キーが適していると考えられます。
  - 多くの場合、マッチ キーはマッチングするすべてのフィールドの一部で構成されます。
  - 欠落しているデータが及ぼす影響に注意してください。
- データ密度に関して、以下のことを考慮してください。
  - 例えば、住所マッチングにおいて、すべてのレコードがある国のデータセットではなくある 1 つの町にある場合、マッチ キーはより厳格になる可能性があります。
  - 平均ではなく最大のマッチ キューを検討してください。Match Summary Report を精査して、最大マッチ キューを判断します。
- Transactional Match を使用する場合、同じ考慮事項が Candidate Finder の SELECT 文にも当てはまります。

Express マッチ キー

一般的なファイルでは、大部分の重複レコードが完全に、またはほぼ完全に一致します。Express マッチ キーを定義すると、マッチング エンジンが Experss マッチ キーを最初に比較して、2 つのレコードが重複かどうかを判断できます。これにより、フィールド レベルのマッチ ルールをすべて評価する必要はなくなるので、パフォーマンスを大幅に向上させることができます。

Intraflow Match の方法

デフォルトの Intraflow Match マッチング方法では、同じマッチ キーを持つすべてのレコードが比較されます。マッチ キューのサイズが N の場合、デフォルトの方法では N−1 から N×(N−1) までのいずれかの回数の比較が実行されます。すべてのレコードがマッチした場合、比較回数は N-1 です。レコードがまったくマッチしなかった場合、比較回数は N×(N−1) です。たいていの場合、比較回数はこの範囲の後半部分のどこかの値になります。

パフォーマンスが優先事項である場合は、デフォルトの方法ではなく、スライディング ウィンドウ マッチング方法の使用を検討してください。スライディング ウィンドウ マッチング方法では、各レコードを次の W レコード (W はウィンドウ サイズ) と比較します。ファイル サイズが N の場合、スライディング ウィンドウ方法で実行される比較回数は最大で N×W です。この方法によってパフォーマンスは向上しますが、一部のマッチが検出されなくなることがあります。

Candidate Finder の最適化

Candidate Finder は、Transactional Match で比較する候補レコードをデータベースから選択します。サスペクト レコードを Candidate Finder が返すすべての候補レコードと比較するため、Transactional Match のパフォーマンスは比較回数に比例します。

しかし、Candidate Finder のパフォーマンス向上のためにできることがいくつかあります。Candidate Finder のパフォーマンスを最大化するには、データベース管理者またはデータベース スキーマおよびインデックスについて豊富な知識を持つ開発者が、Candidate Finder の SQL

SELECT 文をチューニングする必要があります。パフォーマンスに関する最も一般的な問題の 1 つは、テーブル全体のスキャンが必要となる JOIN を含んだクエリです。この場合は、インデックスの追加や、JOIN の代わりに UNION を使用することを検討してください。原則として、適任者が SQL クエリを確認し、最適化してください。

トランスフォームの最適化

Transformer ステージには、入力データに対して実行できる定義済みの操作セットがあります。これらの定義済みトランスフォームは既にコンパイルされているため、通常はカスタム トランスフォームより実行速度が速くなります。ただし、多数のトランスフォームを定義すると、カスタム トランスフォームの方がしばしば実行速度が速くなります。例えば、多数のフィールドをトリムする場合、通常は、9つの個別のトリム トランスフォームよりも、次のカスタム トランスフォームの方が実行速度が速くなります。

```
data['AddressLine1'] = (data['AddressLine1'] != null) ?
data['AddressLine1'].trim() : null;
data['AddressLine2'] = (data['AddressLine2'] != null) ?
data['AddressLine2'].trim() : null;
data['AddressLine3'] = (data['AddressLine3'] != null) ?
data['AddressLine3'].trim() : null;
data['AddressLine4'] = (data['AddressLine4'] != null) ?
data['AddressLine4'].trim() : null;
data['City'] = (data['City'] != null) ? data['City'].trim() : null;
data['StateProvince'] = (data['StateProvince'] != null) ?
data['StateProvince'].trim() : null;
data['PostalCode'] = (data['PostalCode'] != null) ?
data['PostalCode'].trim() : null;
data['LastName'] = (data['LastName'] != null) ? data['LastName'].trim()
 : null;
data['FirstName'] = (data['FirstName'] != null) ?
data['FirstName'].trim() : null;
```

Write to DB の最適化

デフォルトでは、Write to DB ステージは各行をテーブルに挿入した後で確定されます。ただし、パフォーマンスを向上するためには、**[一括確定]** オプションを有効にします。このオプションが有効になっている場合は、指定された数のレコードを処理した後に確定が行われます。データベースによっては、これによって書き出しパフォーマンスを大幅に向上させることができます。

バッチ サイズの選択時には、以下の点を考慮してください。

- **Write To DB ステージへのデータ到着速度:** データの到着速度がデータベースの処理速度よりも遅い場合は、バッチ サイズを変更してもデータフローの全般的なパフォーマンスは改善しません。例えば、データフローで住所検証またはジオコーディングが行われている場合は、バッチ サイズを大きくしても効果が得られないことがあります。

- **ネットワーク トラフィック:** 低速のネットワークでは、バッチ サイズを中程度 (1,000 ～ 10,000) まで大きくすると、パフォーマンスが改善します。
- **データベースの負荷や処理速度:** データベースの処理能力が高い場合は、バッチ サイズを大きくすると、パフォーマンスが改善します。
- **複数の実行時インスタンス:** Write to DB の複数の実行時インスタンスを使用している場合、バッチ サイズを大きくすると大量のメモリが消費されるので、小または中程度のバッチ サイズ (100 ～ 10,000) を使用してください。
- **データベース ロールバック:** いずれかの文が失敗すると、バッチ全体がロールバックされます。バッチ サイズを大きくすると、ロールバックの実行時間が長くなります。

住所検証の最適化

Validate Address は、入力レコードが郵便番号でソートされている場合にパフォーマンスが最高になります。これは、参照データが郵便番号でソートされてメモリ内にロードされているためです。入力がソートされていると、ソートされていない入力に比べ、実行速度が数倍速くなることもあります。郵便番号フィールドにデータが入っていないレコードもあるため、次のソート順を推奨します。

1. Country (複数の国のレコードを処理する場合のみ必要)
2. PostalCode
3. StateProvince
4. City

ジオコーディングの最適化

ジオコーディング ステージでは、入力レコードが郵便番号でソートされている場合に最大のパフォーマンスが得られます。これは、参照データが郵便番号でソートされてメモリ内にロードされているためです。入力がソートされていると、ソートされていない入力に比べ、実行速度が数倍速くなることもあります。郵便番号フィールドにデータが入っていないレコードもあるため、次のソート順を推奨します。

1. PostalCode
2. StateProvince
3. City

異なるマッチ モードを試してみることによっても、ジオコーディング ステージを最適化できます。マッチ モードは、ジオコーディング結果が近似一致であるかどうかを、ジオコーディング ステージが判定する方法を制御します。マッチ モードを **緩和** に設定して、要件を満たす結果が得られるかどうか確認してみてください。一般的に **緩和** モードは、他のマッチ モードよりもパフォーマンスが向上します。

Geocode US Address の最適化

Geocode US Address ステージには、パフォーマンスに影響を与える複数のオプションがあります。以下のオプションがこのファイル内にあります。

SpectrumLocation\server\modules\geostan\java.properties

egm.us.multimatch.max.records	返されるマッチの最大数を指定します。この数値を小さくするとパフォーマンスは向上しますが、マッチ数は少なくなります。
egm.us.multimatch.max.processing	実行する検索の数を指定します。この数値を小さくするとパフォーマンスは向上しますが、マッチ数は少なくなります。
FileMemoryLimit	最初にメモリにロードする参照データの量を制御します。

データフローのバージョン

Enterprise Designer のバージョン機能を使用すると、データフローの改訂履歴を保持できます。データフローの以前のバージョンを表示したり、実行のために古いバージョンをエクスポーズしたりできます。また、データフローを以前のバージョンに戻す必要がある場合に備えて、変更履歴を保持することもできます。

データフローのバージョンの保存

Enterprise Designer では、データフローのバージョンを 2 つの方法で保存できます。

- データフローをエクスポーズします。**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択するか、ツール バーの電球をクリックして、データフローをエクスポーズするたびに、Enterprise Designer によってデータフローのバージョンが自動的に保存されます。
- Enterprise Designer の **[バージョン]** ウィンドウで、バージョンを手動で保存します。

注： データフローを保存するだけでは、データフローのバージョンは作成されません。

次の手順では、Enterprise Designer の **[バージョン]** ウィンドウでバージョンを手動で保存する方法について説明します。

1. Enterprise Designer で、データフローを開きます。
2. **[バージョン]** ウィンドウが表示されない場合は、**[表示] > [バージョン]** を選択します。

3. **[バージョン]** リストで最終保存バージョンが選択されていることを確認します。最終保存バージョンはリストの一番上にあります。
4. **[バージョン]** ウィンドウの緑のプラス アイコンをクリックします。

データフローの新しいバージョンが保存され、**[バージョン]** ウィンドウに追加されます。

データフローのバージョンの表示

データフローの以前のバージョンを表示できます。データフローの以前のバージョンを表示すると、変更を加える前の状態のデータフローの設計を確認できます。以前のバージョンは表示のみが可能で、変更することはできません。以前のバージョンを変更するには、まず、そのバージョンを最終保存バージョンにプロモートする必要があります。

1. Enterprise Designer で、データフローを開きます。
2. **[バージョン]** ウィンドウが表示されない場合は、**[表示] > [バージョン]** を選択します。
3. 表示するバージョンを選択します。

選択したバージョンが、データフローのキャンバスに表示されます。

データフローのバージョンの編集

データフローの以前のバージョンを編集するには、そのバージョンを最終保存バージョンにプロモートします。データフローのバージョンをプロモートすると、そのバージョンは最終保存バージョンになって編集できるようになります。

> **注：** バージョンをプロモートして編集すると、既存の最終保存バージョンが編集後のバージョンで上書きされます。そのことを踏まえたうえで、次の手順を実行してください。既存の最終保存バージョンのコピーを保持する場合は、古いバージョンをプロモートする前のバージョンとして保存します。

1. Enterprise Designer で、データフローを開きます。
2. **[バージョン]** ウィンドウが表示されない場合は、**[表示] > [バージョン]** を選択します。
3. 編集するバージョンを選択します。
4. プロモート アイコンをクリックします。

選択したバージョンが、最終保存バージョンにプロモートされます。これで、データフローを編集できます。

バージョンのプロパティの編集

データフローのバージョンを保存すると、デフォルトのバージョン番号が割り当てられます。バージョン番号を変更し、コメントを追加してバージョンの変更や用途を記録できます。

1. Enterprise Designer で、データフローを開きます。
2. **[バージョン]** ウィンドウが表示されない場合は、**[表示] > [バージョン]** を選択します。
3. 変更するバージョンを選択します。
4. プロパティ アイコンをクリックします。

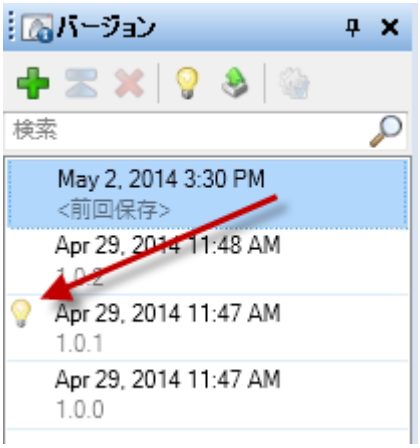
5. **[名前]** フィールドに、バージョンの名前を入力します。バージョン番号を入力することも、わかりやすい名前を入力することもできます。任意の名前にすることができます。
6. **[コメント]** フィールドでは、変更したバージョンの用途を詳しく説明する長めのコメントを入力できます。コメントの追加はオプションです。
7. **[OK]** をクリックします。

バージョンのエクスポーズ

複数のバージョンのデータフローを保存している場合、実行のためにエクスポーズするバージョンを選択できます。

1. Enterprise Designer で、データフローを開きます。
2. **[バージョン]** ウィンドウが表示されない場合は、**[表示] > [バージョン]** を選択します。
3. **[バージョン]** ウィンドウで、エクスポーズするデータフローのバージョンを選択します。
4. **[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。

これで、選択したバージョンをエクスポーズして実行できるようになります。次に示すように、横に電球が表示されているバージョンが、エクスポーズされるバージョンです。



データフローがエクスポーズされると、次のように、Enterprise Designer ツール バーの電球ボタンがデータフローのエクスポーズを示します。

エクスポーズするバージョン以外のバージョンを表示している場合でも、電球はデータフローのエクスポーズを示します。アンエクスポーズするバージョンを表示しているときに電球をクリックすると、エクスポーズするバージョンが現在表示中のバージョンに切り替わります。エクスポーズするバージョンの表示中に電球をクリックすると、データフローがアンエクスポーズされます。

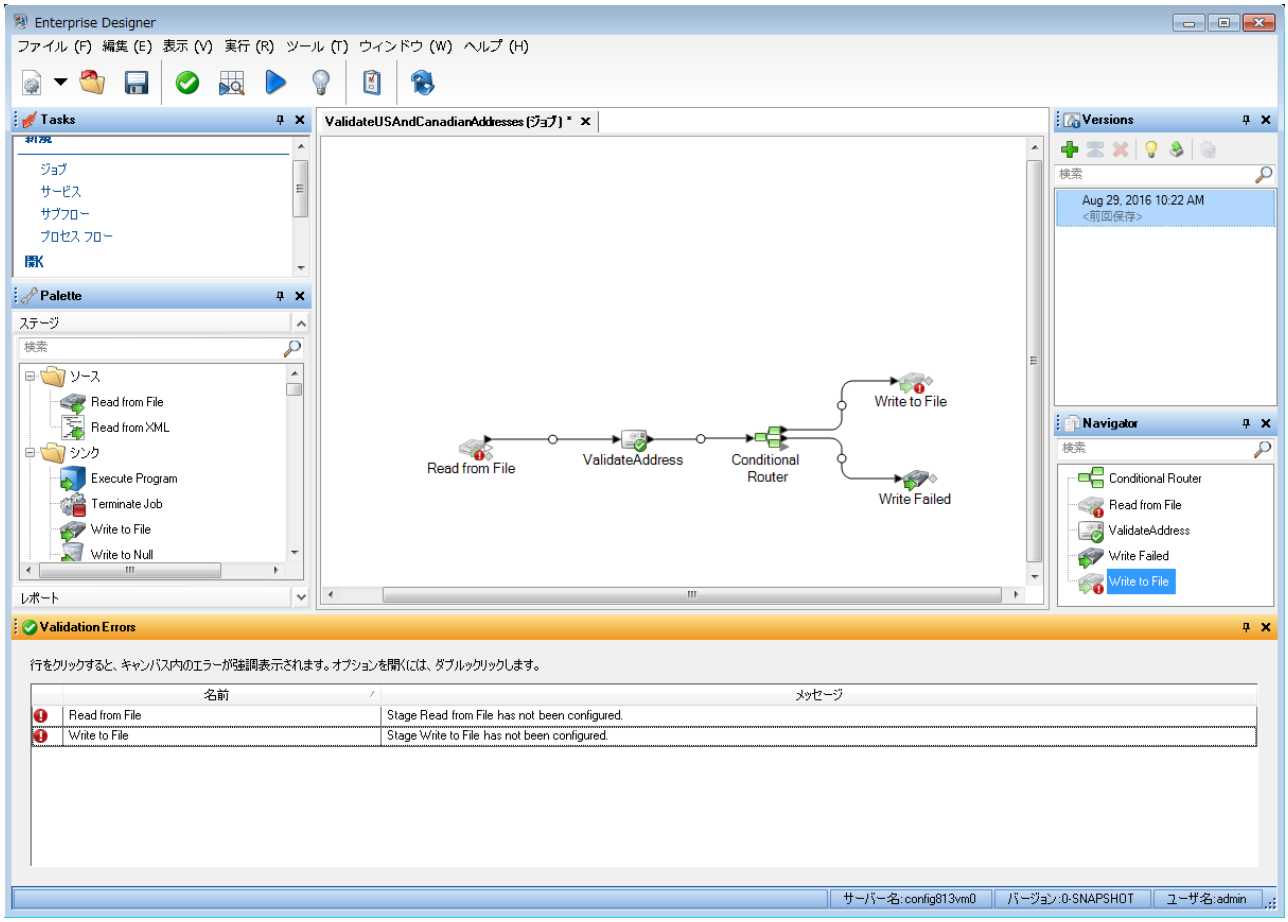
3 - インスペクションとテスト

このセクションの構成

フローのエラーの確認

Enterprise Designer では、フローの実行時、インスペクションの実行時、フローの公開時、および公開したフローの保存時に自動的にフローのエラーがチェックされます。また、検証ボタン をクリックすることにより、フローのエラーをチェックできます。

エラーが見つかると、Enterprise Designer ウィンドウの一番下に **[Validation (検証)]** ウィンドウが表示されます。エラーをクリックすると、キャンバス内のエラーが強調表示されます。エラーをダブル クリックすると、そのエラーを含むアイテムのオプション ウィンドウが開きます。



データフローのインスペクション

データフロー内のさまざまなポイントで入力データに対するデータフローの影響を確認するには、Enterprise Designer のインスペクション ツールを使用します。インスペクションを使うと、データフローがデータに与えている作用が期待どおりか確認したり、問題を切り分けたり、不具合があるレコードを特定することができます。

注： データフロー インスペクションは、埋め込まれたデータフローで使用できません。

1. インスペクションに使用するデータを指定します。

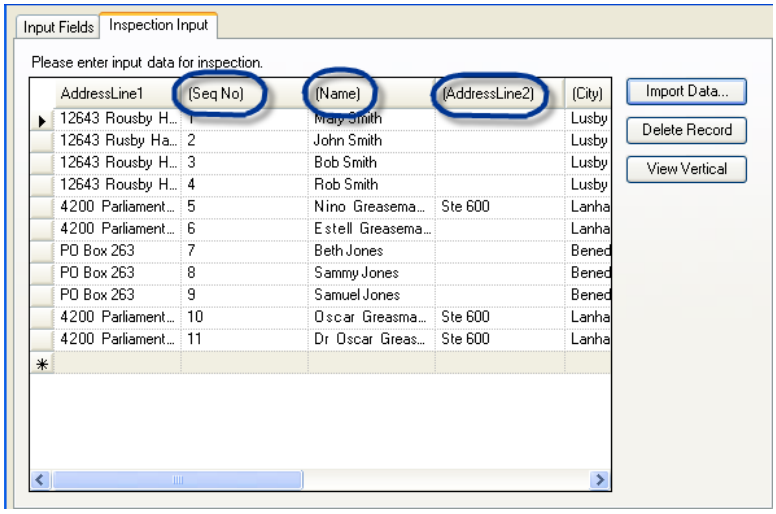
   実際のデータを代表するデータ、または特定の問題をトラブルシューティングする場合はその問題の原因であるデータを指定する必要があります。サービスのインスペクションを行うか、ジョブのインスペクションを行うかによって、インスペクションに使用するデータを指定する方法は 2 通りあります。

シナリオ	説明
ジョブのインスペクション データを指定するには	ジョブのインスペクションを行う場合、インスペクションに使用するデータは、ソース ステージで指定されているデータです。インスペクション ツールは最大 50 レコードを処理でき、デフォルトでは入力ファイルまたはデータベースの最初の 50 レコードです。最初以外のレコードからデータの使用を開始するには、[Read From File] ステージをダブルクリックし、[実行時] タブの **[開始レコード]** フィールドに入力します。
サービスのインスペクション データを指定するには	サービス データフローは、Input ステージを使用してデータフローへの入力を定義します。データフローの編集時には、Input ステージからデータにアクセスできないため、Input ステージ内のインスペクション データを **[インスペクション データ]** タブで定義する必要があります。最大 50 個のレコードを指定できます。 いくつかの方法によって、Input ステージにインスペクション データを入力できます。 • インスペクションに数個のレコードのみを使用する場合は、データを手動で入力できます。 **ヒント：[データのエクスポート]** をクリックしてインスペクション データをテキスト ファイルにエクスポートすることにより、入力したインスペクション データを保存して別のステージで再利用することができます。

シナリオ	説明

- CSV または TXT ファイルのデータがある場合は、**[データのインポート]** をクリックすることによってそのデータをインポートできます。データには、次のいずれかの区切り文字を使用する必要があります。
  - \t
  - |
  - ,
  - ;
- 他のアプリケーションからの区切り文字で区切られたデータをコピーし、インスペクション データ エディタに貼り付けることができます。

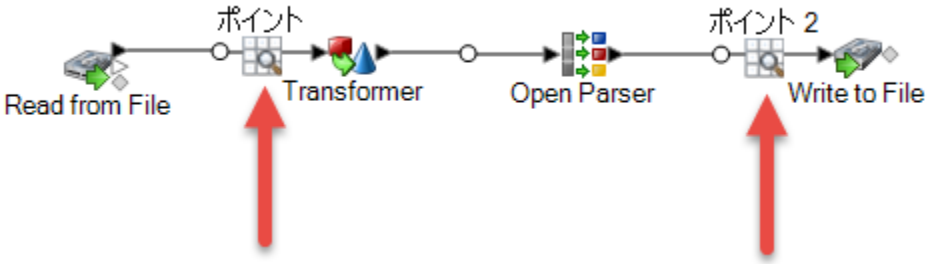
以下に示すように、**[インスペクション入力]** タブでは、フィールド名を括弧で囲むことによってパススルー データを示します。

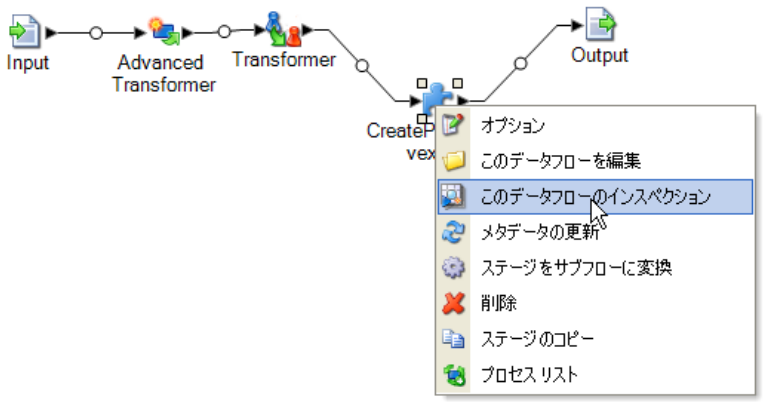


注： 一部のフィールド タイプには、インスペクションでの使用に関して制約があります。

- double 型および float 型のフィールドには、数値データのみが格納されている必要があります。フィールドは最大 16 桁で、小数点以下桁数は 6 桁までです。指数表記は、インスペクションにおいてサポートされていません。
- integer 型および long 型のフィールドには、数値データのみが格納されている必要があります。

2. データフロー内でデータを表示するポイントを示します。

シナリオ	説明
インスペクション ポイントをチャネルに追加するには	チャネルで [名前の変更] ノードの左側を右クリックし、**[インスペクション ポイントの追加]** を選択します。  ポイントがジョブに追加されます。 
データフロー内の 2 つのポイントでレコードを比較するには	データフロー内の比較したい 2 つのポイントにインスペクション ポイントを追加します。  **ヒント：** 問題を特定するためにインスペクションを使用する場合は、最初にデータフローの広い範囲を検査するようにポイントを設定してから、問題がありそうな位置に向かってポイントを狭めていきます。
ジョブまたはサービスに埋め込まれているサ	サブフロー ステージを右クリックし、**[このデータフローのインスペクション]** を選択します。

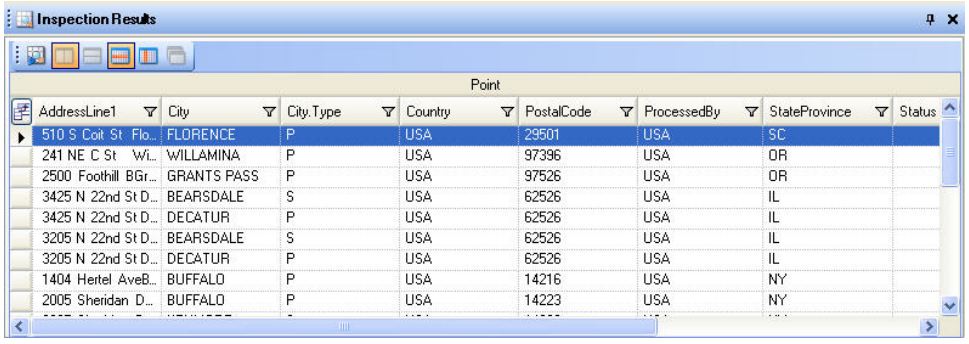
シナリオ	説明
ブフローのインスペクションを行うには	

入力データ (ジョブの場合) またはインスペクション データ (サービスの場合) がサブフローに自動的に渡されるので、サブフローの Input ステージでインスペクション データを入力する必要はありません。

> **注 :** サブフローのインスペクションを行うとき、エクスポーズされているバージョンのサブフローが表示されます。サブフローを変更して、インスペクションを再実行したい場合は、新しいバージョンをエクスポーズする必要があります。

3. **[実行] > [現在のフローのインスペクション]** を選択するか、またはツールバーの **[現在のフローのインスペクション]** ボタンをクリックします。

