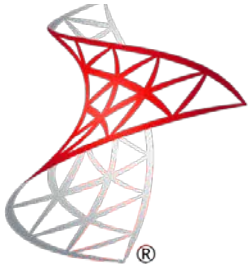
Microsoft



Microsoft® SQL Server® 2012

SQL Server 2012 自習書シリーズ No.8

Integration Services 応用

Published: 2008 年 6 月 20 日
SQL Server 2012 更新版: 2012 年 9 月 30 日
有限会社エスキューエル・クオリティ

SQLQuality
データベース コンサルティング

目次

STEP 1. 本自習書の概要と自習書を試す環境について

この STEP では、自習書の概要について説明します。

この STEP では、次のことを学習します。

- ✓ 自習書の内容について
- ✓ 自習書を試す環境について

1.1 本自習書の内容について

本自習書の内容について

本自習書では、**SQL Server Integration Services**（**SSIS**）の応用的／実践的な利用方法を説明します。Integration Services の基本的な操作方法については、本自習書シリーズの「**Integration Services 入門**」編で説明しています。

Integration Services 入門編で説明した内容は、次のとおりです。

- **インポート／エクスポート ウィザードによる単純なデータ転送**（テキスト ファイル／Access データベース／Excel ファイルからのインポート方法など）
- **SSIS デザイナーの基本操作**（データ フロー タスクによるデータ転送、派生列コンポーネントによるデータ変換、参照コンポーネントによる別テーブルのデータ取得など）
- **データ ビューアーによる転送中データの表示方法**
- **SSIS パッケージ（.dtsx）の実行方法**（dtexecUI、dtexec）
- **SSIS パッケージの定期的な実行方法**（SQL Server Agent ジョブとしての登録）

本自習書では、次の内容を説明します。

- **スクリプト タスクと変数の利用方法**
- **SQL 実行タスクの結果を変数へ格納**
- **データ フロー タスク内の SQL ステートメントのパラメーター化**
- **ログ記録機能**
- **イベント ハンドラー**
- **データ転送時にエラーが発生した場合の動作**（エラーの無視、エラー情報のファイルへの書き込みなど）
- **スクリプト コンポーネントによるエラー メッセージの取得**
- **Foreach Loop コンテナーによる複数ファイルの読み取り**
- **ブレークポイントを利用したステップ実行**
- **パッケージの別マシンへの配置方法**（パッケージの配置、パラメーター化など）

1.2 自習書を試す環境について

必要な環境

この自習書の手順を試すために必要な環境は次のとおりです。

- **OS**

  Windows Server 2008 SP2 以降 または
  Windows Server 2008 R2 SP1 以降 または
  Windows Server 2012 または
  Windows Vista SP2 以降 または Windows 7 SP1 以降 または Windows 8

- **ソフトウェア**

  SQL Server 2012

  SQL Server 2012 のインストール時には、次のコンポーネントを選択して、インストールしておく必要があります。

  - **データベース エンジン サービス**
  - **Integration Services**
  - **SQL Server Data Tools**
  - **管理ツール**

  この自習書内での画面やテキストは、OS に Windows Server 2008 R2（x64）SP1、ソフトウェアに SQL Server 2012 Enterprise エディション（x64）を利用して記述しています。

サンプル スクリプト

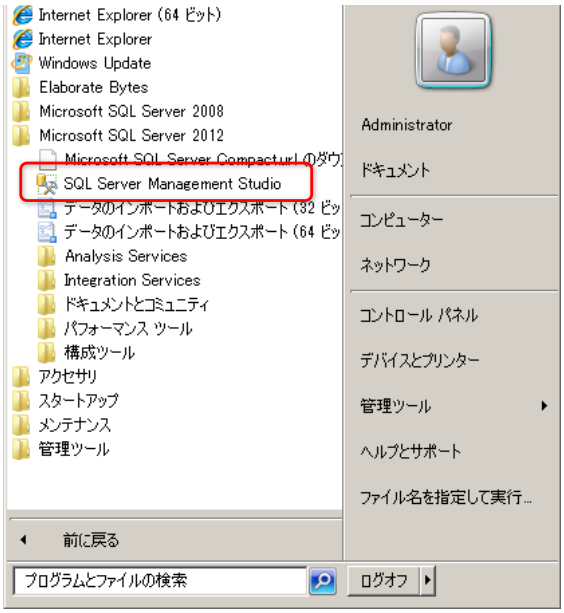
この自習書を試すには、サンプル スクリプトをダウンロードしておく必要があります。サンプル スクリプトには、各 STEP のインポートの実習で使用するファイル（CSV ファイルや SQL スクリプト）が含まれています。

1.3 事前作業：サンプル スクリプトのセットアップ

Step 2 で使用するテーブルの作成

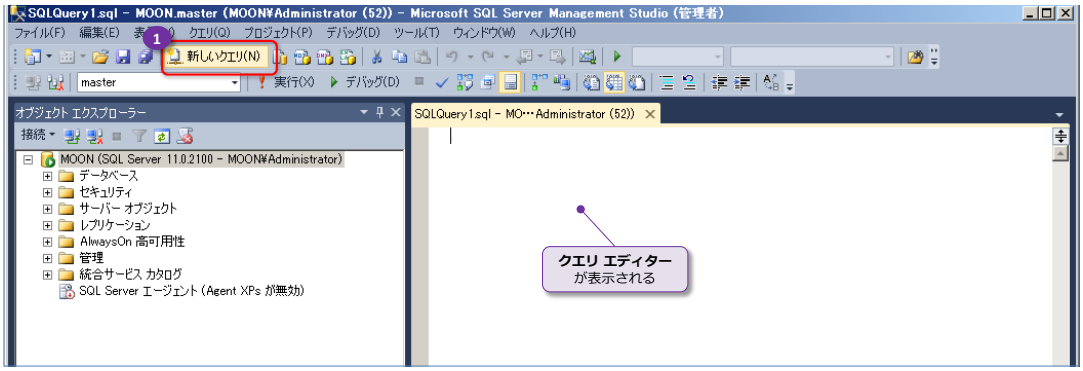
本自習書の Step 2 を試すには、サンプル スクリプト内にある「**CreateTable.txt**」を実行して、「**sampleDB**」データベースと「**社員**」テーブルを作成しておく必要があります（実行手順は、次のとおりです）。

1. まずは、スクリプトを実行するために、**Management Studio** を起動します。[**スタート**] メニューの [**すべてのプログラム**] から、[**Microsoft SQL Server 2012**] を選択して [**SQL Server Management Studio**] をクリックします。

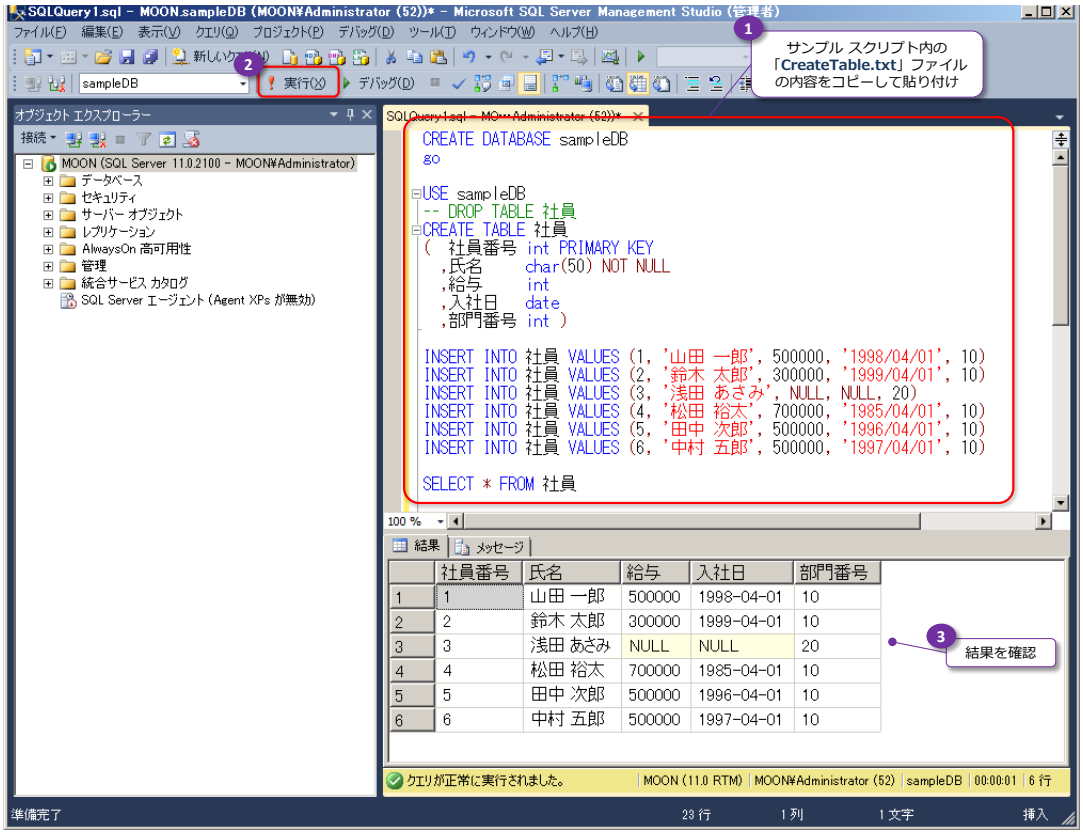


2. [**サーバーへの接続**] ダイアログが表示されたら、[**サーバー名**] へ SQL Server の名前を入力して、[**接続**] ボタンをクリックします。

3. 接続完了後、Management Studio が開いたら、次のようにツールバーの [**新しいクエリ**] をクリックして、**クエリ エディター**を開きます。



4. 次に、**Windows エクスプローラー**を起動して、**サンプル スクリプト**をダウンロードしたフォルダーを展開し、このフォルダー内の「**CreateTable.txt**」ファイルをダブル クリックして開きます。ファイルの内容をすべてコピーして、クエリ エディターへ貼り付けます。



貼り付け後、ツールバーの [**！実行**] ボタンをクリックしてクエリを実行します。これにより、「**sampleDB**」という名前のデータベースが作成され、その中へ「**社員**」テーブルが作成されます。実行後、「**社員**」テーブルの 6 件のデータが表示されれば、実行が完了です。

STEP 2. スクリプト タスクと変数の利用

この STEP では、スクリプト タスクや変数の使い方などについて説明します。これらは、本格的な Integration Services パッケージを作成する上で欠かせない機能になります。

この STEP では、次のことを学習します。

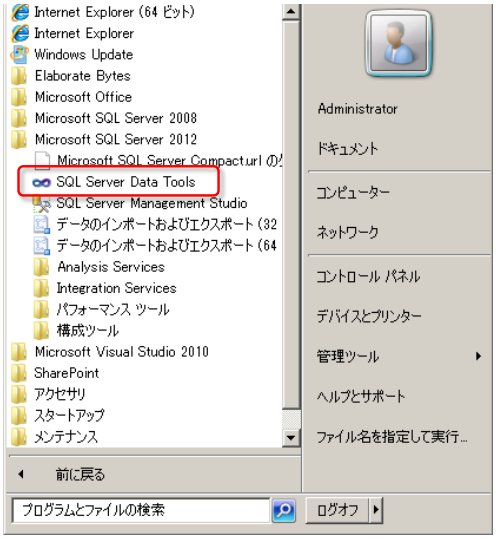
- ✓ スクリプト タスクとは
- ✓ スクリプト タスクによるシステム変数の表示
- ✓ スクリプト タスクによるファイル書き込み
- ✓ ユーザー定義の変数の利用
- ✓ SQL 実行タスク結果の変数への格納
- ✓ SQL ステートメントのパラメーター化と変数の代入

2.1 スクリプト タスクの実行

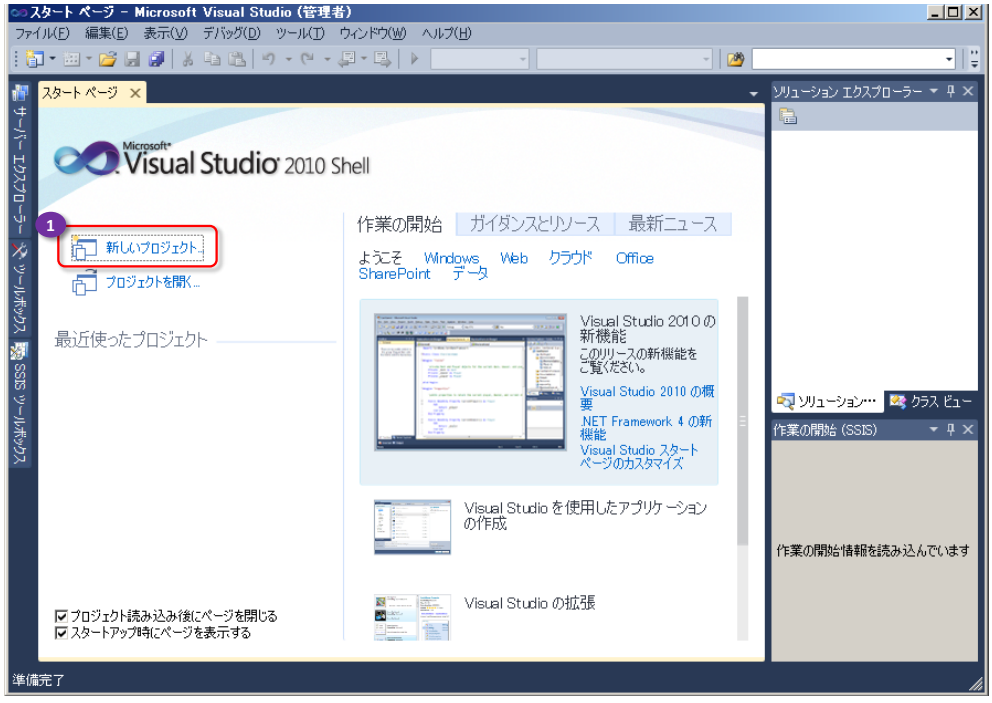
スクリプト タスクの実行

スクリプト タスクを利用すると、**Visual C# 2010** や **Visual Basic 2010** で記述したスクリプトを実行できるようになります。ここでは、Visual Basic 2010 を利用して簡単なスクリプトを記述し、実行できることを確認してみましょう。

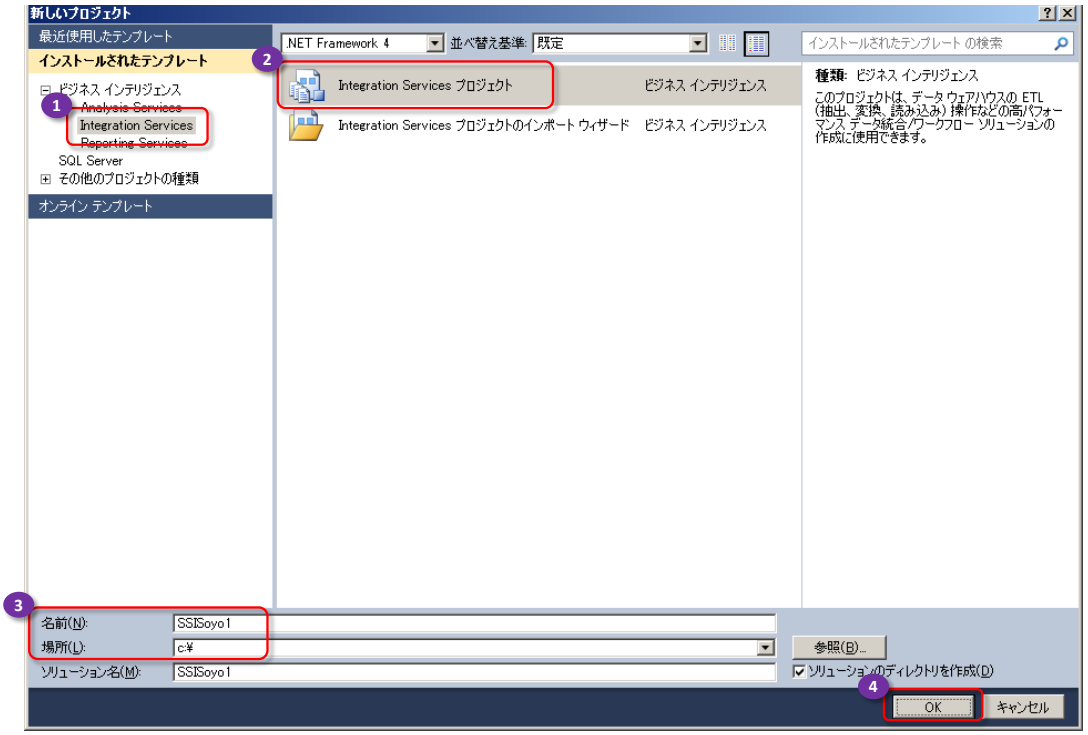
1. まずは、[**スタート**]メニューの[**すべてのプログラム**]から、[**Microsoft SQL Server 2012**]の［**SQL Server Data Tools**］をクリックして、**SQL Server Data Tools** を起動します。



2. **SQL Server Data Tools**（Visual Studio 2010）が起動したら、［**スタート ページ**］の［**新しいプロジェクト**］をクリックして、新しいプロジェクトを作成します。



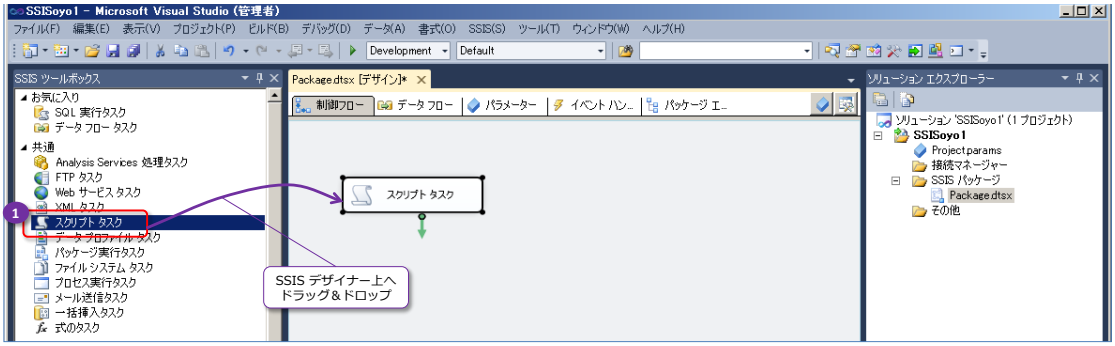
3. ［**新しいプロジェクト**］ダイアログが表示されたら、次のように［**インストールされたテンプレート**］で［**ビジネス インテリジェンス**］の［**Integration Services**］をクリックして、「**Integration Services プロジェクト**」を選択します。



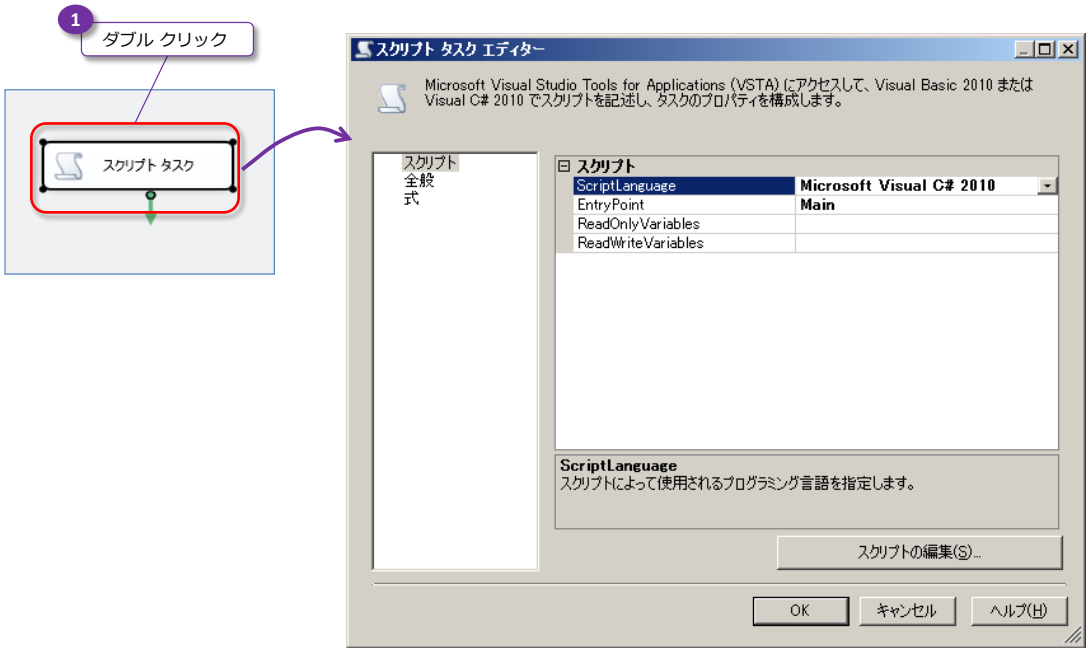
［**名前**］へ任意のプロジェクト名（画面は **SSISoyo1**）、［**場所**］へ任意の保存場所（画面は **C:¥**）を入力して、［**OK**］ボタンをクリックします。これにより、Integration Services プロジェクトが作成されます。

スクリプト タスクの追加

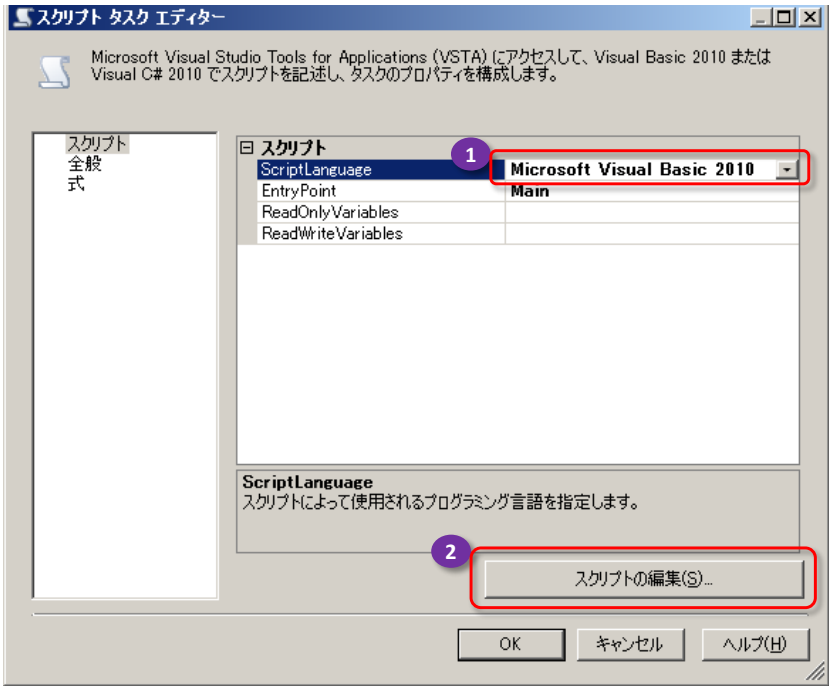
4. 次に、SSIS ツールボックスの［**共通**］カテゴリから［**スクリプト タスク**］を SSIS デザイナー上へドラッグ＆ドロップして配置します。



5. 続いて、配置した［**スクリプト タスク**］をダブル クリックして、［**スクリプト タスク エディター**］ダイアログを表示します。



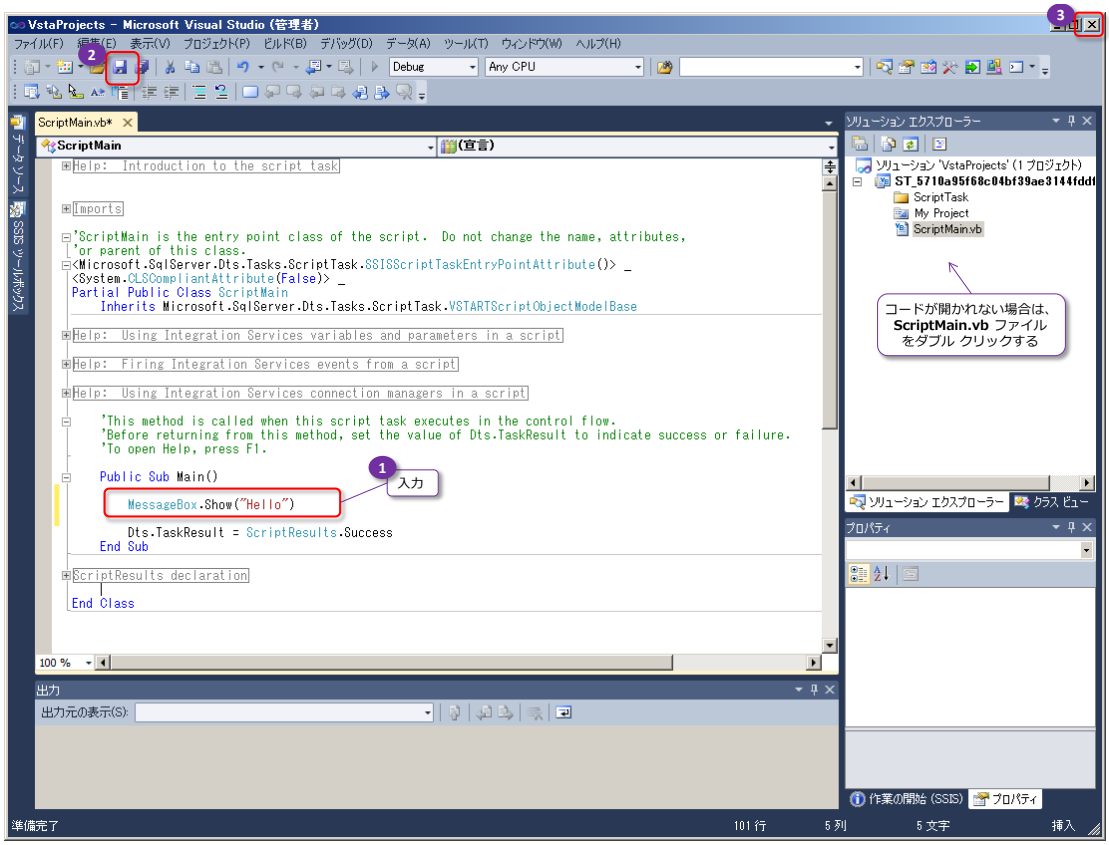
6. このダイアログでは、次のように［**ScriptLanguage**］で「**Microsoft Visual Basic 2010**」を選択して、［**スクリプトの編集**］ボタンをクリックします。



これで、Visual Basic 2010 でスクリプトを記述できるようになります。

7. 数秒後、次のように「**スクリプト エディター**」が起動されるので、「**public Sub Main( )**」内へ、以下のコードを記述します（コードが開かれていない場合は、**プロジェクト エクスプローラー**で［**ScriptMain.vb**］ファイルをダブル クリックして開きます）。

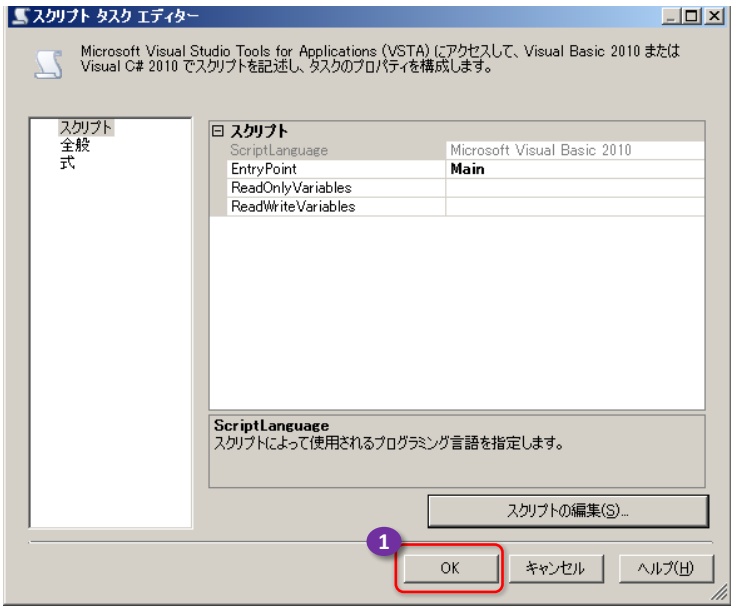
```
MessageBox.Show("Hello")
```



このコードにより、「**Hello**」という文字をメッセージ ボックスで表示できるようになります。

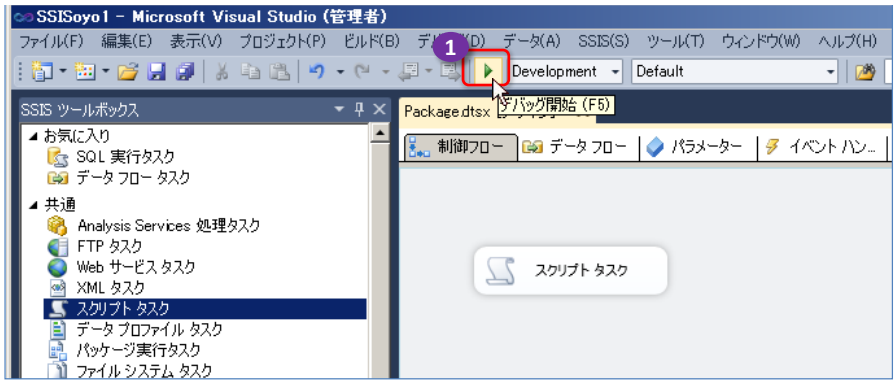
コードを記述後、ツールバーの［**保存**］ボタンをクリックしてスクリプトを保存し、スクリプト エディターを閉じます（［**×**］ボタンで終了します）。

8. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。

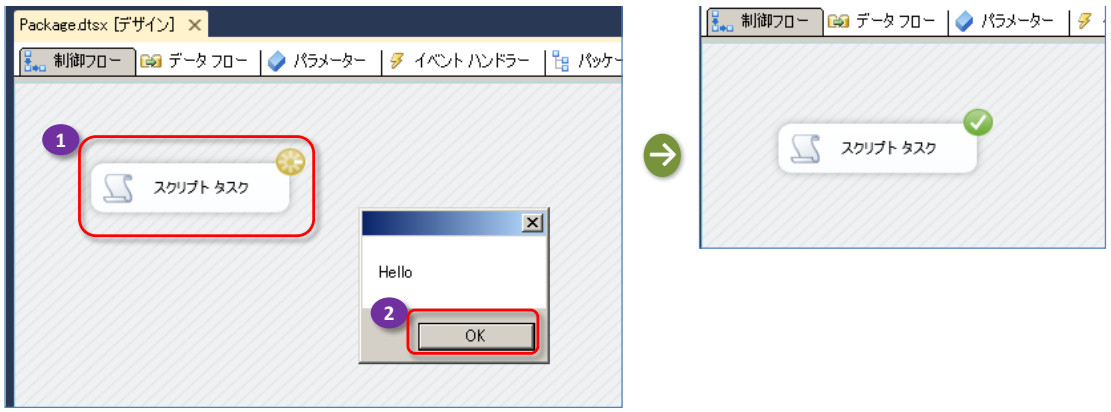


スクリプト タスクの実行

9. 次に、スクリプトを実行するために、ツールバーの［**デバッグ開始**］ボタンをクリックして、パッケージを実行します。

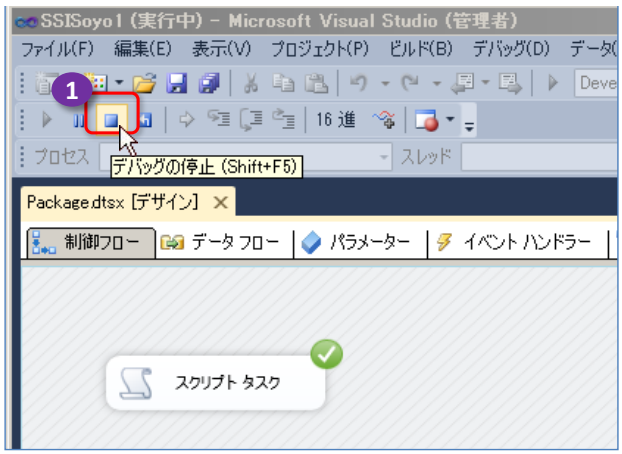


配置されている［**スクリプト タスク**］が実行中に変わって、メッセージ ボックスに「**Hello**」と表示されることを確認できます。



［**OK**］ボタンをクリックすると、［**スクリプト タスク**］に**緑のチェックマーク**が付いて、スクリプトの実行が成功したことを確認できます。

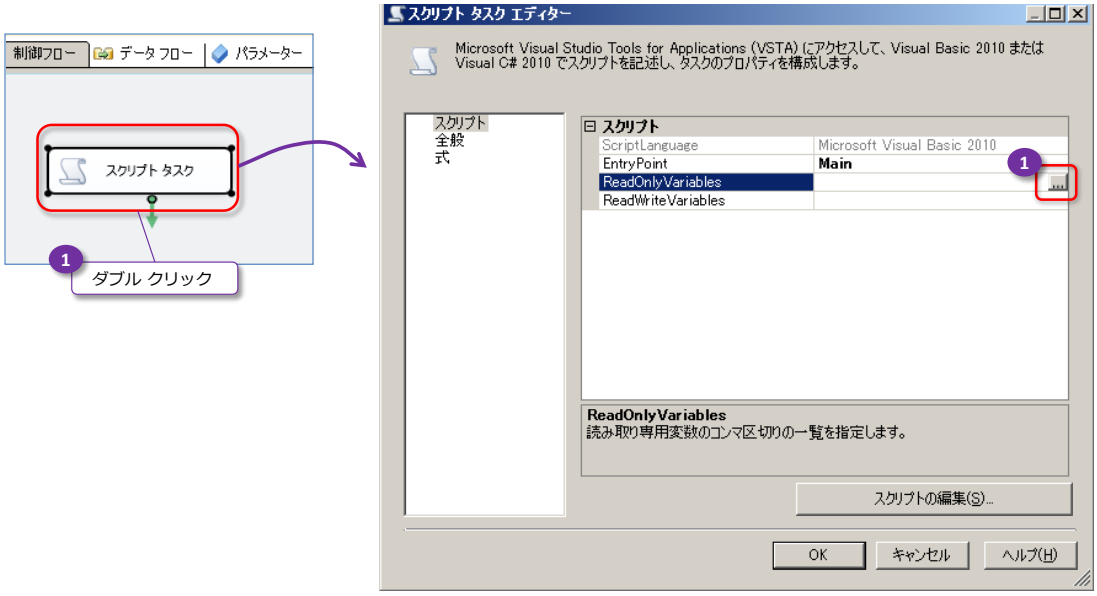
10. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。



システム変数の利用

次に、スクリプト内で Integration Services の**システム変数**を利用してみましょう。

1. 配置した［**スクリプト タスク**］をダブル クリックします。

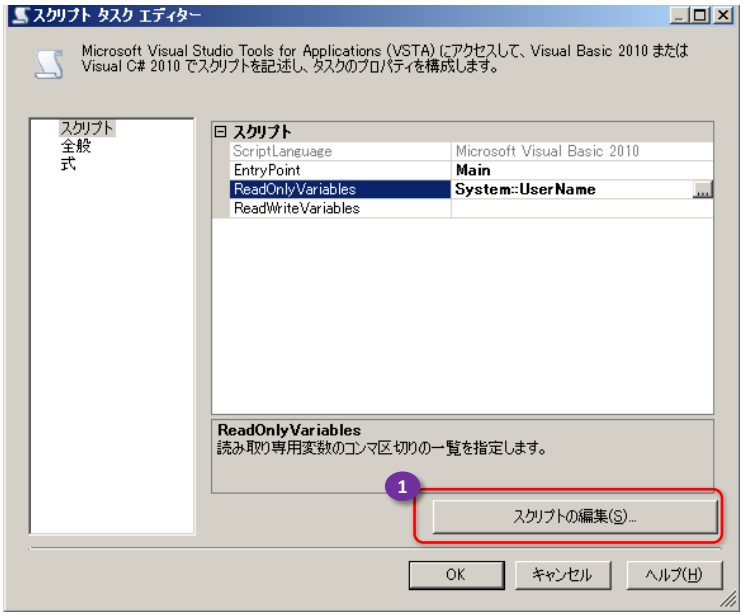


［**スクリプト タスク エディター**］ダイアログが表示されたら、［**ReadOnlyVariables**］の［**...**］ボタンをクリックします。

2. これにより、次のように［**変数の選択**］ダイアログが表示されて、「**System::**」で始まるシステム変数が一覧されます。

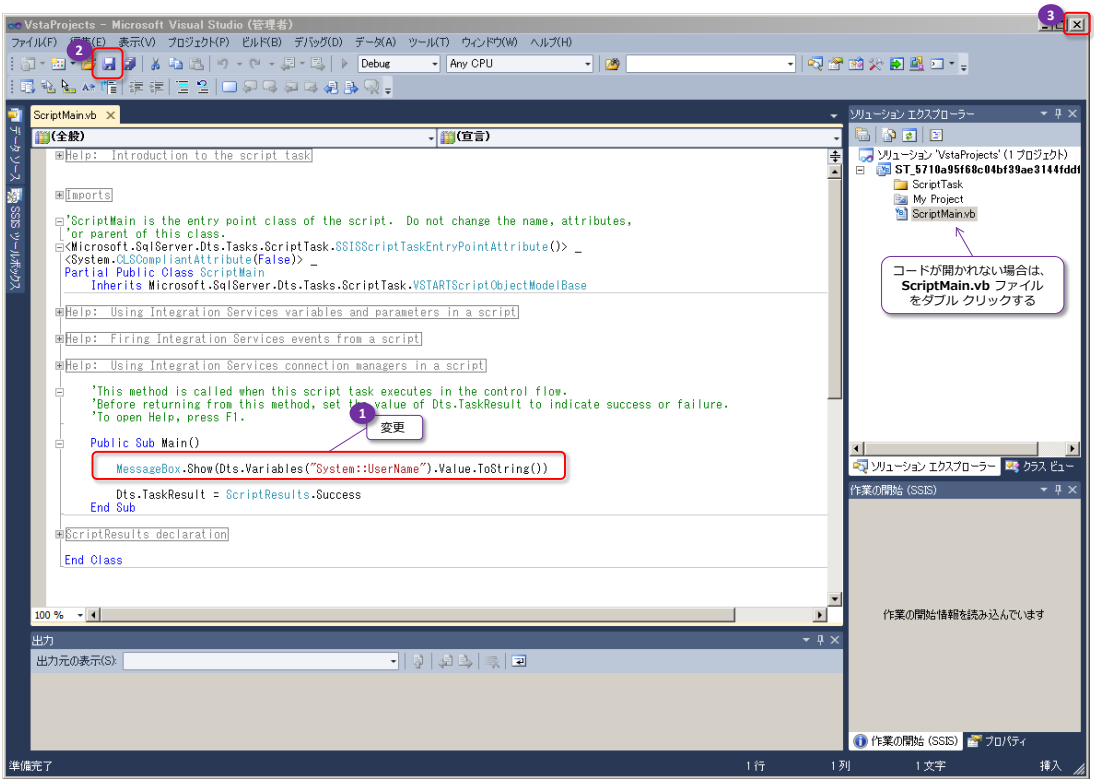
今回は、この中から「**System::UserName**」をチェックして、［**OK**］ボタンをクリックします。**UserName** システム変数は、パッケージを実行しているユーザーの名前を取得することができます。

3. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**スクリプトの編集**］ボタンをクリックします。



4. ［**スクリプト エディター**］が開いたら、「**public Sub Main( )**」内のコードを、次のように変更します。

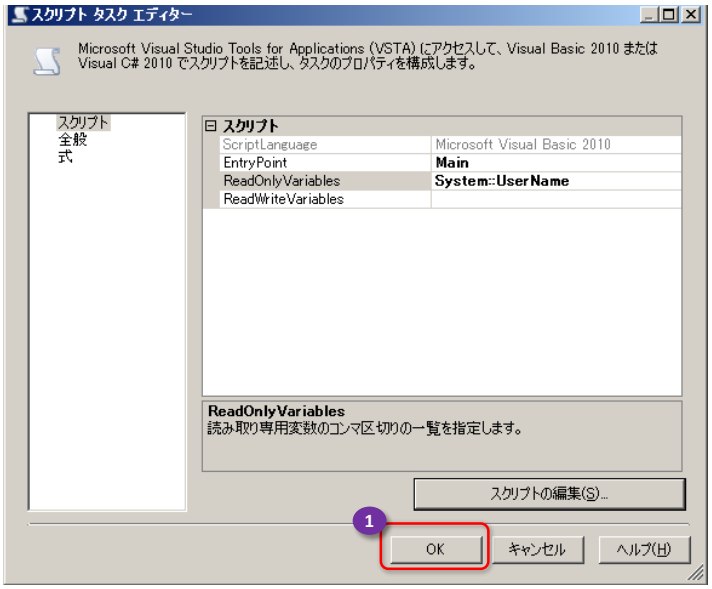
```
MessageBox.Show(Dts.Variables("System::UserName").Value.ToString())
```



スクリプト タスク内では、「**Dts.Variables("変数名")**」と指定することで、変数を利用することができます。変数名の「**System::UserName**」は、大文字と小文字を区別するので、注意してください。このコードにより、メッセージ ボックスでユーザー名を表示できるようになります。

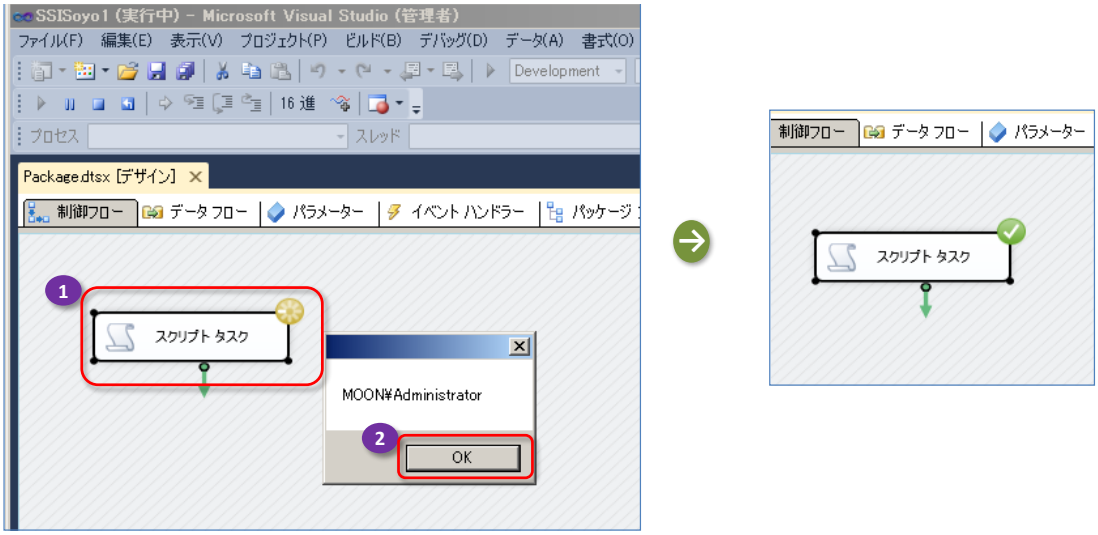
入力後、保存してスクリプト エディターを閉じます。

5. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。



6. 次に、スクリプトを実行するために、ツールバーの［**デバッグ開始**］ボタンをクリックして、パッケージを実行します。

7. 配置されている［**スクリプト タスク**］が実行中に変わって、実行しているユーザー名（画面は **MOON¥Adminitrator**）がメッセージ ボックスで表示されることを確認できます。



［**OK**］ボタンをクリックすると、［**スクリプト タスク**］に緑のチェックマークが付いて、スクリプトの実行が成功したことを確認できます。

8. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。

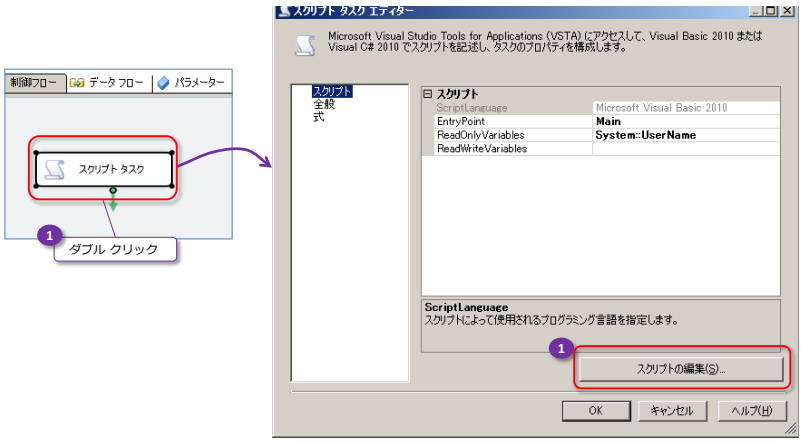
このように、スクリプト タスクを利用すると、変数を扱うことができます。スクリプト タスクでは、ユーザー定義の変数を扱うこともできますが、これについては Step2.3 で説明します。

2.2 スクリプト タスクからファイルへの書き込み

スクリプト タスクからファイルへの書き込み

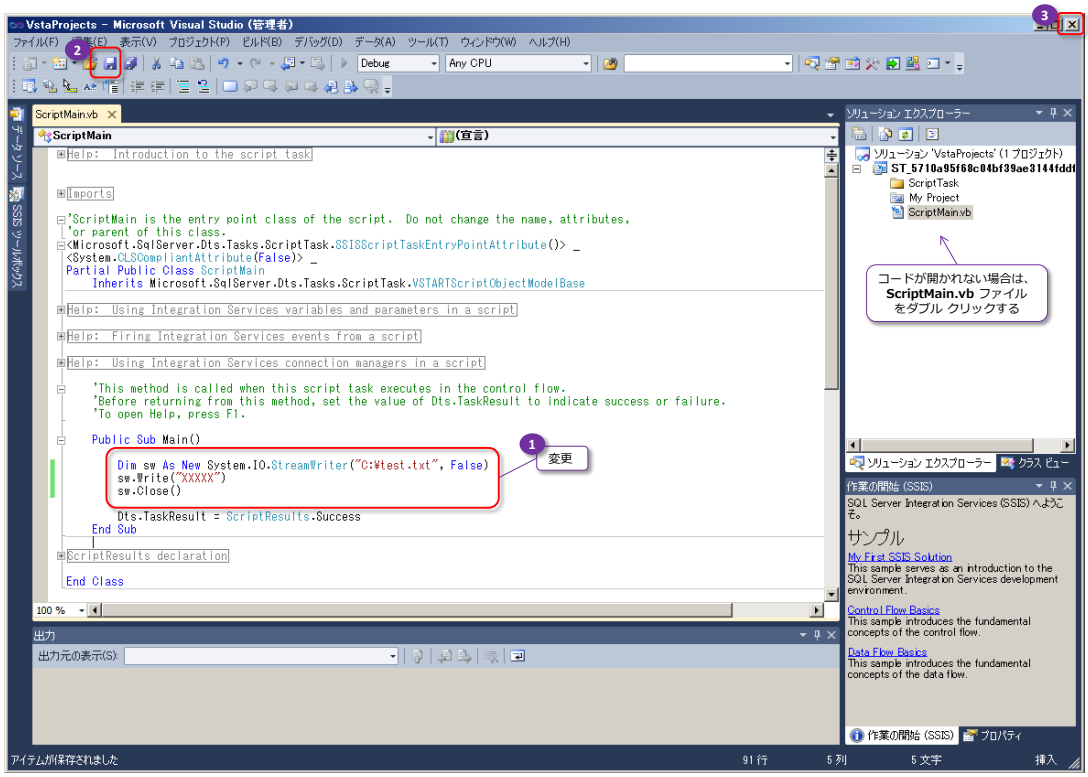
次に、ファイルへの書き込みを行うスクリプトを作成してみましょう。

1. まずは、配置した［**スクリプト タスク**］をダブル クリックして、［**スクリプト タスク エディター**］ダイアログを表示し、［**スクリプトの編集**］ボタンをクリックします。



2. ［**スクリプト エディター**］が表示されたら、「**public Sub Main( )**」内のコードを、次のように変更します。

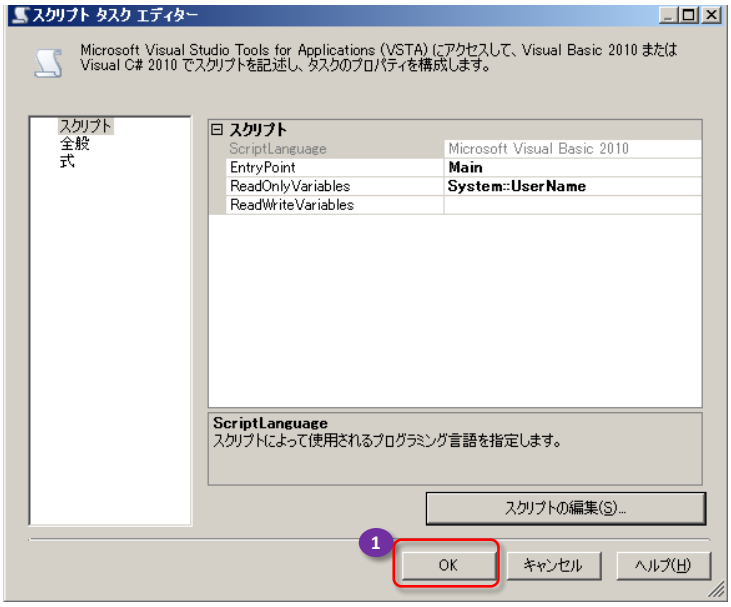
```
Dim sw As New System.IO.StreamWriter("C:¥test.txt", False)
sw.Write("XXXXX")
sw.Close()
```



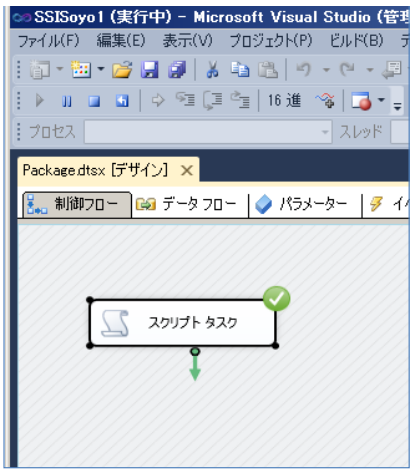
このコードにより、「**C:¥text.txt**」というファイルを作成して、「**XXXXX**」という文字を書き込むことができます。

入力後、保存してスクリプト エディターを閉じます。

3. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。



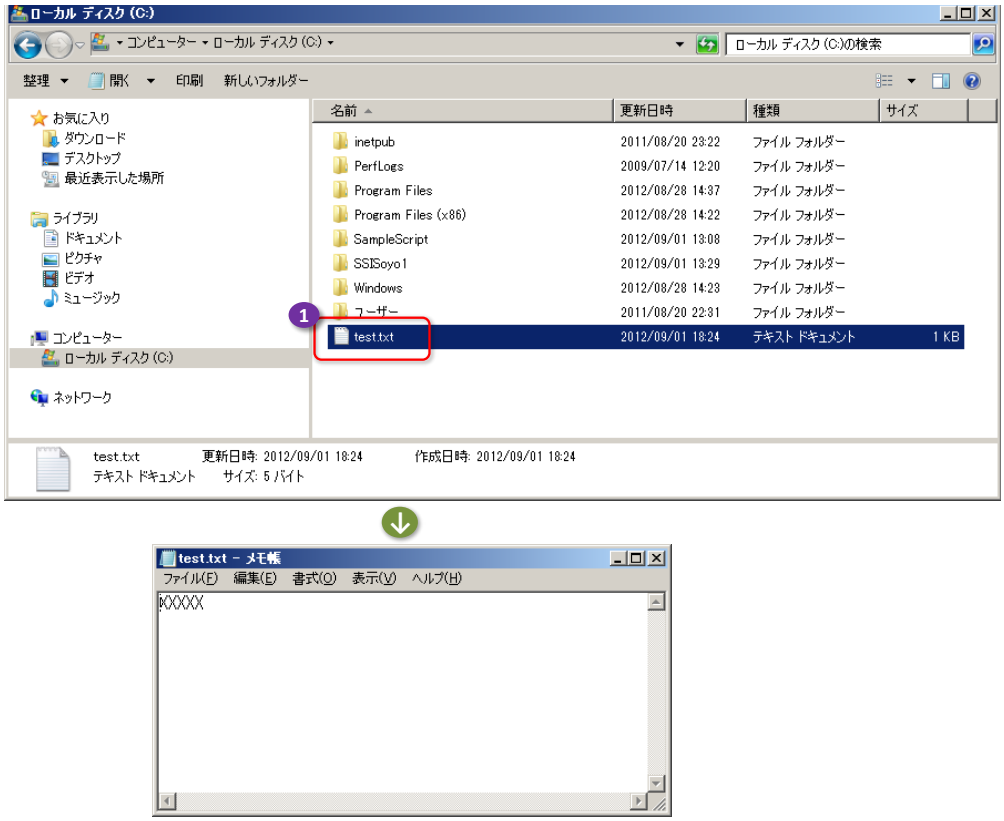
4. 次に、スクリプトを実行するために、ツールバーの［**デバッグ開始**］ボタンをクリックして、パッケージを実行します。



配置されている［**スクリプト タスク**］に緑のチェックマークが付いて、スクリプトの実行が成功したことを確認できます。

5. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。

6. 次に、**Windows エクスプローラー**を起動して、**C:¥text.txt** ファイルをダブル クリックして開きます。



「**XXXXX**」という文字が書き込まれていることを確認できます。

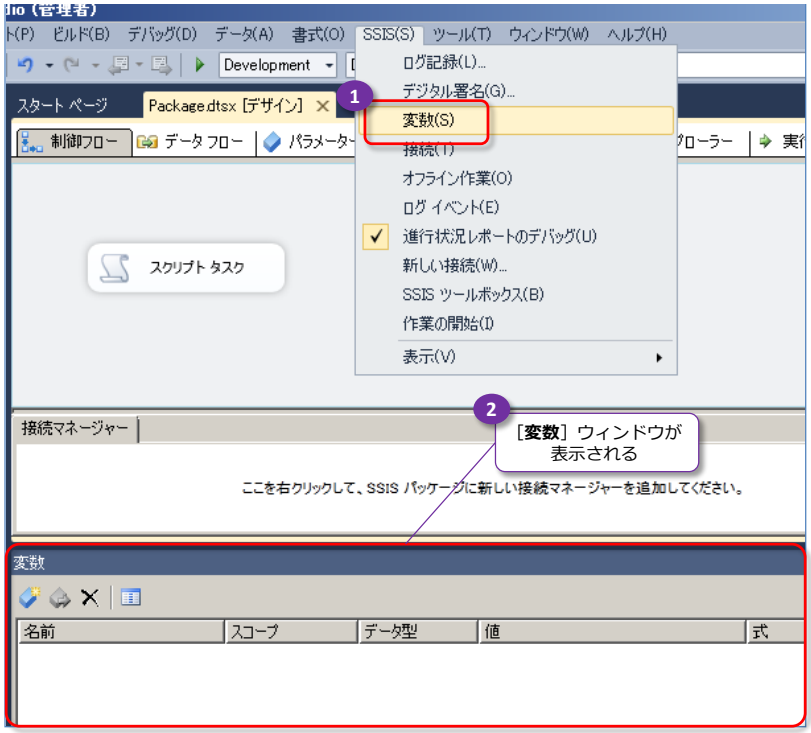
このように、スクリプト タスクでは、Visual Basic 2010 または Visual C# 2010 を利用して、任意の .NET コードを記述することができます。

2.3 ユーザー定義の変数の利用

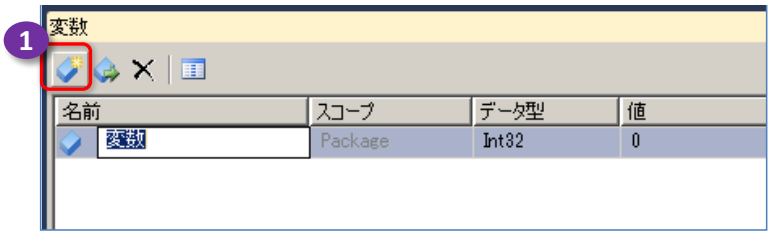
ユーザー定義の変数の利用

次に、ユーザー定義の変数を利用してみましょう。

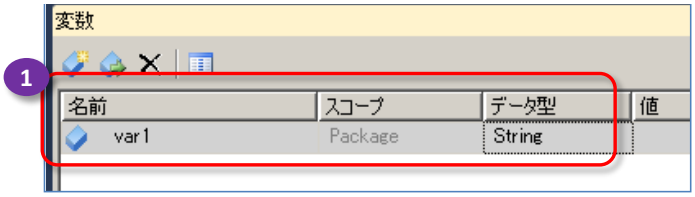
1. まずは、変数を定義します。次のように、[**SSIS**] メニューから [**変数**] をクリックして、[**変数**] ウィンドウを表示します。