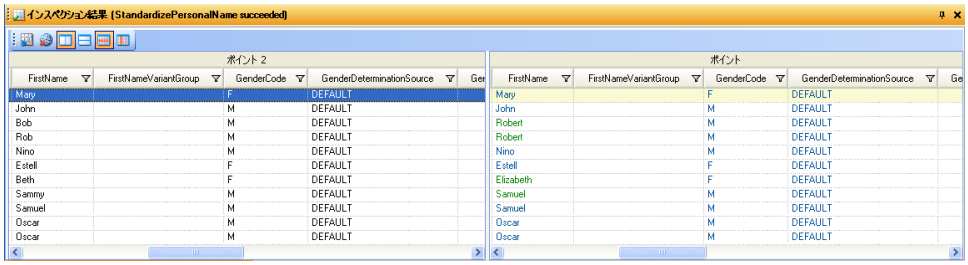
   1 つのインスペクション ポイントを指定すると、**[インスペクション結果]** ウィンドウには検査済みデータが横方向に表示されます。表示レイアウトを変更するには、テーブルの上にあるツールバーのアイコンを使用します。インスペクション データが階層的な場合は、縦に表示することはできません。



> **注 :** タイプ変換オプションで指定された形式で日付および時間データが表示されます。

ヒント： インスペクション ポイントを移動するには、別のチャネルにドラッグします。インスペクション データが自動的に更新されます。

2 つのインスペクション ポイントを指定すると、**[インスペクション結果]** ウィンドウにその 2 つのポイントの位置にあるレコードが表示されます。左のウィンドウにはデータフローで左側にあるインスペクション ポイントが表示され、右のウィンドウにはデータフローで右側にあるインスペクション ポイントが表示されます。右のウィンドウでレコードをクリックして左のウィンドウの対応するレコードをハイライト表示すると、2 つのインスペクション ポイントの間でレコードがどのように変化したのかがわかります。



各列は、データフローの 1 フィールドを表します。列は、アルファベット順に並べられます。インスペクション ポイントの間に追加された新しいフィールドは、右のウィンドウで元の列の直後に表示されます。列を並べ替えるには、その列をクリックして、目的の位置にドラッグします。

2 つのインスペクション ポイントによるインスペクション結果の表示は、以下の条件によって影響されます。

- 2 つのインスペクション ポイントの間に Sorter ステージがある場合、インスペクション結果に含まれるレコードは Sorter ステージの前にあるものとしてソートされます。2 番目のインスペクション ポイントについてはソートは無視されるため、各インスペクション ポイントに対応するレコードはその並びのまま比較できます。
- 2 つのインスペクション ポイントの間に、Aggregator ステージのように新規のレコードを作成するステージがある場合、2 番目のインスペクション ポイントに表示されるレコードには、最初のインスペクション ポイントで対応するレコードが含まれません。
- 2 番目のインスペクション ポイントの位置に存在するが、最初のポイントの位置には存在しないレコードは、2 番目のインスペクション ポイントのレコードの一覧で最後に表示されます。

4. データフローを更新または変更した場合は、**[実行] > [現在のフローのインスペクション]** をクリックして、インスペクション結果を更新してください。
5. [インスペクション結果] ウィンドウを閉じると、インスペクション データは失われます。同様に、ジョブを閉じても、インスペクション ポイントとインスペクション データは失われます。インスペクション結果をファイルに保存するには
   a) インスペクション結果グリッドで、保存する行を選択します。いずれかのウィンドウで右クリックして **[すべて選択]** をクリックすると、すべてのデータを選択できます。

b) コンテキスト メニューから **[コピー]** を選択します。
c) データを保存するアプリケーション (Microsoft Excel、メモ帳など) を開きます。
d) アプリケーション内にデータを貼り付けます。
e) ファイルを保存します。

Management Console を使ったサービスのテスト

Management Console にはプレビュー機能があり、テスト データをサービスに送信し、結果を確認することができます。

1. Web ブラウザで次の URL を表示します。

   http://*server*:*port*/managementconsole

   ここで *server* は、 Spectrum™ Technology Platform サーバーの名前または IP アドレスで、*port* はSpectrum™ Technology Platformが使用する HTTP ポートです。デフォルトの HTTP ポートは 8080 です。

2. **[サービス]** メニューに移動し、テストするサービスが含まれるモジュールをクリックします。
3. テストするサービスをクリックします。
4. **[プレビュー]** をクリックします。
5. テストで使用する入力データを入力します。データをファイルからインポートするには、[インポート] ボタン をクリックします。
6. **[プレビューを実行]** をクリックします。

4 - フローの実行

このセクションの構成

ジョブまたはプロセス フローの実行

Enterprise Designer でのデータフローの実行

ジョブまたはプロセス フローを手動で実行する手順を以下に示します。

1. Enterprise Designer で、**[ファイル] > [開く]** を選択し、実行するデータフローを開きます。
2. 実行の前にデータフローの検証を行い、エラーが存在しないことを確認します。データフローを検証するには、**[実行] > [検証]** を選択します。
3. **[実行] > [現在のフローを実行]** を選択します。

コマンド ラインからのジョブの実行

コマンド ラインからジョブを実行するには、その前に、ジョブをエクスポーズする必要があります。ジョブをエクスポーズするには、Enterprise Designer でジョブを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。

コマンド ラインからジョブを実行するには、ジョブを実行するシステムに Job Executor ユーティリティをインストールする必要があります。Job Executor は Spectrum™ Technology Platform サーバー上の Spectrum™ Technology Platform Welcome ページ (例えば、http://myserver:8080) にあります。

使用方法

```
java -jar jobexecutor.jar -uUserID -p Password-j Job [オプションの引数]
```

必須	引数	説明
いいえ	`-?`	使用方法を出力します。
いいえ	`-d` *delimiter*	インスタンス/ステータス区切り文字を設定します。これは、同期出力にのみ表示されます。
いいえ	`-e`	Spectrum™ Technology Platform サーバーとの通信にセキュアな HTTPS 接続を使用します。

必須	引数	説明
いいえ	`-f` *property file*	ジョブ プロパティ ファイルへのパスを指定します。ジョブ プロパティ ファイルには Job Executor の引数が含まれています。ジョブ プロパティ ファイルの詳細については、ジョブ プロパティ ファイルの使用（162ページ）を参照してください。
いいえ	`-h` *host name*	Spectrum™ Technology Platform サーバーの名前または IP アドレスを指定します。
いいえ	`-i` *poll interval*	完了したジョブを確認する間隔を秒数で指定します。これは、同期モードにのみ適用されます。
はい	`-j` *job name*	実行するジョブをカンマで区切って指定します。ジョブ名は大文字と小文字が区別されます。ジョブは、列挙された順序で開始されます。
いいえ	`-n` *email list*	ジョブ通知設定に対する追加の電子メール アドレスをカンマで区切って指定します。

いいえ	`-o` *property file*	データフロー オプション プロパティ ファイルへのパスを指定します。データフロー オプション プロパティ ファイルは、データフローのステージのオプションを設定するために使います。プロパティ ファイルを使用してデータフロー オプションを設定するには、実行時にステージ オプションをエクスポーズするようにデータフローを設定する必要があります。詳細については、「データフロー実行時オプションの追加（181ページ）」を参照してください。
		
		例えば、Assign GeoTAX Info ステージを含むデータフローのデータフロー オプション プロパティ ファイルは、次のようなものになります。
		
		`OutputCasing=U`
		`UseStreetLevelMatching=N`
		`TaxKey=T`
		`Database.GTX=gsl`

必須	引数	説明
はい	`-p` *password*	ユーザのパスワードです。
いいえ	`-r`	ジョブについての詳細レポートを返すには、この引数を指定します。このオプションは、`-w` も指定している場合にのみ機能します。レポートには、次の情報が含まれます。 • **位置 1** — ジョブの名前 • **位置 2** — ジョブ プロセス ID • **位置 3** — ステータス • **位置 4** — 開始日時 (MM/DD/YYYY HH:MM:SS) • **位置 5** — 終了日時 (MM/DD/YYYY HH:MM:SS) • **位置 6** — 成功したレコードの数 • **位置 7** — 失敗したレコードの数 • **位置 8** — 形式に誤りのあるレコードの数 • **位置 9** — 現在未使用 例: `MySimpleJob|4|succeeded|04/09/2010 14:50:47|04/09/2010 14:50:47|100|0|0|` 情報は、`-d` 引数で指定された区切り文字で区切られて出力されます。
いいえ	`-s` *port*	Spectrum™ Technology Platform サーバーが実行されているソケット (ポート)。デフォルト値は 8080 です。
いいえ	`-t` *timeout*	同期モードの場合のタイムアウト (秒単位) を設定します。デフォルト値は 3600 です。最大数は 2147483 です。これは、すべてを合わせたグローバルなタイムアウト値で、spawn されたすべてのジョブが完了するまでの最大待機時間を表します。
はい	`-u` *user name*	ユーザのログイン名を指定します。
いいえ	`-v`	詳細な出力を返します。

必須	引数	説明
いいえ	`-w`	Job Executor を同期モードで実行します。この場合、Job Executor はジョブが完了するまで実行されたままになります。 `-w` を指定しない場合は、Job Executor はジョブの開始後に終了します。ただし、ジョブがサーバー上のファイルを読み書きする場合は、Job Executor はローカル ファイルがすべて処理されるまで実行され、その後で終了します。
いいえ	*StageName=Protocol:FileName*	Read from File または Write to File で指定された入力または出力ファイルをオーバーライドします。詳細については、「ジョブ ファイルの場所のオーバーライド（157ページ）」を参照してください。
いいえ	*StageName*:schema=*Protocol*:*SchemaFile*	Read from File または Write to File で指定されたファイル レイアウト定義を、スキーマ ファイルで定義されたファイル レイアウト定義でオーバーライドします。詳細については、「コマンド ラインでのファイル フォーマットのオーバーライド（160ページ）」を参照してください。

Job Executor の使用例

以下は、コマンド ラインによる起動とその出力の例です。

```
D:\spectrum\job-executor>java -jar jobexecutor.jar -u user123
 -p "mypassword" -j validateAddressJob1 -h spectrum.example.com
 -s 8888 -w -d "%" -i 1 -t 9999

validateAddressJob1%105%succeeded
```

この例の出力からは、'validateAddressJob1' という名前のジョブ (識別子 105) がエラーを生じることなく実行したことがわかります。その他の実行結果としては、"失敗" と "実行中" があります。

ジョブ ファイルの場所のオーバーライド

Job Executor または管理ユーティリティを使用してコマンド ラインでジョブを実行する際に、データフローのソース ステージ (Read from File など) で指定された入力ファイル、およびデータ

フローのシンク ステージ (Write to File など) で指定された出力ファイルをオーバーライドできます。

Job Executor でこれを行うには、Job Executor コマンド ライン コマンドの終わりに以下を指定します。

```
StageName=Protocol:FileName
```

管理ユーティリティでは、`job execute` コマンドで次のように `--l` 引数を使用します。

```
--l StageName=Protocol:FileName
```

WHERE:

StageName

Enterprise Designer で、データフローのステージのアイコンの下に表示されるステージ ラベル。例えば、ステージのラベルが "Read from File" の場合は、ステージ名として `Read from File`と指定します。

埋め込まれたデータフローまたはサブフロー内のステージを指定するには、ステージ名の前に埋め込まれたデータフローまたはサブフローの名前とピリオドを付けて、次のようにします。

```
EmbeddedOrSubflowName.StageName
```

例えば、"Subflow1" という名前のサブフローの中の "Write to File" という名前のステージを指定するには、次のようにします。

```
Subflow1.Write to File
```

別の埋め込まれたデータフローの中にある埋め込まれたデータフロー内のステージを指定するには、親データフローをそれぞれピリオドで区切って追加します。例えば、Embedded Dataflow 2 が Embedded Dataflow 1 の中にあり、Embedded Dataflow 2 内の Write to File ステージを指定する場合は、次のようにします。

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

通信プロトコル。次のいずれかです。

file ファイルが Spectrum™ Technology Platformサーバーと同じコンピュータ上にある場合は、file プロトコルを使用します。例えば、Windows では、

```
"file:C:/myfile.txt"
```

と指定し、Unix または Linux では、

```
"file:/testfiles/myfile.txt"
```

と指定します

esclient ファイルがジョブを実行するコンピュータ上にあり、そのコンピュータがSpectrum™ Technology Platformサーバーを実行するコンピュータと異なる場合は、esclient プロトコルを使用します。次の形式を使用します。

`esclient:`*コンピュータ名/ファイルへのパス*

例を次に示します。

```
esclient:mycomputer/testfiles/myfile.txt
```

> **注：** ジョブをサーバー自体で実行している場合は、fileプロトコルを使用してもesclient プロトコルを使用しても構いませんが、file プロトコルを使用した方がパフォーマンスが高くなる可能性があります。

Job Executor を実行しているコンピュータがSpectrum™ Technology Platformサーバーのホスト名を解決できない場合、"ファイルのアクセスでエラーが発生しました" というエラーが表示される可能性があります。この問題を解決するには、サーバー上で次のファイルを開きます:*SpectrumLocation*`/server/app/conf/spectrum-container.properties`。プロパティ `spectrum.runtime.hostname` にサーバーの IP アドレスを設定します。

esfile ファイルが ファイル サーバー上にある場合は、esfile プロトコルを使用します。このファイル サーバーは Management Console でリソースとして定義する必要があります。次の形式を使用します。

`esfile://`*ファイル サーバー/ファイルへのパス*

例を次に示します。

```
esfile://myserver/testfiles/myfile.txt
```

ここで、myserver は、Management Console で定義された FTP ファイル サーバー リソースです。

webhdfs ファイルが Hadoop Distributed File Server 上にある場合は、webhdfs プロトコルを使用します。HDFS サーバーは Management Console でリソースとして定義する必要があります。次の形式を使用します。

`webhdfs://`*ファイル サーバー/ファイルへのパス*

例を次に示します。

```
webhdfs://myserver/testfiles/myfile.txt
```

ここで、myserver は、Management Console で定義された HDFS ファイル サーバー リソースです。

FileName

入力または出力として使用するファイルへのフル パス。

注： ファイルのパスには、バックスラッシュではなくスラッシュ (/) を使用する必要があります。

複数のオーバーライドを指定する場合は、カンマで区切ってください。

ファイルのオーバーライドの例

以下の Job Executor コマンドでは、ファイル `C:/myfile_input.txt`を Read from File ステージの入力ファイルとして使用し、ファイル`C:/myfile_output.txt` を Write to File ステージの出力ファイルとして使用します。

```
java -jar jobexecutor.jar -j Job1 -u Bob1234 -p "" "Read from
File"="file:C:/myfile_input.txt" "Write to
File"="file:C:/myfile_output.txt"
```

コマンド ラインでのファイル フォーマットのオーバーライド

Job Executor または管理ユーティリティを使用してジョブを実行する際に、データフローの Read from File ステージおよび Write to File ステージで指定されたファイルのファイル レイアウト (またはスキーマ) をオーバーライドできます。

Job Executor でこれを行うには、Job Executor コマンド ライン コマンドの終わりに以下を指定します。

StageName:schema=*Protocol*:*SchemaFile*

管理ユーティリティでは、`job execute` コマンドで次のように `--l` 引数を使用します。

`--l`*StageName*:schema=*Protocol*:*SchemaFile*

説明：

StageName

Enterprise Designer で、データフローのステージのアイコンの下に表示されるステージ ラベル。例えば、ステージのラベルが "Read from File" の場合は、ステージ名として `Read from File`と指定します。

埋め込まれたデータフローまたはサブフロー内のステージを指定するには、ステージ名の前に埋め込まれたデータフローまたはサブフローの名前とピリオドを付けて、次のようにします。

EmbeddedOrSubflowName.StageName

例えば、"Subflow1" という名前のサブフローの中の "Write to File" という名前のステージを指定するには、次のようにします。

```
Subflow1.Write to File
```

別の埋め込まれたデータフローの中にある埋め込まれたデータフロー内のステージを指定するには、親データフローをそれぞれピリオドで区切って追加します。例えば、Embedded Dataflow 2 が Embedded Dataflow 1 の中にあり、Embedded Dataflow 2 内の Write to File ステージを指定する場合は、次のようにします。

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

通信プロトコル。次のいずれかです。

file ファイルが Spectrum™ Technology Platformサーバーと同じコンピュータ上にある場合は、file プロトコルを使用します。例えば、Windows では、

```
"file:C:/myfile.txt"
```

と指定し、Unix または Linux では、

```
"file:/testfiles/myfile.txt"
```
と指定します

esclient ファイルがジョブを実行するコンピュータ上にあり、そのコンピュータがSpectrum™ Technology Platformサーバーを実行するコンピュータと異なる場合は、esclient プロトコルを使用します。次の形式を使用します。

`esclient:`*コンピュータ名/ファイルへのパス*

例を次に示します。

```
esclient:mycomputer/testfiles/myfile.txt
```

> **注：** ジョブをサーバー自体で実行している場合は、fileプロトコルを使用してもesclient プロトコルを使用しても構いませんが、file プロトコルを使用した方がパフォーマンスが高くなる可能性があります。

Job Executor を実行しているコンピュータがSpectrum™ Technology Platformサーバーのホスト名を解決できない場合、"ファイルのアクセスでエラーが発生しました" というエラーが表示される可能性があります。この問題を解決するには、サーバー上で次のファイルを開きます:*SpectrumLocation*`/server/app/conf/spectrum-container.properties`。プロパティ `spectrum.runtime.hostname` にサーバーの IP アドレスを設定します。

esfile ファイルが ファイル サーバー上にある場合は、esfile プロトコルを使用します。このファイル サーバーは Management Console でリソースとして定義する必要があります。次の形式を使用します。

`esfile://`*ファイル サーバー/ファイルへのパス*

例を次に示します。

```
esfile://myserver/testfiles/myfile.txt
```

ここで、myserver は、Management Console で定義された FTP ファイル サーバー リソースです。

webhdfs ファイルが Hadoop Distributed File Server 上にある場合は、webhdfs プロトコルを使用します。HDFS サーバーは Management Console でリソースとして定義する必要があります。次の形式を使用します。

```
webhdfs://ファイル サーバー/ファイルへのパス
```

例を次に示します。

```
webhdfs://myserver/testfiles/myfile.txt
```

ここで、myserver は、Management Console で定義された HDFS ファイル サーバー リソースです。

SchemaFile

使用するレイアウトを定義したファイルへのフル パス。

> **注：** ファイルのパスには、バックスラッシュではなくスラッシュ (/) を使用する必要があります。

スキーマ ファイルを作成するには、目的のレイアウトを Read from File または Write to File で定義し、**[エクスポート]** ボタンをクリックして、レイアウトを定義する XML ファイルを作成します。

> **注：** Job Executor を使用する場合は、スキーマ ファイル内のフィールドのデータ タイプをオーバーライドできません。<FieldSchema> 要素の子である <Type> 要素の値は、データフローの Read from File ステージまたは Write to File ステージで指定されたフィールドのタイプと一致している必要があります。

ファイル フォーマットのオーバーライドの例

以下の Job Executor コマンドでは、ファイル `C:/myschema.xml` を Read from File ステージによって読み込まれるファイルのレイアウト定義として使用します。

```
java -jar jobexecutor.jar -j Job1 -u Bob1234 -p "" "Read from File":schema="file:C:/myschema.xml"
```

ジョブ プロパティ ファイルの使用

ジョブ プロパティ ファイルには、ジョブの実行を制御する引数が含まれており、Job Executor または管理ユーティリティを使用したジョブの実行を制御できます。引数を再使用する場合に、各引数をコマンド ラインで個別に指定するのではなく、単一の引数 (`-f`) を指定するには、ジョブ プロパティ ファイルを使用します。

プロパティ ファイルを作成するには、各行に 1 つの引数を含むテキスト ファイルを作成します。例:

```
d %
h spectrum.mydomain.com
i 30
j validateAddressJob1
u user
p password
s 8888
t 9999
w true
```

ジョブ プロパティ ファイルには次の引数を含めることができます。

必須	引数	説明
いいえ	`?`	使用方法を出力します。
いいえ	`d` *delimiter*	インスタンス/ステータス区切り文字を設定します。これは、同期出力にのみ表示されます。
いいえ	`e`	Spectrum™ Technology Platformサーバーとの通信にセキュアな HTTPS 接続を使用します。
いいえ	`h` *hostname*	Spectrum™ Technology Platformサーバーの名前または IP アドレスを指定します。
いいえ	`i` *pollinterval*	完了したジョブを確認する間隔を秒数で指定します。これは、同期モードにのみ適用されます。
はい	`j` *jobname*	実行するジョブをカンマで区切って指定します。ジョブ名は大文字と小文字が区別されます。ジョブは、列挙された順序で開始されます。
いいえ	`n` *emaillist*	ジョブ通知設定に対する追加の電子メール アドレスをカンマで区切って指定します。
はい	`p` *password*	ユーザのパスワードです。
いいえ	`r`	標準出力に出力されるジョブに関する以下の情報を、区切り文字で区切って返します。 • **位置 1** — ジョブの名前 • **位置 2** — ジョブ プロセス ID • **位置 3** — ステータス • **位置 4** — 開始日時 (MM/DD/YYYY HH:MM:SS) • **位置 5** — 終了日時 (MM/DD/YYYY HH:MM:SS) • **位置 6** — 成功したレコードの数

必須	引数	説明
		• **位置 7** — 失敗したレコードの数 • **位置 8** — 形式に誤りのあるレコードの数 • **位置 9** — 現在未使用 情報は、-d 引数で指定された区切り文字で区切られて出力されます。例: `MySimpleJob|4|succeeded|04/09/2010 14:50:47|04/09/2010 14:50:47|100|0|0|`
いいえ	`s` *port*	Spectrum™ Technology Platformサーバーが実行されているソケット (ポート)。デフォルト値は 8080 です。
いいえ	`t` *timeout*	同期モードの場合のタイムアウト (秒単位) を設定します。デフォルト値は 3600 です。最大数は 2147483 です。これは、すべてを合わせたグローバルなタイムアウト値で、spawn されたすべてのジョブが完了するまでの最大待機時間を表します。
はい	`u` *username*	ユーザのログイン名を指定します。
いいえ	`v`	詳細な出力を返します。
いいえ	`w`	同期モードにおいてジョブの完了を待つように指定します。

コマンドライン引数とプロパティ ファイルの両方の使用

コマンド ライン入力とプロパティ ファイル入力を組み合わせて使用することもできます。例:

```
java -jar jobexecutor.jar -f /dcg/job.properties -j job1
```

この場合、コマンドライン引数の方が、プロパティ ファイルで指定された引数よりも優先されます。上の例では、ジョブ job1 の方が、プロパティ ファイルで指定されたジョブよりも優先されることになります。

コマンド ラインからのプロセス フローの実行

コマンド ラインからプロセス フローを実行するには、Process Flow Executor を使用します。Process Flow Executor は、Spectrum™ Technology PlatformWelcome ページ (例えば、http://myserver:8080 など) からインストールできます。

> **注:** 管理ユーティリティを使用して、コマンド ラインからプロセス フローを実行することもできます。

使用方法

```
java -jar pflowexecutor.jar-rProcessFlowName -u UserID-p Password [Optional Arguments]
```

必須	引数	説明
いいえ	`-?`	使用方法を出力します。
いいえ	`-d` *DelimiterCharacter*	コマンド実行時にコマンド ラインに表示されるステータス情報を区切るのに使用する区切り文字を設定します。デフォルトは "|" です。例えば、デフォルト文字を使用して、"MyProcessflow" というプロセス フローを実行した場合、コマンド ラインには次のように表示されます。 `MyProcessflow|1|Succeeded`
いいえ	`-e`	Spectrum™ Technology Platformサーバーとの通信に HTTPS 接続を使用します。 **注：** ファイル オーバーライドを指定する場合は、この引数を最後の引数として指定しないでください。
いいえ	`-f` *PropertyFile*	プロパティ ファイルへのパスを指定します。プロパティ ファイルの詳細については、プロセス フローのプロパティ ファイルの使用（166ページ）を参照してください。
いいえ	`-h` *HostName*	Spectrum™ Technology Platformサーバーの名前または IP アドレスを指定します。
いいえ	`-i` *PollInterval*	完了したジョブを確認する間隔を秒数で指定します。デフォルト値は "5" です。
はい	`-p` *Password*	ユーザのパスワードです。必須
はい	`-r` *ProcessFlowNames*	実行するプロセス フローのリストをカンマで区切って指定します。必須 **注：** ファイル オーバーライドを指定する場合は、この引数を最後の引数として指定しないでください。
いいえ	`-s` *Port*	Spectrum™ Technology Platformサーバーが実行されているソケット (ポート)。デフォルト値は 8080 です。
いいえ	`-t` *Timeout*	このオプションは非推奨で、将来は無視されます。
はい	`-u` *UserName*	ユーザのログイン名を指定します。必須
いいえ	`-v` *Verbose*	詳細な出力を返します。ここで、*Verbose* は次のいずれかです。 **true** 詳細な出力を返します。

必須	引数	説明
		false 詳細な出力を返しません。 **注：** ファイル オーバーライドを指定する場合は、この引数を最後の引数として指定しないでください。
いいえ	`-w` *WaitToComplete*	このオプションは非推奨で、将来は無視されます。
いいえ	*StageName=FileName*	ジョブで指定されている入力または出力ファイルをオーバーライドします。詳細については、プロセス フロー ファイルの場所のオーバーライドを参照してください。

例

プロセス フロー名、ユーザ ID、パスワードを指定した基本的なコマンド ライン エントリを次に示します。

```
java -jar pflowexecutor.jar -r MyFlow1 -u Bob1234 -p
"mypassword1"
```

次の例では、前の例と同じ情報に加えて引数を指定しています。

```
java -jar pflowexecutor.jar -r Flow1 -u Bob1234 -p
"mypassword1" -h spectrum.example.com -s 8888 -w -d "%" -i
1 -t 9999
```

次の例では、コマンド ラインでの起動と出力を示します。

```
D:\spectrum\pflow-executor>java -jar pflowexecutor.jar -u
Bob1234 -p "mypassword1" -r
validateAddressFlow1 -h spectrum.example.com -s 8888 -w -d
"%" -i
1 -t 9999
validateAddressJob1%111%succeeded
```

この例では、validateAddressFlow1 という名前のプロセス フローが実行しました (ID は 111)。エラーは発生していません。その他の実行結果としては、"失敗" と "実行中" があります。

プロセス フローのプロパティ ファイルの使用

プロパティ ファイルには引数が含まれ、Process Flow Executor で`-f`引数を使ってプロパティ ファイルのパスを指定することにより、引数を再利用できます。プロパティ ファイルには、少なくともプロセス フロー (`r`)、ユーザ ID (`u`)、およびパスワード (`p`) が含まれる必要があります。

1. テキスト エディタを起動します。
2. 次の例で示すように、1 行で 1 つの引数を指定します。引数の一覧については、コマンド ラインからのプロセス フローの実行（164ページ）を参照してください。

   注： プロパティ ファイルを使用して、入力ファイルまたは出力ファイルをオーバーライドすることはできません。入力ファイルまたは出力ファイルのオーバーライドは、コマンド ライン引数を使用することによってのみ行うことができます。

```
d=%
h=myserver.mydomain.com
i=30
u=user
p=password
r=MyFlow1
s=8888
```

3. .properties という拡張子をつけてこのファイルを保存します ("example.properties" など)。
4. Process Flow Executor を実行するときに、`-f`引数を使用してプロパティ ファイルのパスを指定します。コマンド ライン入力とプロパティ ファイル入力を組み合わせて使用することもできます。コマンド ラインでの引数の方が、プロパティ ファイルで指定されている引数より優先されます。

   ```
   java -jar pflowexecutor.jar -f /dcg/flow.properties -r MyFlow2
   ```

   この例では、プロセス フロー MyFlow2 が、プロパティ ファイルで指定されているプロセス フローより優先されます。

データフローのスケジュール

データフロー スケジュールは、ジョブまたはプロセス フローを特定の時間に自動的に実行します。データフローを 1 回実行するようにスケジュールするか、繰り返し実行するように設定することができます。

1. ジョブまたはプロセス フローをエクスポーズしていない場合は、エクスポーズします。

   ジョブやプロセス フローをエクスポーズするには、Enterprise Designer でジョブまたはプロセス フローを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。

2. Management Console を開きます。
3. **[実行]** を表示し、**[スケジューリング]** をクリックします。
4. **[追加]** をクリックして新しいスケジュールを作成するか、既存のスケジュールを変更する場合は、スケジュールを選択して **[変更]** をクリックします。
5. **[タスクの追加]** または **[タスクの変更]** ウィンドウで、タスクに対する設定を選択します。

- **[タスク名]** — スケジューリングするこのタスクの名前を指定します。これがタスク一覧に表示される名前になります。
- **[フロー タイプ]** — スケジューリングするプロセスの種類 (ジョブまたはプロセス フロー) を選択します。
- **[フロー名]** — スケジューリングするジョブまたはプロセス フローを選択します。保存およびエクスポーズされたジョブとプロセス フローだけが、ここで使用可能です。選択したいジョブまたはプロセス フローが表示されない場合は、Enterprise Designer でジョブまたはプロセス フローを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。
- **[タスクの有効化]** — ジョブまたはプロセス フローを指定した時間に実行するには、このボックスをオンにします。スケジュールを一時的に停止するにはこのボックスをオフにします。
- **[スケジュール]** — ジョブまたはプロセス フローを実行する日時を指定します。

6. フローで入力または出力にファイルを使用する場合は、これらのファイルはSpectrum™ Technology Platformサーバー上に存在するか、Management Console で外部リソースとして定義されたファイル サーバー上に存在する必要があります。これはジョブと、プロセス フロー内のジョブ アクティビティの両方に適用されます。ソース ステージまたはシンク ステージがクライアント コンピュータ上のファイルを参照している場合は、次の手順のいずれかを実行します。

オプション	説明
オプション 1: データフローを変更する	ファイルを Spectrum™ Technology Platformサーバーまたはファイル サーバーに移動し、データフローを変更します。 1. Enterprise Designer でフローを開きます。 2. ソースまたはシンク ステージをダブルクリックします。 3. **[ファイル名]** フィールドで、参照ボタンをクリックします。 4. **[リモート マシン]** をクリックして、必要なファイルを選択します。 **注：** Spectrum™ Technology Platformサーバーと同じコンピュータ上で Enterprise Designer を実行している場合、[リモート マシン] のクリックは、[マイ コンピュータ] をクリックするのと同じであるように見えます。しかし、ファイルが Spectrum™ Technology Platformサーバー上に存在することをシステムに認識させるためには、[リモート マシン] を使用してファイルを選択する必要があります。

オプション	説明
オプション 2: このスケジュールの実行時にデータフローのファイルの場所をオーバーライドする	データフローのソース ステージ (Read from File など) で指定された入力ファイルや、データフローのシンク ステージ (Write to File など) で指定された出力ファイルをオーバーライドできます。 1. **[オプション]** をクリックします。 2. **[ステージ ファイルの場所]** の下で、ローカル ファイルを参照するステージを選択します。 3. **[変更]** をクリックし、Spectrum™ Technology Platform サーバー上のファイルを選択します。

7. ジョブまたはプロセス フローを、繰り返し実行するようにスケジュールする場合は、**[タスクの繰り返し]** チェック ボックスをオンにし、**[繰り返し]** ボタンをクリックして、フィールドを設定します。
8. データフローが電子メール通知用に設定されている場合は、データフローの実行時に通知が送信される受信者を追加で指定できます。
   a) **[オプション]** をクリックします。
   b) [通知] の下の **[追加]** をクリックします。
   c) 通知を送信する電子メール アドレスを入力します。例えば、me@mycompany.com と入力します。
   d) **[OK]** をクリックします。

   注： 電子メール通知を機能させるには、Management Console で通知を設定する必要があります。また、データフローが通知をサポートするように設定されていることも確認してください。そのためには、Enterprise Designer でデータフローを開き、**[編集] > [通知]** を選択します。

9. **[OK]** をクリックします。

コントロール ファイルによるフローのトリガー

モニタリング対象のディレクトリ内でコントロール ファイルが検出されたときに、フローを自動的に実行することができます。この機能は、フローの実行の前に別のプロセスの完了が必要という状況において便利です。例えば、別のビジネス プロセスによって生成される入力ファイルを必要とするフローが考えられます。コントロール ファイルをフォルダに配置するように別のプロセスを設定し、そのコントロール ファイルを検出したらフローを実行するように Spectrum™ Technology Platformを設定することができます。

注: 必ず、フローで必要なすべてのファイルの準備が整い、処理できる状態になった後に、コントロール ファイルをモニタリング対象のディレクトリに配置するようにしてください。

1. フローをエクスポーズしていない場合は、エクスポーズします。

   フローをエクスポーズするには、Enterprise Designer でフローを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。

2. Management Console を開きます。
3. **[フロー] > [スケジュール]** に移動します。
4. [追加] ボタン をクリックします。
5. **[名前]** フィールドに、このスケジュールに付ける名前を入力します。これがスケジュールの一覧に表示される名前になります。
6. **[フロー]** フィールドに、実行するジョブまたはプロセス フローを入力します。保存およびエクスポーズされたジョブとプロセス フローだけが、ここで使用可能です。
7. フローを指定した後で、追加のフィールドが**[フロー]**フィールドの下に表示されます。フローの各ソース ステージ (Read from File など) と各シンク ステージ (Write to File など) に対応するフィールドが提供されます。これらのフィールドは、このスケジュールに従ってフローが実行されるときに使われるファイルを示します。デフォルトで、フローのソースとシンクで指定されたファイルが使用されます。スケジュールで使われるファイルを変えるには、ファイル パスを別のファイルへのパスに置き換えます。例えば、フローで使う Read from File ステージでデータが`C:\FlowInput\Customers.csv`から読み込まれるが、このスケジュールの実行に `C:\FlowInput\UpdatedCustomers.csv` のデータを使いたい場合は、`C:\FlowInput\UpdatedCustomers.csv` を [Read from File] フィールドに指定します。

   注: ソース ステージやシンク ステージで使われるファイルを変更するには、**リソース - ファイル サーバー** セキュア エンティティ タイプの読み取り権限が必要です。

フローがスケジュールによって実行される場合、フローで使われるファイルは Management Console で外部リソースとして定義されたSpectrum™ Technology Platformサーバーまたはファイル サーバーに存在する必要があります。これはジョブと、プロセス フロー内のジョブ アクティビティの両方に適用されます。ソース ステージまたはシンク ステージがクライアント コンピュータ上のファイルを参照している場合は、次の手順のいずれかを実行します。

オプション	説明
オプション 1: データフローを変更する	ファイルを Spectrum™ Technology Platformサーバーまたはファイル サーバーに移動し、データフローを変更します。 1. Enterprise Designer でデータフローを開きます。 2. ソースまたはシンク ステージをダブルクリックします。 3. **[ファイル名]** フィールドで、参照ボタンをクリックします。

オプション	説明
	4. **[リモート マシン]** をクリックして、必要なファイルを選択します。 **注：** Spectrum™ Technology Platformサーバーと同じコンピュータ上で Enterprise Designer を実行している場合、[リモート マシン] のクリックは、[マイ コンピュータ] をクリックするのと同じであるように見えます。しかし、ファイルが Spectrum™ Technology Platformサーバー上に存在することをシステムに認識させるためには、[リモート マシン] を使用してファイルを選択する必要があります。
オプション 2: このスケジュールの実行時にデータフローのファイルの場所をオーバーライドする	このスケジュールを実行するときに、フロー内のファイル参照をオーバーライドします。これを行うには、ソース フィールドとシンク フィールドに表示されているデフォルト ファイルを Management Console で定義された Spectrum™ Technology Platformサーバー上のファイルまたはファイル サーバー リソースへのパスで置き換えます。

8. **[トリガー]** フィールドで、**[コントロール ファイル]** を選択します。
9. **[コントロール ファイル]** フィールドで、フローをトリガするコントロール ファイルのフル パスと名前を指定します。ファイル名は、そのまま指定するか、ワイルドカードとしてアスタリスク (*) を使用できます。たとえば、`*.trg` と指定すると、フォルダに拡張子 `.trg` の任意のファイルが現れたときフローがトリガされます。

   コントロール ファイルの存在は、フローで必要とされるファイルがすべて揃っていて、フローで使用される準備が整っていることを意味します。

   コントロール ファイルは空白のファイルでもかまいません。ジョブの場合、コントロール ファイルは、Write to File ステージまたは Read from File ステージで設定されるファイル パスのオーバーライドを指定できます。コントロール ファイルを使用してファイル パスをオーバーライドするには、Read from File ステージまたは Write from File ステージの名前と、その入力ファイルまたは出力ファイルを最後の引数として指定します。

```
stagename=filename
```

例:

```
Read\ from\ File=file:C:/myfile_input.txt
Write\ to\ File=file:C:/myfile_output.txt
```

コントロール ファイルで指定するステージ名は、データフロー内のステージのアイコンの下に表示されるステージ ラベルと一致する必要があります。例えば、入力ステージのラベルが "Read From File" である場合は、次のように指定します。

```
Read\ From\ File=file:C:/inputfile.txt
```

入力ステージのラベルが "Illinois Customers" である場合は、次のように指定します。

```
Illinois\ Customers=file:C:/inputfile.txt
```

Read from File または Write to File の場所をオーバーライドする場合は、次のガイドラインに従ってください。

- パスの先頭に "file:" プロトコルを付けます。例えば、Windows では、"`file:C:/myfile.txt`" と指定し、Unix または Linux では、"`"file:/testfiles/myfile.txt"`" と指定します。
- ファイルのコンテンツには、ASCII ベースの ISO-8559-1 (Latin-1) 互換文字エンコーディングを使用する必要があります。
- ファイルのパスには、バックスラッシュではなくスラッシュ (/) を使用する必要があります。
- ステージ名に含まれるスペースは、バックスラッシュでエスケープする必要があります。
- ステージ名は大文字と小文字を区別します。

> **注：** コントロール ファイルによるトリガーを使うスケジュールが複数ある場合は、それぞれで異なるコントロール ファイルをモニタリングする必要があります。そうしなければ、同じコントロール ファイルで複数のジョブまたはプロセス フローがトリガされ、予期せぬ動作を引き起こす恐れがあります。構造上の理由から、すべての必要なファイルとコントロール ファイルを専用ディレクトリに配置することをお勧めします。

10. **[ポーリング間隔]** フィールドに、コントロール ファイルの有無をチェックする間隔を指定します。例えば、10 を指定するとモニタリング対象フォルダ内にコントロール ファイルがないか 10 秒間隔でチェックされます。

    デフォルトでは、60 秒に設定されています。

11. **[作業フォルダ]** フィールドに、フローの実行中にコントロール ファイルを一時的に配置するフォルダを指定します。Spectrum™ Technology Platformは、フローの実行前に、モニタリング対象フォルダから作業フォルダにファイルをコピーします。これによりモニタリング対象フォルダが空になるため、同じコントロール ファイルによって再びフローが開始されることがなくなります。

12. **[作業フォルダのオプション]** フィールドで、データフローの実行終了時に作業フォルダ内のファイルをどう処理するかを指定します。

Keep	ファイルを現在の場所に現在の名前で保持します。このオプションを選択した場合、作業フォルダ内のファイルはこのスケジュールが実行されるたびに上書きされます。
Move to	ファイルを作業フォルダから指定のフォルダに移動します。これにより、作業フォルダにあったファイルを別の場所に移動し、次回ファイル モニターが実行される際に上書きされないよう、これらのファイルを保持することができます。また、このオプションを使用して、別のデータフローなどの下流プロセスをトリガするように、ファイルを別のモニタリング対象フォルダに移動することもできます。
Rename with time stamp	作業フォルダ内のファイル名にタイムスタンプを付加します。これにより、作業フォルダ内のファイルのコピーを保持できます。フォルダの名前を変更すれば一意の名前が与えられるので、ファイル モニターがデータフローを次回実行してもファイルが上書きされなくなるからです。
削除	フローの実行終了時に作業フォルダ内のファイルを削除します。

13. フローが電子メール通知用に設定されている場合は、スケジュールの実行時に通知が送信される受信者を追加で指定できます。ここで指定した受信者は、フローの通知設定で指定された受信者に追加される形で通知を受け取ります。通知を送信するフローを設定するには、Enterprise Designer でフローを開き、**[編集] > [通知]** を選択します。
14. **[保存]** をクリックします。

例: モニタリング対象フォルダと作業フォルダ

例えば、あなたは自動車修理店を営んでいるとします。あなたは、毎日、前日に訪れた顧客に、今後のサービスの割引クーポンを送付したいと考えています。これを達成するために、この日の顧客のリストを取得し、顧客名の大文字と小文字の区別が正しいことを確認し、住所を検証するデータフローを用意しているとします。また、別のシステムによって、その日の顧客のリストが毎日夕方に生成されます。あなたは、このシステムにより生成される顧客リストを含むファイルを、データフローへの入力として使用しようと考えています。

顧客リストを生成するシステムにより、顧客リストが`DailyCustomerReport`という名前のフォルダに格納されます。さらに、完了すると、空白のトリガ ファイルがこのフォルダに格納されます。このフォルダを監視するようにSpectrum™ Technology Platformを設定するには、トリガ ファイルとして次を指定します。

```
C:\DailyCustomerReport\*.trg
```

これにより、このフォルダに拡張子 .trg の任意のファイルが現れると常に、Spectrum™ Technology Platformよりデータフローが実行されます。特定のファイル名を指定することも可能ですが、この例ではワイルド カードを使用しています。

`DailyCustomerReport`フォルダで .trg ファイルが検出された場合、Spectrum™ Technology Platform は、データフローを実行する前にこのファイルを別のフォルダに移動する必要があります。ファイルを移動しないと、次回のポーリング感覚で再びこのファイルが検出され、結果としてこのデータフローが再び実行されることになるためです。そのため、ファイルは「作業フォルダ」に移され、データフローの実行中はそのフォルダに格納されます。作業フォルダとして、`C:\SpectrumWorkingFolder`を選択します。

顧客リストを処理するデータフローの完了後は、請求プロセスをトリガするための別の場所にトリガ ファイルを移動させるとします。そのため、**[フォルダへ移動]** オプションを選択し、`C:\DailyBilling` という名前のフォルダを選択します。

この例では、トリガ ファイルは最初 `C:\DailyCustomerReport`に配置され、次に作業フォルダ`C:\SpectrumWorkingFolder` に移動されます。データフローの官僚が、トリガ ファイルは `C:\DailyBilling`に移動され、請求プロセスを開始します。

フロー ステータスと履歴の表示

ジョブ、プロセス フロー、およびサービス実行の履歴は、Management Console と Enterprise Designerで表示できます。

Management Console の場合

Management Console でフロー ステータスや履歴を表示するには、**[フロー] > [履歴]** を選択します。**[フロー]** タブにはジョブとプロセス フローの履歴が表示され、**[トランザクション]** タブにはサービス履歴が表示されます。

> **注:** フローの履歴の場合、**[結果]** 列の上にカーソルを置くと表示されるレコード数は、データフローのすべてのシンクによって出力として書き出されるレコードの総数です。データフローにおいてレコードが結合または分割されたり、新しいレコードが作成されたりする場合は、この数は入力レコード数とは異なる可能性があります。

デフォルトで、トランザクションの履歴は無効になっています。トランザクションの履歴を有効にすると、パフォーマンスに悪影響が出るためです。トランザクションの履歴を表示する場合は、**[トランザクションのログ記録]** スイッチをクリックしてトランザクション履歴のログへの記録を

有効にする必要があります。ユーザの操作履歴を表示するには、**[システム] > [ログ]** でアクセスできる監査ログを使うことをお勧めします。

フローの履歴リストは、30 秒ごとに自動的に更新されます。それよりも先に更新するには、更新ボタン をクリックします。

Enterprise Designer の場合

Enterprise Designerでフロー ステータスや履歴を表示するには、**[表示] > [実行履歴]** を選択します。

フローの履歴リストは、30 秒ごとに自動的に更新されます。実行履歴の表示が遅いと感じる場合は、**[自動更新]** チェックボックスをオフにします。

[ジョブ] タブでは、ジョブ ステータスの監視、実行中のジョブの一時停止、再開、またはキャンセル、完了したジョブの削除を行います。

> **注：[ジョブ]** タブに表示されるレコード数は、データフローのすべてのシンクによって出力として書き出されるレコードの総数です。データフローにおいてレコードが結合または分割されたり、新しいレコードが作成されたりする場合は、この数は入力レコード数とは異なる可能性があります。

- **[Succeeded]** 列には、データフローのすべてのシンクによって出力として書き出されるレコードで、[ステータス] フィールドの値が空であるものの総数が表示されます。
- **[Failed]** 列には、データフローのシンクによって出力として書き出されるレコードで、[ステータス] フィールドの値が F であるものの総数が表示されます。
- **[Malformed]** 列には、ソース ステージのすべてのエラー ポートからのレコードの総数が表示されます。

[プロセス フロー] タブでは、プロセス フロー ステータスの監視、実行中のプロセス フローのキャンセル、完了したプロセス フローの削除を行います。各プロセス フローの横にある **[+]** 記号をクリックすると、そのプロセス フローのアクティビティ ステータス情報が表示されます。このエリアには、次の情報が表示されます。

アクティビティ名	成功アクティビティを含む、プロセス フローを構成するすべてのアクティビティの名前が含まれます	
ステータス	アクティビティのステータスです (失敗、成功、実行中、キャンセル)	
リターン コード	プロセス フローの結果を示すコード。次のいずれかです。	
	1	プロセス フローは失敗しました。
	0	プロセス フローは正常終了しました。
	-1	プロセス フローはキャンセルされました。
	その他の数値	プロセス フローに「プログラムの実行」アクティビティが含まれる場合は、外部プログラムが独自のコードを返す可能性があ

	ります。ReturnCode 列の値が 1、0、-1 以外である場合、それは外部プログラムによるものです。外部プログラムのリターンコードの説明については、そのプログラムのドキュメントを参照してください。
開始	アクティビティが開始した日時です
終了	アクティビティが終了した日時です
コメント	アクティビティに関するコメントです

フロー履歴のダウンロード

Management Console の [履歴] 画面に表示される情報を Microsoft Excelファイルにダウンロードできます。

1. Management Console を開きます。
2. **[フロー] > [履歴]** に移動します。
3. サービスの履歴情報をダウンロードするには、**[トランザクション履歴]** をクリックします。ジョブとプロセス フローの履歴をダウンロードするには、**[フロー]** タブを選択します。
4. ダウンロード ボタン をクリックします。

   ヒント： 履歴リストから特定の項目だけをダウンロードするには、フィルタ設定を変更してダウンロードしたい履歴だけを表示します。

形式に誤りのあるレコード数のデフォルト値の設定

形式に誤りのあるレコードとは、Spectrum™ Technology Platform が解析できない入力レコードです。デフォルトでは、形式に誤りのあるレコードが 1 つでもジョブの入力データに含まれていると、ジョブは停止します。この設定を変更すると、形式に誤りのあるレコードを複数許容したり、そうしたレコードを無制限に許容したりできます。この手順では、システム上のジョブに対して形式に誤りのあるレコード数の最大値のデフォルトを設定する方法について説明します。

> **注：** Enterprise Designer でジョブを開き、**[編集] > [ジョブ オプション]** に移動して、形式に誤りのあるレコード数の最大数のデフォルトをオーバーライドできます。

1. Management Console を開きます。
2. **[フロー] > [デフォルト]** を選択します。
3. **[形式に誤りのあるレコード]** をクリックします。
4. 以下のオプションのいずれかを選択します。

形式に誤りのあるレコードがこれだけ多く含まれているジョブを停止	形式に誤りのあるレコードが入力データに 1 つ以上含まれている場合は、このオプションを選択します。ジョブ停止のトリガとして設定する、形式に誤りのあるレコードの数を入力します。デフォルト値は 1 です。
形式に誤りのあるレコードによってフローを終了しない	入力データに含まれる、形式に誤りのあるレコードの数を無制限に許容する場合は、このオプションを選択します。

レポートに関するデフォルト値の設定

レポートは、レポート ステージが含まれているジョブによって生成されます。レポートには、処理の概要 (ジョブによって処理されたレコード数など) や郵便フォーム (USPS CASS 3553 フォームなど) を含めることができます。一部のモジュールには、定義済みレポートが付属しています。カスタム レポートを作成することもできます。レポートに関するデフォルト値を設定すると、レポートを保温するためのデフォルト設定が定められます。こうしたデフォルト設定は、Enterprise Designer を使用すると、特定のジョブやあるジョブ内の特定のレポートについてオーバーライドできます。

ここでは、システムに対するデフォルト レポート オプションを設定する方法を示します。

1. Management Console を開きます。
2. **[フロー] > [デフォルト]** を選択します。
3. **[レポート]** をクリックします。
4. レポートの保存に使用する形式を選択します。レポートは HTML、PDF、またはテキストとして保存できます。
5. レポートの保存場所を選択します。

レポートをジョブ履歴に保存	レポートをジョブ履歴の一部としてサーバーに保存します。こうすると、Management Console および Enterprise Designer ユーザはレポートを実行履歴で参照できるようになって便利です。
レポートをファイルに保存	指定した場所にあるファイルにレポートを保存します。これは、Spectrum™ Technology Platform ユーザではない人とレポートを共有したい場合に便利です。また、レポートのアーカイブを別の場所に作成したい場合にも便利です。この方法で保存されたレポートを表示する場合は、そのレポートの形式を開くことができる任意のツール (PDF レポートの場合は PDF ビューア、HTML レポートの場合は Web ブラウザ) を使用できます。

6. **レポートをファイルに保存** を選択したは、次のフィールドに入力します。

レポートの場所 レポートの保存先フォルダです。

レポート名に追加 ファイル名に含める変数の情報を指定します。以下から 1 つ以上を選択できます。

- **ジョブ ID** ジョブの実行に対して割り当てられる一意の ID です。システムで初めて実行したジョブの ID は 1 になります。2 回目にジョブを実行したときには、それが前回と同じジョブでも異なるジョブでも、ジョブ ID は 2 になります (3 回目以降も同様)。
- **ステージ** レポートにデータを提供したステージの名前であり、Enterprise Designer 内のレポート ステージで指定されているものです。
- **Date** レポートが作成された年月日です