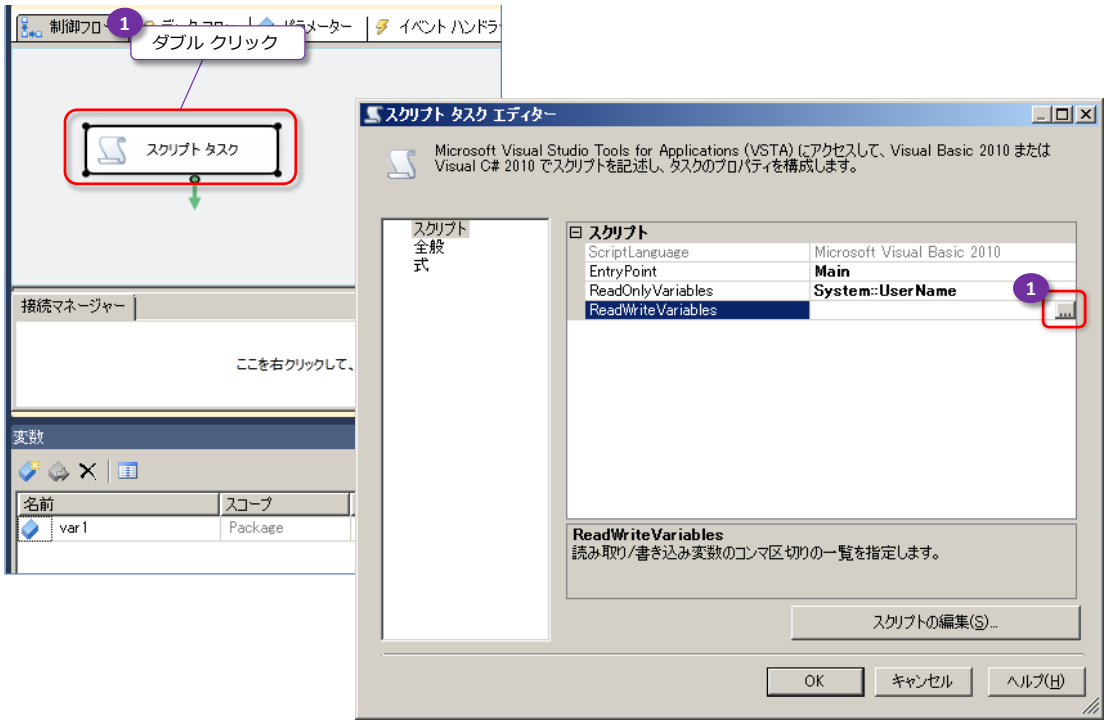
2. 続いて、ツールバーの [**変数の追加**] ボタンをクリックします。



3. これにより、変数が追加できるようになるので、今回は、次のように [**名前**] を「**var1**」、[**データ型**] を「**String**」（文字列）へ指定した変数を追加します。

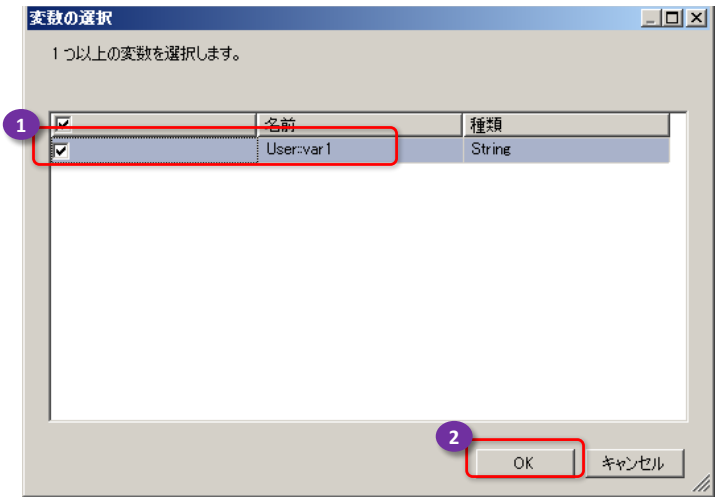


4. 続いて、制御フローに配置している [**スクリプト タスク**] をダブル クリックします。

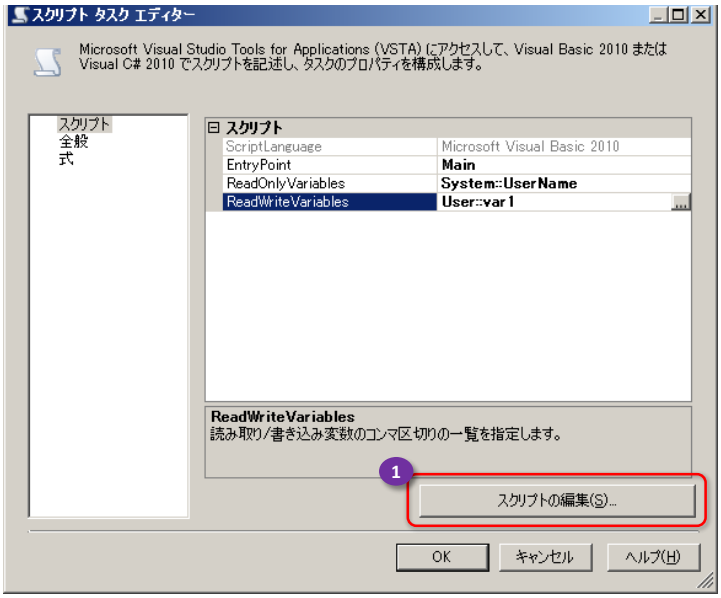


［**スクリプト タスク エディター**］ダイアログが表示されたら、［**ReadWriteVariables**］の［**...**］ボタンをクリックします（スクリプト内で変数へ値を代入するには、ReadOnly ではなく、**ReadWrite～** へ変数を指定する必要があります）。

5. これにより、［**変数の選択**］ダイアログで、変数の一覧が表示されるので、「**User::var1**」をチェックして、［**OK**］ボタンをクリックします（ユーザー定義の変数には、「**User::**」が付きます）。

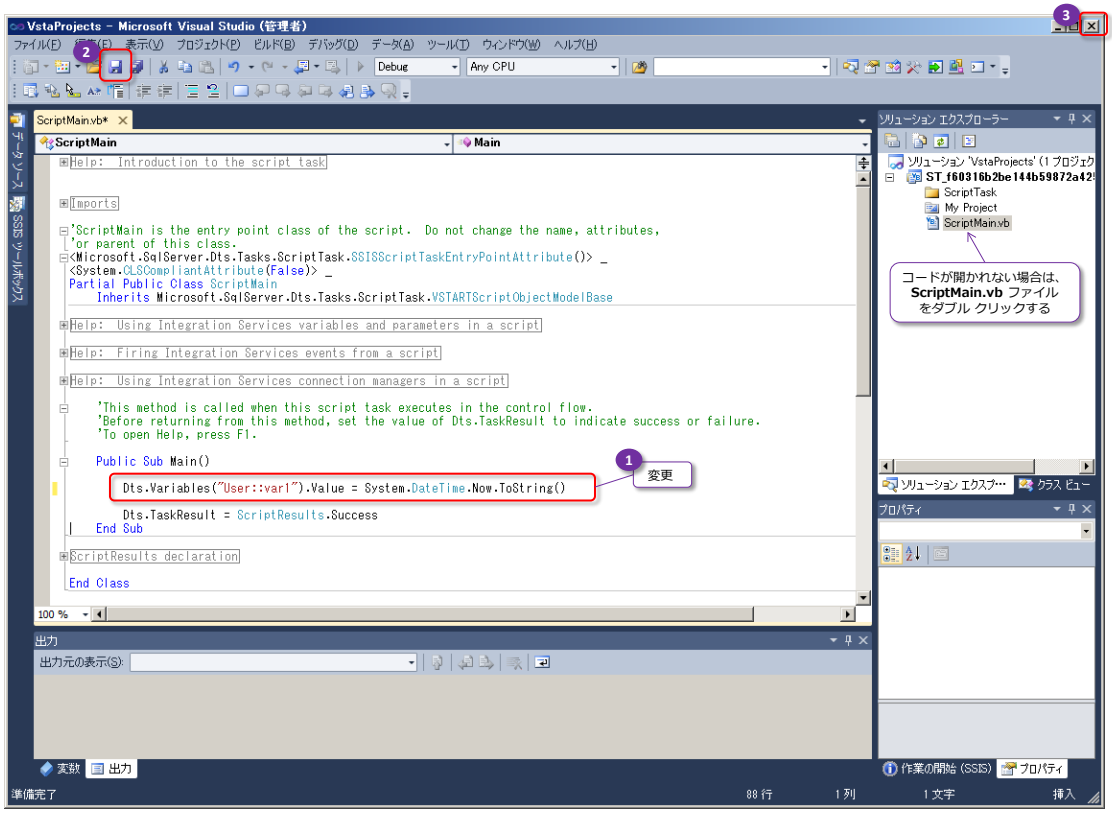


6. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**スクリプトの編集**］ボタンをクリックします。



7. ［**スクリプト エディター**］が表示されたら、「**public Sub Main( )**」内のコードを、次のように変更します。変数名「**User::var1**」は、大文字と小文字を区別するので、注意してください。

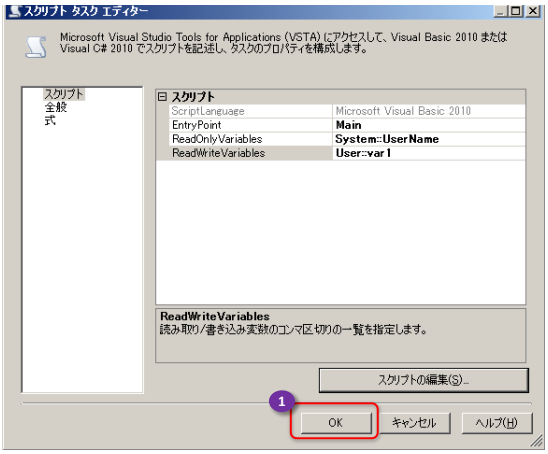
```
Dts.Variables("User::var1").Value = System.DateTime.Now.ToString()
```



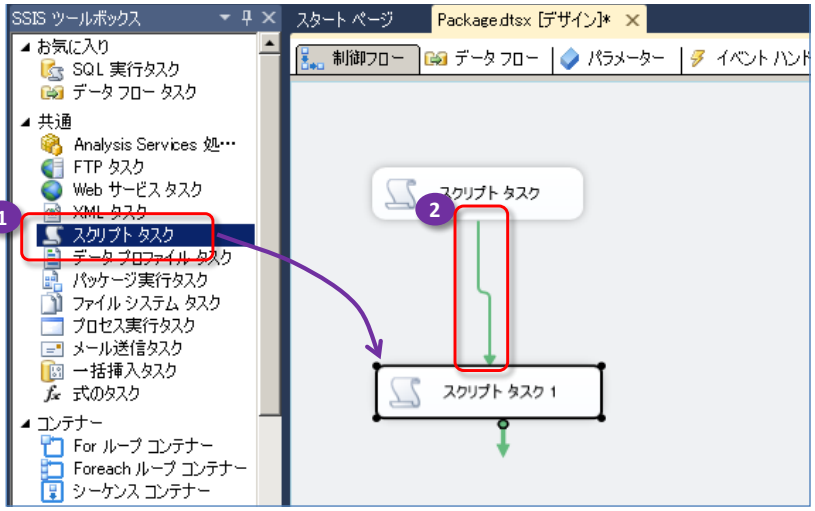
このコードにより、スクリプト実行時の時刻（**Now** プロパティ）を変数「**var1**」へ格納できるようになります。

入力後、保存してスクリプト エディターを閉じます。

8. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。

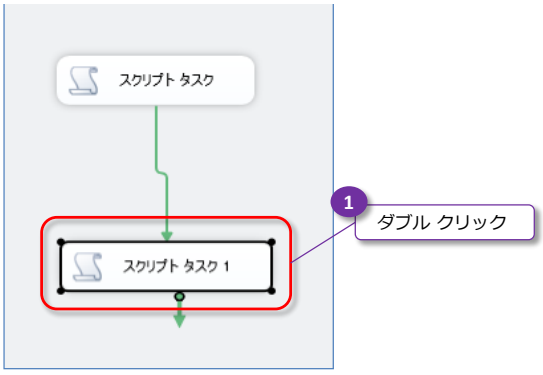


9. 次に、SSIS ツールボックスを開いて、［**スクリプト タスク**］をもうひとつ、SSIS デザイナー上へドラッグ＆ドロップして配置します。



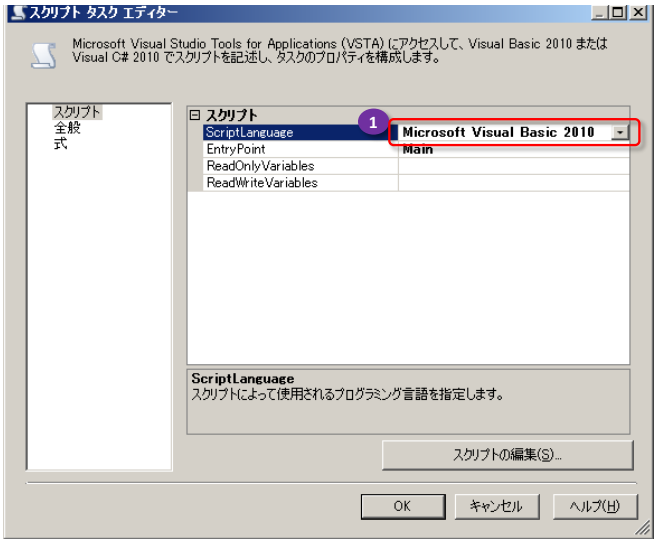
最初の［**スクリプト タスク**］の**緑色の矢印**を、新たに追加した［**スクリプト タスク 1**］まで、ドラッグ＆ドロップして伸ばします。

11. 追加した［**スクリプト タスク 1**］をダブル クリックして、［**スクリプト タスク エディター**］ダイアログを表示します。

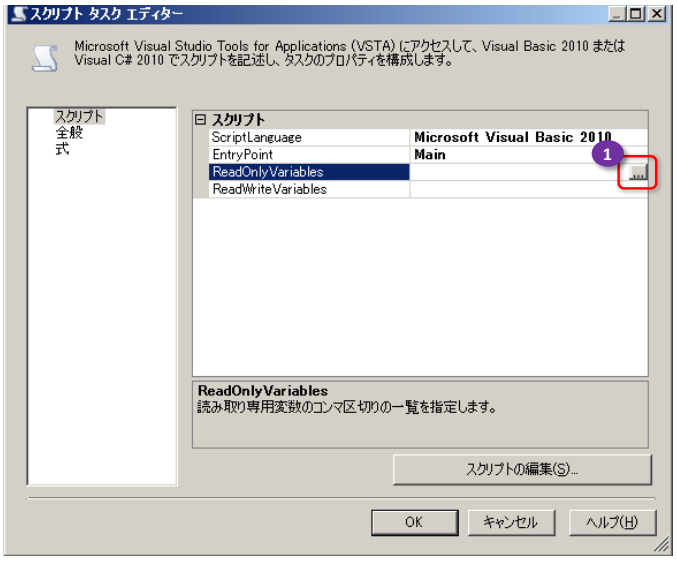


12. ［**スクリプト タスク エディター**］ダイアログが表示されたら、［**ScritLanguage**］で

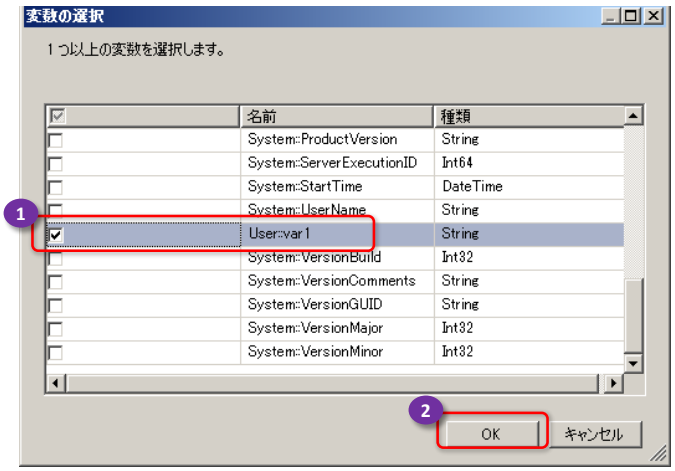
「**Microsoft Visual Basic 2010**」を選択します。



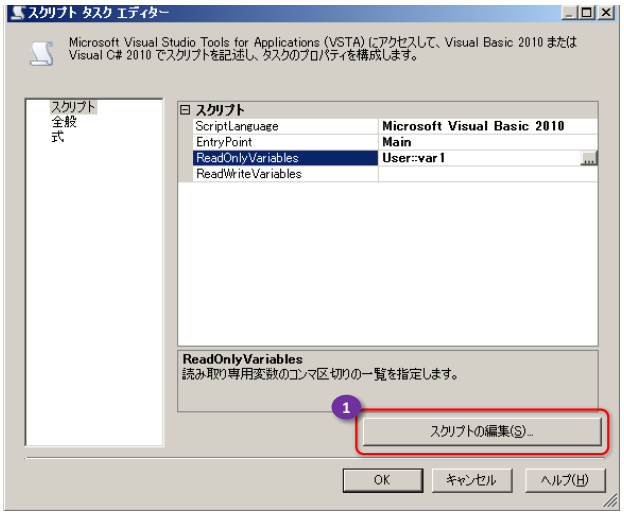
13. 次に、[**ReadOnlyVariables**] の [**...**] ボタンをクリックします。



14. [**変数の選択**] ダイアログが表示されたら、変数の一覧から、「**User::var1**」をチェックして、[**OK**] ボタンをクリックします。

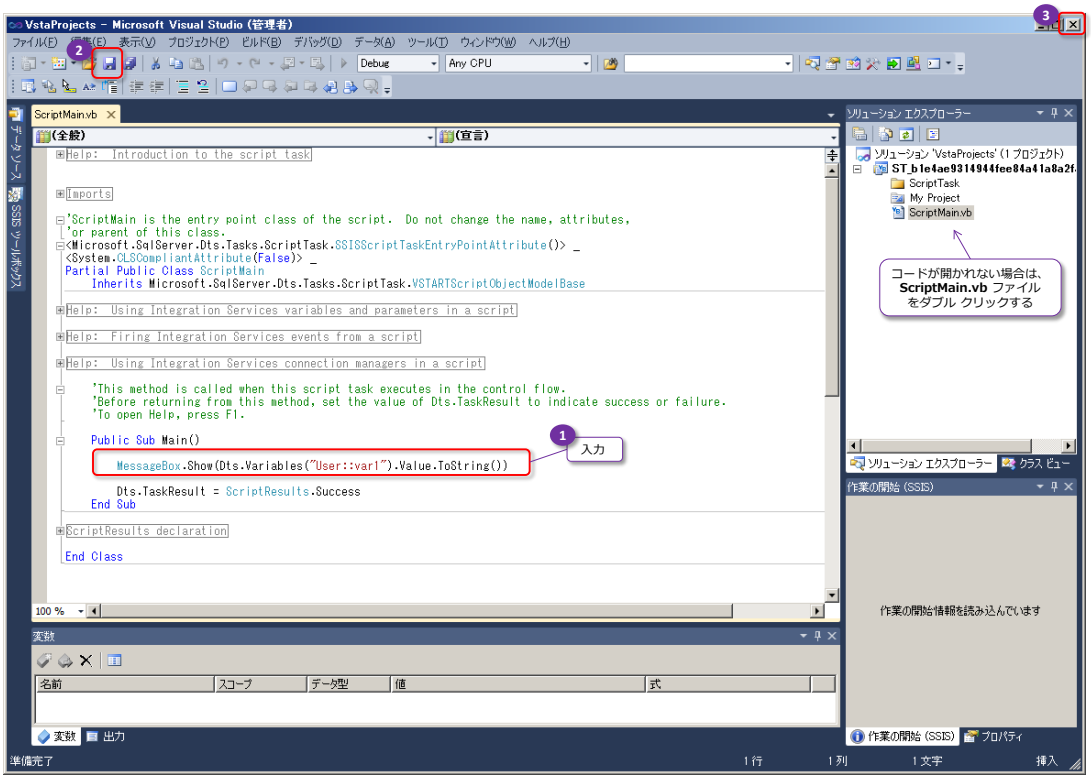


15. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**スクリプトの編集**］ボタンをクリックします。



16. ［**スクリプト エディター**］では、「**public Sub Main( )**」内のコードを、次のように変更します（変数名の大文字と小文字に注意してください）。

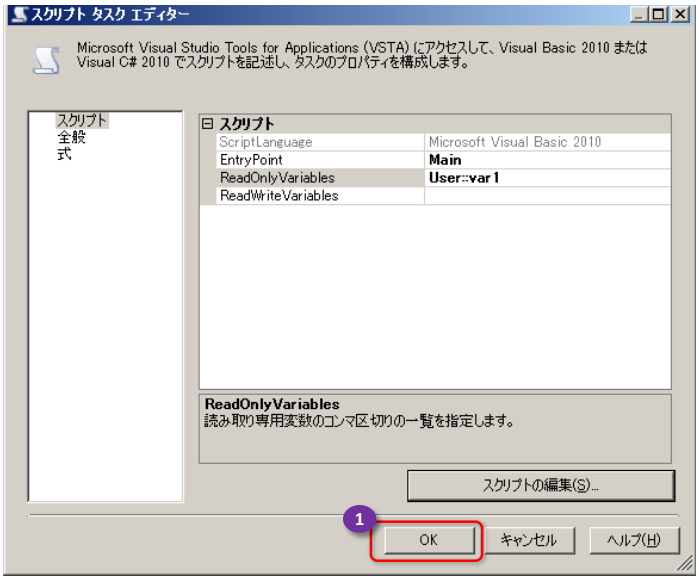
```
MessageBox.Show(Dts.Variables("User::var1").Value.ToString())
```



このコードにより、変数「**var1**」の値をメッセージ ボックスで表示できるようになります。

入力後、保存してスクリプト エディターを閉じます。

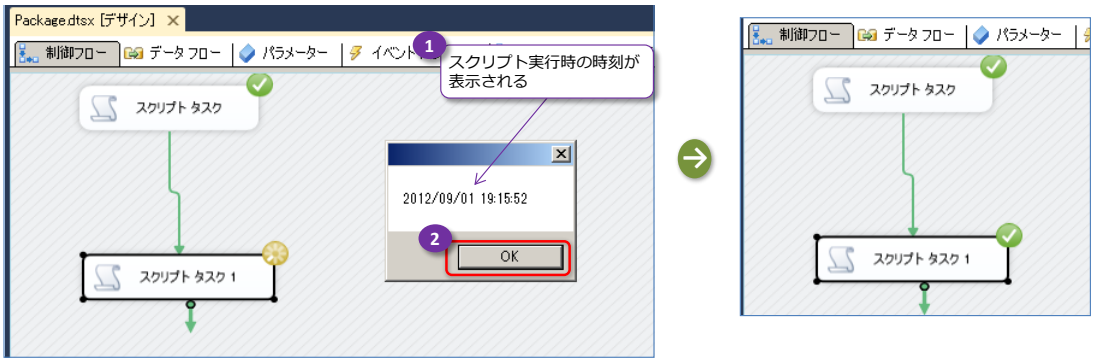
17. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。



パッケージの実行

18. 次に、ツールバーの［**デバッグ開始**］ボタンをクリックして、パッケージを実行します。

1 つ目の［**スクリプト タスク**］が緑色、2 つ目の［**スクリプト タスク 1**］が実行中に変わって、スクリプト タスクを実行した時の日時がメッセージ ボックスで表示されることを確認できます。



［**OK**］ボタンをクリックすると、［**スクリプト タスク 1**］に緑のチェックマークが付いて、スクリプトの実行が成功したことを確認できます。

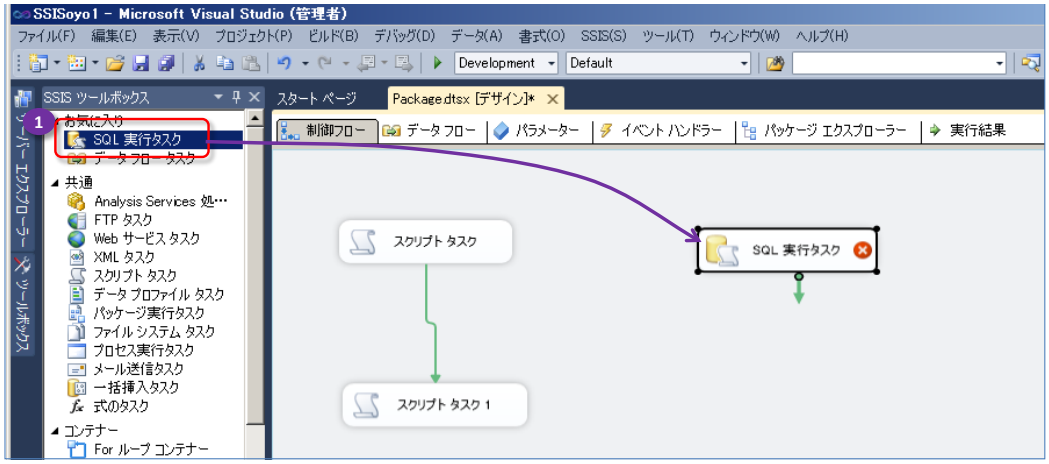
19. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。

2.4 SQL 実行タスクの結果を変数へ格納

SQL 実行タスクの結果を変数へ格納

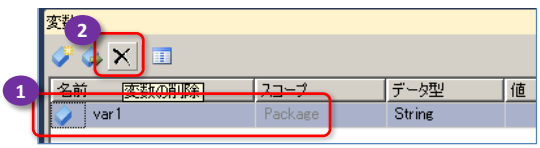
次に、「**SQL 実行タスク**」を利用して SQL を実行し、その結果を変数へ格納したり、その変数をデータ フロー タスクのパラメーターへ受け渡してデータ転送を実行してみましょう。

1. まずは、SSIS ツールボックスの［**お気に入り**］カテゴリから［**SQL 実行タスク**］を SSIS デザイナー上へドラッグ＆ドロップして配置します。

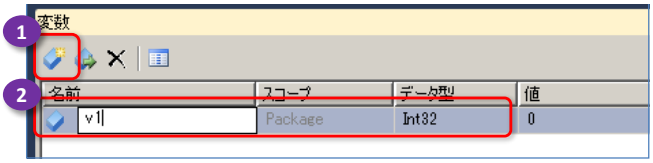


2. 次に、［**SSIS**］メニューから［**変数**］をクリックします。

3. ［**変数**］ウィンドウが表示されたら、前のステップで作成した変数「**var1**」を選択して、［**変数の削除**］ボタンをクリックします。

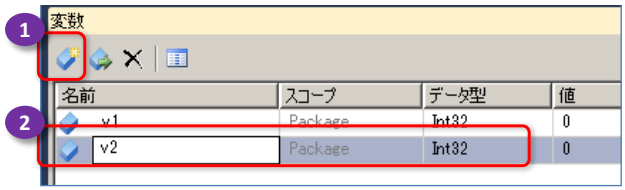


4. 続いて、［**変数の追加**］ボタンをクリックし、新しく変数を追加します。



［**名前**］には「**v1**」と入力して、［**データ型**］は既定値の「**Int32**」を選択します。**Int32** は、SQL Server での **int** 型に相当するデータ型です。

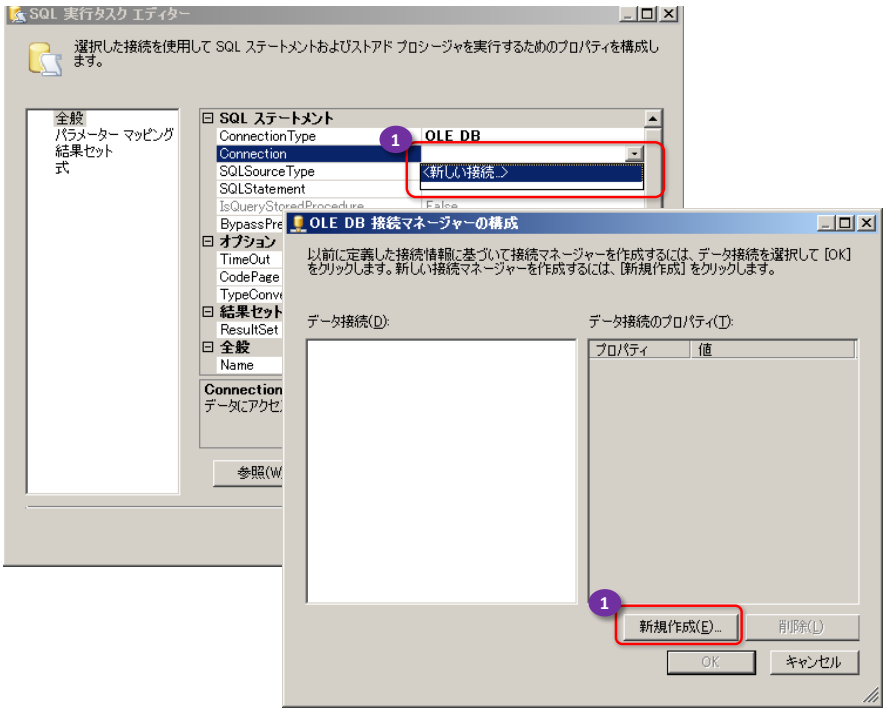
5. 次に、もう一度［**変数の追加**］ボタンをクリックして、もうひとつ変数を追加します。



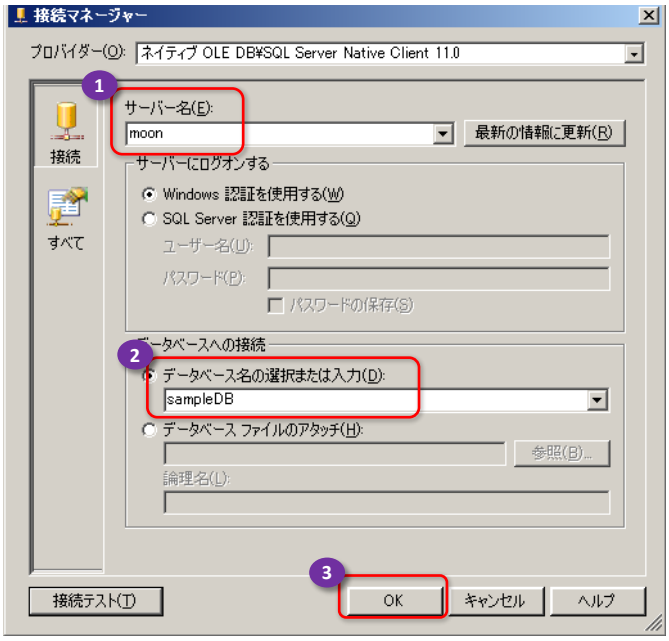
［**名前**］は「**v2**」とし、［**データ型**］は既定値の「**Int32**」を選択します。

6. 次に、SSIS デザイナーへ配置した［**SQL 実行タスク**］をダブル クリックします。

7. ［**SQL 実行タスク エディター**］ダイアログが表示されたら、［**Connection**］で「**新しい接続**」をクリックして、［**OLE DB 接続マネージャーの構成**］ダイアログを表示し、［**新規作成**］ボタンをクリックします。



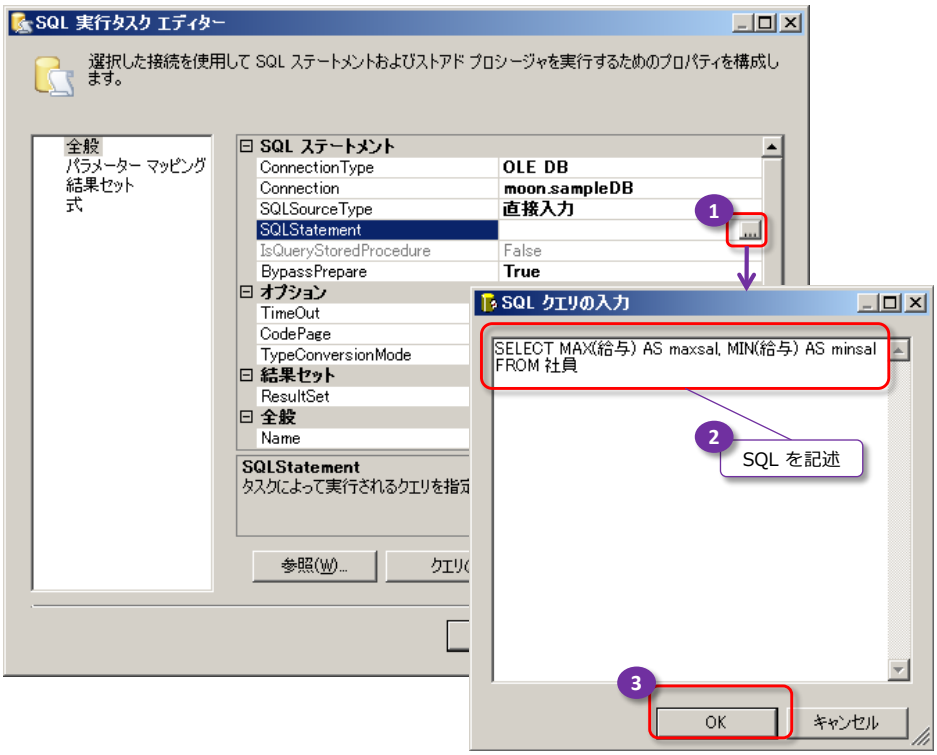
8. ［**接続マネージャー**］ダイアログが表示されたら、［**サーバー名**］へ SQL Server の名前（画面は **moon**）を入力し、［**データベース名の選択または入力**］で「**sampleDB**」データベースを選択して、［**OK**］ボタンをクリックします。



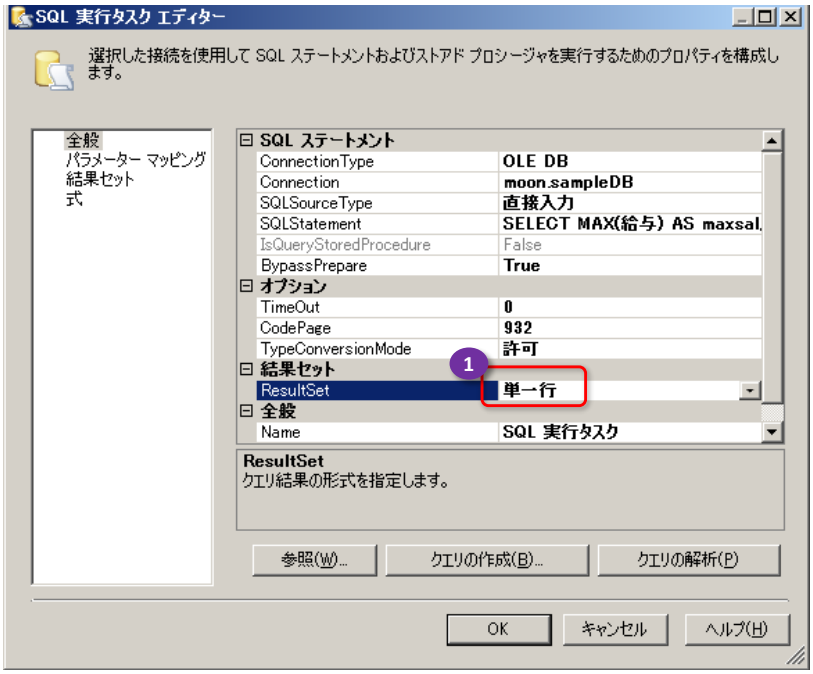
［**OLE DB 接続マネージャーの構成**］ダイアログへ戻ったら、［**OK**］ボタンをクリックします。

9. 次に、［**SQLStatement**］の［**...**］ボタンをクリックして、［**SQL クエリの入力**］ダイアログを表示し、次のように SQL を入力します。この SQL は、sampleDB データベース内にある「**社員**」テーブルから、「**給与**」の**最大値**と**最小値**を取得するものです。

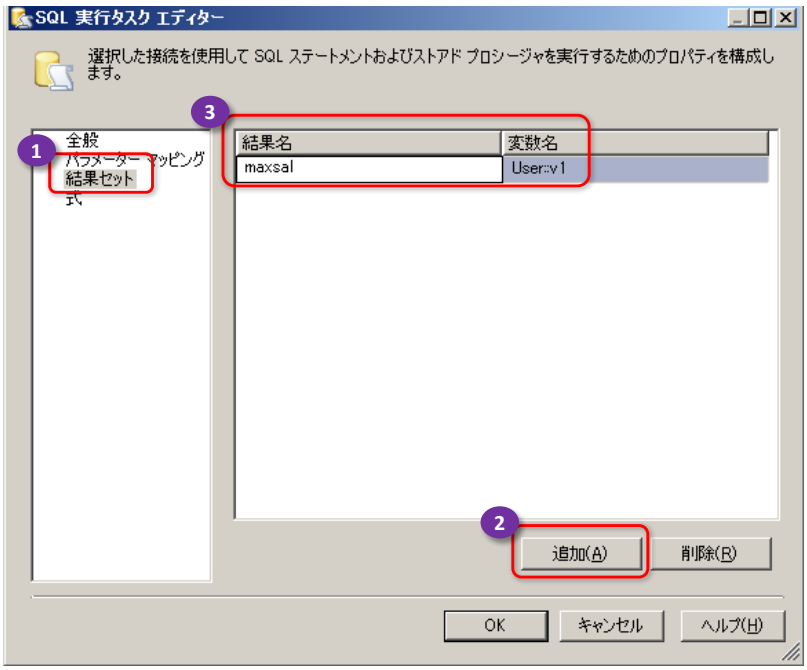
```
SELECT MAX(給与) AS maxsal, MIN(給与) AS minsal FROM 社員
```



10. 続いて、［**ResultSet**］で、クエリ結果の形式を指定します。このクエリの結果は、1 行のみを返すので、「**単一行**」を選択します。

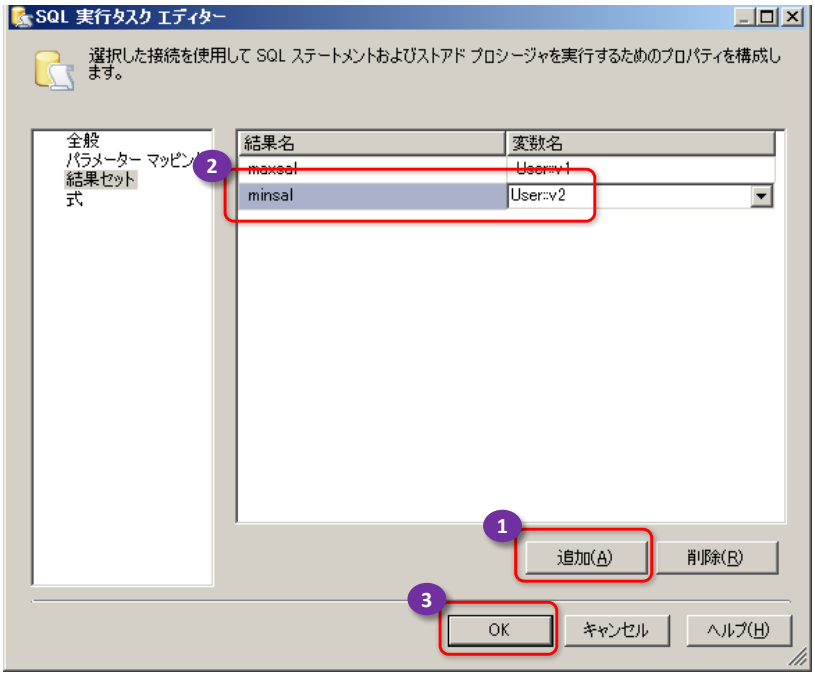


11. 次に、[**結果セット**] ページをクリックして開きます。[**追加**] ボタンをクリックして、[**結果名**] へ給与の最大値の列名である「**maxsal**」を入力し、[**変数名**] で「**User::v1**」が選択されていることを確認します。



これにより、SELECT ステートメントで取得した maxsal（**MAX(給与)**）の値を変数 **v1** へ格納できるようになります。

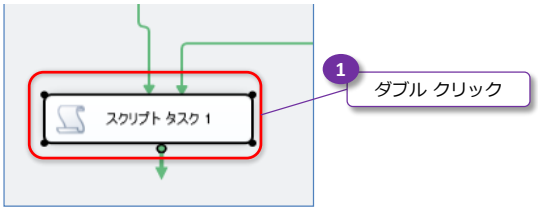
12. 続いて、もう一度 [**追加**] ボタンをクリックして、[**結果名**] へ給与の最小値の列名である「**minsal**」を入力し、[**変数名**] で「**User::v2**」を選択して、[**OK**] ボタンをクリックします。



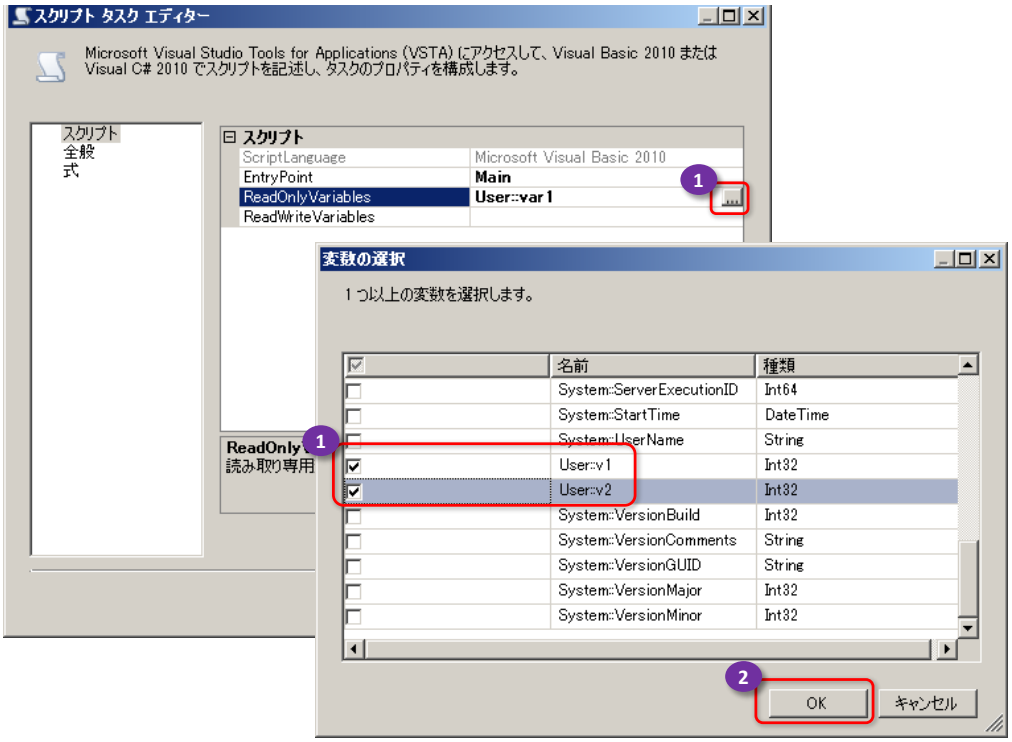
13. 次に、1 つ目の［**スクリプト タスク**］を右クリックして、［**無効化**］をクリックし、［**スクリプト タスク**］を無効化しておきます。

14. 続いて、［**SQL 実行タスク**］の**緑色の矢印**を［**スクリプト タスク 1**］までドラッグ＆ドロップして伸ばします。

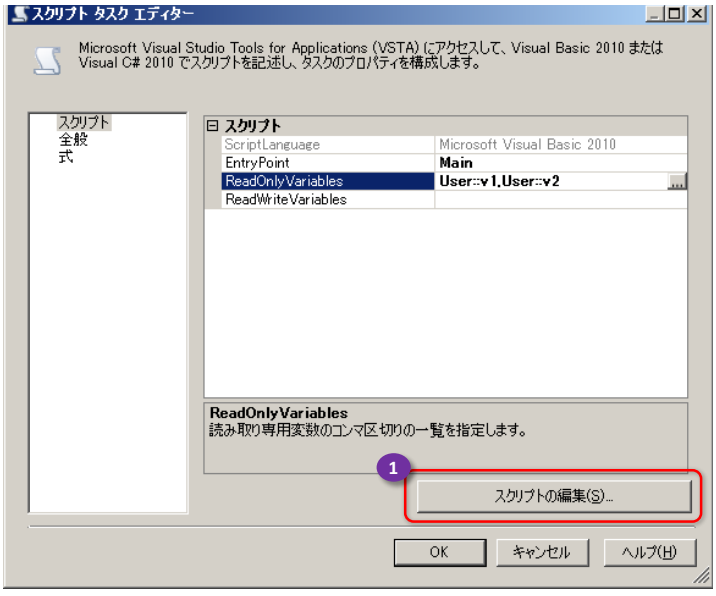
15. 次に、［**スクリプト タスク 1**］をダブルクリックします。



16. ［**スクリプト タスク エディター**］ダイアログが表示されたら、［**ReadOnlyVariables**］で［**…**］ボタンをクリックして、［**変数の選択**］ダイアログを表示し、追加した「**User::v1**」と「**User::v2**」の変数をチェックして、［**OK**］ボタンをクリックします。

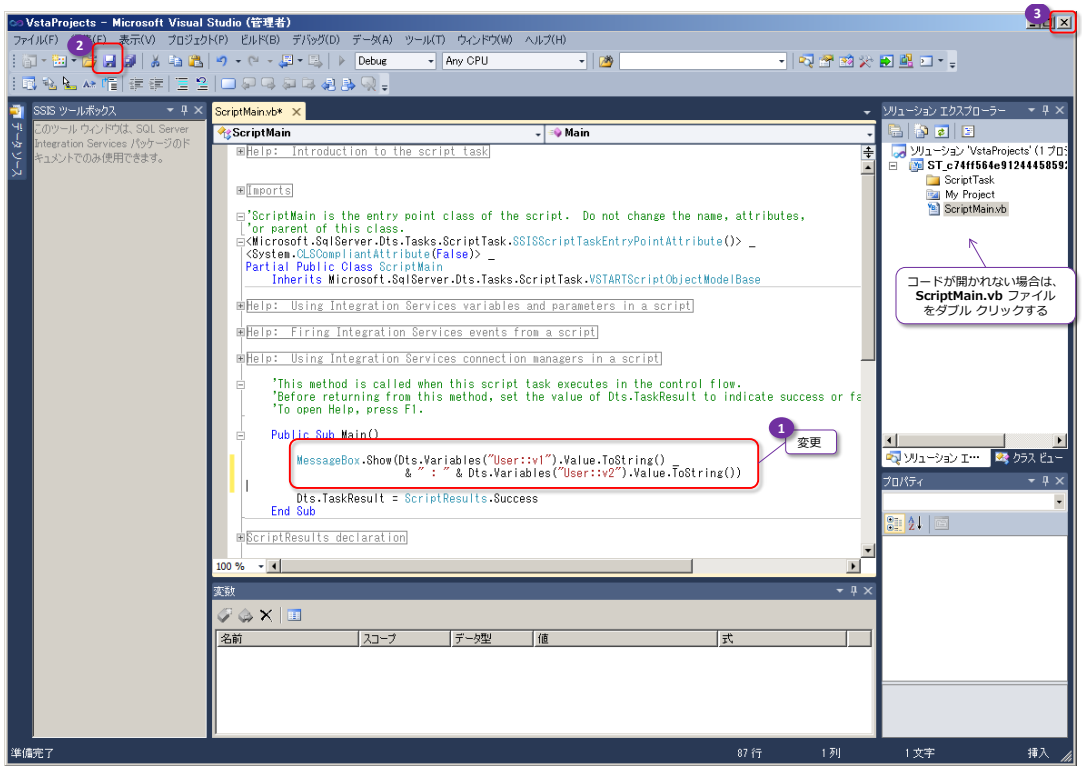


17. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**スクリプトの編集**］ボタンをクリックします。



18. ［**スクリプト エディター**］では、「**public Sub Main( )**」内のコードを、次のように変更します（変数名の、大文字と小文字に注意してください）。

```
MessageBox.Show(Dts.Variables("User::v1").Value.ToString() _
                        & " : " & Dts.Variables("User::v2").Value.ToString())
```



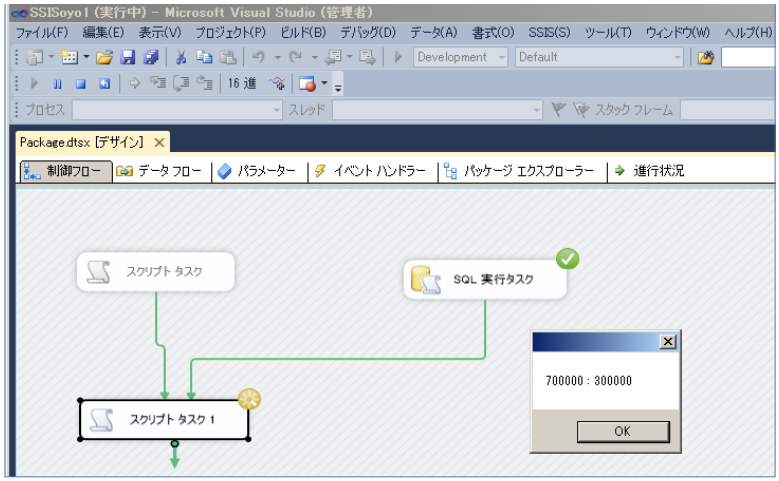
このコードにより、変数「**v1**」と「**v2**」の値をメッセージ ボックスで表示できるようになります。入力後、保存してスクリプト エディターを閉じます。

19. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。

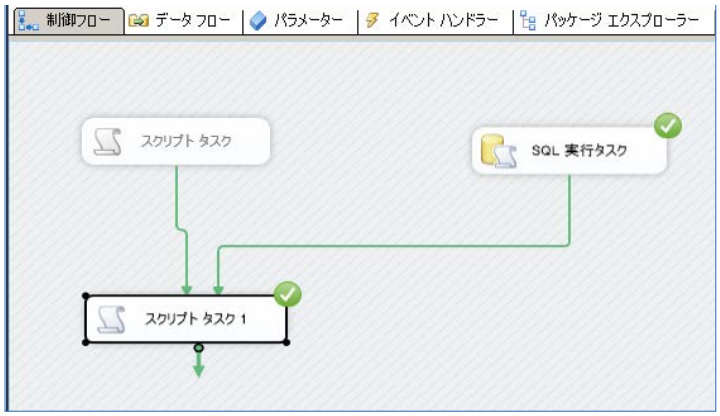
パッケージの実行

20. 次に、スクリプト タスクを実行するために、ツールバーの［**デバッグ開始**］ボタンをクリックして、パッケージを実行します。

次のように、［**SQL 実行タスク**］に緑のチェックマーク、［**スクリプト タスク 1**］が実行中へ変わって、社員テーブルの給与の最大値「**700000**」と最小値「**300000**」がメッセージ ボックスで表示されることを確認できます。



［**OK**］ボタンをクリックすると、［**スクリプト タスク 1**］に緑のチェックマークが付いて、スクリプトの実行が成功したことを確認できます。



このように、SQL 実行タスクで実行した結果は、変数へ格納することができ、それはスクリプト タスクなどパッケージ内で再利用していくことができます。

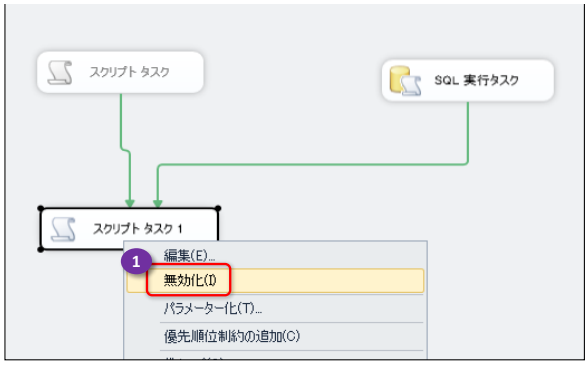
21. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。

2.5 SQL ステートメントのパラメーター化と変数の引き渡し

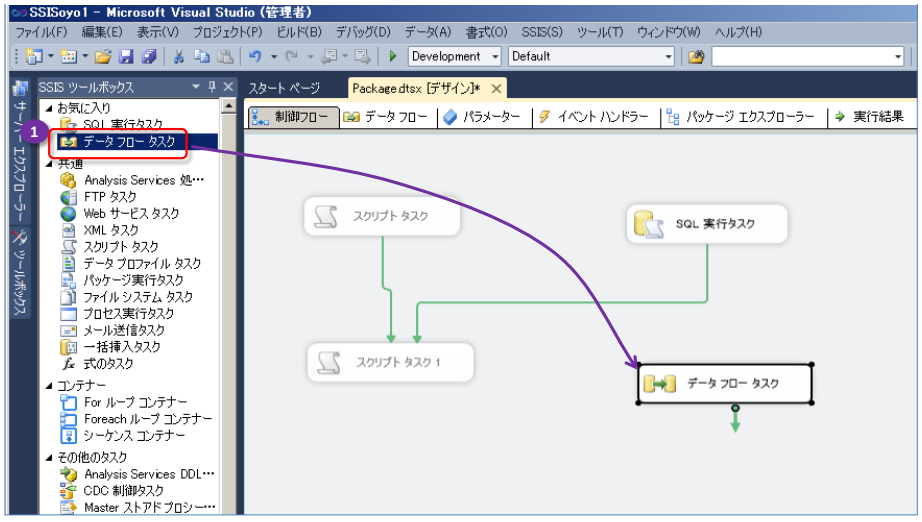
SQL ステートメントのパラメーター化と変数の引き渡し

次に、SQL 実行タスクの結果を格納した変数を、データ フロー タスク内の SQL ステートメントの「**パラメーター**」へ引き渡して、実行できるようにしてみましょう。

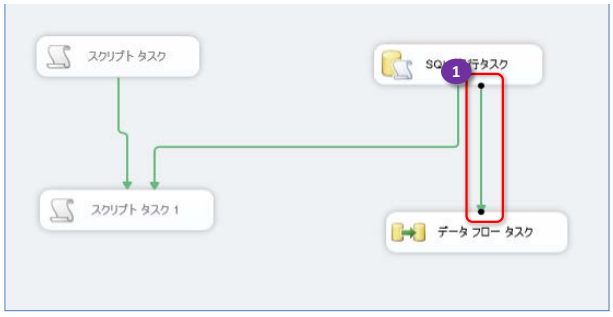
1. まずは、[**スクリプト タスク 1**] を右クリックして、[**無効化**] をクリックし、[**スクリプト タスク 1**] を無効にしておきます。



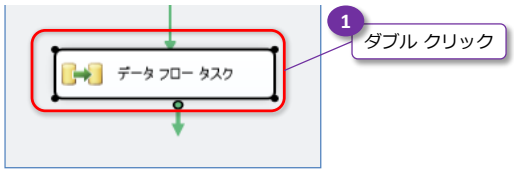
2. 次に、SSIS ツールボックスを開いて、[**お気に入り**] カテゴリから [**データ フロー タスク**] をドラッグ＆ドロップして SSIS デザイナー上へ配置します。



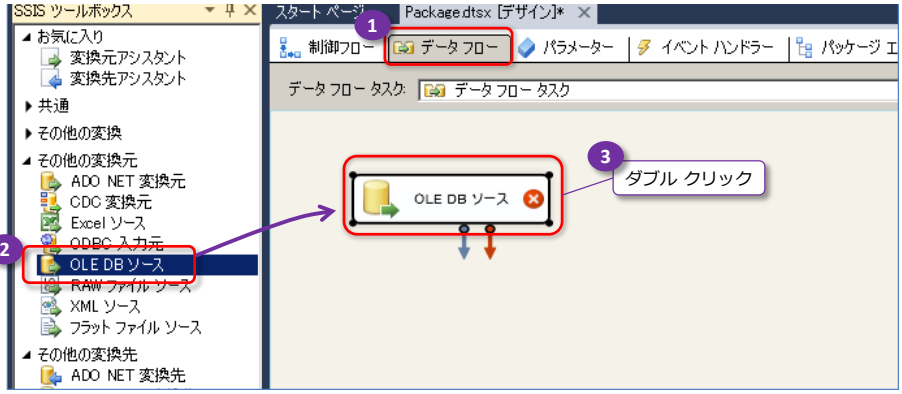
3. 次に、[**SQL 実行タスク**] の**緑色の矢印**を、[**データ フロー タスク**] まで、ドラッグ＆ドロップして伸ばします。



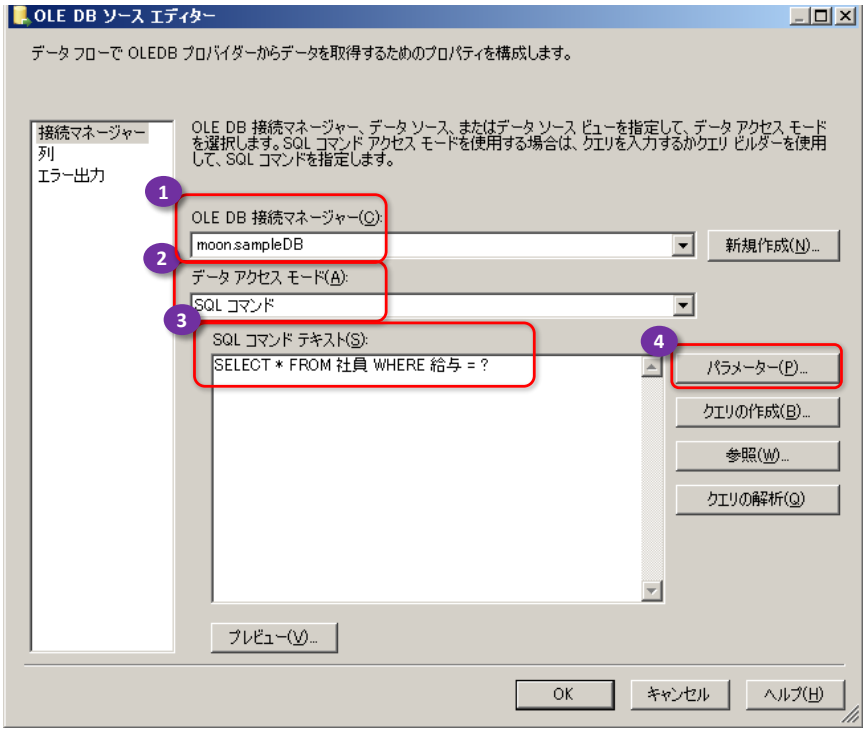
4. 続いて、[**データ フロー タスク**] をダブル クリックして、[**データ フロー**] タブを表示します。



5. [**データ フロー**] タブが表示されたら、SSIS ツールボックスの [**その他の変換元**] カテゴリから [**OLE DB ソース**] を SSIS デザイナー上へドラッグ＆ドロップして配置します。