既存のレポートを上書きする 以前のレポートを、同じファイル名を持つ新しいレポートで置き換えます。新しいレポートと同じ名前の既存レポートがあるのにこのオプションを選択していない場合、ジョブは正常に完了しますが、新しいレポートは保存されません。その場合は、レポートが保存されなかったことを示すコメントが実行履歴に表示されます。

サービスのエクスポーズ

Web サービスとしてのサービスのエクスポーズ

Spectrum™ Technology Platform サービスは、REST または SOAP Web サービスとして使用できます。サービスをサーバー上で Web サービスとして使用できるようにするには

1. Enterprise Designer を開きます。
2. Web サービスとしてエクスポーズするサービスを開きます。
3. **[編集] > [Web サービス オプション]** の順に選択します。
4. サービスを SOAP Web サービスとして使用できるようにするには、**[SOAP Web サービスとして公開]**チェック ボックスをオンにします。
5. サービスを REST Web サービスとして使用できるようにするには、**[REST Web サービスとして公開]** チェック ボックスをオンにして、次の手順を実行します。

a) デフォルトのエンドポイントをオーバーライドする場合は、使用するエンドポイントを **[パス]** フィールドに指定します。

パスの指定はオプションです。REST Web サービスのデフォルトのエンドポイントは次のとおりです。

`http://server:port/rest/service_name/results.qualifier`

別のエンドポイントを使用する場合は、指定したパスがサービス名の後に追加されます。例えば、"`Americas/Shipping`" と **[パス]** フィールドに指定すると、JSON エンドポイントは次のようになります。

`http://myserver:8080/rest/MyService/`**`Americas/Shipping`**`/results.json`

[変数の挿入] ドロップダウンメニューをクリックして、使用するフィールドまたはオプションを選択することにより、データフローからのフィールドとオプションをパス内の変数名として使用できます。変数はパス内において、 `${Option.Name}` (データフロー オプションの場合)、または `${Data.Name}` (データフロー フィールドの場合) という表記で記述されます。

b) REST Web サービスはデフォルトで、GET メソッドをサポートし、XML および JSON 形式でデータを返します。**[追加]** をクリックしてリソースを Web サービスに追加することによって、その他の HTTP メソッドや出力形式を定義できます。

リソースを追加する際に、HTTP メソッド (**GET** または **POST**) を選択できます。サポートされるデータ形式は以下のとおりです。これらの形式のすべてが使用できるとは限りません。一部の形式は、お使いの Spectrum™ Technology Platform サーバー上に特定のモジュールがインストールされている場合のみ使用可能であるためです。

XML デフォルトの XML 形式。XML をリクエストとレスポンスの形式として使用し、処理するデータに特殊な XML 形式が存在しない場合は、この形式を使用します。

JSON デフォルトの JSON 形式。JSON をリクエストとレスポンスの形式として使用し、処理するデータに特殊な JSON 形式が存在しない場合は、この形式を使用します。

GeoJSON 地理的データを処理するサービスに適合した特殊な JSON 形式です。ジオメトリおよび次のネイティブ プラットフォーム 型でのみサポートされています。

- boolean
- double
- float
- integer
- bigdecimal
- long

- string
- date
- time
- datetime
- timespan

その他の型を持つフローをエクスポーズしようとする場合は、GeoJSON を指定できません (設計時にエラーが表示されます)。また、GeoJSON では単一ジオメトリのみを使用できます。出力に複数のジオメトリ フィールドが含まれている場合は、システムは "geometry" というフィールドに続いて "obj" というフィールドを検索します。そうしたフィールドが存在しない場合は、最初のジオメトリ フィールドが選択されます。

c) **[OK]** をクリックします。

新しいリソースが Web サービスに追加されます。

6. Web サービス オプションの設定を終えたら、**[OK]** をクリックします。
7. ツール バーにあるグレーの電球をクリックして、サービスをエクスポーズします。

データフローがエクスポーズされると、次のように、Enterprise Designer ツール バーの電球ボタンがデータフローのエクスポーズを示します。

サービスが Web サービスとしてエクスポーズされているか確認するには、以下のいずれかの URL に移動します。

- REST の場合: `http://server:port/rest`
- SOAP の場合: `http://server:port/soap`

ここで、*server* は Spectrum™ Technology Platform サーバーの名前または IP アドレス、*port* は HTTP 通信に使用するポート番号です。

API アクセスのためにサービスをエクスポーズする

Spectrum™ Technology Platform API を使ってサービスにアクセスできるようにするには、そのサービスをエクスポーズする必要があります。

1. Enterprise Designer でサービスを開きます。
2. ツール バーにあるグレーの電球をクリックして、サービスをエクスポーズします。

データフローがエクスポーズされると、次のように、Enterprise Designer ツール バーの電球ボタンがデータフローのエクスポーズを示します。

実行時オプション

データフロー実行時オプションの追加

データフロー実行時オプションを使用すると、データフロー実行時のステージの動作を制御できます。これは、実行時にデータフローの動作を変更したい場合に役立ちます。例えば、データフローの Read from DB ステージで指定されたデータベースを使用する代わりに、データフローの実行時に Read from DB ステージ用のソース データベースを指定することができます。

以下では、実行時に設定できるオプションをエクスポーズする手順について説明します。この手順を実行すると、実行時に次のような手法でデータフロー オプションを設定できるようになります。

- ジョブの場合、データフロー オプション プロパティ ファイルと Job Executor の -o 引数を使用して実行時オプションを指定できます。
- サービスの場合、実行時オプションを API オプションとして指定できます。
- Web サービスとしてエクスポーズされたサービスの場合、実行時オプションをリクエスト内のパラメータとして指定できます。
- サブフローの場合、親データフロー タイプ (ジョブ、サービス、Web サービスとしてエクスポーズされたサービス) に応じて、実行時オプションは親データフローから継承され、上記のいずれかの方法によってエクスポーズされます。

データフローに実行時オプションを追加するには、以下の手順に従います。

1. Enterprise Designer でデータフローを開きます。
2. 埋め込まれたデータフロー内のステージに対して実行時オプションを設定する場合は、埋め込まれたデータフローを開きます。
3. ツールバー上のデータフロー オプション アイコンをクリックするか、**[編集] > [データフロー オプション]** をクリックします。**[データフロー オプション]** ダイアログ ボックスが表示されます。

4. **[追加]** をクリックします。**[データフロー オプションの定義]** ダイアログ ボックスが表示されます。
5. **[オプション名]** フィールドに、このオプションに付ける名前を入力します。このオプションを設定するには、実行時にこのオプション名を指定する必要があります。
6. **[ラベル]** フィールドで、別のラベルを指定できます。また、オプション名と同じラベルを使用することもできます。
7. **[説明]** フィールドにオプションの説明を入力します。
8. **[ターゲット]** フィールドで、このオプションをデータフローのすべてのステージに適用するか、特定のステージにのみ適用するかを選択します。

   選択ステージ

   指定するステージにのみオプションを適用する場合は、このオプションを選択します。

   すべてのステージ

   データフローのすべてのステージにオプションを適用する場合は、このオプションを選択します。

   トランスフォームを含む

   データフローの Transformer ステージの Custom Transform で実行時オプションを使用できるようにする場合は、このオプションを選択します。このオプションを選択すると、Custom Transform の Groovy スクリプトの中で、実行時に指定された値にアクセスできます。次の構文を使用します。

   ```
   options.get("optionName")
   ```

   例えば、`casing` という名前のオプションにアクセスするには、Custom Transform スクリプトに次の式を含めます。

   ```
   options.get("casing")
   ```

9. **[ターゲット]** フィールドで **[選択ステージ]** を選択した場合は、**[データフロー オプションをステージにマッピングします]** テーブルにデータフローのステージの一覧が表示されます。データフロー オプションとしてエクスポーズするオプションを選択します。最初の項目を選択すると、**[デフォルト値]** フィールドと **[有効値]** フィールドにデータが表示されます。

   注： 複数のオプションを選択すると、データフロー オプションによって複数のステージ オプションを制御できます。この場合、選択するどのステージ オプションも有効値を共有する必要があります。例えば、あるオプションの値が「Y」と「N」である場合、追加されるどのオプションも値のセットに「Y」または「N」のいずれかを含む必要があり、実行時に使用できる値も共通する値に限られます。つまり、値が「Y」と「N」であるオプションを選択した場合、値が「E」、「T」、「M」、および「L」であるオプションを選択することはできませんが、値が「P」、「S」、および「N」であるオ

プションは、「N」という値が共通であるため、選択できます。ただし、このオプションに対して使用可能な値は「N」のみで、「Y」、「P」、または「S」は使用できません。

10. 実行時に指定できる値を制限するには、**[有効値]** フィールドのすぐ右にあるアイコンをクリックしてオプションを編集します。
11. デフォルト値を変更する場合は、**[デフォルト値]** フィールドに別の値を指定します。

    注： サービスの場合、デフォルト値はサービスを最初にエクスポーズする前にしか変更できません。サービスをエクスポーズした後は、Enterprise Designer を使用してデフォルト値を変更することはできません。代わりに、Management Console を使用する必要があります。詳細については、「デフォルト サービス オプションの指定（183ページ）」を参照してください。

12. **[OK]** をクリックします。
13. 必要に応じて、オプションの追加を続けます。
14. オプションの追加を終えたら、**[データフロー オプション]** ダイアログ ボックスの **[OK]** をクリックします。
15. 埋め込まれたデータフローに実行時オプションを追加した場合は、実行時にそのオプションを使用可能にするために、その実行時オプションを親データフローとすべての先祖データフローで定義する必要があります。これを行うには、埋め込まれたデータフローを含むデータフローを開き、作成したオプションをエクスポーズします。必要に応じて、そのデータフローの親を開き、そこでこのオプションを定義します。すべての先祖でこのデータフロー オプションが定義されるまで、これを続けます。

    例えば、"A" という名前のデータフローに、"B" という名前の埋め込まれたデータフローが含まれており、その中に "C" という名前の埋め込まれたデータフローが含まれているとします。つまり、埋め込まれたデータフローは A > B > C という階層構造を持ちます。Casing という名前のオプションを埋め込まれたデータフロー "C" 内のステージでエクスポーズしたい場合、埋め込まれたデータフロー "C" を開いてこれを定義します。続いて、埋め込まれたデータフロー "B" を開いて、このオプションを定義します。最後に、データフロー "A" を開いてオプションを定義することで、このオプションが実行時に使用可能になります。

これで、実行時にオプションを指定できるようにデータフローが設定されました。

デフォルト サービス オプションの指定

デフォルト サービス オプションは、システム上の各サービスのデフォルトの動作を制御します。サービス内の各オプションのデフォルト値を指定できます。デフォルト オプション設定は、API

呼び出しまたは Web サービス リクエストで、オプションの値が明示的に定義されていない場合に有効です。また、デフォルト サービス オプションは、Enterprise Designer でこのサービスを使用してデータフローを作成する際にデフォルトとして使用される設定です。

> **注：** サービスの場合、デフォルト値はサービスを最初にエクスポーズする前にしか変更できません。サービスをエクスポーズした後は、Enterprise Designer を使用してデフォルト値を変更することはできません。代わりに、Management Console を使用する必要があります。

1. Management Console を開きます。
2. **[サービス]** をクリックします。
3. 必要なサービスの横にあるチェックボックスをオンにして、編集ボタン をクリックします。
4. サービスのオプションを設定します。サービスのオプションの詳細については、そのサービスのモジュールのソリューション ガイドを参照してください。
5. **[保存]** をクリックします。

データフロー実行時オプションの削除

データフロー実行時オプションを使用すると、ステージ オプションを実行時に設定できます。ステージ オプションは、プロセス フローから、または Job Executor コマンドライン ツールから、ジョブを呼び出すときに設定できます。以下では、データフローの実行時オプションを削除して、そのオプションを実行時に設定できないようにする方法を説明します。

1. ジョブ、サービス、またはサブフローを開きます。
2. **[データフロー オプション]** アイコンをクリックするか、**[編集] > [データフロー オプション]** をクリックします。**[データフロー オプション]** ダイアログ ボックスが表示されます。
3. 削除するオプションをハイライト表示し、**[削除]** をクリックします。

データフローの電子メール通知の設定

データフローを設定して、ジョブ ステータス情報を電子メールで通知することができます。例えば、データフローがエラーになったことを電子メールで通知する場合にこの機能を利用できます。電子メールによる通知には、フロー名、開始時間と終了時間、処理されたレコード数などの情報を含めることができます。

注： データフローの通知を設定するには、まず、Management Console でメール サーバーを設定する必要があります。詳細については、*『Spectrum™ Technology Platform 管理ガイド』*を参照してください。

1. Enterprise Designer でデータフローまたはプロセス フローを開くには、**[編修] > [通知]** を選択します。
2. **[追加]** をクリックします。
3. **[通知送信先]** フィールドに、通知を送信する電子メール アドレスを入力します。
4. 通知するイベントを選択します。
5. **[件名]** フィールドに、電子メールの件名の行に表示するテキストを入力します。
6. **[メッセージ]** フィールドに、電子メールの本文に表示するテキストを入力します。

   ジョブに関する情報を電子メールに含めたければ、**[ここをクリックして件名またはメッセージにフィールドを挿入]**をクリックします。ジョブ情報には、開始時間、終了時間、失敗したレコードの数などがあります。

7. 通知の内容を確認するには、**[プレビュー]** をクリックします。
8. **[OK]** をクリックします。**[通知]** ダイアログ ボックスが再表示され、その中に新しい通知が表示されます。
9. **[OK]** をクリックします。

5 - フローのプロセス フローへの結合

このセクションの構成

プロセス フローの概要

プロセス フローでは、ジョブや外部アプリケーションなどの一連のアクティビティが実行されます。プロセス フロー内の各アクティビティは、1 つ前のアクティビティが終了した後に実行されます。プロセス フローは、複数のデータフローを連続して実行したり、外部プログラムを実行したりする場合に便利です。例えば、プロセス フローでは、名前を正規化し、住所を照合してから、郵便料金の割引が受けられるようにレコードを正しい順序にソートするための外部アプリケーションを呼び出すようなジョブを実行することができます。次の図はそのようなプロセス フローを示したものです。

この例では、ジョブ Standardize Names と Validate Addresses は Spectrum™ Technology Platform サーバーでエクスポーズされたジョブです。プログラムの実行は外部アプリケーションを起動し、成功アクティビティはプロセス フローの終了を示します。

プロセス フローの設計

プロセス フローの作成

プロセス フローを作成するには、Enterprise Designer を使用して、ジョブや外部アプリケーションを実行する連続したアクティビティを作成します。

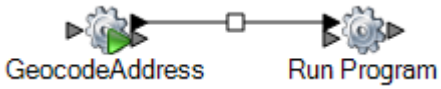
1. Enterprise Designer を開きます。
2. **[ファイル] > [新規作成] > [プロセス フロー]** の順に選択します。
3. プロセス フローで実行する最初のアクションを追加します。以下のいずれかの方法を実行します。
   - ジョブを実行するには、ジョブのアイコンをパレット内の **[アクティビティ]** フォルダからキャンバスにドラッグします。
   - 外部プログラムを実行するには、プログラムの実行アイコンをパレット内の **[アクティビティ]** フォルダからキャンバスにドラッグします。

4. プロセス フローで実行する 2 番目のアクションを追加します。

   ジョブのアイコンをキャンバスにドラッグしてジョブを追加したり、プログラムの実行アイコンをキャンバスにドラッグして外部プログラムを追加したりできます。

5. 最初のアイコンの右側にあるグレーの三角形をクリックし、ドラッグして 2 番目のアイコンの左側にあるグレーのアイコンに接続することにより、2つのアクティビティを接続します。

   例えば、最初に GeocodeAddress というジョブを実行してから、外部プログラムを実行するプロセス フローを作成する場合、プロセス フローは次のようになります。



6. その他のアクティビティを必要に応じて追加します。
7. プロセス フローで実行するすべてのアクテビティを追加したら、成功アクティビティをキャンバスにドラッグし、プロセス フローの最後のアクティビティに接続します。

   例えば、このプロセス フローには、2 つのジョブ ("Standardize Names" と "Validate Addresses") と 1 つのプログラムの実行アクティビティが含まれています。このプロセス フローの最後は成功アクティビティです。

8. 実行時オプションを設定するには、キャンバスに配置したアクティビティをダブルクリックします。アクティビティ間の接続をダブルクリックして、分岐オプションを設定することもできます。

変数を使用したファイルの参照

プロセス フローの変数は、プロセス フローの複数のアクティビティで同じファイルを参照する場合に便利です。変数を使用して、ある場所にあるファイルを定義し、そのファイルを参照する必要がある下流のアクティビティすべてでその変数を参照することができます。ファイルが変更された場合は、下流のアクティビティすべてを変更することなく、変数定義を変更することができます。

ジョブ アクティビティをプロセス フローに追加すると、データフロー内のソースおよびシンクごとに、変数が自動的に作成されます。ジョブのソースまたは シンクで定義されていないファイルをプロセス フローで使用したい場合は、変数を作成することができます。

プログラムの実行アクティビティを追加する場合、デフォルトでは変数は作成されません。プログラムの実行アクティビティで変数を使用する場合は、変数を作成する必要があります。

この手順では、ジョブ アクティビティまたはプログラムの実行アクティビティに変数を作成する方法を説明します。

1. Enterprise Designer でプロセス フローを開きます。
2. 変数を定義するジョブ アクティビティまたはプログラムの実行アクティビティをダブルクリックします。

   注： 変数は、その変数を定義するアクティビティ以降のアクティビティで参照できます。したがって、変数を使用するアクティビティより前のアクティビティで変数を定義してください。

3. **[変数]** タブをクリックします。
4. 変数を作成します。

オプション	説明
入力ファイルの変数を作成する場合	**[入力]** の横にある **[追加]** をクリックします。**[名前]** フィールドに、変数の名前を入力します。下流のアクティビティはこの名前を参照します。**[場所]** フィールドで、次のいずれかを選択します。 **ジョブで指定されているファイルを使用する**：ジョブのソース ステージで定義されているファイルを使用するには、このオプションを選択します。このオプションは、ジョブ アクティビティ用の変数を定義する場合にのみ使用できます。 **サーバー上のファイルを参照する**：この変数に割り当てるファイルを選択する場合に選択します。 **上流のアクティビティのファイルを参照する**：上流のステージから既存の変数に割り当てられているファイルを使用する場合に選択します。
出力ファイルの変数を作成する場合	**[入力]** の横にある **[追加]** をクリックします。**[名前]** フィールドに、変数の名前を入力します。下流のアクティビティはこの名前を参照します。**[場所]** フィールドで、次のいずれかを選択します。 **サーバー上のファイルを参照する**：この変数に割り当てるファイルを選択する場合に選択します。 **サーバーが管理する一時ファイル**：必要に応じて自動的に作成および削除される一時ファイルを参照する場合に選択します。このオプションは、ファイルがプロセス フローの中間ステッ

オプション	説明
	プとしてのみ使用され、プロセス フローが完了した後では必要ない場合に便利です。

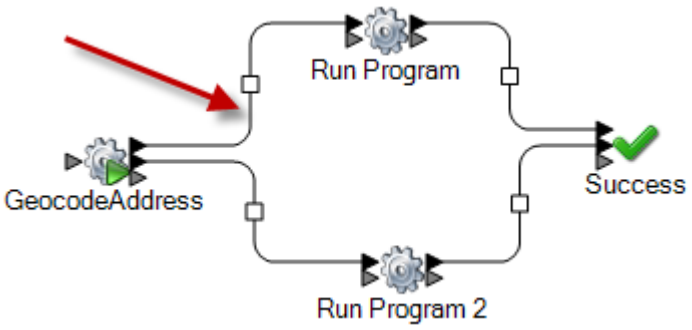
5. **[OK]** をクリックして **[変数の追加]** ウィンドウを閉じます。
6. **[OK]** をクリックしてアクティビティ オプション ウィンドウを閉じます。
7. 下流のアクティビティで変数を参照するには
   a) 変数を参照するアクティビティをダブルクリックします。
   b) 変数を参照する入力ステージを選択し、**[変更]** をクリックします。
   c) **[場所]** フィールドで、Reference an upstream activity's file... を選択します。

プロセス フローへの条件付きロジックの追加

条件付きロジックをプロセス フローに追加し、前のアクティビティのリターン コードに基づいて、異なるアクティビティを実行することができます。例えば、ジョブがリターン コード 1 を返した場合はあるアクティビティを、リターン コード 0 を返した場合は別のアクティビティを実行することができます。このようにして、プロセス フローに条件付き分岐を組み込むことができます。

1. Enterprise Designer でプロセス フローを開きます。
2. フローの 2 つのアクティビティの間にある分岐をダブルクリックします。

   *分岐*は、2 つのアクティビティを接続する線です。例えば、次に示す GeocodeAddress アクティビティとプログラムの実行アクティビティの間の線は分岐です。



   [分岐オプション] ウィンドウが表示されます。

3. 追加する分岐の種類を選択します。

単純	プロセス フロー内で常にこのパスを実行する場合に選択します。

条件付き 上流のアクティビティが特定のリターン コードまたは一定範囲のリターン コードを返す場合のみ、プロセス フロー内でこのパスを実行する場合に選択します。

それ以外の場合 アクティビティから出ている他の分岐の条件を満たさない場合のみ、プロセス フロー内でこのパスを実行する場合に選択します。

注 : 1 つのアクティビティから出ている分岐の中で、**[それ以外の場合]** 分岐は 1 つしか存在できません。

4. **[OK]** をクリックします。
5. アクティビティをトリガする分岐を設定するには、アクティビティを右クリックして、**[入力モード]** を選択し、次のいずれかを選択します。

最初の入力 **[単純]**、**[条件付き]**、または **[それ以外の場合]** 分岐からこのアクティビティに入ってくる最初の分岐が、アクティビティの実行をトリガします。その他の分岐は無視されます。

すべての入力 このアクティビティに入ってくるすべての分岐が使用されるまで、アクティビティは実行されません。

6. アクティビティから出ていくどの分岐を使用するかを設定するには、アクティビティを右クリックして、**[出力モード]** を選択し、次のいずれかを選択します。

最初の出力 true として評価される最初の分岐が使用されます。その他の分岐はj条件が true として評価された場合でも無視されます。

すべての出力 true として評価されるすべての分岐が使用されます。

プロセス フローの削除

1. **[ファイル] > [管理]** を選択します。**[管理]** ダイアログ ボックスが表示されます。
2. 削除するプロセス フローを右クリックし、**[削除]** を選択します。
3. **[OK]** をクリックします。

アクティビティ

ジョブ

ジョブ アクティビティでは、ジョブがプロセス フローの一部として実行されます。次の例は、Standardize Names と Validate Addresses の 2 つのジョブ アクティビティを実行するプロセス フローを示します。

ジョブ アクティビティをプロセス フローに追加するには、ジョブ アクティビティをパレット内の [アクティビティ] フォルダからキャンバスにドラッグします。

> **注 :** ジョブをプロセス フローで使用できるようにするには、ジョブをエクスポーズする必要があります。希望するジョブがエクスポーズされていない場合は、そのジョブを Enterprise Designer で開いて **[ファイル] > [エクスポーズ]** を選択します。

ジョブ アクティビティをダブルクリックして、**[オプション]** タブと **[変数]** タブを設定します。

[オプション] タブ

[オプション] タブには、そのジョブに使用可能な実行時オプションが表示されます。実行時オプションを変更して、このプロセス フローでジョブが実行されるときに、ここで指定したオプションをジョブに使用することができます。例えば、ジョブのデータフロー オプションの 1 つが距離に使用する単位を制御し、デフォルトがマイルに設定されている場合、ここでそのオプションをオーバーライドし、このプロセス フローでジョブが実行されるときに、代わりにキロメートル単位で距離を返すことができます。

[変数] タブ

[変数] タブには、データフローのソース ステージとシンク ステージが表示されます。データフロー内で指定されている入出力ファイルをオーバーライドし、このプロセス フローでジョブが実行されるときに、別の入出力ファイルを使用することができます。