配置した [**OLE DB ソース**] をダブル クリックして、[**OLE DB ソース エディター**] を開き、[**OLE DB 接続マネージャー**] で「**サーバー名.sampleDB**」が選択されていることを確認します。



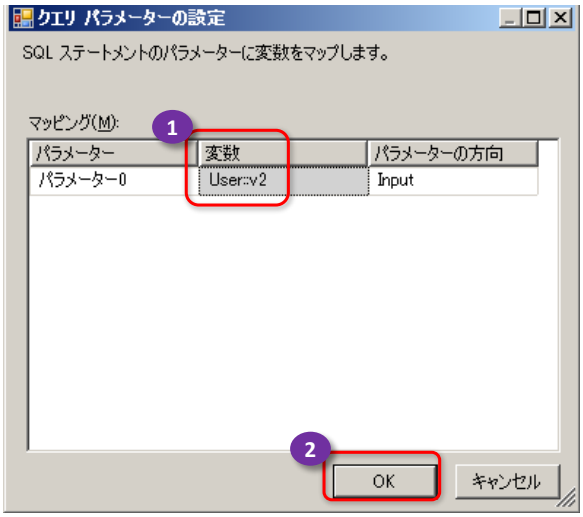
[**データ アクセス モード**] では、「**SQL コマンド**」を選択します。これにより、[**SQL コマンド テキスト**] が表示され、SQL が入力できるようになるので、次のように SQL を記述します。

```
SELECT * FROM 社員 WHERE 給与 = ?
```

「**給与 = ?**」と記述することで、パラメーターが 1 つ（**パラメーター0**）自動作成されます。

SQL を記述後、［**パラメーター**］ボタンをクリックします。

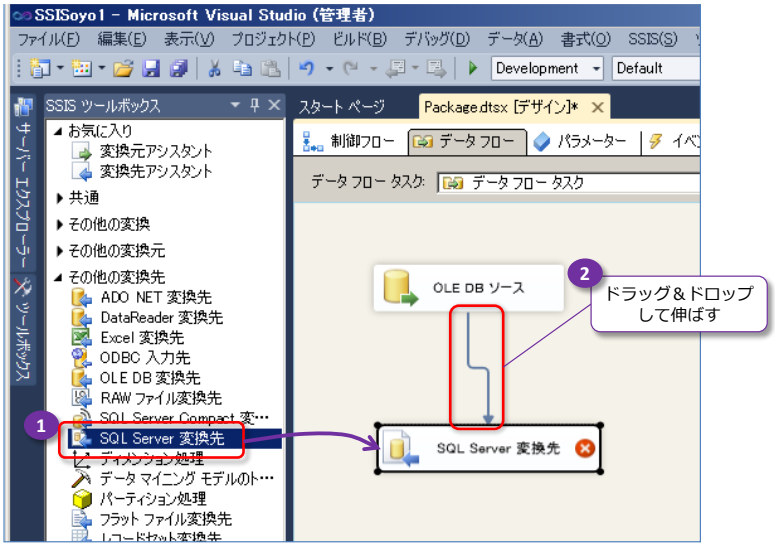
6. ［**クエリ パラメーターの設定**］ダイアログが表示されたら、自動作成された「**パラメーター0**」の［**変数**］で「**User::v2**」変数を選択して割り当てます。



これで、「**給与 = ?**」の **?** へ代入する値を変数 **v2**（最小給与が格納されている変数）にすることができます。

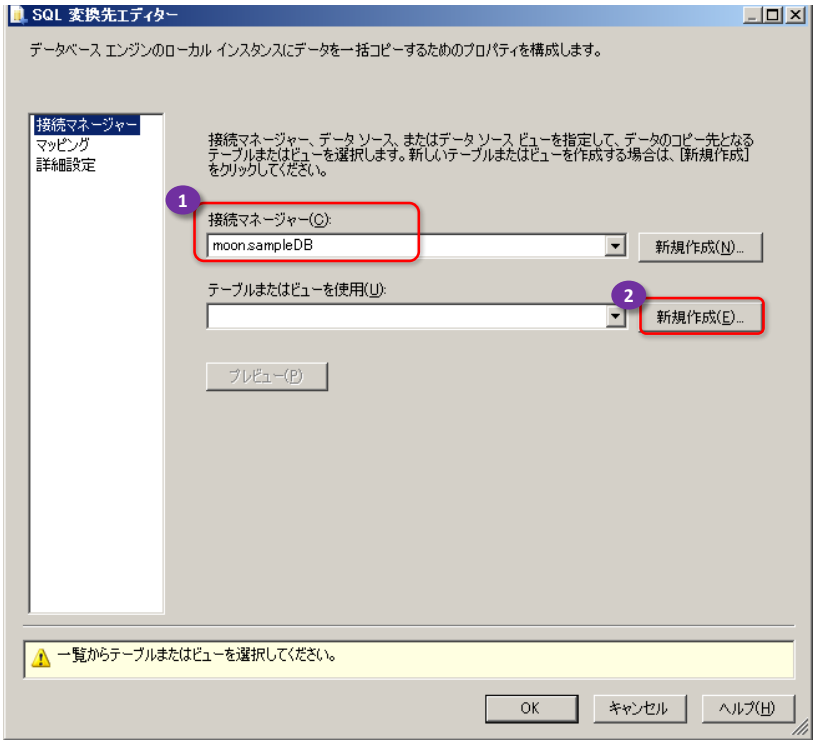
設定後、［**OK**］ボタンをクリックして、ダイアログを閉じます。［**OLE DB ソース エディター**］へ戻ったら、［**OK**］ボタンをクリックして閉じます。

7. 次に、SSIS ツール ボックスの［**その他の変換先**］カテゴリから［**SQL Server 変換先**］を SSIS デザイナー上へドラッグ＆ドロップして配置します。



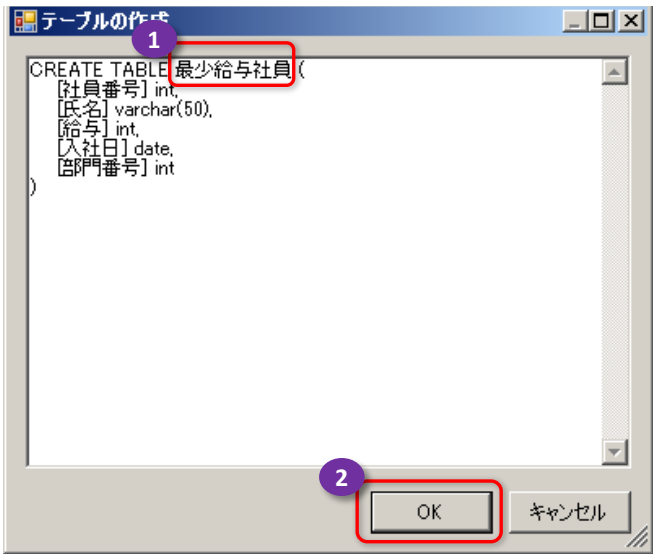
続いて、［**OLE DB ソース**］の**青色の矢印**を［**SQL Server 変換先**］まで、ドラッグ＆ドロップして伸ばします。

8. 次に、[**SQL Server 変換先**] をダブル クリックして、[**SQL 変換先エディター**] を開き、[**接続マネージャー**] で「**サーバー名.samplDB**」が選択されていることを確認します。

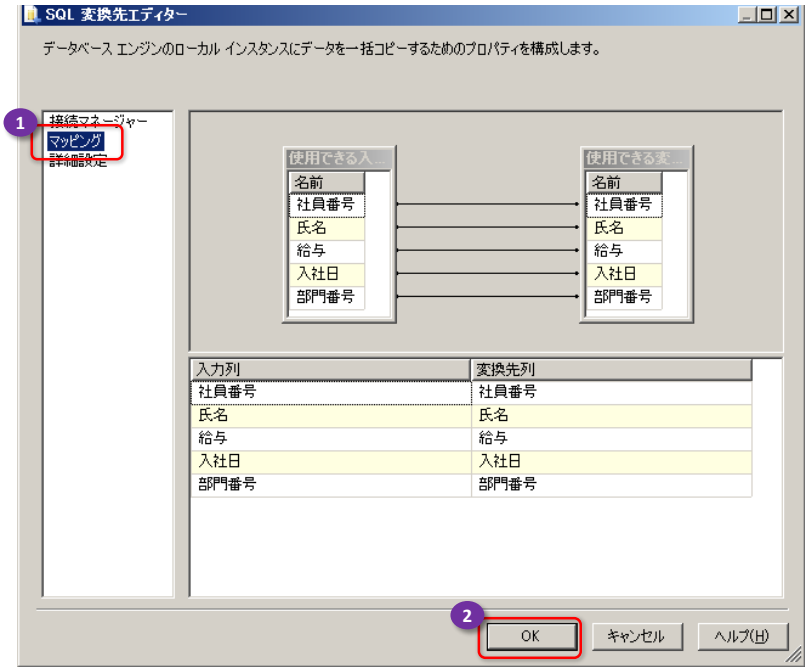


[**テーブルまたはビューを使用**] で [**新規作成**] ボタンをクリックします。

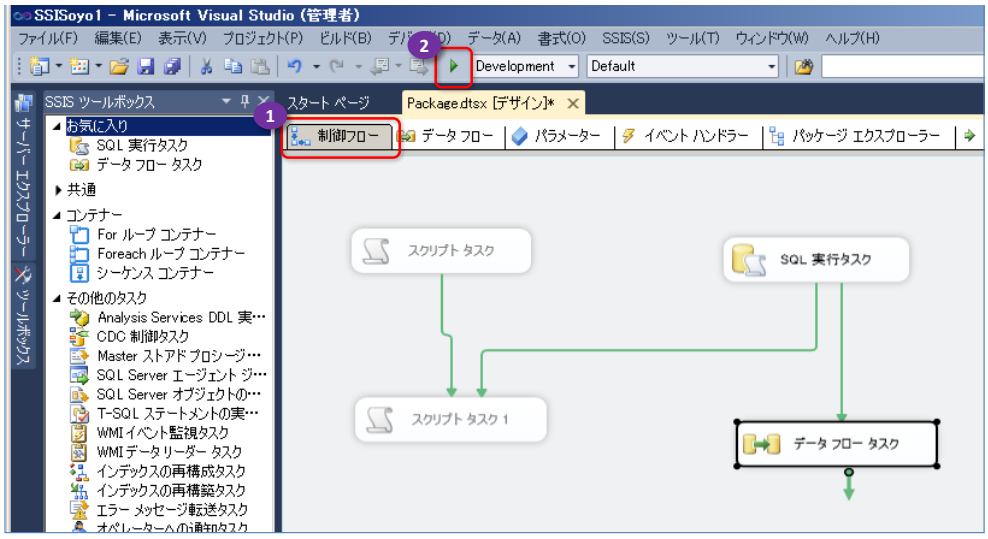
9. これにより、[**テーブルの作成**] ダイアログが表示されるので、テーブル名を「**最小給与社員**」へ変更して、[**OK**] ボタンをクリックします。



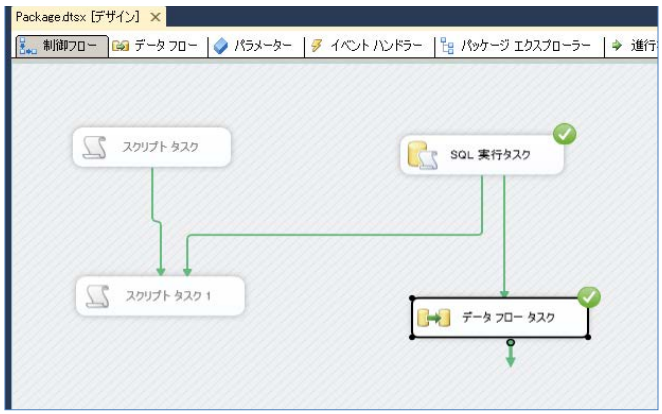
10. [**SQL 変換先エディター**] へ戻ったら、[**マッピング**] ページをクリックして開き、[**OK**] ボタンをクリックします。



11. 次に、[**制御フロー**] タブをクリックして開いてから、ツールバーの [**デバッグ開始**] ボタンをクリックして、パッケージを実行します。



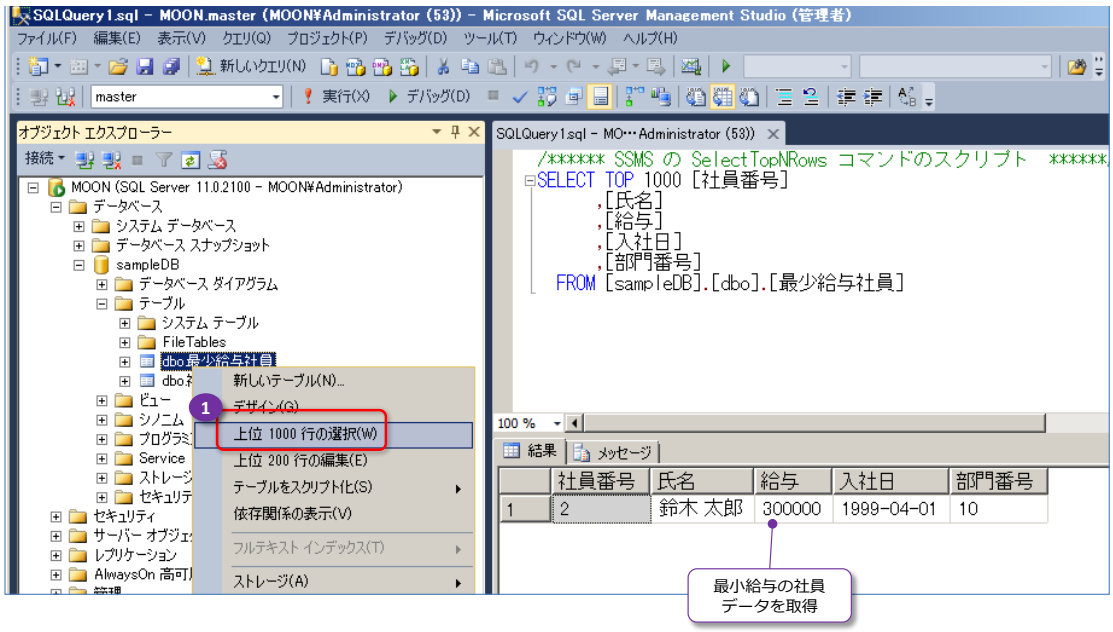
[**SQL 実行タスク**] と [**データ フロー タスク**] に緑のチェックマークが付いて、パッケージの実行が成功したことを確認できます。



12. 確認後、ツールバーの［**デバッグの停止**］ボタンをクリックして、デバッグを終了します。

転送されたデータの確認

13. 転送されたデータを確認するには、Management Studio から、［**最小給与社員**］テーブルを右クリックして、［**上位 1000 行の選択**］をクリックし、データを確認します。

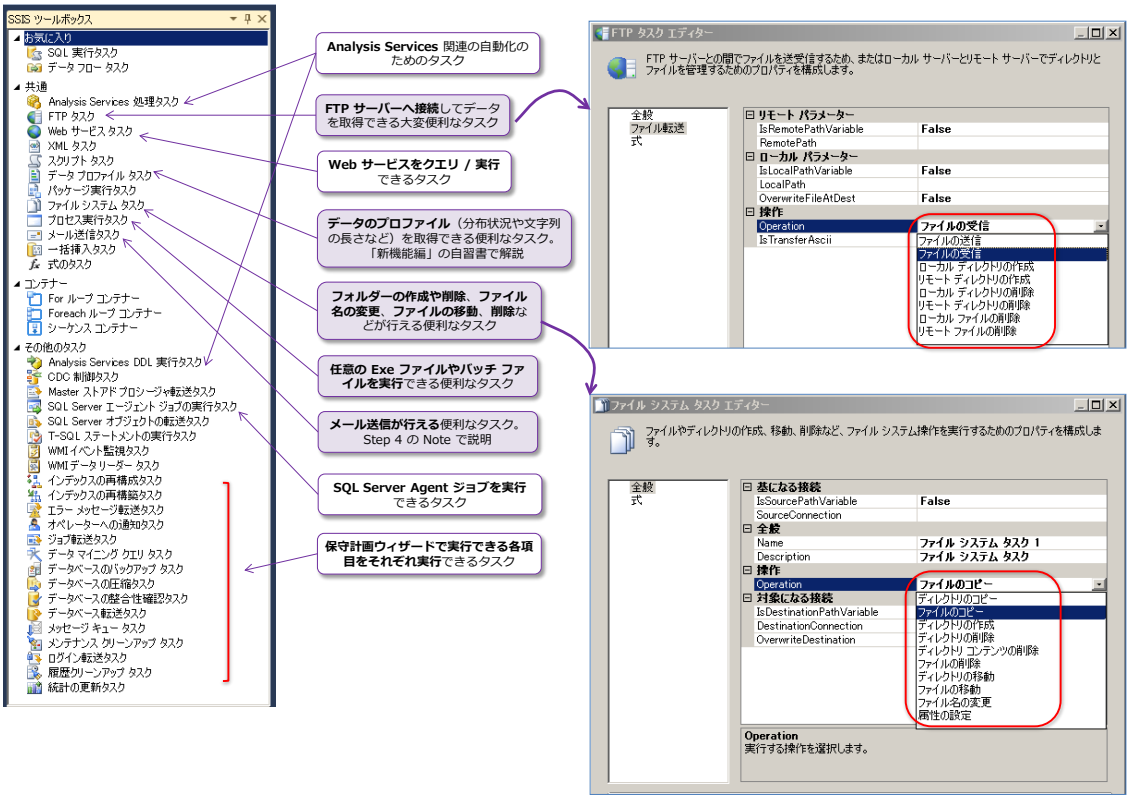


給与が最小の社員が格納されていることを確認できます。

このように、データ フロー タスク内の SQL ステートメントは、パラメーター化することができ、また、それに対して変数を代入して実行することができます。

Note： そのほかの便利なタスク

Integration Services には、そのほかにも便利なタスクが多くあります。その代表的なものは、次のとおりです。



中でも、FTP サーバーへ接続して、データを取得できる「**FTP タスク**」と、ファイル名の変更や削除などが行える「**ファイル システム タスク**」、任意の EXE やバッチ ファイルを実行できる「**プロセス実行タスク**」、メールを送信できる「**メール送信タスク**」などは、大変便利なので、オンライン ブックなどを参考にぜひ活用してみてください（メール送信タスクについては、Step4 の Note でも説明してします）。

STEP 3. ログ記録

この STEP では、パッケージの実行状態（実行時間やエラーなど）をログへ記録する方法について説明します。

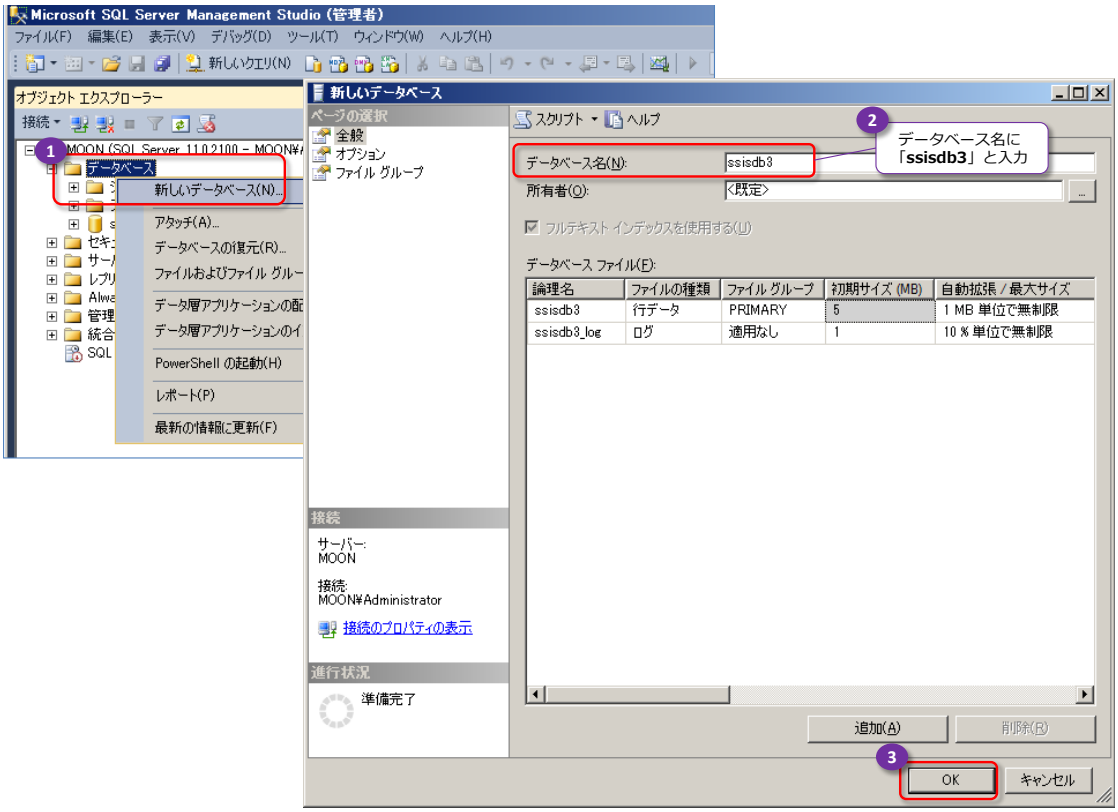
この STEP では、次のことを学習します。

- ✓ ログ記録の設定
- ✓ ログ記録の詳細設定
- ✓ エラーが発生した場合のログ記録の確認

3.1 実習を始める前に

転送先となるデータベースの作成

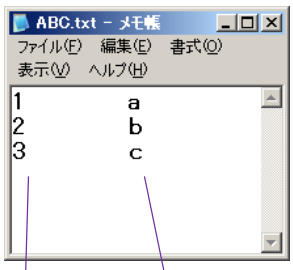
この Step では、SSIS デザイナーを使用して、テキスト データを SQL Server 内へ取り込んでいくので、取り込み先（転送先）となる空のデータベースを作成しておきます。次のように Management Studio で［**データベース**］フォルダーを右クリックして、［**新しいデータベース**］をクリックし、「**ssisdb3**」という名前のデータベースを作成します。



インポートするファイル

インポートするファイル（**ABC.txt** と **MISS.txt**）は、次のように、タブ区切りのファイルで、サンプル スクリプトの **import** フォルダー内へ格納してあります。

正常に転送されるファイル（ABC.txt）



数値列　文字列

エラーが発生するファイル（MISS.txt）

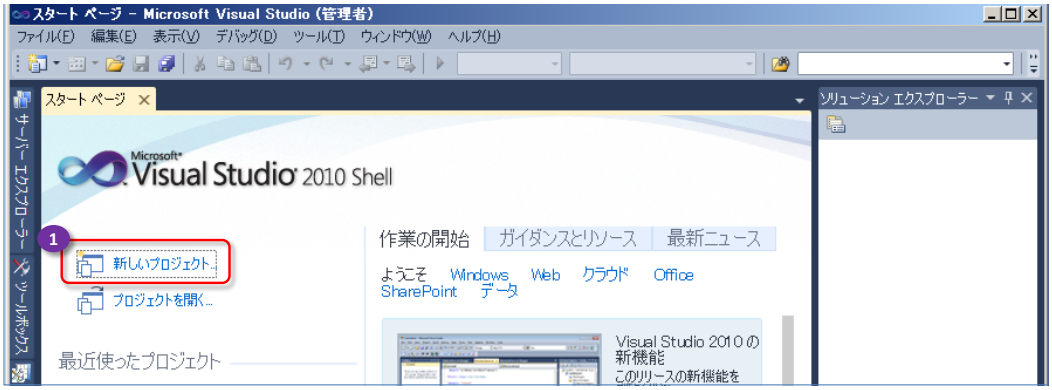
文字と数値が混在している列　文字列

3.2 ログ記録の設定

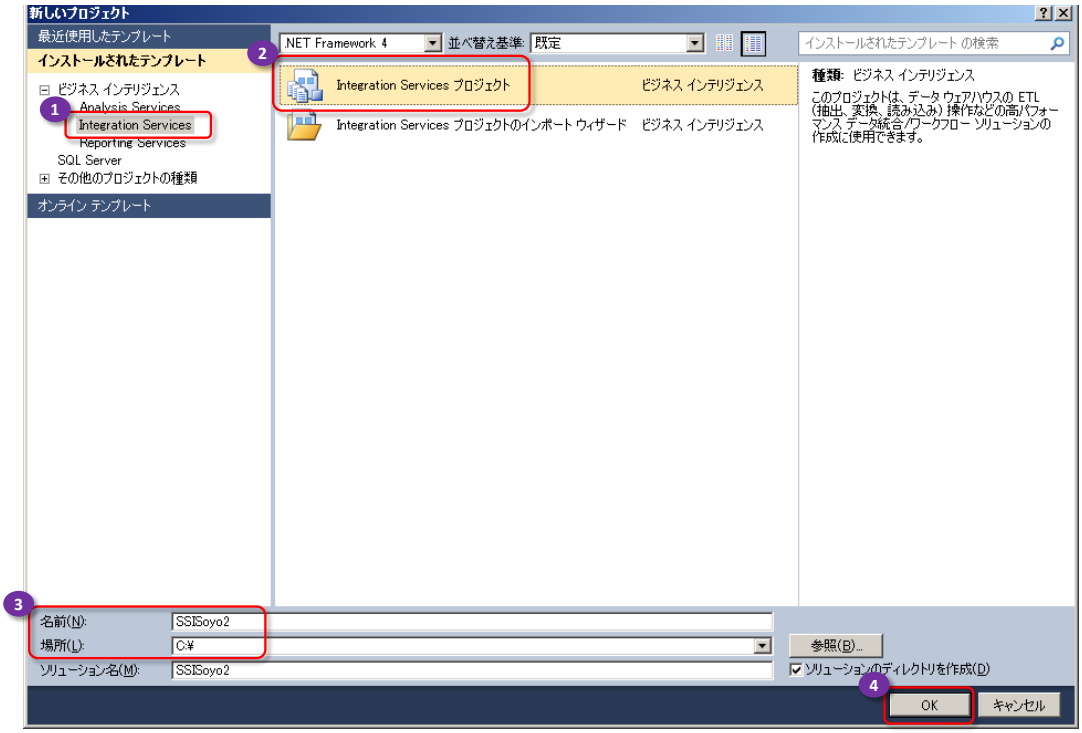
ログ記録の設定

まずは、テキスト ファイル（**ABC.txt**）を SQL Server へインポートするパッケージを作成して、実行時にログが記録される設定にしてみましょう。

1. SQL Server Data Tools を起動して、[**スタート ページ**] の [**新しいプロジェクト**] をクリックし、新しいプロジェクトを作成します。



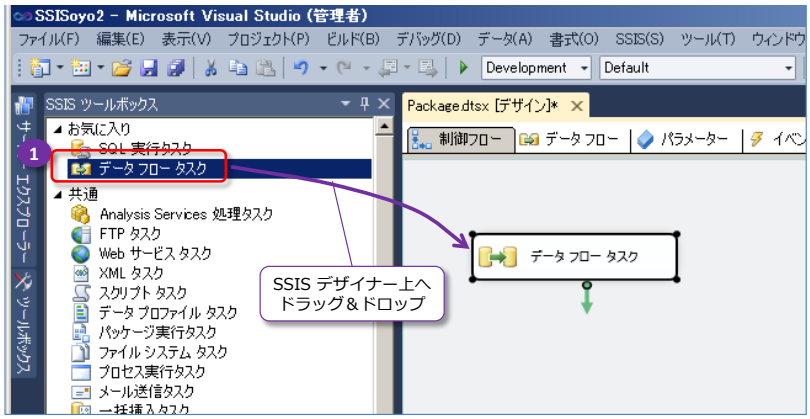
2. [**新しいプロジェクト**] ダイアログでは、[**インストールされたテンプレート**] で「**ビジネス インテリジェンス プロジェクト**」の [**Integration Services**] をクリックし、「**Integration Services プロジェクト**」を選択します。



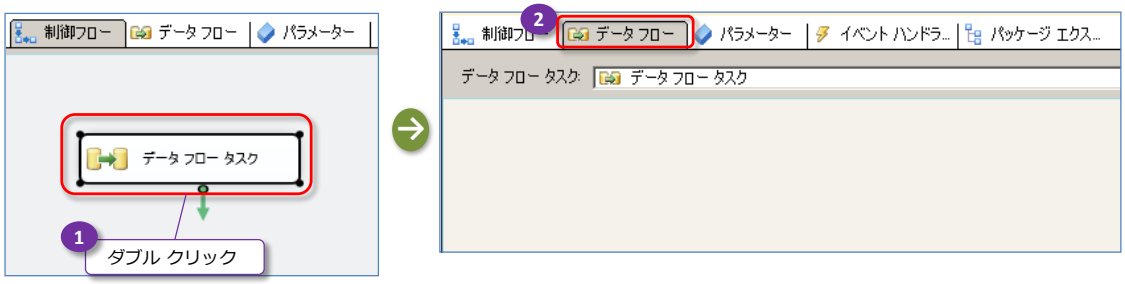
[**名前**] へ任意のプロジェクト名（画面は **SSISoyo2**）、[**場所**] へ任意の保存場所（画面は **C:¥**）を入力して、[**OK**] ボタンをクリックします。

データ フロー タスクの追加

3. 次に、データ転送を設定するために、SSIS ツールボックスの [**お気に入り**] カテゴリから [**データ フロー タスク**] を SSIS デザイナー上へドラッグ&ドロップします。

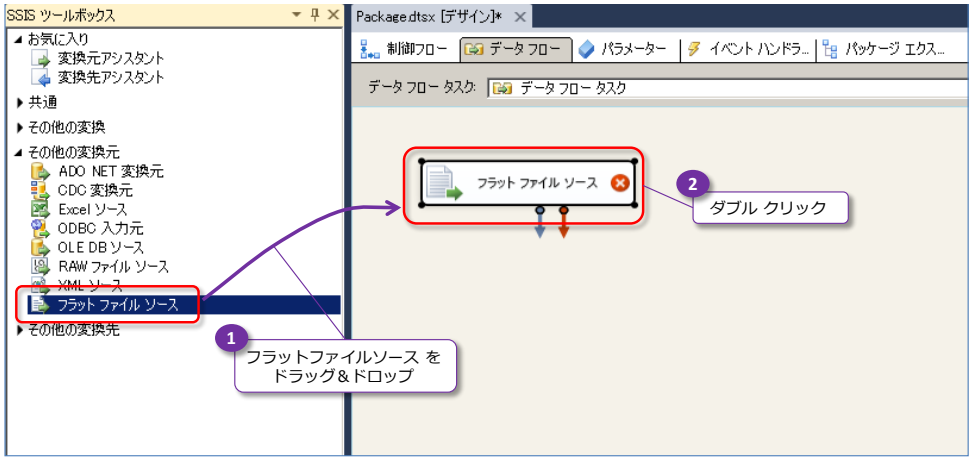


4. 配置した [**データ フロー タスク**] をダブル クリックして、[**データ フロー**] タブを表示します。



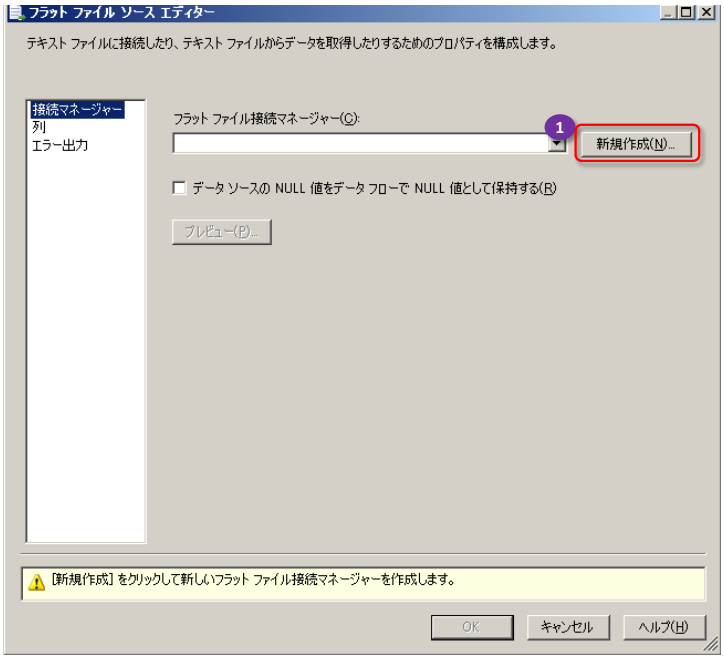
転送元テキスト ファイルの指定

5. 次に、転送元データとしてテキスト ファイルを指定するため、SSIS ツールボックスの [**その他の変換元**] カテゴリから、[**フラット ファイル ソース**] を SSIS デザイナー上へドラッグ&ドロップします。



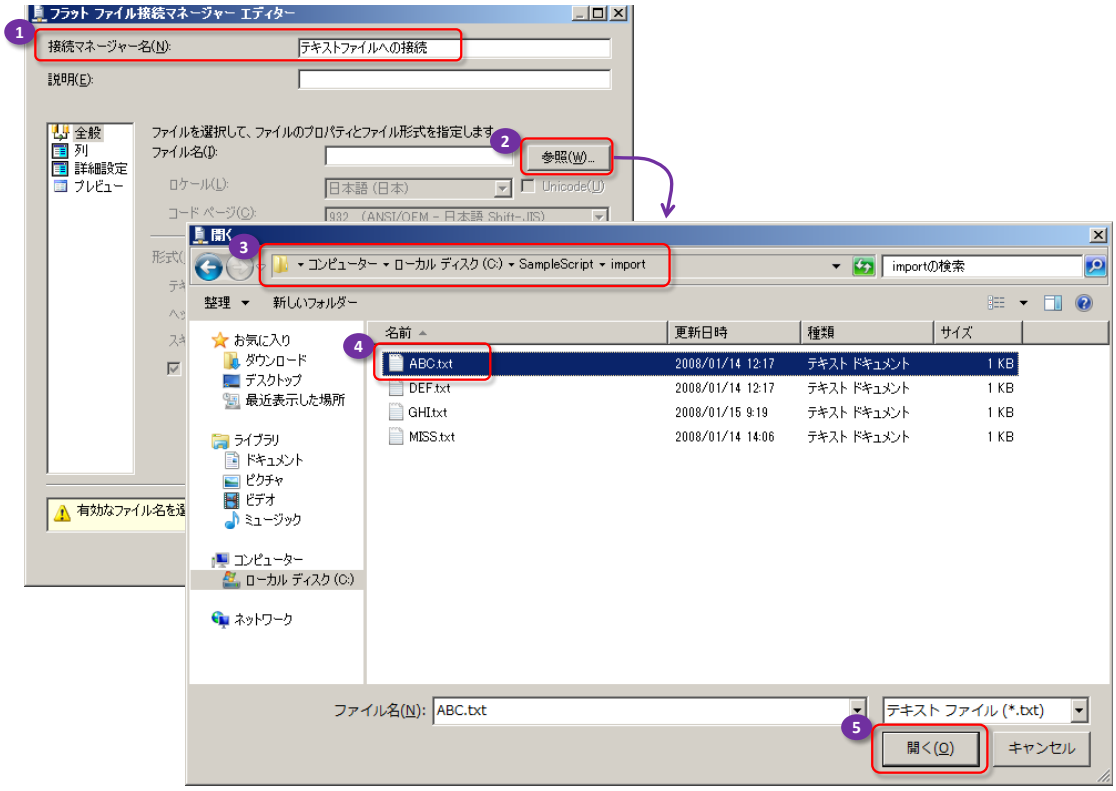
6. 配置した [**フラット ファイル ソース**] をダブル クリックして、[**フラット ファイル ソース**

エディター］ダイアログを表示します。



このダイアログでは、転送元のテキスト ファイルを指定するために［**新規作成**］ボタンをクリックします。

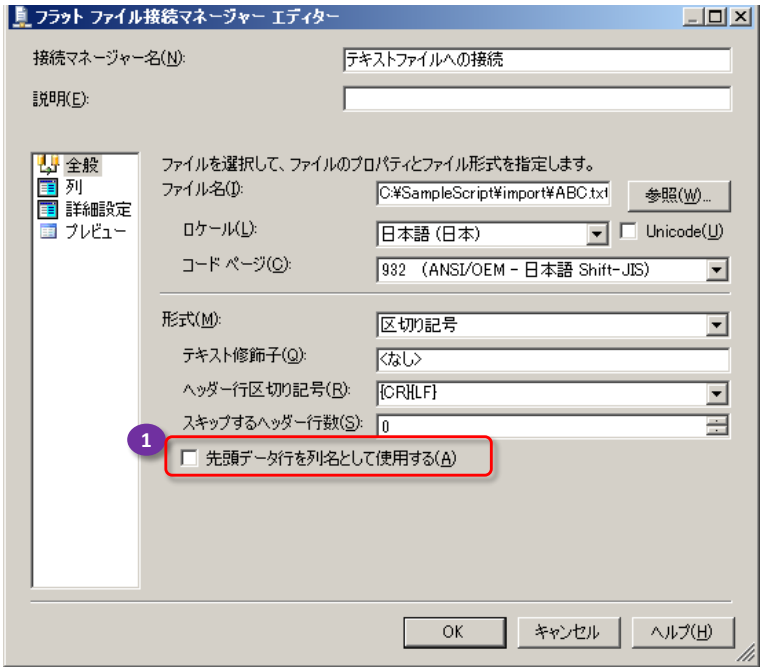
7. これにより、［**フラット ファイル接続マネージャー エディター**］ダイアログが表示されるので、［**接続マネージャー名**］へ任意の名前（**テキストファイルへの接続** など）と入力し、［**参照**］ボタンをクリックします。



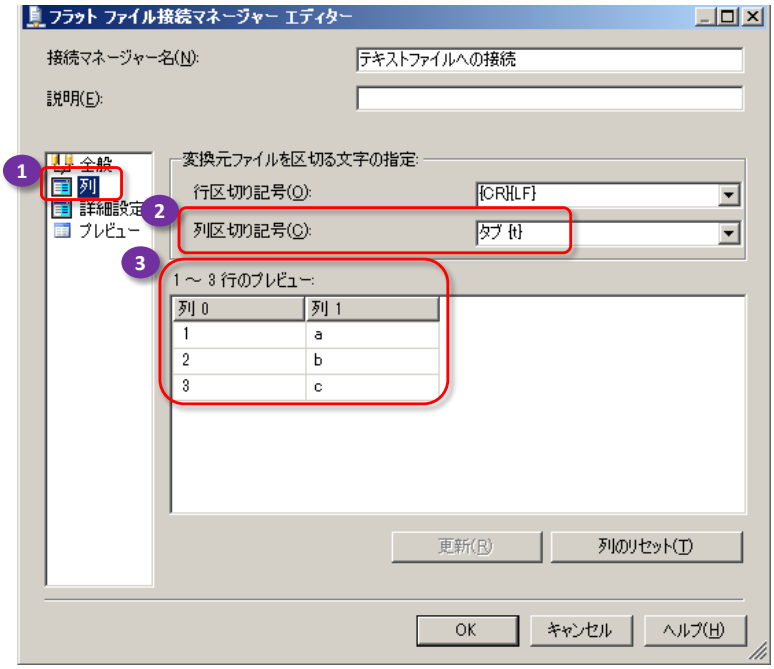
［**開く**］ダイアログが表示されたら、サンプル スクリプト内の「**import**」フォルダー内にあ

る「**ABC.txt**」ファイルを選択して、[**開く**] ボタンをクリックします。

8. [**フラット ファイル接続マネージャー エディター**] ダイアログへ戻ったら、[**先頭データ行を列名として使用する**] のチェックを外します。

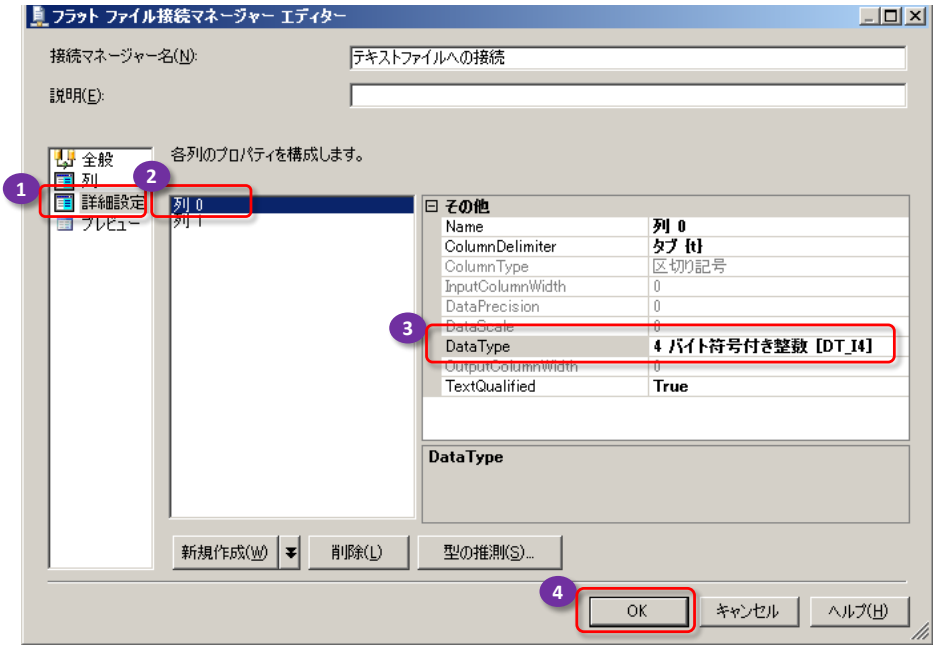


9. 続いて、「**列**」ページをクリックして、[**列区切り記号**] で「**タブ {t}**」が選択されていることを確認します（**ABC.txt** ファイルはタブ区切りのファイルです）。



[**1～3 行のプレビュー**] で転送元のデータを確認します。このファイルには、列見出しの行がないので、列名が自動的に「**列 0**」、「**列 1**」と設定されています。

10. 次に、[**詳細設定**] ページをクリックして開きます。

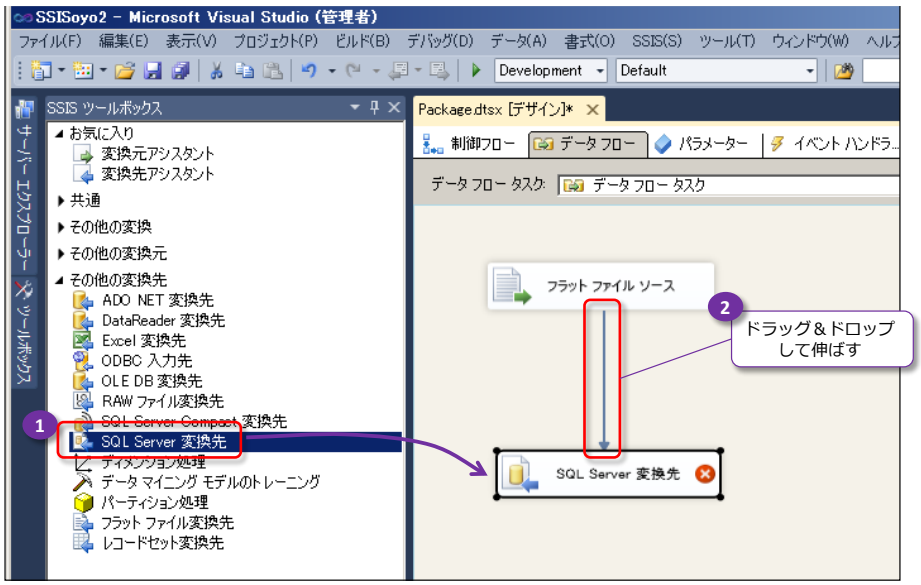


テキスト ファイルの場合、デフォルトでは「**文字列**」（**DT_STR : string**）データ型としてデータが転送されますが、今回は、「**列 0**」列（1 番左の列）のデータを数値として転送するように、「**列 0**」列の［**DataType**］を「**4 バイト符号付き整数**」（SQL Server での **int** 型に相当するデータ型）へ変更します。変更後、［**OK**］ボタンをクリックします。

［**フラット ファイル ソース エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。

転送先データベースの指定

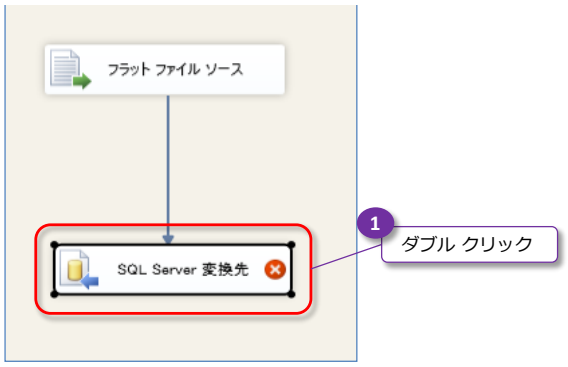
11. 次に、転送先のデータベースを指定するために、SSIS ツールボックスの［**その他の変換先**］カテゴリから、［**SQL Server 変換先**］を SSIS デザイナー上へドラッグ＆ドロップします。



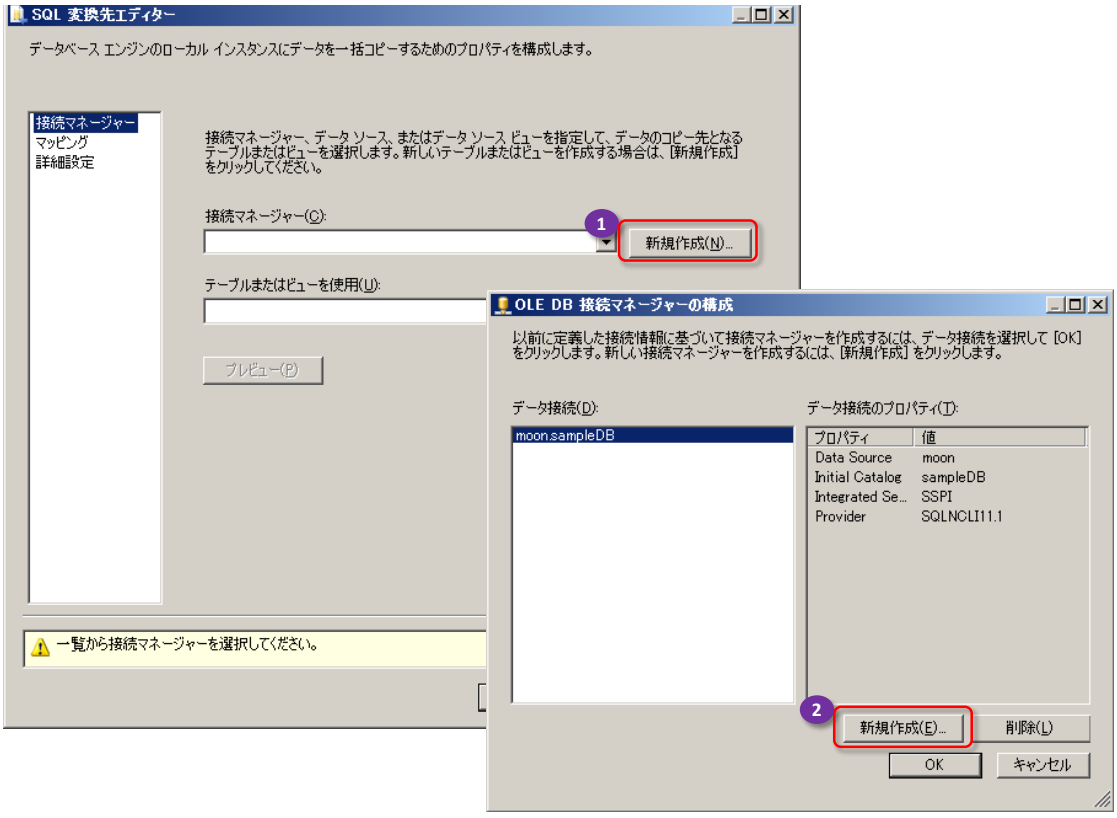
続いて、転送元と転送先を関連付けるために、［**フラット ファイル ソース**］の**青色の矢印**を

［**SQL Server 変換先**］まで、ドラッグ＆ドロップして伸ばします。

12. 次に、転送先の詳細設定を行うために、［**SQL Server 変換先**］をダブル クリックします。

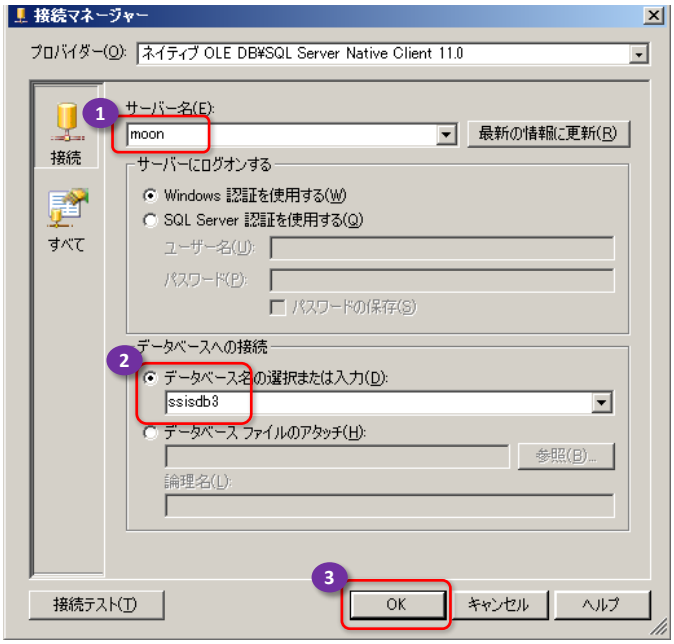


13. ［**SQL 変換先エディター**］ダイアログが表示されたら、［**接続マネージャー**］で［**新規作成**］ボタンをクリックします。



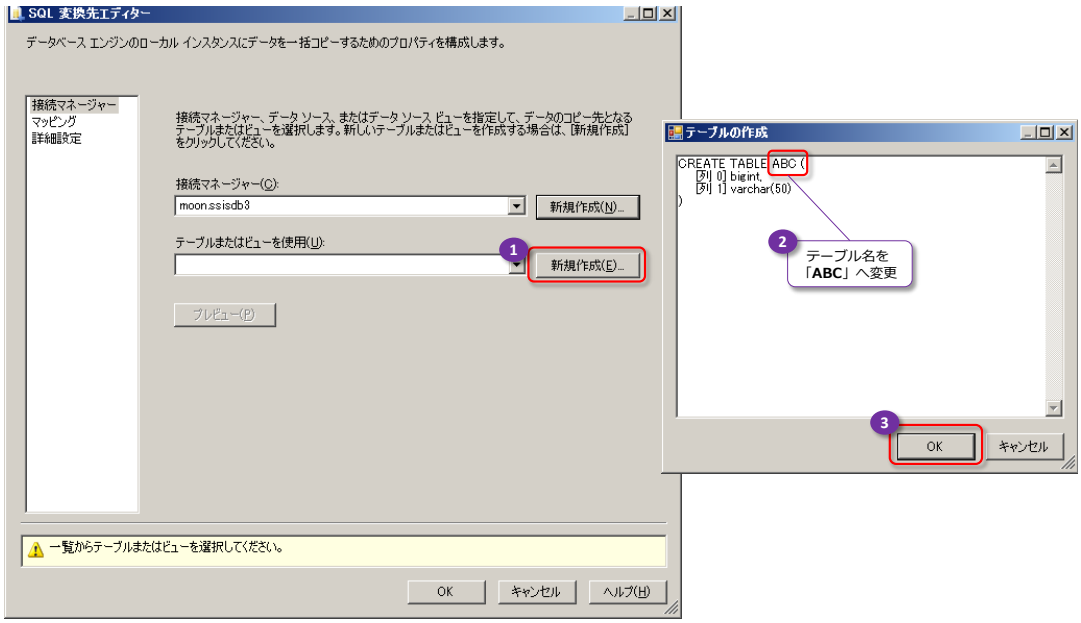
［**OLE DB 接続マネージャーの構成**］ダイアログが表示されたら、［**新規作成**］ボタンをクリックします。

14. ［**接続マネージャー**］ダイアログでは、［**サーバー名**］へ SQL Server の名前、［**データベース名の選択または入力**］で「**ssisdb3**」データベースを選択して、［**OK**］ボタンをクリックします。



［**OLE DB 接続マネージャーの構成**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして、閉じます。これで、**ssisdb3** データベースへ接続できるようになります。

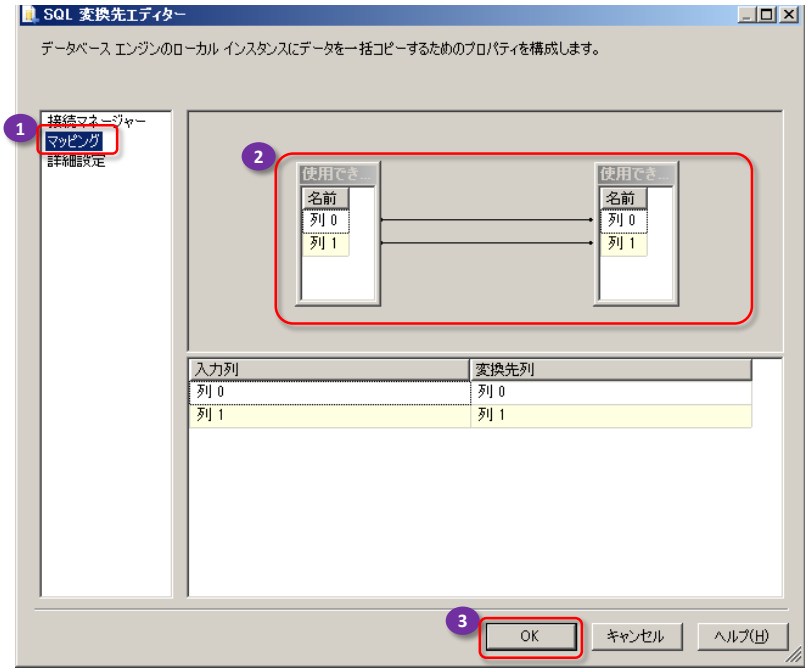
15. 次に、データの転送先となるテーブルを作成するために、次のように［**SQL 変換先エディター**］の［**テーブルまたはビューを使用**］で［**新規作成**］ボタンをクリックします。



これにより、［**テーブルの作成**］ダイアログが表示されて、転送元のデータをもとにテーブルを作成するための **CREATE TABLE** ステートメントが表示されます。今回は、テーブル名を「**ABC**」へ変更して、［**OK**］ボタンをクリックします。

［**OK**］ボタンをクリックしたタイミングで、このステートメントが実行されて、**ssisdb3** データベース内へ「**ABC**」テーブルが作成されます。

16. ［**SQL 変換先エディター**］ダイアログへ戻ったら、［**マッピング**］ページをクリックし、転送元のデータと転送先のデータのマッピングを設定します。



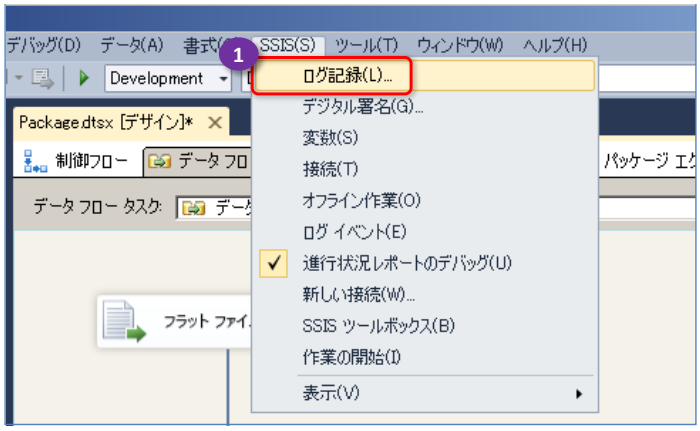
同じ名前の列同士がマッピングされていることを確認して、[**OK**] ボタンをクリックします。

これで、**ABC.txt** ファイルのデータが **ssisdb3** データベースへインポートすることができるようになりました。

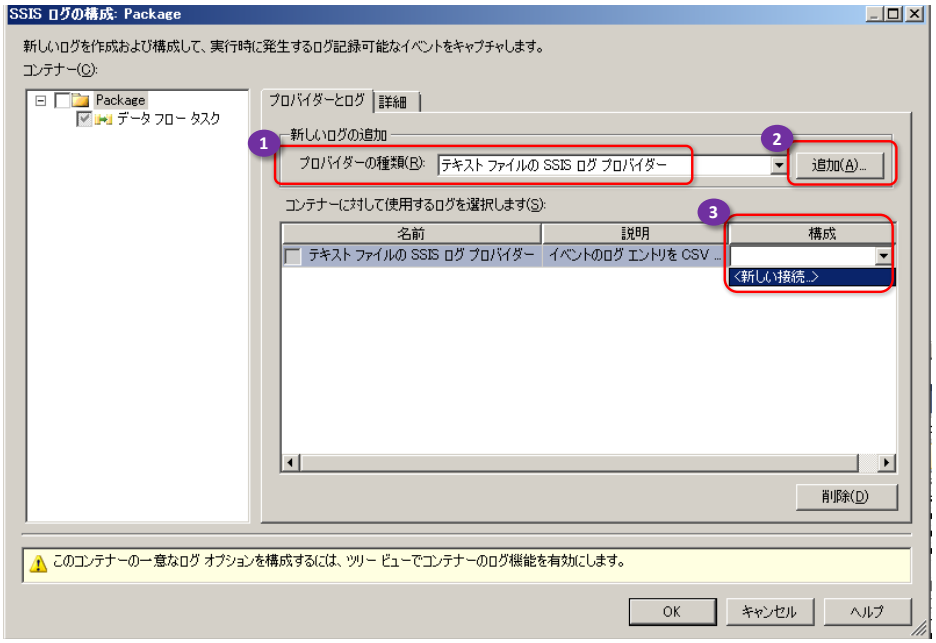
ログ記録の設定

続いて、パッケージの実行時に、ログがテキスト ファイルに記録されるようにしてみましょう。

1. 次のように、[**SSIS**] メニューから [**ログ記録**] をクリックします。

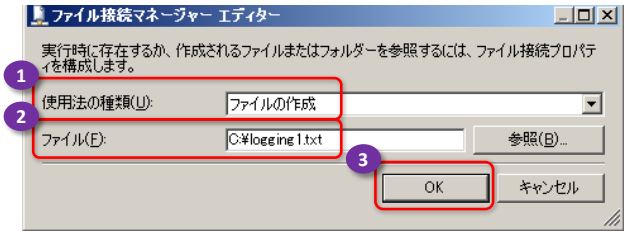


2. これにより、[**SSIS ログの構成**] ダイアログが表示されるので、[**プロバイダーの種類**] で「**テキスト ファイルの SSIS ログ プロバイダー**」を選択して、[**追加**]ボタンをクリックします。