入出力ファイルのオーバーライド

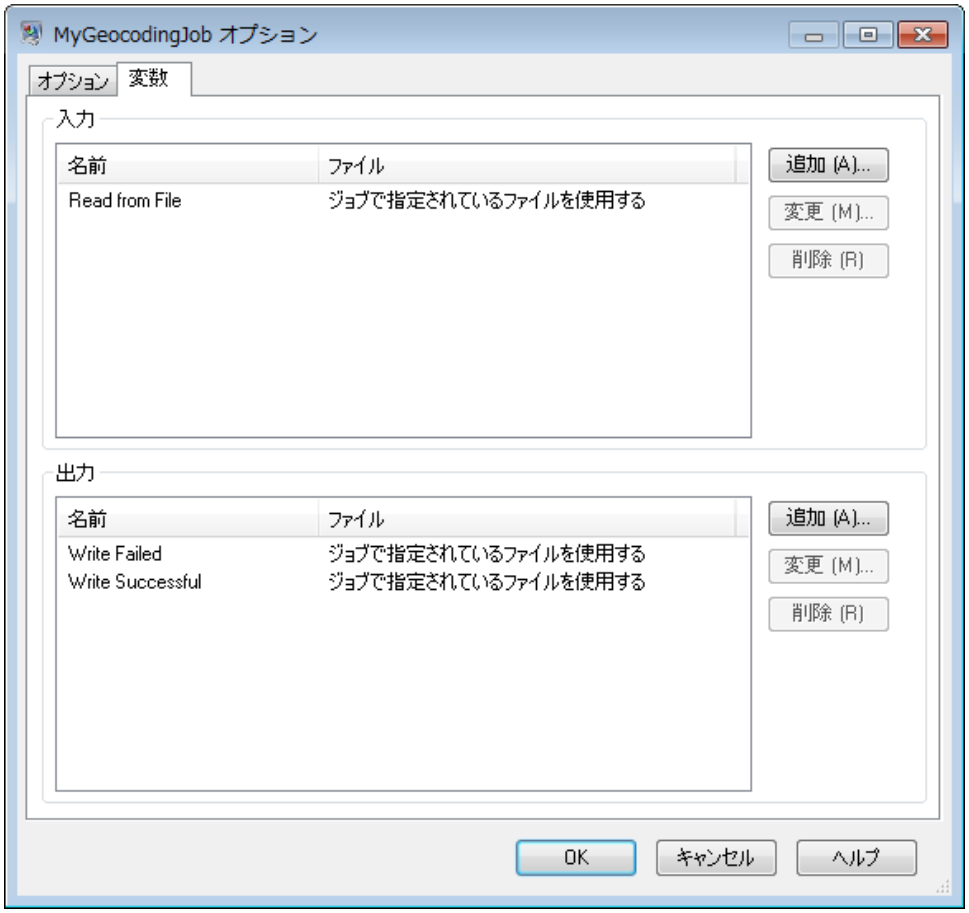
デフォルトでは、プロセス フローで実行されるジョブには、ジョブのソース ステージとシンク ステージで定義されている入出力ファイルを使用します。ただし、ジョブで定義されている入出力ファイルをオーバーライドして、プロセス フローの実行時に、ジョブのソース ステージまたはシンク ステージで指定されているファイルの代わりに、プロセス フロー内のジョブ アクティビティで指定する入出力ファイルをジョブで使用することができます。入出力ファイルは、特定の

ファイルの指定、または上流のアクティビティで定義されているファイルを参照する変数の使用によって、オーバーライドできます。

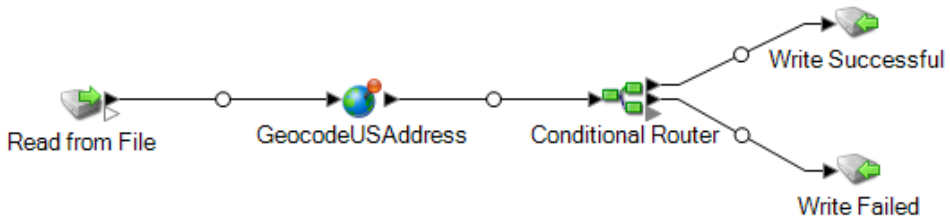
1. Enterprise Designer でプロセス フローを開きます。
2. 入出力ファイルをオーバーライドするジョブ アクティビティをダブルクリックします。
3. **[変数]** タブをクリックします。

   [変数] タブで、**[入力]** の下に一覧表示される変数は、ジョブのソース ステージに対応しています。**[出力]** の下に一覧表示される変数は、ジョブのシンク ステージに対応しています。

   例えば、MyGeocodingJob というジョブに対するジョブ アクティビティを含むプロセス フローがあるとします。アクティビティ オプションの **[変数]** タブは、次のようになっています。



   一覧表示される各変数は、MyGeocodingJob データフローのソース ステージまたはシンク ステージの名前に対応しています。この例では、**[変数]** タブに 1 つのソース (Read from File) と 2 つのシンク (Write Failed と Write Successful) が示されます。Enterprise Designer で MyGeocodingJob データフローを開いた場合は、次のようになります。



4. **[変数]** タブで、オーバーライドするソース ステージまたは出力ステージを選択し、**[変更]** をクリックします。
5. 以下のいずれかの方法を実行します。

オプション	説明
入力ファイルをオーバーライドする場合	**[場所]** フィールドで、次のいずれかを選択します。
	ジョブで指定されているファイルを使用する ジョブのソース ステージで定義されているファイルを使用するには、このオプションを選択します。
	サーバー上のファイルを参照する ジョブで定義されているファイルをオーバーライドし、選択した別のファイルを使用する場合に選択します。
	上流のアクティビティのファイルを参照する 上流のアクティビティの Read from File ステージまたは Write to File ステージ、または上流のアクティビティの変数で名前と場所が定義されているファイルでジョブに定義されているファイルをオーバーライドする場合に選択します。前のアクティビティからの出力ファイルがこのアクティビティの入力になる場合は、このオプションを使用してください。このオプションの利点は、上流のアクティビティの Write to File ステージが別のファイルを使用するように変更した場合でも、このアクティビティが正しいファイルを示すことです。もう 1 つの利点は、上流のアクティビティの入力ファイルのファイル パスと名前を知らなくても、ファイルを参照できることです。
出力ファイルをオーバーライドする場合	**[場所]** フィールドで、次のいずれかを選択します。
	サーバー上のファイルを参照する ジョブで定義されているファイルをオーバーライドし、選択した別のファイルを使用する場合に選択します。

オプション	説明
サーバーが管理する一時ファイル	必要に応じて自動的に作成および削除される一時ファイルを参照する場合に選択します。このオプションは、ファイルがプロセス フローの中間ステップとしてのみ使用され、プロセス フローが完了した後では必要ない場合に便利です。

6. **[OK]** をクリックして **[変数の変更]** ウィンドウを閉じます。
7. **[OK]** をクリックしてアクティビティ オプション ウィンドウを閉じます。

プロセス フローを実行すると、ジョブ自体で指定したファイルの代わりに、プロセス フロー アクティビティで指定したファイルが使用されます。

Clear Cache

[キャッシュのクリア] アクティビティでは、プロセス フローの一部として、グローバル キャッシュ データがクリアされます。キャッシュは削除されず、キャッシュ データのみがクリアされます。

1. **[キャッシュのクリア]** アクティビティをキャンバスにドラッグします。
2. [キャッシュのクリア] をダブルクリックします。
3. キャッシュを選択します。複数のキャッシュを選択してデータをクリアすることもできます。
4. プロセス フローを実行します。

Execute SQL

Execute SQL を使うと、データフローまたは外部プログラムの実行前後で SQL 文を実行できます。

Hive に読み込み

Apache Hive は、データの要約、クエリ、分析用に Hadoop 上に構築されたデータ ウェアハウス インフラストラクチャです。Hive を使用して基盤のデータ ソースをクエリするには、それ独自のクエリ言語である HiveQL を使用します。

Hive は、以下の Hadoop ファイル形式をサポートします。

- TEXTFILE
- SEQUENCE FILE
- ORC
- RCFILE
- PARQUET

- AVRO

  注： AVRO ファイル形式は、Hive バージョン 0.14 以降でサポートされています。

[Hive に読み込み] アクティビティでは、JDBC 接続を使用してデータを Hive テーブルに読み込むことができます。この接続を使用して、指定された Hadoop ファイルからデータが読み取られ、選択された接続の既存のテーブルか、または選択された接続の新しく作成されたテーブルに読み込まれます。

データを新しいテーブルに読み込むには、テーブルのスキーマを定義する必要があります。

注： Hive は階層データをサポートしますが、Spectrum はサポートしません。

Hive 接続の作成

1. **[Hive に読み込み]** アクティビティを開きます。
2. **[ファイル名]** フィールドに、読み取るファイルの名前を入力します。参照 [...] をクリックして、読み取るファイルを選択します。
3. **[ファイル タイプ]** フィールドで、読み取るファイルの形式を選択します。デフォルトでは、［区切り記号付き］のファイル形式が選択されています。

   選択された **[ファイル タイプ]** が［区切り記号付き］または［シーケンス］の場合は、**[フィールド区切り文字]** と **[レコード区切り文字]** のフィールドが表示されます。それ以外の場合は、これらのフィールドは表示されません。
4. **[フィールド区切り文字]** フィールドで、レコードの連続する各フィールドを区切る文字を選択します。
5. **[接続]** フィールドで、使用する Hive データベースへの接続を選択します。
   a) 接続を追加、変更、削除するには、**[管理]** をクリックします。
   [Database Connection Manager] ウィンドウが開きます。
   b) **[追加]** をクリックして新しい接続を作成するか、**[変更]** をクリックして既存の接続を編集します。
   [接続プロパティ] ウィンドウが開きます。
   c) **[接続名]** を入力します。
   d) **[データベース ドライバ]** フィールドで、接続用の Hive データベース ドライバを選択します。
   e) **[ユーザ]**、**[パスワード]**、**[ホスト]**、**[ポート]**、**[インスタンス]** といった、接続のすべての詳細情報を指定します。
   f) 接続の詳細情報をテストするには、**[テスト]** をクリックします。
   g) 接続テストが正常に終了したら、**[OK]** をクリックします。
   [接続プロパティ] ウィンドウが閉じます。
   h) **[OK]** をクリックします。

[Database Connection Manager] ウィンドウが閉じます。

6. **[テーブル/ビュー]** フィールドで、書き込み先のテーブルを選択するか、作成する新規テーブルの名前を入力します。

   [テーブル/ビュー] フィールドで新しいテーブルを作成すると、**[外部]** チェックボックスが有効になります。既存のテーブルを選択する場合は、**[外部]** チェックボックスは無効のままです。

7. Hive データベースの外部に新しいテーブルを作成するには、**[外部]** チェックボックスをオンにします。

   重要： 外部テーブルの場合:

   1. 既存のレコードは上書きできません。また、新しいレコードは追加できません。新しい外部テーブルを作成し、それにレコードを設定することだけが可能です。
   2. 特定のフォルダにある 1 つのファイルを選択すると、そのフォルダの中のすべてのファイルが自動的に選択されます。したがって、特定のフォルダに置くファイルはすべて、同じ形式である必要があります。

   Hive の EXTERNAL テーブルの詳細については、こちらを参照してください。

8. テーブルの既存のレコードをすべて上書きする場合は、**[上書き]** チェックボックスをオンにします。これによって、選択されたテーブルの既存のレコードが削除され、ファイルから読み取られたレコードがテーブルに追加されます。
9. グリッドには、選択されたテーブルの列の名前とデータタイプが表示されます。

   [テーブル/ビュー] フィールドで新しいテーブルを指定した場合は、グリッドの横にある **[追加]**、**[変更]**、**[削除]** の各ボタンを使用して、テーブルを定義するための列を追加し、その各データタイプを指定します。テーブル列の順序を指定するには、**[上へ移動]** ボタンと **[下へ移動]** ボタンを使用します。

   > **注：** **[追加]**、**[変更]**、**[削除]**、**[上へ移動]**、**[下へ移動]** の各ボタンは、**[テーブル/ビュー]** フィールドで既存のテーブルを選択した場合は無効のままとなります。

   重要：

   1. すべてのデータタイプが String である場合以外は、ファイル内のすべてのフィールドのデータタイプが、各テーブル列のデータタイプと一致することを確認してください。一致しない場合は、データの読み込みによってデータの矛盾が生じる恐れがあります。
   2. ファイル内のフィールド数が、テーブル列の数と一致することを確認してください。一致しない場合、ファイル内の余分なフィールドは破棄されます。
   3. Hive は、名前がすべて小文字のテーブルおよび列のみを受け付けます。大文字を使った名前を入力した場合は、Hive によって小文字に変換されます。その結果得られるスキーマには、すべての名前が小文字で表示されます。

10. **[OK]** をクリックします。

注： 新しいテーブルを作成してその列を定義することを選択した場合は、実行時にそれが作成されます。**[Hive に読み込み]** アクティビティで行われるのは、テーブル構造の設計だけです。実行時に、設計されたテーブルが作成され、ファイルから読み取られたデータがそれに書き込まれます。

Hadoop MapReduce ジョブの実行

[Hadoop MapReduce ジョブの実行] アクティビティでは、関連する JAR ファイルをマッピングすることにより、Hadoop クラスタ上で任意の MapReduce ジョブを実行できます。このアクティビティは、Spectrum™ Big Data Quality SDK の MapReduce ジョブまたは任意の外部 MapReduce ジョブを実行するために使うことができます。

注： MapReduce ジョブの実行がエラーになった場合、ジョブ実行のステータスがエラーメッセージの一部として表示されます。

フィールド	説明
Hadoop サーバー	設定された Hadoop サーバーのリスト。 Management Console から HDFS ファイル サーバーをマッピングする方法については、『*管理ガイド*』を参照してください。
Jar パス	実行する Hadoop MapReduce ジョブに関連する JAR ファイルのパス。 **注：** JAR は、外部クライアント側または Spectrum サーバー上に存在する必要があります。Hadoop クラスタに配置することはできません。
ドライバ クラス	次のいずれかを選択します。 **既定値** ジョブのクラス名と*引数*を入力するだけで外部ジョブを実行するには、［デフォルト］を選択します。 ［デフォルト］を選択すると、**[クラス名]** フィールドと **[引数]** フィールドが表示されます。 **設定** いずれかの外部ジョブの追加のジョブ プロパティを入力するか、Spectrum Big Data Quality SDK ジョブのいずれか 1 つを実行するには、［設定］を選択します。 ［設定］を選択すると、**[ジョブの種類]** フィールドが表示されます。

フィールド	説明
ジョブの種類	次のいずれかを選択します。 **Spectrum** Spectrum Big Data Quality SDK ジョブのいずれか 1 つを実行するには、[Spectrum] を選択します。 [Spectrum] を選択すると、**[Spectrum ジョブ]** フィールドが表示されます。 **汎用** いずれかの外部ジョブの追加のジョブ プロパティを指定するには、[汎用] を選択します。
Spectrum ジョブ	Spectrum Big Data Quality SDK ジョブの一覧から Spectrum ジョブを 1 つ選択します。 必要な Spectrum ジョブを選択すると、次のようになります。 1. **[ジョブ名]**、**[クラス名]**、**[引数]** の各フィールドは自動的に設定されます。 **[クラス名]** フィールドを除き、自動設定されたフィールドはすべて、必要に応じて編集できます。 **重要：** 選択した Spectrum ジョブに対し、自動設定された **[クラス名]** を編集してはいけません。変更すると、ジョブは実行できません。 2. **[プロパティ]** グリッドには、選択した Spectrum ジョブの必須設定プロパティがデフォルト値とともに自動的に設定されます。 必要に応じて、他のプロパティを追加またはインポートしたり、自動設定されたプロパティを変更したりできます。
クラス名	ジョブのドライバ クラスの完全修飾名。

フィールド	説明

引数	スペースで区切られた引数のリスト。これらは実行時にドライバ クラスに渡されて、ジョブの実行に使用されます。
	
	例を次に示します。
	
	`23Dec2016 /home/Hadoop/EYInc.txt`
	
	1. これらの変数を引数リストの引数として渡すことができます。ソース ステージまたはプロセス フローの現在のステージで実行時の値を受け取るように、引数を定義します。
	
	例えば、プロセス フローの前ステージの出力で変数 `SalesStartRange`が定義されていれば、次のように指定して、この変数を `${SalesStartRange}` として他の必要な変数と共に含めることができます。
	
	`23Dec2016 /home/Hadoop/EYInc.txt ${SalesStartRange}`
	
	2. 引数名にスペースが含まれる場合は、変数全体を二重引用符で囲みます。
	
	例えば、 `"/home/Hadoop/Sales Records"`です。
	
	Spectrum Big Data Quality SDK ジョブ - 引数:
	
	Spectrum Big Data Quality SDK *MapReduce* ジョブを実行するには、各種設定ファイルを引数リストとして渡します。各引数キーに、1 つの設定プロパティ ファイルのパスが指定できます。各ファイルには、複数の設定プロパティが含まれます。
	
	設定プロパティの引数リストの構文は以下のとおりです。
	
	`[-config <Path to configuration file>] [-debug] [-input <Path to input configuration file>] [-conf <Path to MapReduce configuration file>] [-output <Path of output directory>]`
	
	例えば、MapReduce MatchKeyGenerator ジョブの場合は次のようになります。
	
	`-config /home/hadoop/matchkey/mkgConfig.xml -input /home/hadoop/matchkey/inputFileConfig.xml -conf /home/hadoop/matchkey/mapReduceConfig.xml -output /home/hadoop/matchkey/outputFileConfig.xml`
	
	注 : 同じ設定プロパティ キーが **[引数]** フィールドと **[プロパティ]** グリッドの両方で指定されており、両者が異なる設定ファイルを参照している場合は、**[プロパティ]** グリッドで指定されているファイルがこのプロパティに対して適用されます。
	
	設定プロパティのサンプルは、Big Data Quality SDK に付属しており、`<Big Data Quality bundle>\samples\configuration`にあります。

[全般] タブ

フィールド	説明	要件:
ジョブ名	Hadoop MapReduce ジョブの名前。	必須
入力パス	ジョブの入力ファイルのパス。	必須
出力パス	ジョブの出力ファイルのパス。	必須
出力の上書き	指定した出力パスが既に存在する場合にそれを上書きするかどうかを指定します。 **注:** このチェック ボックスをオフのままにすると、設定された出力パスが既に存在することが実行時に判明した場合に、Hadoop から例外がスローされ、プロセス フローが中止されます。	オプション
Mapper クラス	ジョブの Mapper 機能を処理するクラスの完全修飾名。	必須
Reducer クラス	ジョブの Reducer 機能を処理するクラスの完全修飾名。	オプション
Combiner クラス	ジョブの Combiner 機能を処理するクラスの完全修飾名。	オプション
Partitioner クラス	ジョブの Partitioner 機能を処理するクラスの完全修飾名。	オプション
リデューサー数	MapReduce ジョブを実行するために使用するリデューサー数。	オプション
入力フォーマット	入力データのフォーマット。	必須
出力フォーマット	出力データのフォーマット。	必須
出力キー クラス	出力キー/値ペアのキーのデータ タイプ。	必須
出力値クラス	出力キー/値ペアの値のデータ タイプ。	必須

[プロパティ] タブ

必要なジョブを実行するために追加のプロパティを指定する場合は、このタブでプロパティと値のペアを必要なだけ定義します。必要なプロパティを 1 つずつ直接グリッドに追加できます。

または、**[インポート]**をクリックしてファイルからプロパティをインポートします。各プロパティ ファイルの場所に移動し、XML 形式のファイルを選択します。インポートされたファイルに含まれるプロパティがグリッドにコピーされます。プロパティ ファイルは XML 形式で、次の構文に従う必要があります。

```
<configuration>
    <property>
        <name>key</name>
        <value>some_value</value>
        <description>A brief description of the
            purpose of the property key.</description>
    </property>
</configuration>
```

Hadoop プロパティ ファイル *mapred.xml* を直接インポートするか、上記の XML 形式で独自のファイルを作成することができます。

注：

1. 同じプロパティがこの場所と Management Console で定義されている場合、この場所で定義された値が Management Console で定義された値に優先します。
2. 同じプロパティがグリッド内とインポートされたプロパティ ファイル内にある場合は、ファイルからインポートされた値がグリッド内の同じプロパティの既存の値を上書きします。
3. 必要であれば、複数のプロパティ ファイルを 1 つずつインポートすることができます。インポートされた各ファイルからプロパティがグリッドに追加されます。
4. プロパティ ファイルが Spectrum™ Technology Platformサーバー上に存在することを確認してください。
5. `<description>`タグは、設定プロパティ ファイル内の各プロパティ キーに対して省略可能です。

[依存関係] タブ

このタブでは、ジョブの実行に必要な一連の入力ファイルと Jar ファイルを追加します。

ジョブの実行が終わると、ここで追加した参照ファイルと参照 Jar ファイルが、ジョブの分散キャッシュから使用できるようになります。

参照ファイル

ジョブの実行に必要な各種のファイルを追加するには、**[追加]**をクリックし、ローカル システムまたはクラスタ上のそれぞれの場所に移動し、特定のファイルを選択します。

リストに追加したファイルを削除するには、そのファイルを選択し、**[削除]** をクリックします。

参照 Jar

ジョブの実行に必要な Jar ファイルを追加するには、**[追加]** をクリックし、ローカル システムまたはクラスタ上のそれぞれの場所に移動し、特定の Jar ファイルを選択します。

リストに追加したファイルを削除するには、そのファイルを選択し、**[削除]** をクリックします。

変数

入力
このグリッドでは、ソース アクティビティから受け取ったフィールドのなかから、このアクティビティで使用するフィールドを選択します。

出力
このグリッドでは、このプロセス フロー内のデスティネーション アクティビティへの出力に含めるフィールドを選択します。

Run Hadoop Pig

Run Hadoop Pig は、Apache Pig スクリプトを実行します。Apache Pig は、データ分析プログラムを表現するための高レベル言語で、これらのプログラムを評価するためのインフラストラクチャを持ちます。Pig プログラムは並列化が可能で、それによって非常に大規模なデータ セットを処理できます。

Run Hadoop Pig により、Pig 操作を選択し、必要なパラメータを入力して、Pig スクリプトをシステムに自動生成させることができます。Pig スクリプトは、任意の Hadoop サーバー上で実行できます。

Run Hadoop Pig は、Hadoop ファイル サーバー上でのみ動作します。Apache Hadoop 1.x と 2.x の両方がサポートされています。

Run Hadoop Pig オプションを設定するには

1. **[Run Hadoop Pig]** アクティビティを、キャンバスにドラッグ アンド ドロップします。
2. **[Run Hadoop Pig]** アクティビティを右クリックして、**[オプション]**を選択します。
3. サーバー名フィールドには、処理するファイルが存在する Hadoop サーバーが表示されます。
4. 参照ボタン ([...]) をクリックして、処理するファイルを参照します。
5. ファイル タイプを選択します。Run Hadoop Pig は、区切り記号付きファイルと区切り記号付きシーケンシャル ファイルの両方をサポートします。
6. 必要に応じて区切り記号とテキスト修飾子を選択します。
7. [フィールド] セクションで **[追加]** をクリックして、処理するファイルに存在するフィールドを追加します。シーケンシャル ファイルの場合は、最初のフィールドがキーとみなされ、その他のフィールドは区切られた値の一部であるとみなされます。

8. 必要に応じて、**[トリム]** 操作を選択します。トリム操作により、処理の前に入力フィールドの空白がトリムされます。
9. [操作] タブに移動します。**[追加]** をクリックして、ファイルに対して実行する Pig 操作の追加を開始します。これにより、操作エディタが開きます。
10. 実行する操作を選択します。以下のようなさまざまな操作があります。
    - Sort - データをアルファベット順にソートします。
    - Filter - データを要件に応じてフィルタできます。
    - Aggregate - データに対して Sum (合計) や Count (総数) などの統計操作を実行できます。
    - Distinct - 指定されたフィールドから一意のレコードをすべて選択します。
    - Limit - 処理するレコード数の上限を指定できます。
11. 操作の順序を変更するには、**[上へ移動]** ボタンと **[下へ移動]** ボタンを使用します。
12. 操作を選択して、操作の処理に必要な入力を設定したら、**[追加]** をクリックして選択を保存し、Pig オプション エディタに戻ります。
13. Pig スクリプトが、選択された操作に基づいて自動的に生成されます。

    エディタにおいて、生成された Pig スクリプトを必要に応じて独自のスクリプトで上書きできます。**[スクリプトを編集]** オプションをクリックして、独自のスクリプトを [Pig スクリプト] テキスト ボックスに入力します。この場合は **[再生成]** ボタンが有効になります。システムに再度スクリプトを生成させたい場合は、[Pig スクリプト] セクションの **[再生成]** をクリックして、Pig スクリプトを生成します。
14. 出力ファイルは、**[変数]** タブで指定できます。出力ファイルは、それ以降のアクティビティで使用できます。
15. **[OK]** をクリックして Pig スクリプトを保存します。デフォルトでは、出力ファイル形式は入力ファイル形式と同じです。生成された Pig スクリプトを使用して、これを変更できます。

Hadoop Pig 操作

以下のようなさまざまな Pig 操作があります。

1. Sort: データをアルファベット順にソートします。ソート操作の詳細については、入力レコードのソートを参照してください。
2. Filter: データを要件に応じてフィルタできます。フィルタ操作の詳細については、入力レコードをフィルタリングを参照してください。
3. Aggregate: データに対して Sum (合計) や Count (総数) などの統計操作を実行できます。

   必要に応じて各フィールドに対する集計操作を選択します。
   - Sum: フィールド内の値の合計を計算します。
   - Average: フィールド内のすべての値の平均値を計算します。
   - Max: フィールド内の値の最大値を求めます。

- Min: フィールド内の値の最小値を求めます。
- Count: フィールド内の値の総数を計算します。

  注： Distinct 操作を選択する場合は、一意の値のみがカウントされます。

4. Distinct: このオプションを選択すると、Aggregate Count 操作においてフィールドの一意の値のみがカウントされます。
5. Limit: 0 より大きい値を入力して、処理するレコード数の上限値を指定します。

プログラムの実行

プログラムの実行アクティビティでは、外部アプリケーションがプロセス フローの一部として実行されます。

表 9 : プログラムの実行のオプション

オプション名	説明
プログラム名	実行する実行可能ファイルのパスです。
引数	**[プログラム名]** フィールドで指定したプログラムに渡すコマンド ライン引数を指定します。複数の引数はスペースで区切ります。**[変数の挿入]** をクリックすることで、[変数] タブで定義されている変数を引数として使用できます。変数の詳細については、**変数を使用したファイルの参照**（188ページ）を参照してください。
タイムアウト (秒)	**[プログラム名]** フィールドで指定したプログラムが応答するのを待機する時間を指定します。指定した時間内にプログラムが応答しない場合、プロセス フローは失敗します。

オプション名	説明
環境変数	このプログラムを実行するときに使用する環境変数の値を指定します。この値を指定した場合、プログラムはシステムに固有の値の代わりにこれらの環境変数を使用します。指定しないと、システムで指定されている環境変数を使用します。呼び出すプログラムで複数の環境変数が使用されている場合は、すべての環境変数の値を定義するか、またはまったく定義しないようにする必要があることに注意してください。ここで値を指定しても、システム上の環境変数定義は変更されません。 **[追加]** をクリックし、**[変数名]** フィールドに変数の名前を入力します。例えば、"JAVA_HOME" と入力します。変数の値を **[変数値]** フィールドに入力します。例えば、"C:\Program Files\Java\jdk1.6.0_17" と入力します。値を入力する代わりに、**[変数の挿入]** をクリックして、**[変数]** タブで定義されている変数の値に設定することもできます。

入出力ファイルの指定

プログラムの実行アクティビティを使用して、プロセス フローから外部アプリケーションを呼び出すことができます。外部アプリケーションに送りたいデータを含むファイルや、外部アプリケーションによって書き込みを行いたい出力ファイルを指定できます。

1. プロセス フローで、プログラムの実行アクティビティをダブルクリックします。
2. **[変数]** タブをクリックします。
3. 入力ファイルを指定するには、
   a) **[入力]** セクションの下の **[追加]** ボタンをクリックします。
   b) **[名前]** フィールドに、このファイルのわかりやすい名前を入力します。任意の名前にすることができます。
   c) **[場所]** フィールドで、次のいずれかを選択します。

サーバー上のファイルを参照する	必要な入力ファイルを参照して選択する場合に選択します。
上流のアクティビティのファイルを参照する	上流のステージから既存の変数に割り当てられているファイルを使用する場合に選択します。

   d) **[OK]** をクリックします。
4. 出力ファイルを指定するには、
   a) **[出力]** セクションの下の **[追加]** ボタンをクリックします。

b) **[名前]** フィールドに、このファイルのわかりやすい名前を入力します。任意の名前にすることができます。

c) **[場所]** フィールドで、次のいずれかを選択します。

サーバー上のファイルを参照する	必要な出力ファイルを参照して選択する場合に選択します。
サーバーが管理する一時ファイル	必要に応じて自動的に作成および削除される一時ファイルにプログラムからの出力を書き込む場合に選択します。このオプションは、ファイルがプロセス フローの中間ステップとしてのみ使用され、プロセス フローが完了した後では必要ない場合に便利です。

d) **[OK]** をクリックします。

5. **[OK]** をクリックして **[プログラムの実行オプション]** ウィンドウを閉じます。

外部プログラムでのコントロール ファイルの使用

プログラムの実行アクティビティを使用して、プロセス フローから外部アプリケーションを呼び出すことができます。その際、外部アプリケーションの設定を含むコントロール ファイルを使用することができます。例えば、プログラムの実行アクティビティで VeriMove™ を呼び出して、実行時に使用するコントロール ファイルを指定できます。

1. プロセス フローで、プログラムの実行アクティビティをダブルクリックします。
2. **[変数]** タブをクリックします。
3. **[コントロール ファイル]** セクションで、**[追加]** をクリックします。
4. **[名前]** フィールドに、コントロール ファイルの名前を入力します。任意の名前にすることができます。
5. **[内容]** フィールドに、コントロール ファイルの内容を指定します。

   [入力] または **[出力]** で定義されている変数を使用してファイルを参照するには、**[変数の挿入]** をクリックします。

6. **[OK]** をクリックして **[コントロール ファイルの追加]** ウィンドウを閉じます。
7. **[オプション]** タブをクリックします。
8. **[変数の挿入]** をクリックします。
9. 作成したコントロール ファイルの名前を選択し、**[OK]** をクリックします。

   コントロール ファイルを示す変数が **[引数]** フィールドに追加されます。

10. コントロール ファイルを示すために必要なコマンド ライン引数を使用するには、必要に応じて **[引数]** フィールドを変更します。詳細については、外部アプリケーションのドキュメントを参照してください。

11. **[OK]** をクリックして **[プログラムの実行オプション]** ウィンドウを閉じます。

Spark Sorter

Spark Sorter アクティビティを使用すると、大部分のレコードをソートすることができます。このアクティビティは Apache Spark ライブラリを使用して機能強化されており、Spectrum™ Technology Platform サーバーで実行されます。

現時点では、Spectrum™ Technology Platform サーバー上に存在する*区切り記号付き*ファイルをレコードの読み込みに利用できます。

> **注：** リモート サーバー上に存在するファイルはサポートされていません。

フィールド	説明
サーバ名	入力用に選択するファイルの場所を示します。 **Spark Sorter** アクティビティで利用できるのは Spectrum™ Technology Platform 上に存在するファイルのみなので、このフィールドには Spectrum™ Technology Platform が表示されます。
ファイル名	ファイルへのパスを指定します。省略記号ボタン (...) をクリックし、目的のファイルを選択します。 ワイルド カード文字を使用して、ディレクトリ内の複数のファイルからデータを読み込むことができます。サポートされているワイルド カード文字は、*と?です。例えば、`*.csv` と指定して、ディレクトリ内にある、拡張子が `.csv` のファイルをすべて読み込むことができます。複数のファイルを正常に読み込むには、各ファイルが同じレイアウト (同じ位置に同じフィールド) を持つ必要があります。**[フィールド]** タブで指定したレイアウトに一致しないレコードは、形式に誤りのあるレコードとして扱われます。 **重要：** なお、Spectrum™ Technology Platform サーバーを実行しているプラットフォームが Unix または Linux の場合、これらのプラットフォームでファイル名およびパスの大文字と小文字が区別されることに注意してください。
レコード タイプ	ファイル内のレコードのフォーマット。現時点で入力用に使用できるのは、*区切り記号付き*ファイル フォーマットです。 **区切り記号付き** ファイル内の各レコードが復帰または改行 (CR または LF) などの行末 (EOL) 文字で区切られ、レコード内の各フィールドがカンマ (,) などの特定の文字で区切られているテキスト ファイル。

フィールド	説明
文字エンコード	入力ファイルの文字エンコーディング。 エンコーディング UTF-8 がサポートされています。UTF の詳細については、**unicode.org/faq/utf_bom.html** を参照してください。
フィールド区切り文字	区切り記号付きファイル内のフィールドを区切るのに使用する文字を指定します。 例えば、次のレコードでは (|) 記号がフィールド区切り文字として使われています。 `7200 13TH ST|MIAMI|FL|33144` フィールド区切り文字として定義できるのは次の文字です。 • スペース • タブ • カンマ • ピリオド (.) • セミコロン • パイプ (|) これ以外の文字がフィールド区切り文字として使われているファイルについては、省略記号ボタン (...) をクリックし、別の文字を区切り文字として選択してください。
Text qualifier	区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 例えば、次のレコードでは二重引用符 (") がテキスト修飾子として使われています。 `"7200 13TH ST"|"MIAMI"|"FL"|"33144"` テキスト修飾子として定義できるのは次の文字です。 • 一重引用符 (') • 二重引用符 (") これ以外の文字がテキスト修飾子として使われているファイルについては、省略記号ボタン (...) をクリックし、別の文字をテキスト修飾子として選択してください。

フィールド	説明
レコード区切り文字	順次ファイルまたは区切り記号付きファイル内のレコードを区切るのに使用する文字を指定します。**[デフォルトの EOL を使用]** チェック ボックスをオンにすると、このフィールドは使用できません。 使用できるレコード区切り文字の設定は次のとおりです。 **Unix (U+000A)** 改行 (LF) 文字でレコードを区切ります。これは Unix システムの標準のレコード区切り文字です。 **Macintosh (U+000D)** 復帰 (CR) 文字でレコードを区切ります。これは Macintosh システムの標準のレコード区切り文字です。 **Windows (U+000D U+000A)** 復帰改行 (CR+LF) でレコードを区切ります。これは Windows システムの標準のレコード区切り文字です。 これ以外の文字がレコード区切り文字として使われているファイルについては、省略記号ボタン (...) をクリックし、別の文字をレコード区切り文字として選択してください。
デフォルトの EOL を使用	Spectrum™ Technology Platform サーバーが実行されているオペレーティング システムのデフォルトの行末 (EOL) 文字をファイルのレコード区切り文字として使用します。 ファイルの EOL 文字がサーバーのオペレーティング システムで使われているデフォルトの EOL 文字と異なる場合は、このオプションをオンにしないでください。例えば、ファイルで Windows の EOL が使われていて、サーバーの動作プラットフォームが Linux の場合は、このオプションをオンにしないでください。代わりに、**[レコード区切り文字]** フィールドで [Windows] オプションを選択します。

最初の行はヘッダ レコード	区切り記号付きファイルの先頭レコードの内容がデータではなくヘッダ情報であるかどうかを指定します。
	
	次のファイル スニペットは、先頭レコードのヘッダー行の例です。
	
	`"AddressLine1"|"City"|"StateProvince"|"PostalCode"`
	`"7200 13TH ST"|"MIAMI"|"FL"|"33144"`
	`"One Global View"|"Troy"|"NY"|12180`

出力	Spectrum™ Technology Platform サーバー上の出力ファイルのパスを指定します。省略記号ボタン (...) をクリックし、出力ファイルのディレクトリとファイル名を選択します。 **重要：** なお、Spectrum™ Technology Platform サーバーを実行しているプラットフォームが Unix または Linux の場合、これらのプラットフォームでファイル名およびパスの大文字と小文字が区別されることに注意してください。

フィールド	説明
Overwrite	**[出力]**フィールドに指定したファイルと同じ名前のファイルが既に存在する場合、出力ファイルでそれを上書きすることを示します。
連結	すべての Spark パート ファイルを、指定した **[出力]** 場所の 1 つの出力ファイルに連結することを示します。
プレビュー	**[ファイル名]** フィールドで入力ファイルを選択すると、**[プレビュー]** グリッドに現在の出力ファイルの最初の 100 件のレコードが表示されます。 すべての列の値を正しく表示するには、**[フィールド]** タブの **[再生成]** をクリックします。

[フィールド] タブ

[フィールド] タブでは、ファイルの各フィールドの名前、タイプ、および位置を定義します。詳細については、以下を参照してください。

区切り記号付き入力ファイルのフィールドの定義（212ページ）

[ソート] タブ

[ソート] タブでは、データフローに送出される前の入力レコードのソートに使うフィールドを定義します。詳細については、「**レコードのソート**（214ページ）」を参照してください。

[構成] タブ

必要なジョブを実行するために追加のプロパティを指定する場合は、このタブでプロパティと値のペアを必要なだけ定義します。必要なプロパティを 1 つずつ直接グリッドに追加できます。

または、**[インポート]**をクリックしてファイルからプロパティをインポートします。各プロパティ ファイルの場所に移動し、XML 形式のファイルを選択します。インポートされたファイルに含まれるプロパティがグリッドにコピーされます。プロパティ ファイルは XML 形式で、次の構文に従う必要があります。

```
<configuration>
    <property>
        <name>key</name>
        <value>some_value</value>
        <description>A brief description of the
            purpose of the property key.</description>
    </property>
</configuration>
```

注：

1. 同じプロパティがこの場所と Management Console で定義されている場合、この場所で定義された値が Management Console で定義された値に優先します。
2. 同じプロパティがグリッド内とインポートされたプロパティ ファイル内にある場合は、ファイルからインポートされた値がグリッド内の同じプロパティの既存の値を上書きします。
3. 必要であれば、複数のプロパティ ファイルを1つずつインポートすることができます。インポートされた各ファイルからプロパティがグリッドに追加されます。
4. プロパティ ファイルが Spectrum™ Technology Platformサーバー上に存在することを確認してください。
5. `<description>`タグは、設定プロパティ ファイル内の各プロパティ キーに対して省略可能です。

[実行時] タブ

フィールド名	説明
ファイル名	最初のタブで選択したファイル名が表示されます。
開始レコード	レコードをデータフローに読み込むときファイルの先頭部分にあるレコードをスキップしたければ、読み込みたい最初のレコードを指定します。例えば、最初の50個のレコードをスキップする場合は51と指定します。これで51番目のレコードがデータフローに読み込まれる最初のレコードとなります。
すべてのレコード	**[開始レコード]** フィールドで指定したレコードからファイルの最後までのレコードをすべて読み込む場合は、このオプションをオンにします。
最大レコード数	**[開始レコード]** フィールドで指定したレコードを起点にそこから一定の数のレコードを読み込む場合は、このオプションをオンにします。例えば、最初の100個のレコードを読み込みたい場合は、このオプションをオンにして100と入力します。

区切り記号付き入力ファイルのフィールドの定義

[フィールド] タブでは、ファイルの各フィールドの名前、タイプ、および位置を定義します。**[ファイル プロパティ]** タブで入力ファイルを定義したら、フィールドを定義できます。

入力ファイルにヘッダ レコードが含まれていない場合、またはフィールドを手動で定義する場合は、**[フィールド]** タブで以下の手順に従います。

1. 入力ファイル内に既に存在するフィールドを定義するには、**[再生成]** をクリックします。その後、**[検出タイプ]** をクリックします。これにより、ファイルの最初の 50 個のレコードに基づいて、各フィールドのデータ タイプが自動的に設定されます。
2. 出力にフィールドを追加するには、**[追加]** をクリックします。
3. **[名前]** フィールドで、追加するフィールドを選択するか、フィールドの名前を入力します。
4. **[タイプ]** フィールドで、データに対して数学的な操作を行う予定がない場合は、データ タイプを`string`のままにしておくことができます。ただし、そのような操作を行う予定がある場合は、適切なデータ タイプを選択します。ファイルに含まれる文字列データは、データフローでの適切なデータ操作を可能にするデータ タイプに変換されます。

   このアクティビティでは、以下のデータ タイプがサポートされています。

   bigdecimal 小数点以下 38 桁の精度をサポートする数値データ タイプ。高い精度が必要な算術計算で使用されるデータ (特に金融データ) には、このデータ タイプを使用してください。bigdecimal データ タイプは、double データ タイプより正確な計算をサポートします。

   double 正と負の倍精度数を含む数値データ タイプ。値の範囲は、2^{-1074} ～ $(2-2^{-52})\times2^{1023}$。指数表記すると、値の範囲は、-1.79769313486232E+308 ～ 1.79769313486232E+308 となります。

   float 正と負の単精度数を含む数値データ タイプ。値の範囲は、2^{-149} ～ $(2-2^{23})\times2^{127}$。指数表記すると、値の範囲は、-3.402823E+38 ～ 3.402823E+38 となります。

   integer 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{31} (-2,147,483,648) ～ 2^{31}-1 (2,147,483,647)。

   long 正と負の整数を含む数値データ タイプ。値の範囲は、-2^{63} (-9,223,372,036,854,775,808) ～ 2^{63}-1 (9,223,372,036,854,775,807)。

   string 文字シーケンス。

5. **[位置]** フィールドで、レコード内のこのフィールドの位置を入力します。

   例えば、この入力ファイルで、AddressLine1 は位置 1、City は位置 2、StateProvince は位置 3、PostalCode は位置 4 です。

```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|12180
```

6. フィールドの値文字列の先頭と末尾から余分なスペース文字を削除するには、**[トリム]** チェック ボックスをオンにします。

レコードのソート

[ソート] タブでは、データフローに送出される前に入力レコードの各種フィールドをソートする条件を定義することができます。

1. **[ソート]** タブで、**[追加]** をクリックします。
2. **[フィールド名]** 列のドロップダウン矢印をクリックし、ソートに使うフィールドを選択します。選択できるフィールドは、この入力ファイルに定義されているフィールドによって異なります。
3. **[順序]** 列で、フィールドを Ascending または Descending のいずれの順序でソートするかを選択します。
4. **[タイプ]** フィールドで、データに対して数学的または日時に関する操作を行う予定がない場合は、データ タイプを`string` のままにしておくことができます。ただし、そのような操作を行う予定がある場合は、適切なデータ タイプを選択します。ファイルに含まれる `string` データは、データフローでの適切なデータ操作を可能にするデータ タイプに変換されます。
5. フィールドの値文字列の先頭と末尾から余分なスペース文字を削除するには、**[トリム]**チェック ボックスをオンにします。
6. **[NULL を次の値とみなす]** フィールドで、レコード内の特定のフィールドの `null` 値を全レコードの［最大値］または［最小値］のいずれと見なすかを選択します。
7. ソートに使用するすべての入力フィールドを追加するまでこれを繰り返します。ソート順序を変更するには、移動するフィールドの行をハイライト表示して、**[上へ]** または **[下へ]** をクリックします。

変数

入力

このグリッドでは、ソース アクティビティから受け取ったフィールドのなかから、このアクティビティで使用するフィールドを選択します。

出力

このグリッドでは、このプロセス フロー内のデスティネーション アクティビティへの出力に含めるフィールドを選択します。

Submit Spark Job

Submit Spark Job アクティビティを使用すると、Spark ジョブを Hadoop クラスタまたは Spark クラスタで実行することができます。このアクティビティを使用すると、Spectrum™ Big Data Quality SDK の Spark ジョブまたは任意の外部 Spark ジョブを実行できます。

現在、Spark ジョブを次の 2 つのクラスタ タイプのどちらかに送信できます。

- YARN

- Spark

展開モード

Spark ジョブは、クラスタまたはクライアント展開モードで実行します。展開モードによって、Spark ジョブ ドライバ クラスがクラスタで実行されるか、それともクライアントSpectrum™ Technology Platformで実行されるかが決まります。

簡単に言えば、Spark ジョブは次のいずれかの展開モードで実行できます。

1. YARN クラスタ モード
2. YARN クライアント モード
3. Spark クライアント モード

重要： Spectrum サーバーをクラスタ環境にインストールして実行している場合は、YARN または Spark クライアント モードで実行することを推奨します。

フィールド	説明
ジョブ名	Spark ジョブの名前。
Hadoop サーバー	設定された Hadoop サーバーのリスト。 Management Console から HDFS ファイル サーバーをマッピングする方法については、『*管理ガイド*』を参照してください。
Jar パス	実行する Spark ジョブに関連する JAR ファイルのパス。 **注：** Jar パスは、Spectrum サーバー コンピュータ上のディレクトリを指している必要があります。
ジョブの種類	次のいずれかを選択します。 **Spectrum** Spectrum Big Data Quality SDK ジョブのいずれか 1 つを実行するには、[Spectrum] を選択します。 [Spectrum] を選択すると、**[Spectrum ジョブ]** フィールドが表示されます。 **汎用** いずれかの外部ジョブの追加のジョブ プロパティを指定するには、[汎用] を選択します。

フィールド	説明
Spectrum ジョブ	Spectrum Big Data Quality SDK ジョブの一覧から Spectrum ジョブを 1 つ選択します。 必要な Spectrum ジョブを選択すると、次のようになります。 1. **[ジョブ名]**、**[クラス名]**、**[引数]** の各フィールドは自動的に設定されます。 **[クラス名]** フィールドを除き、自動設定されたフィールドはすべて、必要に応じて編集できます。 **重要：** 選択した Spectrum ジョブに対し、自動設定された **[クラス名]** を編集してはいけません。変更すると、ジョブは実行できません。 2. **[プロパティ]** グリッドには、選択した Spectrum ジョブの必須設定プロパティがデフォルト値とともに自動的に設定されます。 必要に応じて、他のプロパティを追加またはインポートしたり、自動設定されたプロパティを変更したりできます。
クラス名	ジョブのドライバ クラスの完全修飾名。

フィールド	説明

引数	スペースで区切られた引数のリスト。これらは実行時にドライバ クラスに渡されて、ジョブの実行に使用されます。
	
	例を次に示します。
	
	`23Dec2016 /home/Hadoop/EYInc.txt`
	
	1. これらの変数を引数として渡すことができます。ソース ステージまたはプロセス フローの現在のステージで実行時の値を受け取るように、引数を定義します。
	
	例えば、プロセス フローの前ステージの出力で変数`SalesStartRange`が定義されていれば、次のように指定して、この変数を`${SalesStartRange}`として他の必要な変数と共に含めることができます。
	
	`23Dec2016 /home/Hadoop/EYInc.txt ${SalesStartRange}`
	
	2. 引数名にスペースが含まれる場合は、変数全体を二重引用符で囲みます。例えば、`"/home/Hadoop/Sales Records"`です。
	
	Spectrum Big Data Quality SDK ジョブ - 引数:
	
	Spectrum Big Data Quality SDK *Spark* ジョブを実行するには、各種設定ファイルを引数リストとして渡します。各引数キーに、1 つの設定プロパティ ファイルのパスが指定できます。各ファイルには、複数の設定プロパティが含まれます。
	
	設定プロパティの引数リストの構文は以下のとおりです。
	
	`[-config <Path to configuration file>] [-debug] [-input <Path to input configuration file>] [-conf <Path to Spark configuration file>] [-output <Path of output directory>]`
	
	例えば、Spark MatchKeyGenerator ジョブの場合は次のようになります。
	
	`-config /home/hadoop/spark/matchkey/matchKeyGeneratorConfig.xml -input /home/hadoop/spark/matchkey/inputFileConfig.xml -output /home/hadoop/spark/matchkey/outputFileConfig.xml`
	
	注 : 同じ設定プロパティ キーが **[引数]** フィールドと **[プロパティ]** グリッドの両方で指定されており、両者が異なる設定ファイルを参照している場合は、**[プロパティ]** グリッドで指定されているファイルがこのプロパティに対して適用されます。
	
	設定プロパティのサンプルは、Big Data Quality SDK に付属しており、`<Big Data Quality bundle>\samples\configuration`にあります。
	
	。

一般プロパティ

フィールド	説明
マスター	Spark ジョブの実行に適用されるオプションを選択します。 **YARN** Spark ジョブの起動と管理に YARN を使用します。 **Spark** Spark ジョブの起動と管理に Spark アプリケーションを使用します。
Spark URL	Spark クラスタにアクセスするための URL で、フォーマットは `<hostname of Spark cluster>:<port of Spark cluster>` です。 このフィールドは、**[マスター]** フィールドで `[Spark]` を選択すると表示されます。
展開モード	次のオプションのいずれかを選択します。 **クライアント** Spark ジョブ ドライバをクライアント Spectrum™ Technology Platformで実行する場合。 **Cluster** Spark ジョブ ドライバをクラスタで実行する場合。

フィールド	説明
プロパティ	

フィールド	説明

グリッド内の **[プロパティ]** 列にプロパティの名前を入力し、**[値]** 列にそのプロパティの値を入力します。

[マスター] と **[展開モード]** のタイプによって、特定のプロパティが必須です。

YARN の必須プロパティ	
`yarn.resourcemanager.hostname`	YARN ResourceManager の IP アドレス。
`yarn.resourcemanager.address`	YARN ResourceManager の IP アドレスとポートは、<ホスト名>:<ポート> 形式で指定します。

クライアント展開モードのプロパティ		
`Spark.driver.host`	Spark ドライバを実行するコンピュータの IP アドレス。	必須
`spark.client.mode.temp.location`	Universal Addressing ジョブに使用される Spectrum サーバー上の `temp`フォルダのパス: • Validate Address • Validate Address Global • Validate Address Loqate **注：** このプロパティを Universal Addressing ジョブに使用して、指定された `temp`フォルダが中間結果用に使用されるようにすることを強	オプション

フィールド	説明
クライアント展開モードのプロパティ	く推奨します。

要約すると、

1. YARN クラスタ モード: 最初の 2 つのプロパティが必須。
2. YARN クライアント モード: 3 つのプロパティがすべて必須。
3. SPARK クライアント モード: 3 つ目のプロパティが必須。

注： 上記の必須プロパティは、Management Console で接続を作成するときに定義するか、この Spark アクティビティを使用して定義することができます。同じプロパティが Management Console と Spark Job アクティビティの両方で定義された場合、Spark アクティビティで割り当てられた値が使用されます。

これらの必須プロパティのほかに、ジョブの実行に必要なプロパティをいくつでも入力またはインポートすることができます。

フィールド	説明

インポート	プロパティをファイルからインポートする場合は、**[インポート]** をクリックします。各プロパティ ファイルの場所に移動し、XML 形式のファイルを選択します。インポートされたファイルに含まれるプロパティが、**[プロパティ]** グリッドにコピーされます。
	注：
	1. 同じプロパティがこの場所と Management Console で定義されている場合、この場所で定義された値が Management Console で定義された値に優先します。
	2. プロパティ ファイルは XML 形式で、次の構文に従う必要があります。
	`<configuration>`
	`  <property>`
	`  <name>key</name>`
	`  <value>some_value</value>`
	`  <description>A brief description of the purpose of the property key.</description>`
	`  </property>`
	`</configuration>`
	上記の XML 形式を使用して、独自のプロパティ ファイルを作成します。
	3. 同じプロパティがグリッド内とインポートされたプロパティ ファイル内にある場合は、ファイルからインポートされた値がグリッド内の同じプロパティの既存の値を上書きします。
	4. 必要であれば、複数のプロパティ ファイルを1つずつインポートすることができます。インポートされた各ファイルからプロパティがグリッドに追加されます。
	5. プロパティ ファイルが Spectrum™ Technology Platformサーバー上に存在することを確認してください。
	6. `<description>`タグは、設定プロパティ ファイル内の各プロパティ キーに対して省略可能です。