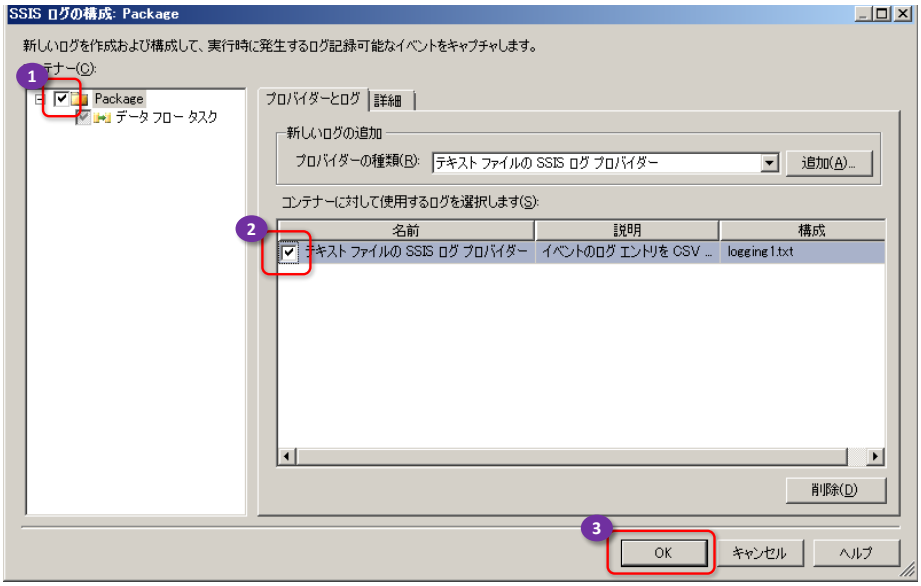


次に、追加したログの［**構成**］で、「**新しい接続**」をクリックします。

3. ［**ファイル接続マネージャー エディター**］が表示されたら、［**使用法の種類**］で、ログ記録用のファイルを新しく作成するために「**ファイルの作成**」を選択し、［**ファイル**］で任意のファイル名（ここでは、**C:¥logging1.txt**）と入力して、［**OK**］ボタンをクリックします。



4. ［**SSIS ログの構成**］ダイアログへ戻ったら、［**コンテナー**］で「**Package**」をチェックして、［**コンテナーに対して使用するログを選択します**］で追加したログをチェックし、［**OK**］ボタンをクリックします。



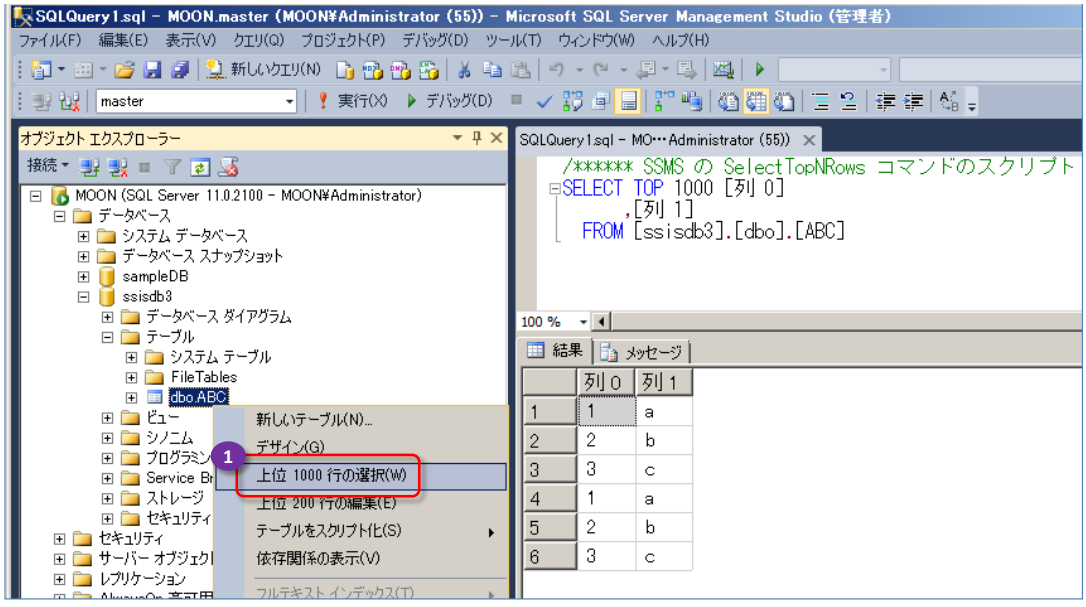
パッケージの実行

5. ツールバーの [**デバッグ開始**] ボタンをクリックしてパッケージを実行し、ここまでの設定を確認します。

配置されているオブジェクトに緑のチェックマークが付いて、データの転送が成功したことを確認したら、ツールバーの [**デバッグの停止**] ボタンをクリックして、デバッグを終了します。

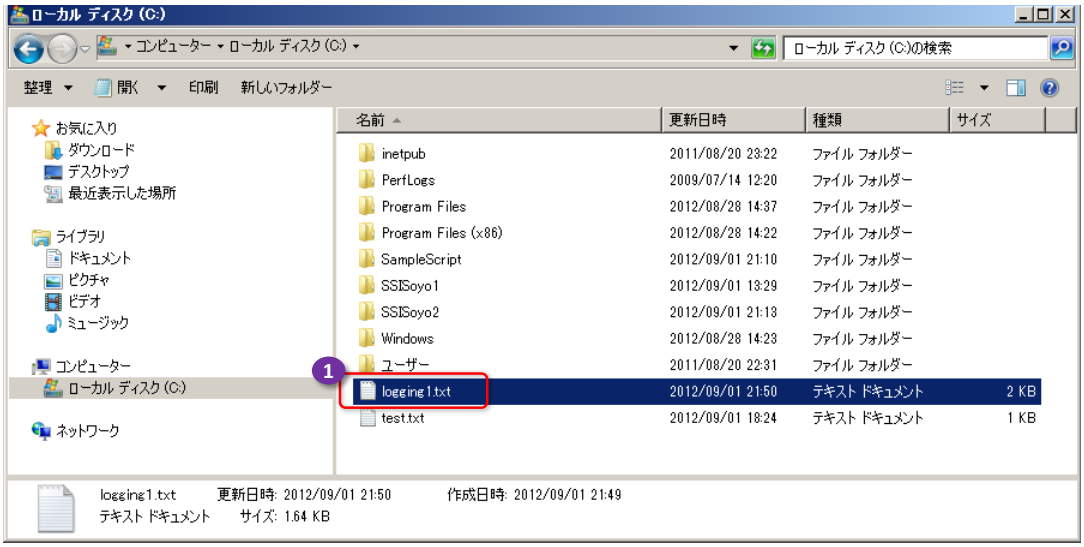
6. もう 1 度、デバッグを開始して、パッケージを合計で 2 回実行します。

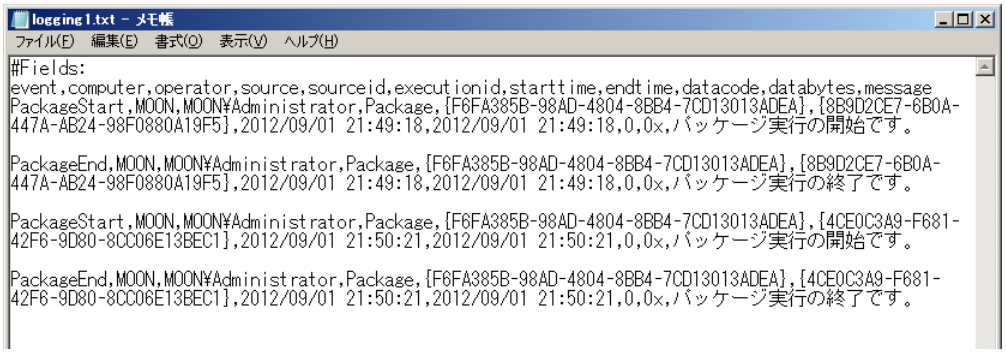
7. データが正常に転送されたことを確認するには、Management Studio から、**ssisdb3** データベースの［**ABC**］テーブルを右クリックして、［**上位 1000 行の選択**］をクリックし、データを確認します。



パッケージを 2 回実行しているので、2 回分のデータがインポートできていることを確認できます。

8. 次に、**Windows エクスプローラー**を起動して、**C:¥logging1.txt** ファイルをダブル クリックして開き、記録されたログを確認します。





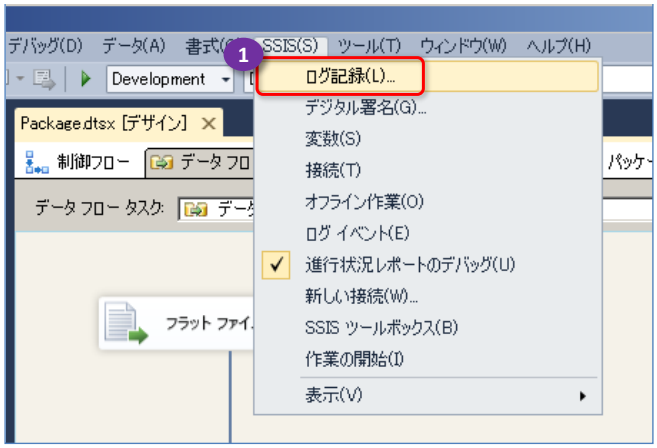
パッケージを実行した開始時刻（**PackageStart**）と終了時刻（**PackageEnd**）が 2 回分記録されていることを確認することができます。このように、ログ記録を設定すると、パッケージの実行に関するログを簡単に取得できるようになります。

3.3 ログ記録の詳細設定

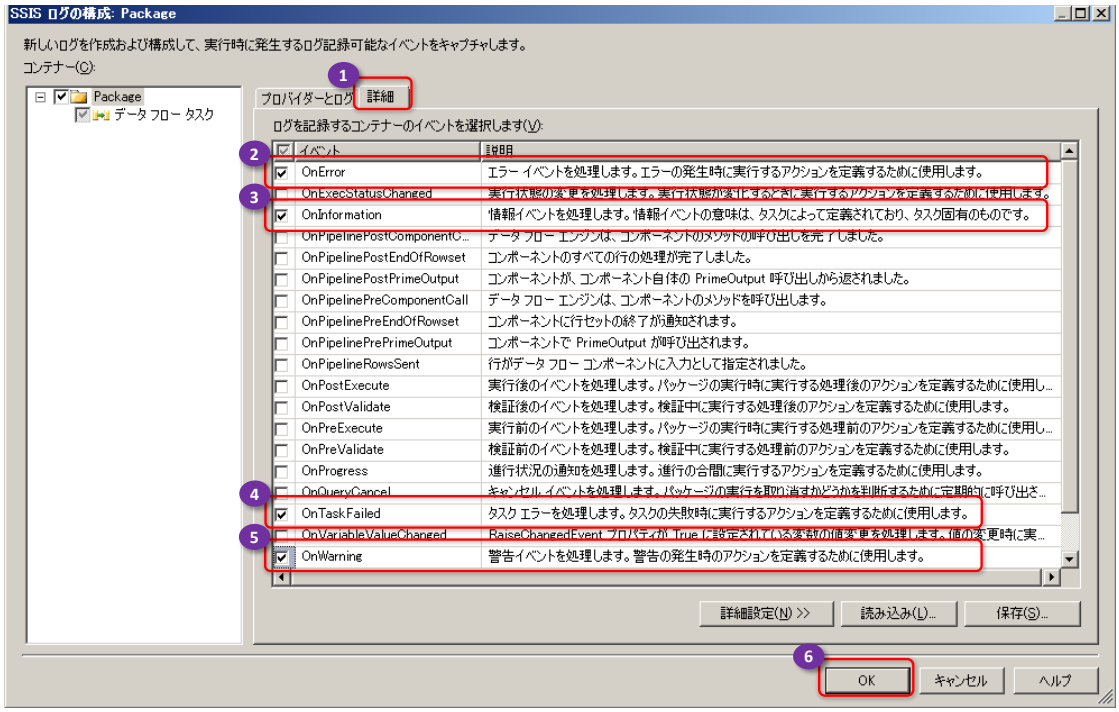
ログ記録の詳細設定

前の手順のログ記録では、パッケージの開始時刻と終了時刻にみの記録でしたが、ログ記録では、それ以外の情報（エラー情報など）も記録することができます。それはこれを試してみましょう。

1. ログ記録の設定を変更するには、[**SSIS**] メニューから [**ログ記録**] をクリックします。



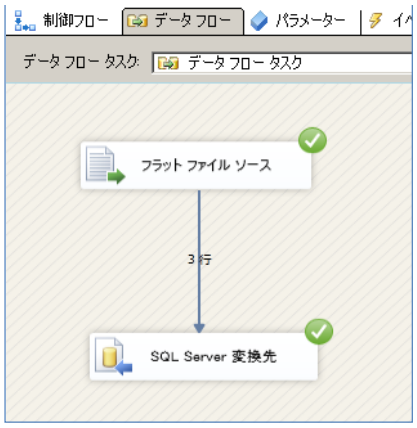
2. [**SSIS ログの構成**] ダイアログが表示されたら、[**詳細**] タブをクリックしてページを開き、ログを記録するイベントを選択します。



ここでは、「**OnError**」（エラーの発生）、「**OnInformation**」（情報メッセージ）、「**OnTaskFailed**」（タスクの失敗）、「**OnWarning**」（警告）をチェックして、[**OK**] ボタンをクリックします。

パッケージの実行

3. パッケージを実行して、ここまでの設定を確認します。



配置されているオブジェクトに緑のチェックマークが付いて、データの転送が成功したことを確認したら、デバッグを終了します。

4. 次に、**Windows エクスプローラー**から、**C:¥logging1.txt** ファイルを開いて、記録されたログを確認します。

パッケージを実行した開始時刻（**PackageStart**）と終了時刻（**PackageEnd**）に加えて、

タスクの情報（**OnInformation**）が記録されていることを確認できます。今回は、正常にデータが転送されているので、**OnError** や **OnTaskFailed**、**OnWarning** については、記録されていません。

Note： ［実行結果］（進捗状況）タブと同じ情報が記録される

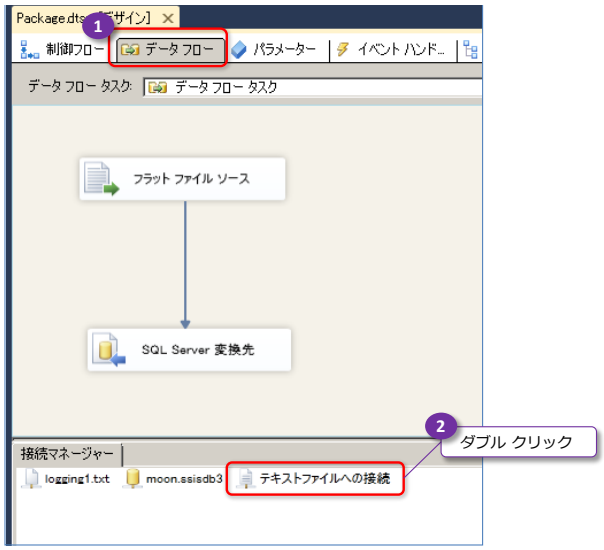
ログ記録では、パッケージの［**実行結果**］タブと同じ結果を取得できます。

3.4 エラー発生時のログ記録の確認

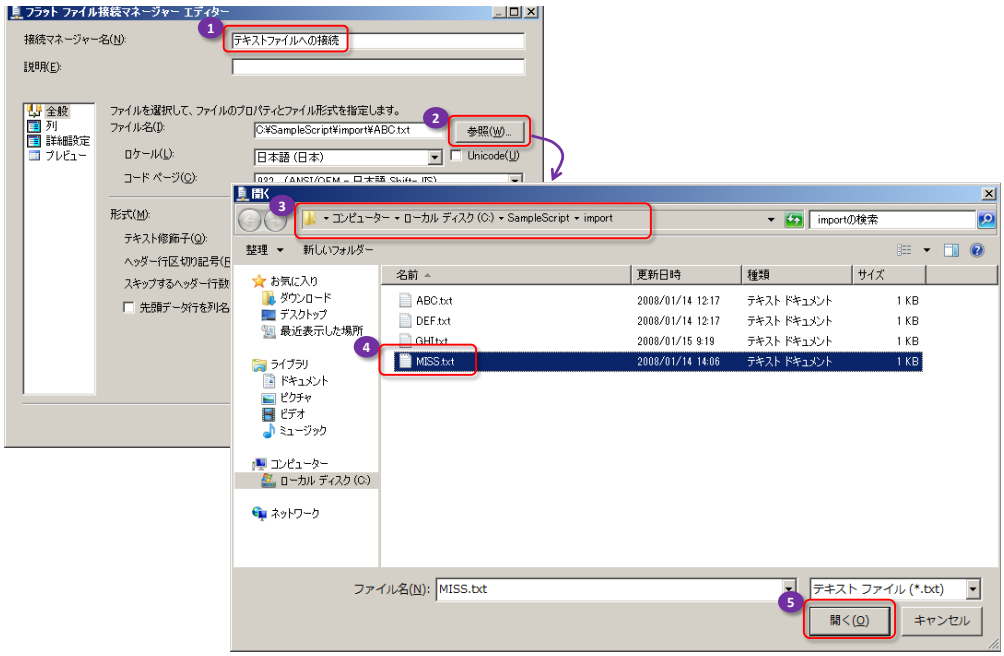
エラー発生時のログ記録の確認

次に、転送元のテキスト ファイルを「**MISS.txt**」ファイルへ変更して、転送時にデータ型に関するエラーが発生するようにして、エラー発生時にどういったログが記録されるのかを確認してみましょう。

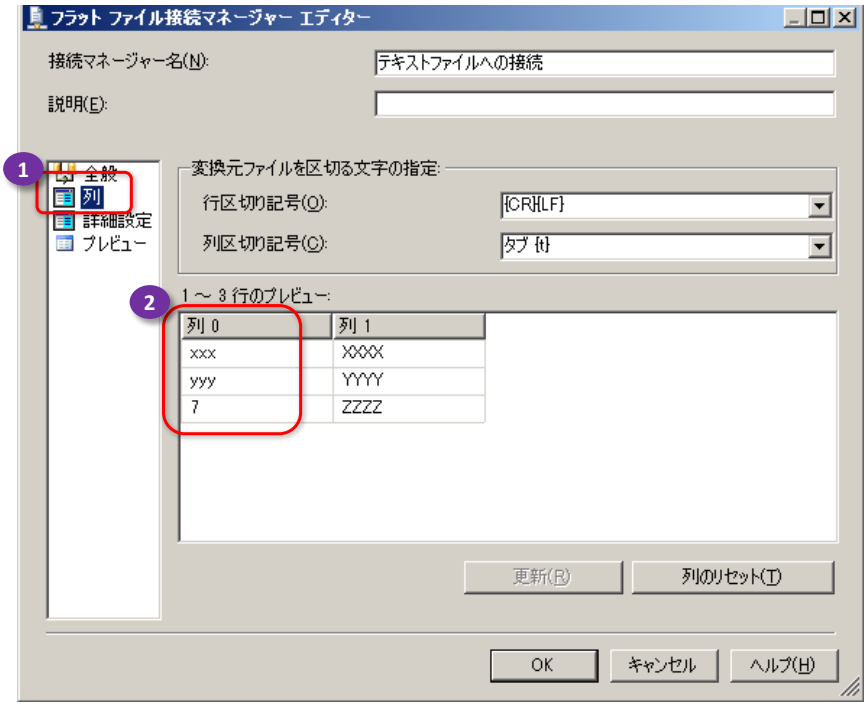
1. 転送元のテキスト ファイルを変更するには、次のように［**データ フロー**］タブの画面下部の［**接続マネージャー**］に一覧されている「**テキストファイルへの接続**」（前の Step で作成したもの）をダブル クリックします。



2. これにより、［**フラット ファイル接続マネージャー エディター**］ダイアログが表示されるので、［**ファイル名**］の［**参照**］ボタンをクリックして、サンプル スクリプト内の「**import**」フォルダー内にある「**MISS.txt**」ファイルへ変更します。

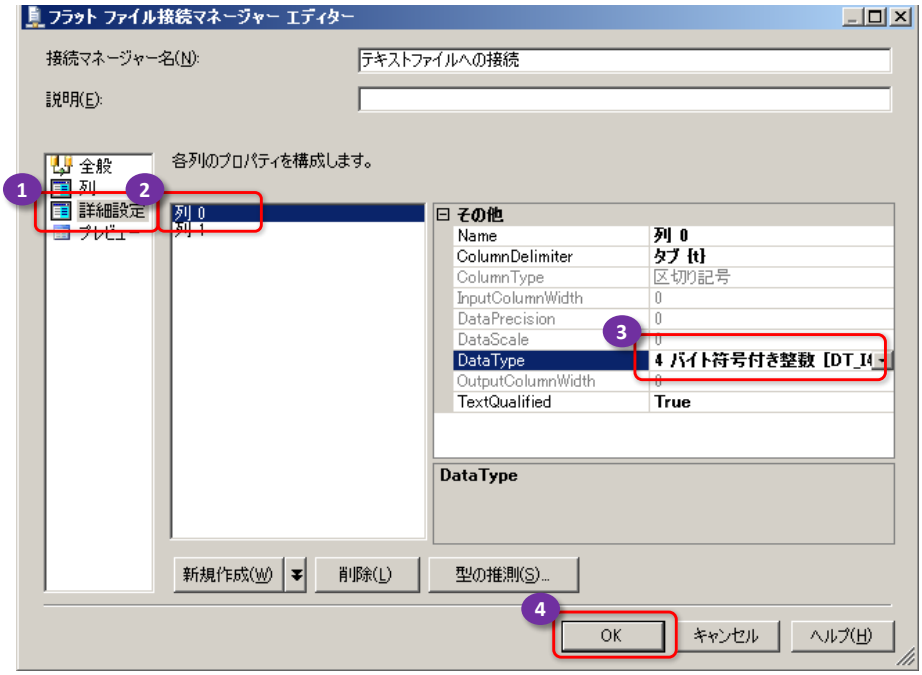


3. 続いて、次のように「**列**」ページをクリックして、[**1～3 行のプレビュー**]で転送元となるデータを確認します。



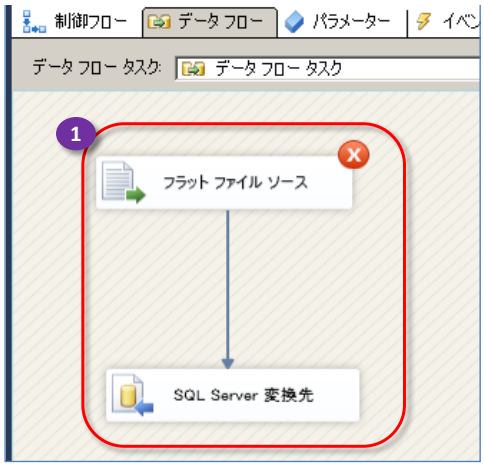
1 列目の「**列 0**」列には、文字データが含まれていることに注目します。

4. 続いて、[**詳細設定**]ページをクリックして開き、前の Step で設定したように「**列 0**」列の[**DataType**]が「**4 バイト符号付き整数**」(SQL Server の **int** 型に相当)になっていることを確認します。確認後、[**OK**]ボタンをクリックして、[**フラット ファイル接続マネージャー エディター**]ダイアログを閉じます。



データ転送の実行

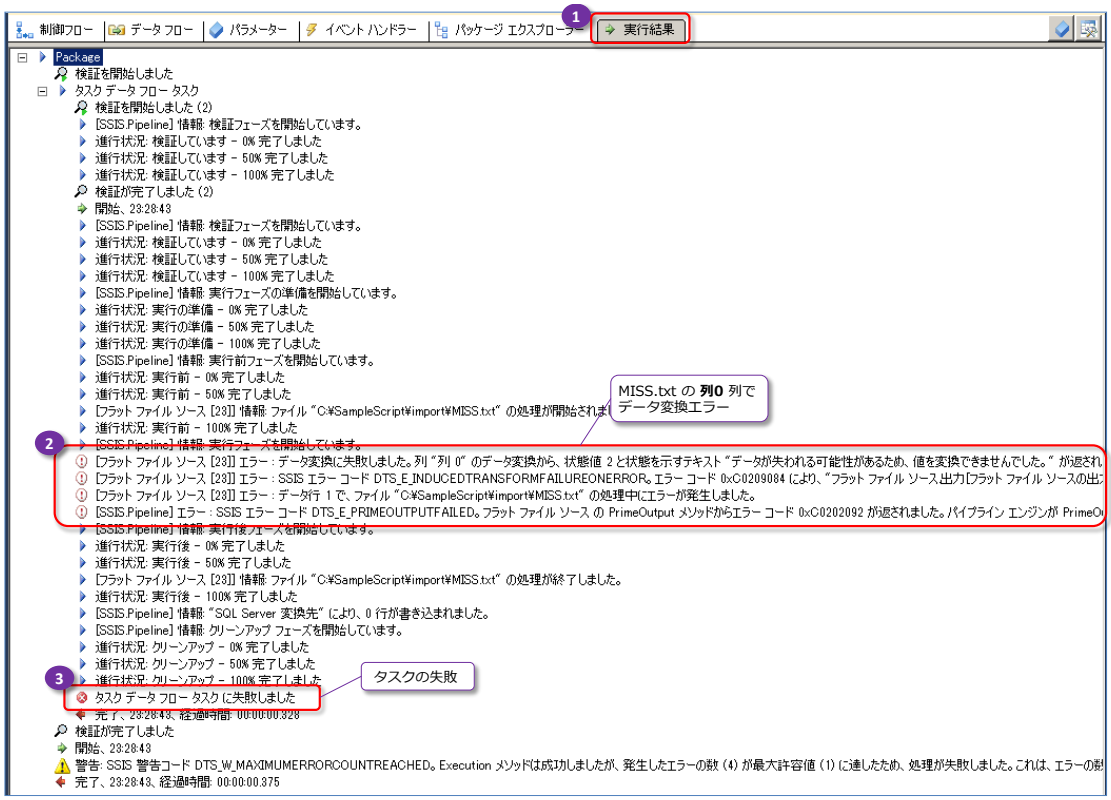
5. パッケージを実行して、結果を確認します。



今度は、[**フラット ファイル ソース**]がエラーになって、データ転送が失敗していることを確認できます。

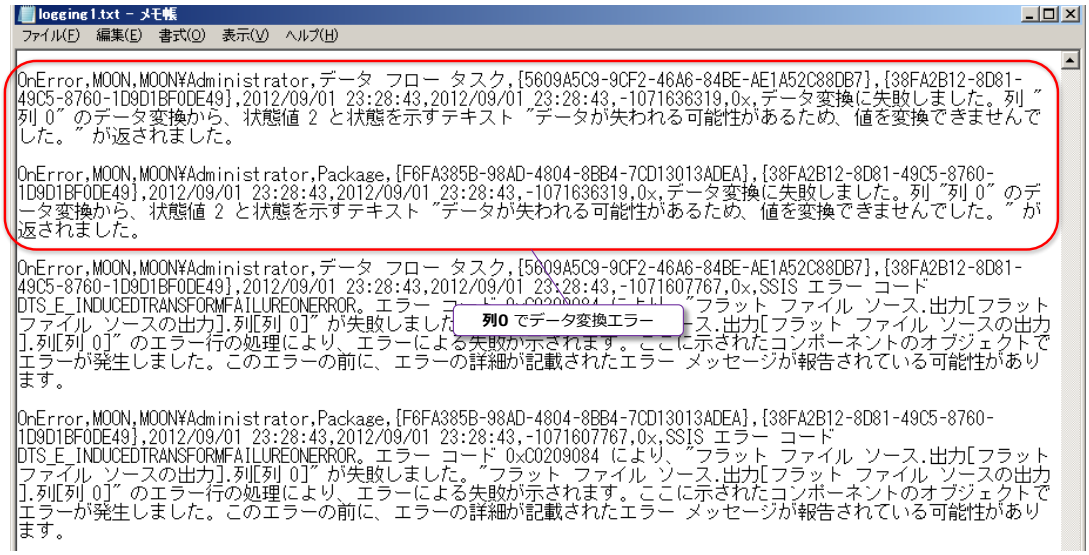
6. 確認後、デバッグを終了します。

7. 続いて、[**実行結果**]タブをクリックして、実行結果を確認します。



「**データ変換に失敗しました。列 "列 0" の～**」というエラーが発生して、**MISS.txt** の「**列 0**」でのデータ型の変換エラーによって、パッケージ実行が失敗したことを確認できます。これは、**列 0**（1 列目）のデータ型を **int** 型に設定しているにも関わらず、**文字データ**が含まれているために発生しています。