依存関係

このタブでは、ジョブの実行に必要な一連の入力ファイルと Jar ファイルを追加します。

ジョブの実行が終わると、ここで追加した参照ファイルと参照 Jar ファイルが、ジョブの分散キャッシュから使用できるようになります。

参照ファイル

ジョブの実行に必要な各種のファイルを追加するには、**[追加]** をクリックし、ローカル システムまたはクラスタ上のそれぞれの場所に移動し、特定のファイルを選択します。

リストに追加したファイルを削除するには、そのファイルを選択し、**[削除]** をクリックします。

参照 Jar

ジョブの実行に必要な Jar ファイルを追加するには、**[追加]** をクリックし、ローカル システムまたはクラスタ上のそれぞれの場所に移動し、特定の Jar ファイルを選択します。

リストに追加したファイルを削除するには、そのファイルを選択し、**[削除]** をクリックします。

注： Jar パスは、Spectrum サーバー コンピュータ上のディレクトリを指している必要があります。

変数

入力

このグリッドでは、ソース アクティビティから受け取ったフィールドのなかから、このアクティビティで使用するフィールドを選択します。

出力

このグリッドでは、このプロセス フロー内のデスティネーション アクティビティへの出力に含めるフィールドを選択します。

成功

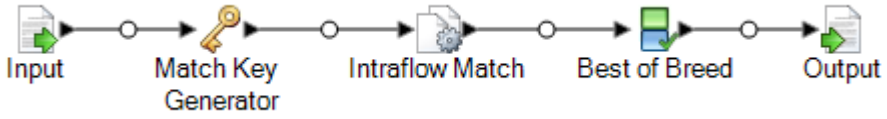
成功アクティビティはプロセス フローの終わりを示します。プロセス フローには少なくとも 1 つの成功アクティビティが必要です。

6 - 再利用可能なフローコンポーネントの作成

このセクションの構成

サブフローの概要

サブフローは、他のデータフロー内で再利用可能なデータフローです。サブフローは、データフローに簡単に組み込むことができる、再利用可能なプロセスを作成する場合に便利です。例えば、各ステージで特定の設定を使用して重複除去を実行するサブフローを作成し、複数のデータフローで同じ重複除去プロセスを使用できるようにしたいことがあります。これを実現するには、次のようなサブフローを作成します。



その後、このサブフローをデータフロー内で使用することができます。例えば、ジオコーディングを実行するデータフロー内で重複除去サブフローを使用し、ジオコーディング操作の前にデータを重複除去することができます。

この例では、データはデータベース内から読み込まれ、重複除去サブフローに渡されます。このサブフローで、データは Match Key Generator、Intraflow Match、Best of Breed の順に処理され、最後にサブフローから親データフロー (この場合は Geocode US Address) の次のステージに送信されます。上記のように、サブフローは、データフロー内ではパズルのピース アイコンとして表されます。

Enterprise Designer の **[ユーザ定義ステージ]** フォルダに表示されるサブフローは、保存してエクスポーズできます。

ソースとしてのサブフローの使用

サブフローをデータフロー内の最初のステージとして使用し、ソースからデータを読み込んだり、データに対して何らかの処理を実行してから親データフローに渡すこともできます。複数のデータフローで再利用できるように設定されている単一のソース ステージのように単純なサブフローを作成したり、データを読み込んで何らかの方法で処理してから親データフローに渡す複雑なサブフローを作成することができます。

1. Enterprise Designer で、**[ファイル] > [新規作成] > [データフロー] > [サブフロー]** を選択します。
2. 適切なデータ ソースをパレットからキャンバス上にドラッグして、設定します。

   例えば、サブフローでカンマ区切りファイルからデータを読み込みたい場合、Read from File ステージをキャンバス上にドラッグします。
3. サブフローで、何らかの方法でデータを処理してから親データフローに渡したい場合は、目的の処理を実行するために必要なその他のステージを追加します。
4. データフローの終わりに Output ステージを追加し、設定します。

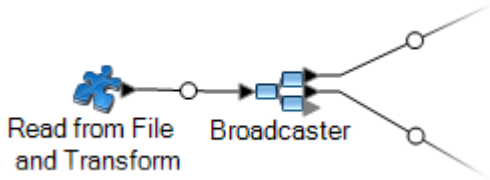
   これにより、サブフローからのデータが親データフローに送信されます。

   例えば、ファイルからデータを読み込み、Transformer ステージを使用して空白をトリムし、フィールドの大文字と小文字の区別を正規化するサブフローを作成した場合、サブフローは次のようになります。

5. Output ステージをダブルクリックし、親データフローに渡したいフィールドを選択します。
6. **[ファイル] > [保存]** を選択し、サブフローを保存します。
7. **[ファイル] > [エクスポーズ]** を選択して、サブフローをデータフロー内に含めて使用できるようにします。
8. サブフローを含めるデータフロー内で、サブフローをパレットからキャンバス上にドラッグします。
9. サブフローを目的のデータフロー ステージに接続します。

   注： サブフローには Input ステージではなくソース ステージが含まれているため、サブフロー アイコンには出力ポートしかありません。これは、データフロー内のソースとしてのみ使用できます。

   親データフローが、入力として作成したサブフローを使用するようになりました。例えば、"Read from File and Transform" というサブフローを作成し、追加して、Broadcaster ステージに接続する場合、データフローは次のようになります。



データフローの途中でのサブフローの使用

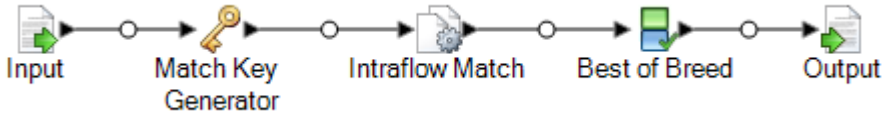
データフローの途中でサブフローを使用して、他のデータフローで再利用できる処理を実行することができます。実際には、サブフローがデータフロー内のカスタム ステージになります。

1. Enterprise Designer で、**[ファイル] > [新規作成] > [データフロー] > [サブフロー]** を選択します。
2. Input ステージをパレットからキャンバスにドラッグします。

   これにより、親データフローのデータがサブフローに送信されます。
3. Input ステージをダブルクリックし、サブフローを使用するデータフローからサブフローが受け取るフィールドを追加します。
4. Input ステージを設定したら、目的の処理を実行するために必要なその他のステージを追加します。
5. データフローの終わりに Output ステージを追加します。

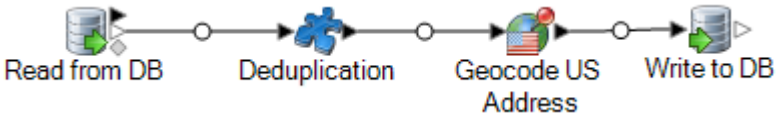
   これにより、サブフローからのデータが親データフローに送信されます。

   例えば、各ステージで特定の設定を使用して重複除去を実行するサブフローを作成し、複数のデータフローで同じ重複除去プロセスを使用できるようにしたいことがあります。これを実現するには、次のようなサブフローを作成します。



6. **[ファイル] > [保存]** を選択し、サブフローを保存します。
7. **[ファイル] > [エクスポーズ]** を選択して、サブフローをデータフロー内に含めて使用できるようにします。
8. サブフローを含めるデータフロー内で、サブフローをパレットからキャンバス上にドラッグします。
9. サブフローを目的のデータフロー ステージに接続します。

   例えば、ジオコーディングを実行するデータフロー内で重複除去サブフローを使用し、ジオコーディング操作の前にデータを重複除去することができます。

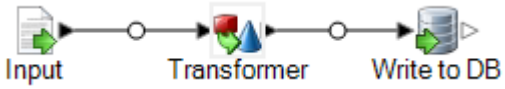


シンクとしてのサブフローの使用

サブフローをデータフロー内の最終ステージとして使用し、ファイルまたはデータベースにデータを書き込んだり、データに対して何らかの処理を実行してから出力デスティネーションに書き込むこともできます。複数のデータフローで再利用できるように設定されている単一のシンク ステージのように単純なサブフローを作成したり、何らかの方法でデータを処理してから出力デスティネーションに書き込む複雑なサブフローを作成することができます。

1. Enterprise Designer で、**[ファイル] > [新規作成] > [データフロー] > [サブフロー]** を選択します。
2. Input ステージをパレットからキャンバスにドラッグします。
3. Input ステージをダブルクリックし、サブフローを使用するデータフローからサブフローが受け取るフィールドを追加します。
4. Input ステージを設定したら、目的の後処理を実行するために必要なその他のステージを追加します。
5. サブフローの終わりに 適切なシンクを追加します。

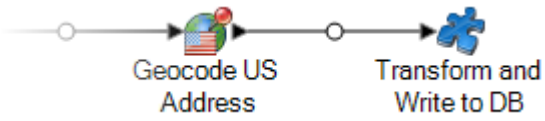
   例えば、Transformer ステージを使用して空白をトリムし、フィールドの大文字と小文字の区別を正規化してからデータベースに書き込むサブフローを作成した場合、サブフローは次のようになります。

6. **[ファイル] > [保存]** を選択し、サブフローを保存します。
7. **[ファイル] > [エクスポーズ]** を選択して、サブフローをデータフロー内に含めて使用できるようにします。
8. サブフローを含めるデータフロー内で、サブフローをパレットからキャンバス上にドラッグし、データフロー内の最終ステージに接続します。

   > **注 :** サブフローには Output ステージではなくシンク ステージが含まれているため、サブフロー アイコンには入力ポートしかありません。これは、データフロー内のシンクとしてのみ使用できます。

   親データフローが、シンクとして作成したサブフローを使用するようになりました。例えば、"Transform and Write to DB" というサブフローを作成し、追加して、Geocode US Address ステージに接続する場合、データフローは次のようになります。



サブフローの変更

1. Enterprise Designer でサブフローを開きます。
2. サブフローを変更する前に、そのサブフローを使用するデータフローに対する影響を検討します。サブフローを使用しているデータフローを確認するには、**[ツール] > [使用箇所]** を選択します。
3. 必要に応じてサブフローを変更します。

   次のことに注意してください。

   - Input または Output ステージを削除したり、新しい Input またはOutput ステージを追加したりすると、他のデータフローがサブフローを使用していることを伝える警告メッセージが表示され、サブフローを使用しているデータフローを確認できます。再利用可能なステージの保存を続行すると、サブフローによって使用されているすべてのデータフローがアンエクスポーズされます。
   - ファイル名やステージ設定の変更など、それ以外の変更をサブフローに対して行うと、他のデータフローがサブフローを使用していることを伝える警告メッセージが表示され、サブフローを使用しているデータフローを確認できます。この場合のデータフローはアンエクスポーズしないで続行できます。

4. 変更が完了したら、**[ファイル] > [保存]** を選択します。
5. **[表示] > [更新]** を選択して、変更を親データフローに反映させます。

   注： サブフローに複数のバージョンがある場合は、親データフロー内で使用されているバージョンがエクスポーズされることに注意してください。サブフローを変更した場合は、最新版を必ずエクスポーズして、サブフローを使用しているデータフロー内で変更を有効にしてください。

サブフローの削除

エクスポーズしたサブフローを削除しようとした場合は、削除しようとしているサブフローを他のデータフローが使用していることを伝える警告メッセージが表示されます。サブフローの削除を続行すると、接続されているすべてのデータフローがアンエクスポーズされます。

サブフローのエクスポーズ/アンエクスポーズ

サブフローをデータフロー内で使用できるようにするには、サブフローをエクスポーズする必要があります。サブフローをエクスポーズするには、Enterprise Designer でサブフローを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。そのサブフローを他のデータフローで使用できるようになります。

> **注 :** サブフローに複数のバージョンがある場合は、親データフロー内で使用されているバージョンがエクスポーズされることに注意してください。サブフローを変更した場合は、最新版を必ずエクスポーズして、サブフローを使用しているデータフロー内で変更を有効にしてください。

サブフローをアンエクスポーズするには、Enterprise Designer でサブフローを開き、**[ファイル] > [エクスポーズ/アンエクスポーズして保存]** を選択します。サブフローをアンエクスポーズするとき、変更しようとしているサブフローを他のデータフローが使用していることを伝える警告メッセージが表示されます。サブフローのアンエクスポーズを続行すると、そのサブフローを使用しているすべてのデータフローがアンエクスポーズされます。

サブフローへのステージの変換

1. 新しいジョブ、サービス、またはサブフローを作成します。
2. ジョブ、サービス、またはサブフローに組み込むステージを追加します。
3. この時点でステージを設定する場合は、ステージを右クリックして **[オプション]** を選択します。必要に応じてステージのオプションを設定した後、**[OK]** をクリックします。

4. 変換するステージを右クリックし、**[ステージをサブフローに変換]** を選択します。**[名前を付けて保存]** ダイアログ ボックスが表示されます。
5. サブフローの名前を入力して **[OK]** をクリックした後、サービスを保存します。システムでユニークな名前を指定する必要があります。次の 3 つのことが行われます。
   - 次のものが含まれる新しいサブフローが作成されます。
     - ユーザが選択したステージ
     - ステージの各入力ポートに対するデータフロー入力
     - ステージの各出力ポートに対するデータフロー出力
     - ステージとその入力および出力の間の接続
   - ユーザが選択したステージが新しいサブフローに置き換えられます。
   - 新しいサブフローがエクスポーズされます。結果は、サーバ エクスプローラおよびツールボックスの [ユーザ定義ステージ] セクションで確認できます。

サブフローを作成し、それを他のデータフローで使用した後は、サブフローを使用している他のデータフローを確認できます。サブフローを開き、**[ツール] > [使用箇所]** を選択します(または、サーバー エクスプローラーでサブフローを右クリックし、**[使用箇所]** を選択します)。現在のサブフローを使用しているデータフローのリストが表示され、現在のサブフローを変更した場合に影響を受けるデータフローを確認できます。

7 - Spectrum™ Technology Platform について

このセクションの構成

Spectrum™ Technology Platform とは

Spectrum™ Technology Platform は、データの正規化、検証、拡張 (価値向上) の 3 つの側面からデータの完全性、妥当性、一貫性、適時性、および正確性を高めるシステムです。データを正確かつ包括的に、最新の状態に維持することで、顧客への理解を深め、顧客とより良好な関連性を構築できます。

Spectrum™ Technology Platform は、以下の機能を実行して、データの品質を高めるビジネス ルールの設計と実装を支援します。

パーシング、名前の正規化、名前のバリデーション

正規化をきわめて正確に実行するには、一連のデータ列を複数のフィールドに分割する必要がある場合があります。Spectrum™ Technology Platform は、個人名、企業名、およびその他多くの語や略語をパースする高度なパーシング機能を備えています。また、スキャン/抽出操作のベースとして使用するカスタム表現のリストを独自に作成することもできます。Universal Name モジュールは、この機能を備えています。

重複除外統合

一意のエンティティを識別することで、レコードを統合する、重複レコードを排除する、および "最良の組み合わせ" レコードを作成できます。"最良の組み合わせ" レコードとは、別のレコードのデータを使用して作成する複合的なレコードです。Advanced Matching モジュールと Data Normalization モジュールは、この機能を備えています。

住所検証

住所検証では、管轄の郵便当局のルールを適用して、住所を標準形式に変換し、その住所が配達可能な住所であるかどうかを確認します。住所検証により、郵便料金の割引を受けやすくなり、郵便物の配達品質を高めることができます。Universal Addressing モジュールと Address Now モジュールは、この機能を備えています。

ジオコーディング

ジオコーディングとは、住所を地図上のポイント（緯度と経度）に変換する処理です。ジオコーディングは、地図製作に使用されますが、それは 1 つの使用例にすぎません。基盤を成すロケーション データがあると、ビジネス上の意思決定を行いやすくなります。処理を逆にすることで、ジオコード (緯度と経度で表現される地図上のポイント) を入力し、そのジオコードに関する住所情報を取得できます。Enterprise Geocoding モジュールは、この機能を備えています。

ロケーションインテリジェンス

ロケーション インテリジェンスは、地理関係を調査、評価、分析、およびモデル化して、データに関する新しい情報を作成します。ロケーション インテリジェンス処理を使用すると、ロケーションを検証し、情報を有益なビジネス インテリジェンスに変換できます。Location Intelligence モジュールは、この機能を備えています。

マスターデータ管理

マスター データ管理では、重要なデータ アセットの関連性を中心に捉えたマスター データ ビューを作成できます。Data Hub モジュールは、インフルエンサーと明白でない関連性の特定、詐欺行為の検出、情報の品質、統合、およびアクセシビリティを高めるのに役立ちます。

税務管轄区域の割り当て

税務管轄区域の割り当てでは、住所の地域に適用される税務管轄区域を判断します。税務管轄区域を最も正確に割り当てると、経済上のリスクや、法的義務を軽減できます。

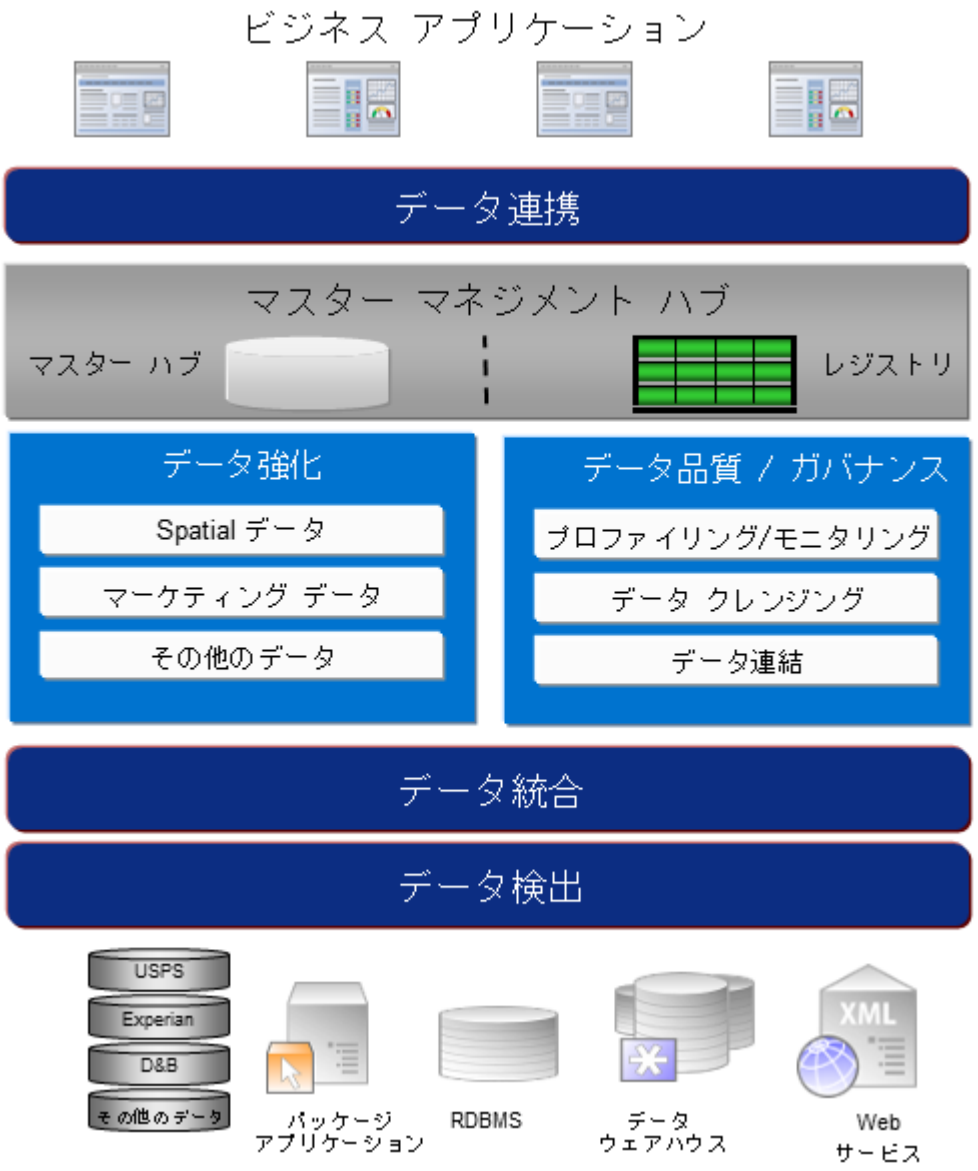
Spectrum™ Technology Platform が提供する Pitney Bowes ソフトウェアでは、最新の税務管轄区域と顧客レコードの正確な住所を統合して正確な州、郡、郡区、市、および特殊な税務管轄区域の情報をレコードに追加できます。税務管轄区域の割り当ての使用例を次に示します。

- 売上税と使用税
- 動産税
- 保険料税

Enterprise Tax モジュールは、この機能を備えています。

エンタープライズ データ管理アーキテクチャ

Spectrum™ Technology Platform では、包括的なエンタープライズ データ管理処理を構築できます。あるいは、対象をさらに絞り込んだソリューションとしてこれを活用することも可能です。次の図は、ソースからデータを取得し、データ強化およびデータ品質処理を経て、マスター データ管理ハブに引き渡す、包括的なソリューションを示したものです。MDM ハブは、データの単一のビューを作成して複数のビジネス アプリケーションに提供します。



データ検出

データ検出は、データ リソースをスキャンしてデータの状況を詳細に把握するプロセスです。Spectrum™ Technology Platform は、さまざまなデータプロファイリング手法を使用して、構造化されたデータ、構造化されていないデータ、および一部分のみ構造化されたデータをスキャンできます。スキャン結果は、会社のデータ アセットを記述するドキュメントのライブラリの生成とメタデータ リポジトリの作成に自動的に使用されます。このドキュメントと付属のメタデータ リポジトリから提供される情報を十分に吟味したうえで、データ統合、データ品質、データ制御、またはマスター データ管理プロジェクトを始めてください。

Spectrum™ Technology Platform のデータ検出モジュールの詳細については、営業担当者にお問い合わせください。

データ統合

データの状況を把握したら、次は、管理する必要があるデータへのアクセス方法を検討する必要があります。Spectrum™ Technology Platform は、複数のソースのデータに直接接続できます。また、既存のデータ アクセス手法を統合した方法で接続することもできます。データ ウェアハウス、データ品質、システム統合、移行といった多様なビジネス ニーズに対応するバッチおよびリアル タイム データ統合機能をサポートします。Spectrum™ Technology Platform は RDBMS データベース、データ ウェアハウス、XML ファイル、フラット ファイルなどのデータにアクセスできます。Spectrum™ Technology Platform は、複雑な結合や集計を含むSQL クエリをサポートし、視覚的なクエリ開発ツールを提供します。また、Spectrum™ Technology Platform は REST および SOAP Web サービスを介してデータにアクセスできます。

Spectrum™ Technology Platform は、指定されたフォルダ内の1つ以上のソースファイルの存在チェック結果に基づいてバッチ処理をトリガできます。この "ホット フォルダ" トリガは、FTP アップロードの監視と、アップロード直後の処理に役立ちます。

これらのデータ統合機能の一部には、Enterprise Data Integration モジュールのライセンスが必要です。詳細については、営業担当者に問い合わせてください。

最後に、Spectrum™ Technology Platform は SAP や Siebel などのパッケージ アプリケーションと統合可能です。

データ品質/ガバナンス

データ品質およびデータ ガバナンス処理では、重複レコード、矛盾した情報、不正確な情報がないか、データを確認します。

重複マッチングは、重複レコードの可能性や、レコード間の関連性を特定します。データが実際の名前や住所であるか、または他の種類の顧客情報であるかは関係ありません。Spectrum™ Technology Platform では、boolean 型マッチング方式、スコアリング方式、しきい値、アルゴリズム、および重みを使用する一貫したビジネスマッチルールを指定して、レコードのグループに重複が含まれているかどうかを調べることができます。Spectrum™ Technology Platform は、多種多様なカスタマイズをサポートしているため、ビジネス固有のニーズに適合するようにルールを調整できます。

重複レコードを特定したら、それらのレコードを統合することもできます。Spectrum™ Technology Platform は、重複レコードをリンクまたは結合して、収集した顧客情報から最も正確かつ包括的なレコードを作成する方法を指定できます。例えば、ある世帯のすべてのレコードに基づいて、1 つの Best-of-Breed (最良の組み合わせ) レコードを作成できます。重複の特定とその排除には、Advanced Matching モジュールを使用します。

データ品質処理では、データの正規化も行われます。正規化は、きわめて重要な処理です。レコードの照合とレコード間の関連性の識別において、最も可能性の高い結果を得るために、正規化データ要素が必要であるためです。モジュールによっては、複数のタイプの正規化を実行するものもありますが、Spectrum™ Technology Platform の Data Normalization モジュールは最も包括的な正規化機能セットを備えています。また、Universal Name モジュールは、個人名や企業名データを処理するための特定のデータ品質機能を提供します。

正規化データは、必ずしも正確なデータではありません。Spectrum™ Technology Platform は、データを既知の最新の参照データと比較して、その妥当性を確認できます。この処理に用いられるソースとしては、米国郵政公社などの規制機関、Experian やD&B などのサードパーティのデータプロバイダ、会計データなどの企業内の参照ソースがあります。Spectrum™ Technology Platform は、住所データの検証に特に優れています。世界中の 250 の国および地域の住所の検証または正規化が可能です。住所の検証を実行するモジュールには、Address Now モジュールと Universal Addressing モジュールの 2 つがあります。

どちらのモジュールがニーズに適しているかは、営業担当者と相談して判断してください。

Spectrum™ Technology Platform は、幅広いデータ品質問題を自動的に処理できますが、データスチュワードによる手動確認が適切である場合が存在します。これをサポートするために、Business Steward モジュールでは、手動確認をトリガするルールを指定するための方法と、例外レコードを確認するための Web ベースのツールが提供されています。確認および解決処理においてデータスチュワードを支援するための、Bing マップや Experian データといったサードパーティ ツールへの統合アクセスも含まれています。

データ強化(データ・エンリッチメント)

データ強化処理は、追加情報によってデータを増補します。強化は、データに詳細情報を追加するためにユーザが使用したいと考える、空間データ、マーケティング データ、または他のソースからのデータに基づいて行うことができます。例えば、顧客住所のデータベースが存在する場合、住所のジオコーディングを行って、住所の緯度/経度座標を特定し、その座標をレコードの一部として保存することができます。これによって顧客データは、顧客に最も近い銀行支店の検索など、多様な空間分析に使用できるようになります。Spectrum™ Technology Platform では、データをさまざまな情報で強化できます。例えば、ジオコーディング (Enterprise Geocoding モジュールを使用)、税務管轄区域の割り当て (Enterprise Tax モジュール)、地理空間分析 (Location Intelligence モジュール)、2 点間の車移動または徒歩経路 (Enterprise Routing モジュール) の情報を利用できます。

マスター データ管理ハブ

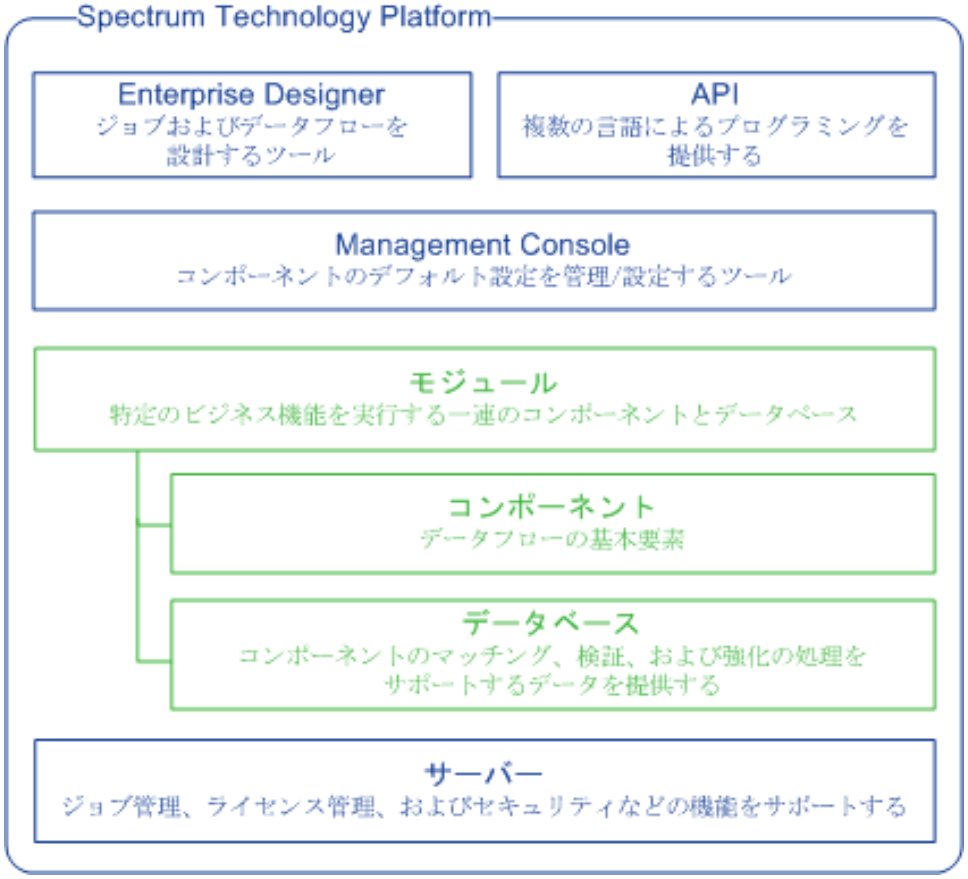
マスター データ管理 (MDM) ハブは、エンティティと、その役割、処理、やり取りの間の複雑な関連性の迅速なモデリングを可能にします。ソーシャル ネットワーク分析機能が組み込まれており、インフルエンサー (influencer) の理解、チャーンの予測、明白でない関係や不正パターンの検出、レコメンデーションを支援します。

Spectrum™ Technology Platform は、MDM ハブに対する 2 つのアプローチをサポートします。マスター ハブのアプローチでは、データは単一の MDM データベースに維持され、アプリケーションは MDM データベースからデータにアクセスします。レジストリのアプローチでは、データは各ビジネス アプリケーションに維持され、MDM ハブ レジストリに、関連レコードの検索に用いられるキーが含まれます。例えば、顧客のレコードが、注文入力データベースと顧客サポートデータベースに存在する場合があります。この場合MDM レジストリには、両方の場所の顧客データへのアクセスに使用できる単一のキーが含まれます。

Data Hub モジュールが、MDM 機能を提供します。

Spectrum™ Technology Platformのアーキテクチャ

Spectrum™ Technology Platformが提供するPitney Bowes は、いくつかのモジュールを実行するサーバーで構成されます。これらのモジュールは、住所のバリデーション、ジオコーディング、高度なパーシング等、さまざまな機能を備えています。次の図に、Spectrum™ Technology Platform のアーキテクチャを示します。



サーバー

このサーバーがSpectrum™ Technology Platformの基盤となります。サーバーは、データを処理し、リポジトリ データを同期し、通信を管理します。また、ジョブ管理およびセキュリティ機能も提供します。

モジュール

モジュールは、特定の機能を実行する機能群です。例えば、Universal Addressing モジュールは、郵便規格に準拠するように住所を正規化します。Enterprise Tax モジュールは、その住所に該当する税務管轄区域を判定します。共通のビジネス問題を解決する各種モジュールがまとめられて、バンドルとしてライセンス供与されています。

コンポーネント

モジュールは、特定の機能をフロー内で実行するか、サービスとして実行するコンポーネントで構成されます。例えば、Enterprise Geocoding モジュールの Geocode US Address コンポーネントは、住所を地図上のポイント (緯度と経度) に変換して返します。Universal Addressing モジュールの Get City State Province は、郵便番号が示す都市および州/省を返します。

システムで利用できるコンポーネントは、Spectrum™ Technology Platformのどのバンドルのライセンスを取得したかによって異なります。

データベース

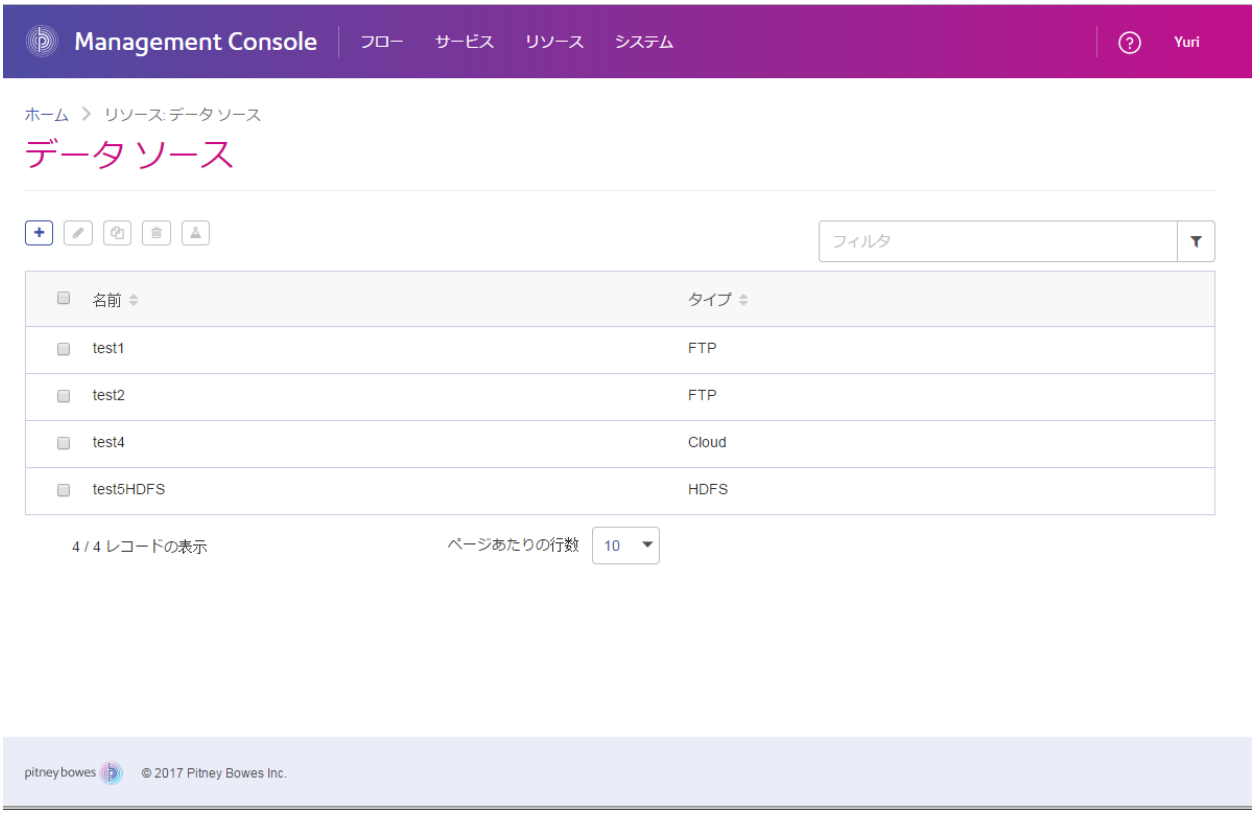
一部のモジュールは、参照データを含むデータベースに依存します。例えば、Universal Addressing モジュールは、米国の住所を検証して正規化するために米国郵政公社 (USPS) のデータにアクセスする必要があります。データベースは個別にインストールされ、一部のデータベースは最新データを提供するために定期的に更新されます。

モジュールには、必須データベースとオプションのデータベースがあります。オプションのデータベースは、Spectrum™ Technology Platformの処理を強化することのできる特定の機能に必要なデータを提供します。

Management Console

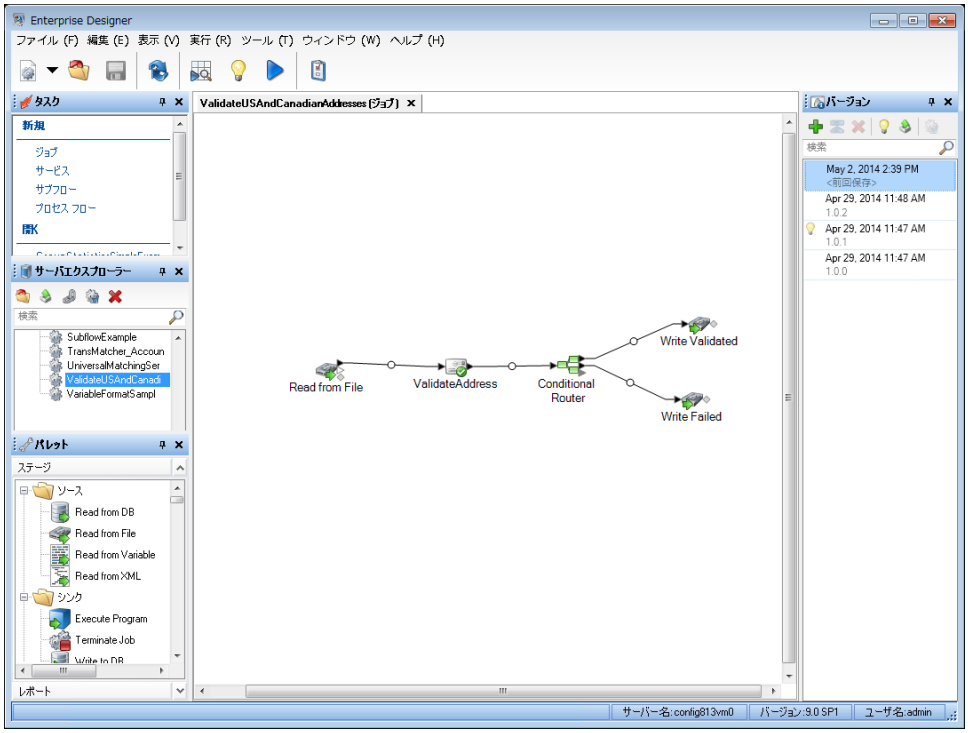
Management Console は、Spectrum™ Technology Platformを管理するためのツールです。Management Console で実行できる操作は、次のとおりです。

- Spectrum™ Technology Platformとデータの間の接続を定義する。
- サービスやフローのデフォルト設定を指定する。
- 権限やパスワード等、ユーザ アカウントを管理する。
- ログを表示する。
- ライセンス有効期限情報を含めて、ライセンスを表示する。



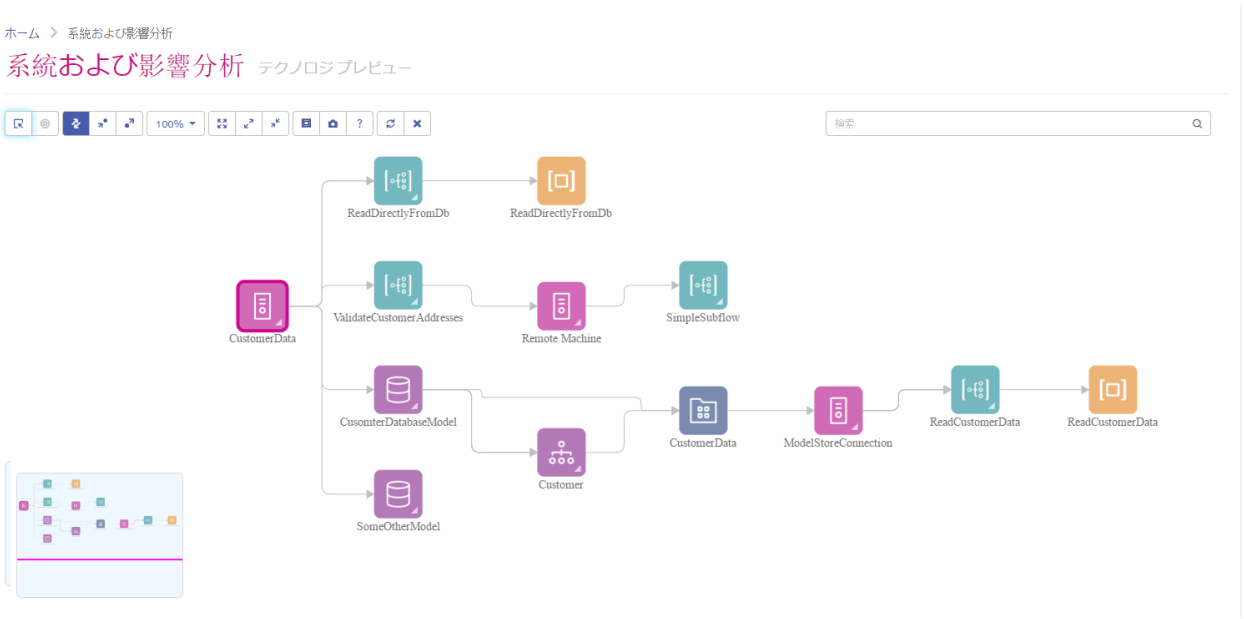
Enterprise Designer

Enterprise Designer は、Spectrum™ Technology Platformのジョブ、サービス、サブフロー、およびプロセス フローを作成するためのツールです。ドラッグアンドドロップ インターフェイスを利用して、複雑なデータフローをグラフィカルに、容易に作成できます。



Metadata Insights

Metadata Insights を使用すると、適切な時間に収集された正確なデータに基づくビジネス分析を得るために必要な制御が可能になります。Metadata Insights を使用して、データ モデルを開発し、ソースからビジネス アプリケーションまでのデータの流れを表示し、プロファイリングによってデータの品質を評価します。この分析を活用すれば、特定のビジネスの課題を解決できるデータ リソースの特定、ビジネス全体でデータの有益性と一貫性を向上するプロセスの適合と最適化、およびデータの問題のトラブルシューティングを行うことができます。



Web サービスと API

Web サービスとプログラミング API を使用して、Spectrum™ Technology Platformの機能を独自のアプリケーションに統合することができます。これらのインターフェイスは、シンプルな統合とレコード処理の効率化を可能とし、将来のバージョンの下位互換性をサポートします。

Spectrum™ Technology PlatformAPI は、以下の言語で使用可能です。

- C
- C++
- COM
- Java
- .NET

Web サービスは、SOAP および REST を介して提供されています。

モジュールとコンポーネント

表 10 : モジュールとコンポーネント

モジュール	説明	コンポーネント
Address Now モジュール	米国以外の住所についての高度なバリデーションと正規化を提供します。また、別の住所処理も提供します。	Build Global Address Get Global Candidate Addresses Validate Global Address
Advanced Matching モジュール	入力ファイル内や入力ファイル間でレコードを照合します。	Best Of Breed Candidate Finder Duplicate Synchronization Filter Interflow Match Intraflow Match Match Key Generator Transactional Match
Analytics Scoring モジュール	入力ファイル内や入力ファイル間でレコードを照合します。	PMML Model Scoring Miner データセットからの読み込み Miner データセットへの書き出し
Business Steward モジュール	例外レコードを特定し、例外レコードを手動で確認するためのブラウザベースのツールを提供します。	Exception Monitor Read Exceptions Write Exceptions
Country Identifier	国名を、または郵便番号と州/省の組み合わせを受け取って、2 文字の ISO 国コード、3 文字の Universal Postal Union (UPU) コード、および国名を英語で返します。	Country Identifier

モジュール	説明	コンポーネント
Data Hub モジュール	データをリンクおよび分析して、関係と傾向を識別します。	Write to Hub Read From Hub Query Hub Graph Visualization
Data Integration モジュール	データ ウェアハウジング、データ品質、システム統合、および移行に便利な機能を備えています。	Field Selector Generate Time Dimension Query Cache Write to Cache
Data Normalization モジュール	データの不整合を取り除きます。	Advanced Transformer Open Parser Table Lookup Transliterator
Enterprise Data Integration	データ ウェアハウジングや、データ品質、システム統合、移行などのさまざまなビジネス ニーズに対応するために複数のソースのデータに接続します。	Call Stored Procedure Field Selector Generate Time Dimension Query Cache Write to Cache
Enterprise Geocoding モジュール	住所を表す地図上のポイント（緯度と経度）に変換します。また、指定された緯度と経度を住所に変換します。	Geocode Address AUS Geocode Address GBR Geocode Address Global Geocode Address World Geocode US Address GNAF PID Location Search Reverse APN Lookup Reverse Geocode Address Global Reverse Geocode US Location

モジュール	説明	コンポーネント
Enterprise Routing モジュール	自動車移動または徒歩の経路を取得し、移動時間および移動距離を計算し、始点から一定の時間または距離内に含まれる領域を特定します。	Get Route Data Get Travel Boundary Get Travel Cost Matrix Get Travel Directions Persistent Update
Enterprise Tax モジュール	特定の地域に適用する税務管轄区域を決定します。	Assign GeoTAX Info Calculate Distance
GeoConfidence モジュール	指定された領域に住所または交差点が含まれる可能性を判定します。	Geo Confidence Surface CreatePointsConvexHull
Global Addressing モジュール	米国以外の住所に対する高度な住所の正規化および検証機能を提供します。また、入力の途中で住所を自動的に予測し、入力に基づく候補を直ちに返します。機械学習の手法を使用して、郵便住所文字列を個々の住所要素に分割します。	Global Address Validation Global Type Ahead グローバル住所パーサー
Global Geocoding モジュール	住所を表す地図上のポイント（緯度と経度）に変換します。また、指定された緯度と経度を住所に変換します。	Global Geocode Global Reverse Geocode
Global Sentry	政府から提供される、各国のデータを含む警戒リストとトランザクションとの照合を試みます。	Global Sentry Global Sentry Address Check Global Sentry ID Number Check Global Sentry Name Check Global Sentry Other Data Check

モジュール	説明	コンポーネント
Location Intelligence モジュール	さまざまな地理空間データベースと照合して Point In Polygon と半径分析を実行します。	Closest Site Find Nearest Point In Polygon Query Spatial Data Read Spatial Data Spatial Calculator Spatial Union
Machine Learning モジュール	数値データをビニングし、教師ありと教師なしの機械学習モデルを適合し、これらのモデルでデータをスコアリングするための機能を提供します。	Binning Binning Lookup Java Model Scoring K-Means Clustering Linear Regression Logistic Regression 主成分分析 Random Forest Classification Random Forest Regression
Metadata Insights	適切な時間に収集された正確なデータに基づくビジネス分析を得るために必要な制御を可能にします。データ モデルを開発し、ソースからビジネス アプリケーションまでのデータの流れを表示し、プロファイリングによってデータの品質を評価できます。この分析を活用すれば、特定のビジネスの課題の解決に使用すべきデータ リソースの特定や、ビジネス全体でデータの有益性と一貫性を向上するプロセスの最適化を行うことができます。	モデル (論理および物理) Model Store プロファイル 系統および影響分析

モジュール	説明	コンポーネント
SAP モジュール	Spectrum™ Technology Platform と SAP Customer Relationship Management モジュール アプリケーションとの連携を有効にします。	SAP Generate Match Key SAP Generate Match Score SAP Generate Search Key SAP Generate Search Key Constant SAP Generate Search Key Metaphone SAP Generate Search Key Substring SAP Validate Address With Candidates
Siebel モジュール	Spectrum™ Technology Platform と Siebel アプリケーションとの連携を有効にします。	Siebel Generate Match Key Siebel Generate Match Score Siebel Generate Search Key Siebel Business Name Standardization Siebel Standardize Name. Siebel Geocode US Address With Candidates Siebel Geocode US Address With No Candidates Siebel Get Global Candidate Addresses Siebel Validate Address With Candidates Siebel Validate Address With No Candidates
Universal Addressing モジュール	郵便当局の規格に従って、住所を正規化してバリデーションを実行します。	Get Candidate Addresses Get City State Province Get Postal Codes Validate Address Validate Address AUS Validate Address Global

モジュール	説明	コンポーネント
Universal Name モジュール	個人名、企業名、住所、およびその他多くの語や略語をパースします。	Name Parser (非推奨) Name Variant Finder Open Name Parser

著作権に関する通知

USPS® 情報

Pitney Bowes Inc. は、ZIP + 4® データベースを光学および磁気媒体に発行および販売する非独占的ライセンスを所有しています。CASS、CASS 認定、DPV、eLOT、FASTforward、First-Class Mail、Intelligent Mail、LACS^{Link}、NCOA^{Link}、PAVE、PLANET Code、Postal Service、POSTNET、Post Office、RDI、Suite^{Link}、United States Postal Service、Standard Mail、United States Post Office、USPS、ZIP Code、および ZIP + 4 の各商標は United States Postal Service が所有します。United States Postal Service に帰属する商標はこれに限りません。

Pitney Bowes Inc. は、NCOA^{Link}® 処理に対する USPS® の非独占的ライセンスを所有しています。

Pitney Bowes Software の製品、オプション、およびサービスの価格は、USPS® または米国政府によって規定、制御、または承認されるものではありません。RDI™ データを利用して郵便送料を判定する場合に、使用する郵便配送業者の選定に関するビジネス上の意思決定が USPS® または米国政府によって行われることはありません。

データ プロバイダおよび関連情報

このメディアに含まれて、Pitney Bowes Software アプリケーション内で使用されるデータ製品は、各種商標によって、および次の 1 つ以上の著作権によって保護されています。

© Copyright United States Postal Service.All rights reserved.

© 2014 TomTom.All rights reserved.TomTom および TomTom ロゴは TomTom N.V の登録商標です。

© 2016 HERE

Fuente: INEGI (Instituto Nacional de Estadística y Geografía)

電子データに基づいています。© National Land Survey Sweden.

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

このプログラムの一部は著作権で保護されています。© Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Second Decimal, LLC

© Copyright Canada Post Corporation

この CD-ROM には、Canada Post Corporation が著作権を所有している編集物からのデータが収録されています。

© 2007 Claritas, Inc.

Geocode Address World データ セットには、**http://creativecommons.org/licenses/by/3.0/legalcode** に存在するクリエイティブ コモンズ アトリビューション ライセンス (「アトリビューション ライセンス」) の下に提供されている GeoNames Project (**www.geonames.org**) からライセンス供与されたデータが含まれています。お客様による GeoNames データ (Spectrum™ Technology Platformユーザ マニュアルに記載) の使用は、アトリビューション ライセンスの条件に従う必要があり、お客様と Pitney Bowes Software, Inc. との契約と、アトリビューション ライセンスの間に矛盾が生じる場合は、アトリビューション ライセンスのみに基づいてそれを解決する必要があります。お客様による GeoNames データの使用に関しては、アトリビューション ライセンスが適用されるためです。

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com