8. 次に、**Windows エクスプローラー**から、**C:¥logging1.txt** ファイルを開いて、記録されたログを確認します。



：　（中略）

今回はデータ転送が失敗したので、エラーの発生（**OnError**）とタスクの失敗（**OnTaskFailed**）が記録されることを確認できます。

このように、ログ記録を設定しておくと、パッケージの実行エラーを記録することができるので、大変便利です。

9. 最後に、［**ファイル**］メニューの［**すべてを保存**］をクリックして、**SSISoyo2** プロジェクトを保存しておきます。このプロジェクトは、次の Step でも引き続き使用します。

STEP 4. イベント ハンドラー

この STEP では、パッケージの実行イベントの内容に応じて、タスクを実行する方法を説明します。

この STEP では、次のことを学習します。

- ✓ イベント ハンドラーの設定
- ✓ エラー発生時のスクリプト実行

4.1 イベント ハンドラーの利用

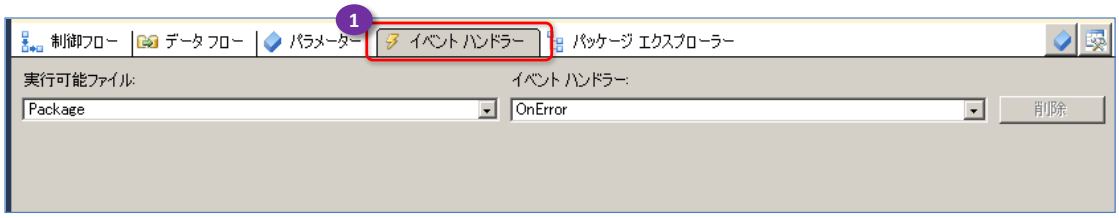
イベント ハンドラーの利用

イベント ハンドラーを利用するとパッケージ実行時のイベント（エラーや警告など）の内容に応じて、タスクを実行できるようになります。この Step では、パッケージのエラー発生時に、スクリプト タスクを実行するようにしてみましょう。

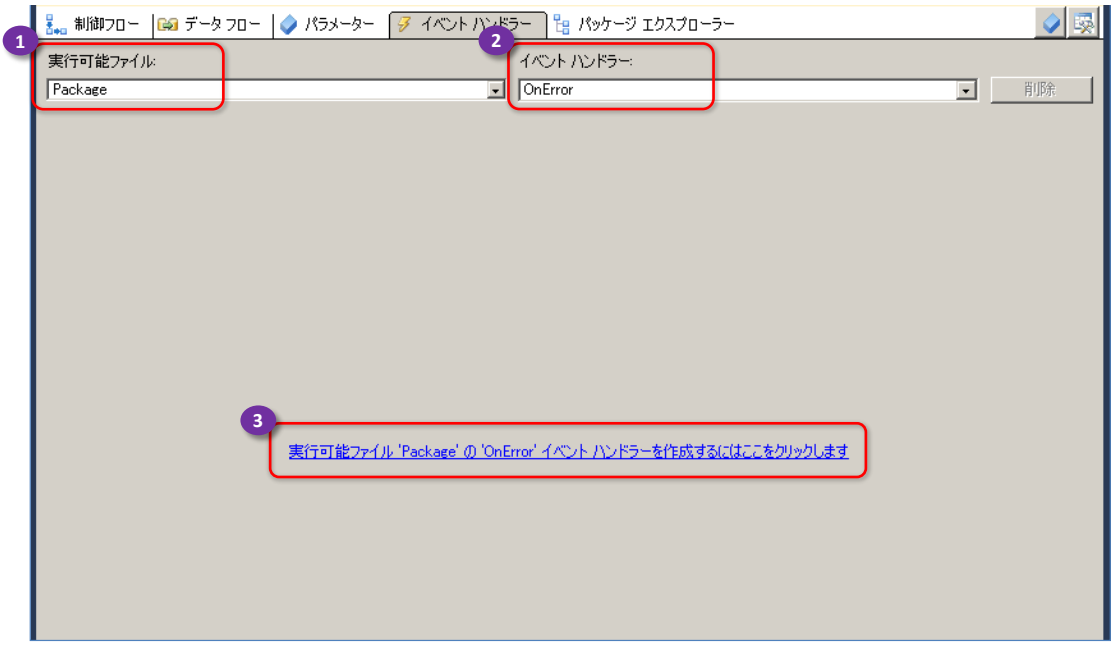
イベント ハンドラーの設定

ここでは、前の Step で作成した **SSISSoyo2** プロジェクトを引き続き利用します。

17. イベント ハンドラーを設定するには、次のように［**イベント ハンドラー**］タブをクリックします。

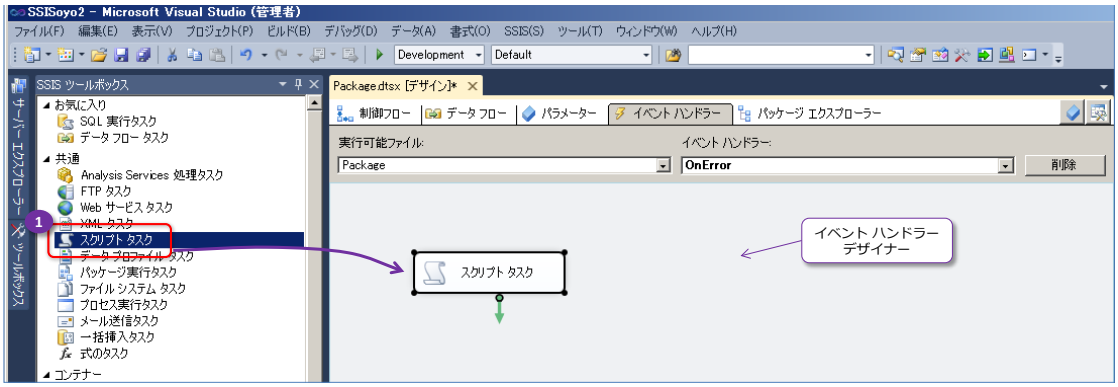


18. 今回は、エラー発生時に実行するタスクを設定するので、［**実行可能ファイル**］で［**Package**］、［**イベント ハンドラー**］で「**OnError**」が選択されていることを確認して、［**実行可能ファイル 'Package' の 'OnError' イベント ハンドラーを作成するにはここをクリックします**］をクリックします。

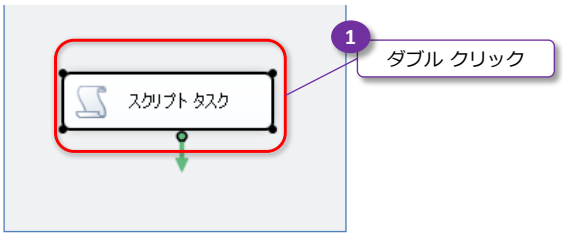


19. これにより、次のように**イベント ハンドラー デザイナー**が表示されるので、ここで、エラー発生時に実行するタスクを設定していきます。今回は、SSIS ツール ボックスの［**共通**］カ

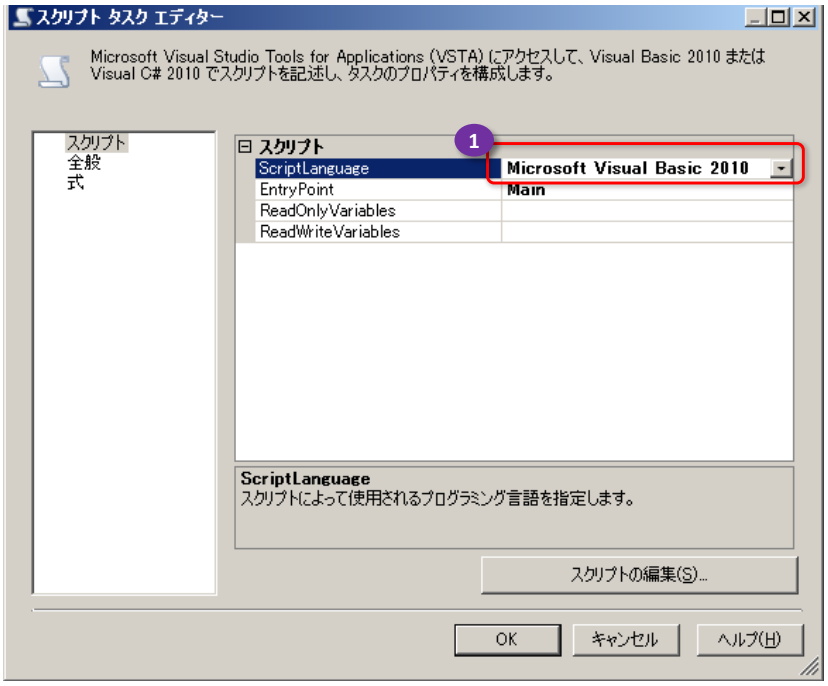
テゴリから［**スクリプト タスク**］をドラッグ＆ドロップして、配置します。



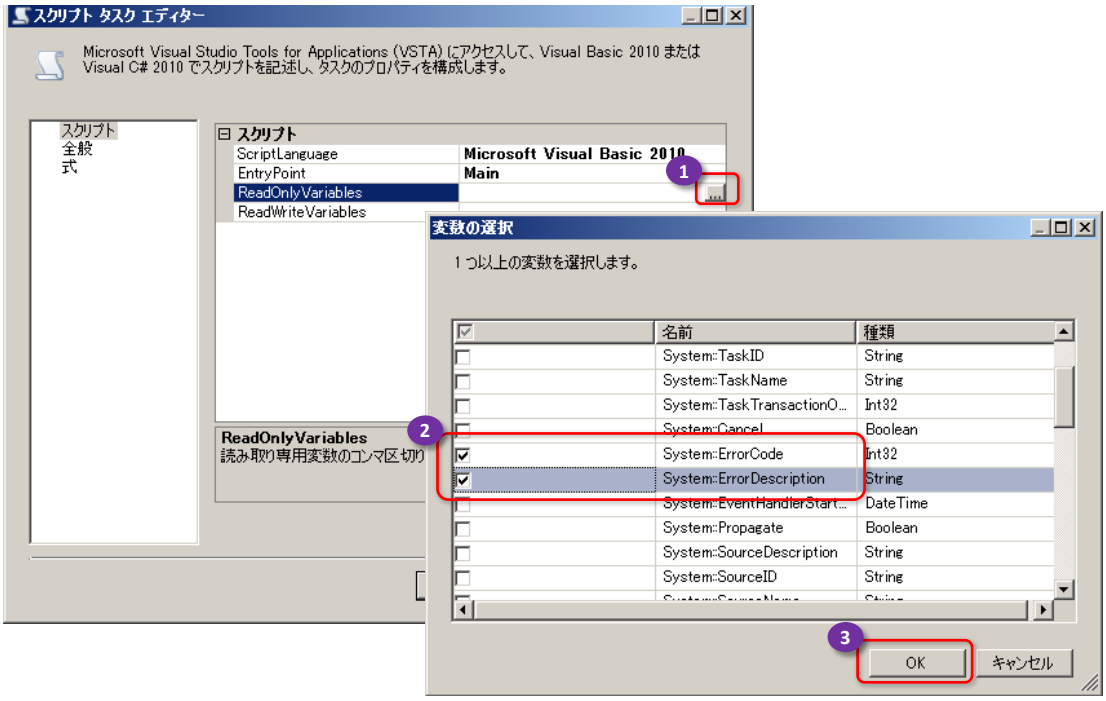
20. 配置した［**スクリプト タスク**］をダブル クリックします。



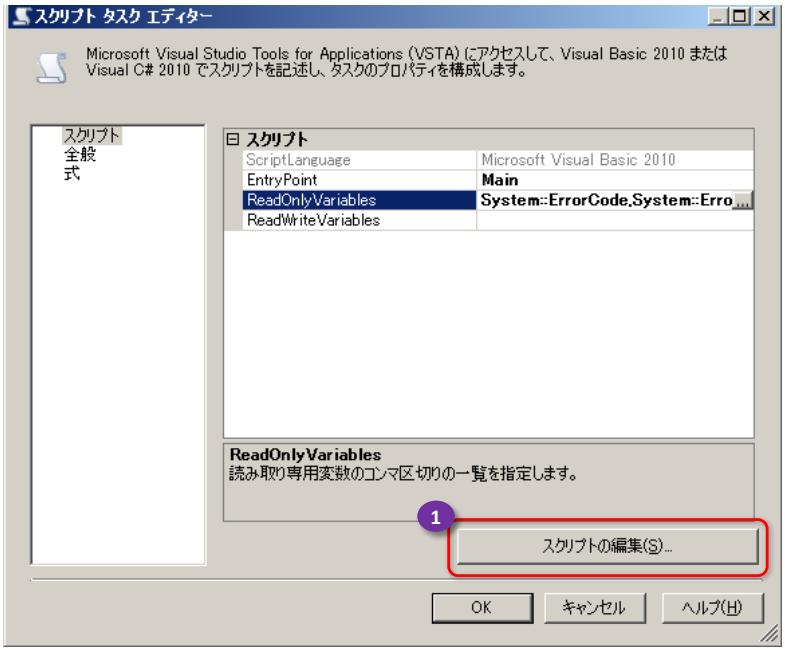
21. ［**スクリプト タスク エディター**］ダイアログが表示されたら、今回も Visual Basic でスクリプトを記述するので、［**ScriptLanguage**］で「**Microsoft Visual Basic 2010**」を選択します。



22. 続いて、［**ReadOnlyVariables**］で［**...**］ボタンをクリックして、［**変数の選択**］ダイアログを表示し、システム変数の一覧から「**System::ErrorCode**」（エラー番号）と「**System::ErrorDescription**」（エラーの説明）をチェックします。

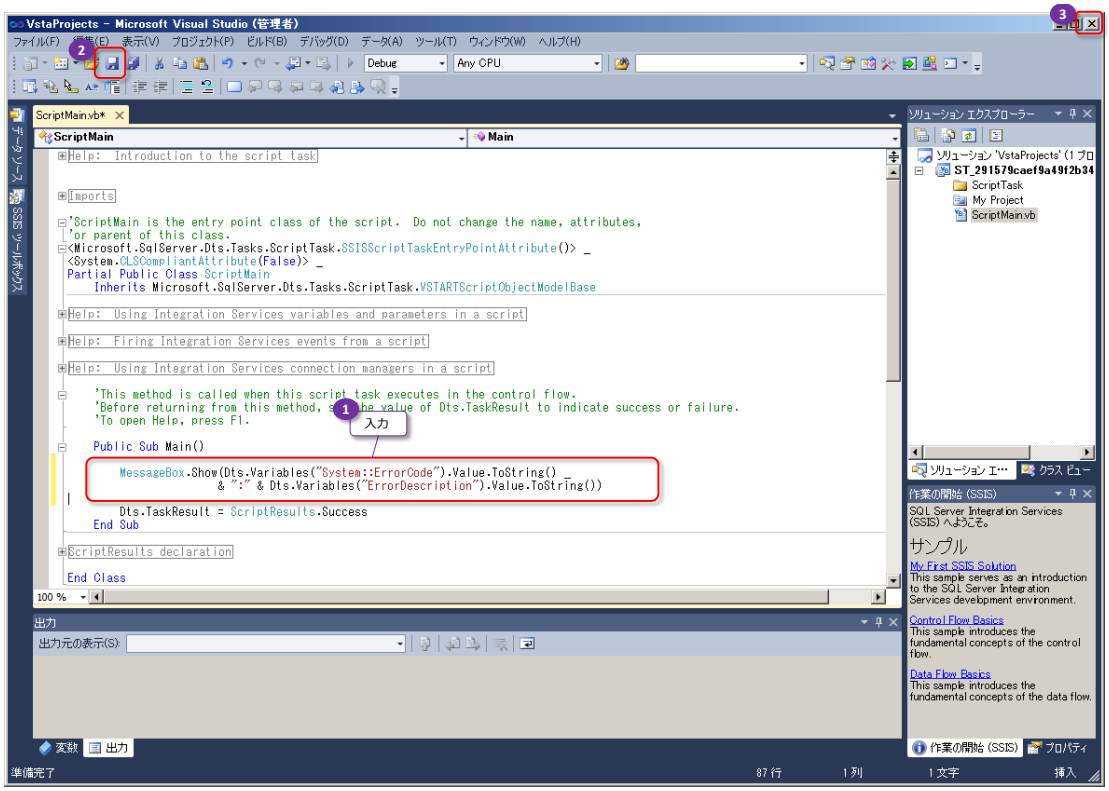


23. ［**スクリプト タスク エディター**］へ戻ったら、［**スクリプトの編集**］ボタンをクリックします。



24. ［**スクリプト エディター**］が表示されたら、「**public Sub Main( )**」内のコードを、次のように入力します（変数名の大文字と小文字に注意してください）。

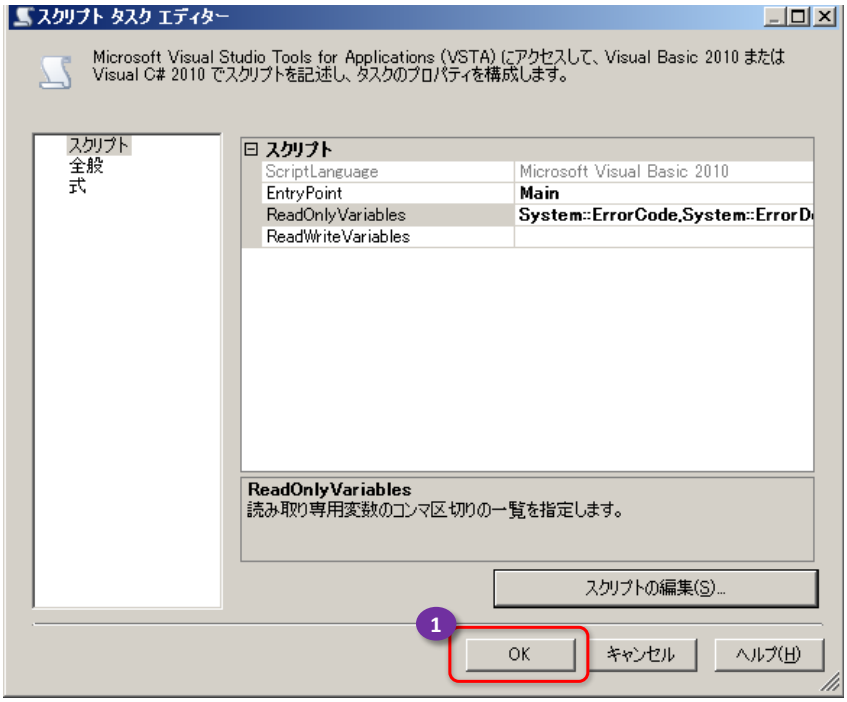
```
MessageBox.Show(Dts.Variables("System::ErrorCode").Value.ToString() _
                & ":" & Dts.Variables("ErrorDescription").Value.ToString())
```



このコードにより、エラー番号とエラーの説明をメッセージ ボックスで表示できるようになります。

入力後、保存してスクリプト エディターを閉じます。

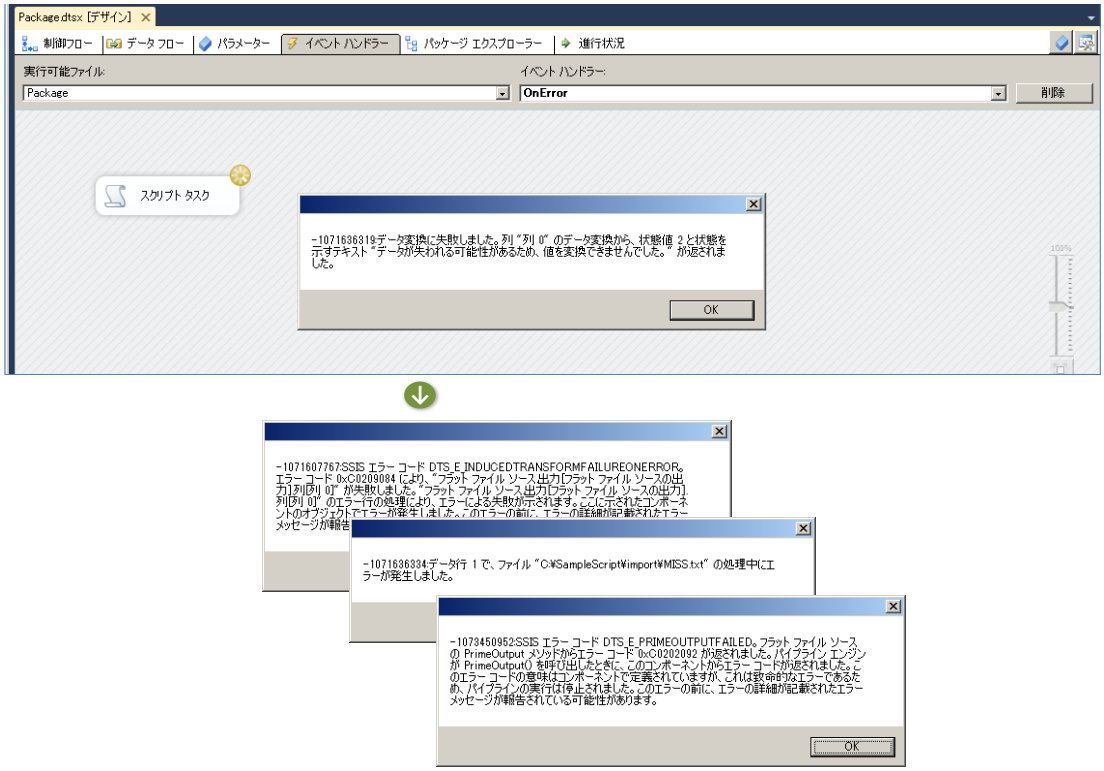
25. ［**スクリプト タスク エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。



パッケージを実行して確認

26. パッケージをデバッグ実行して確認します。

配置されている［**スクリプト タスク**］が実行中に変わり、エラー番号とエラーの説明がメッセージ ボックスで表示されることを確認できます。



このメッセージ ボックスは、エラー（OnError イベント）が発生した数の分だけ表示されるので、その分［**OK**］ボタンをクリックします。

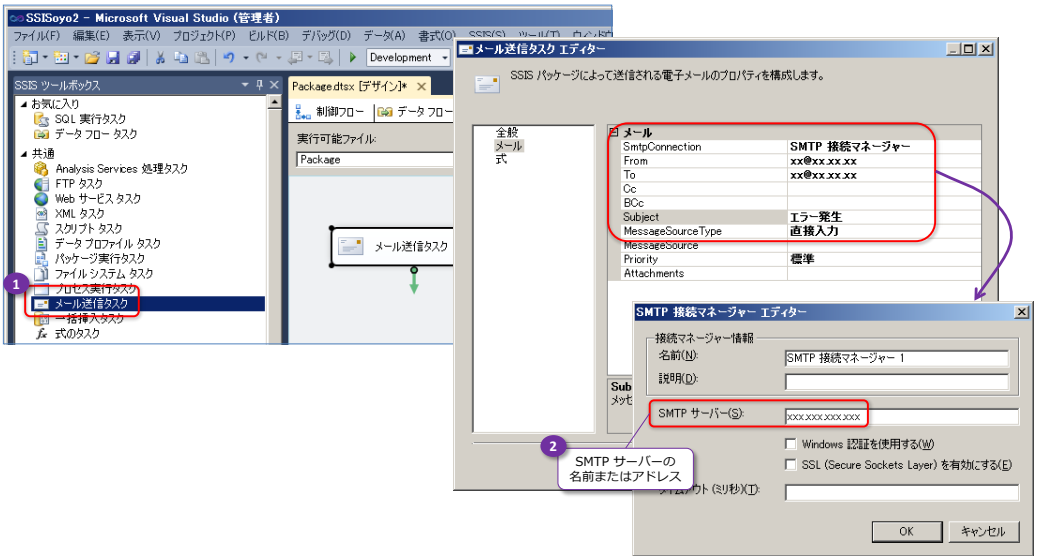
27. すべてのメッセージ ボックスが表示され、［**スクリプト タスク**］に緑のチェックマークが付いたら、デバッグを終了します。

28. 最後に、［**ファイル**］メニューの［**すべてを保存**］をクリックして、**SSISoyo2** プロジェクトを保存しておきます。このプロジェクトは、次の Step でも引き続き使用します。

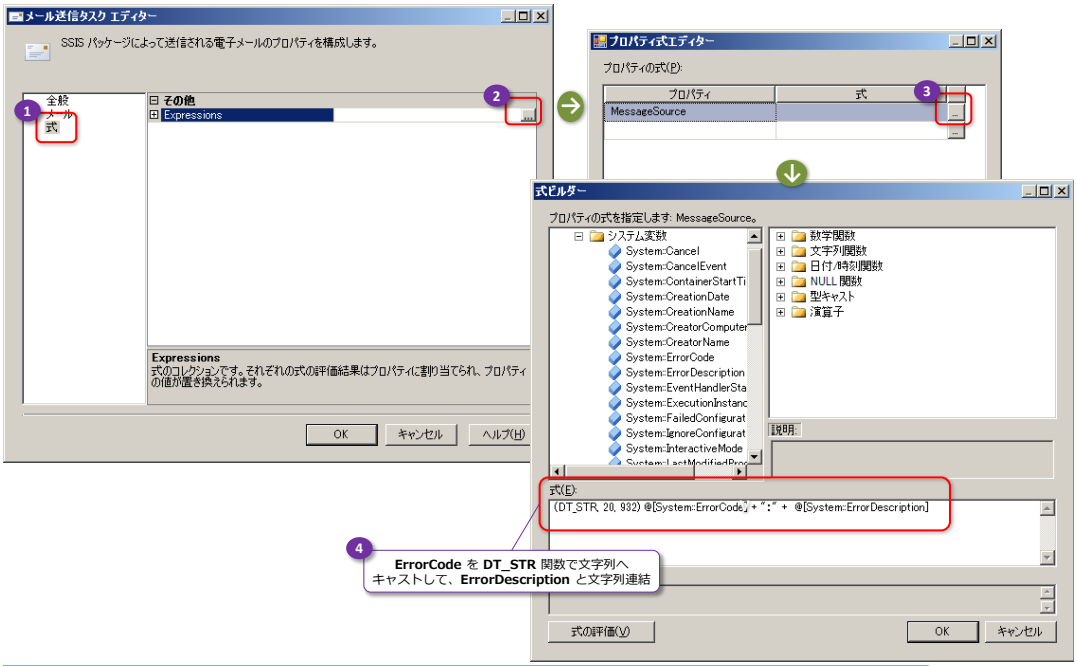
このように、イベント ハンドラーを利用すれば、パッケージのエラー発生時などに応じて、タスクを実行できるようになるので、大変便利です。

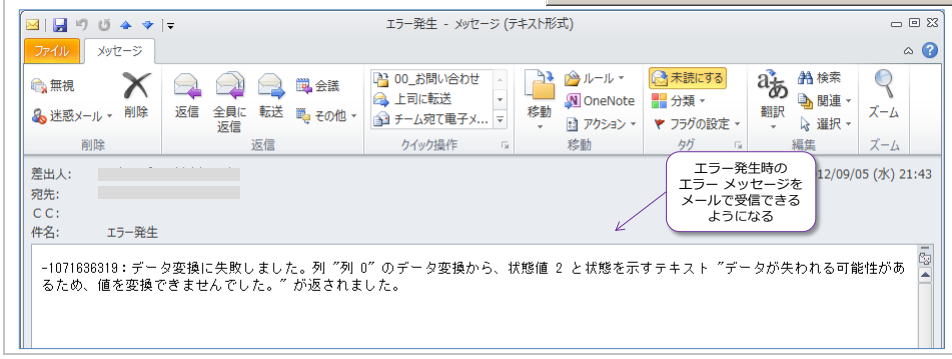
Note： エラー発生時にメール送信（メール送信タスク）

OnError イベント ハンドラーや OnTaskFailed（タスクの失敗）イベント ハンドラーでは、**メール送信タスク**を利用すると、エラー発生時にメールを送信できるようになるので便利です。
メール送信タスクでは、次のように「**SMTP サーバー**」の名前と「**From**」（送信元のメール アドレス）、「**To**」（送信先のメール アドレス）、「**件名**」を設定するだけでメールを送信することができます。



メールの本文として **ErrorCode** や **ErrorDescription** を送信するには、次のように［**式**］ページで［**MessageSource**］プロパティを設定するようにします。





STEP 5. エラー コンポーネントの利用

この STEP では、データ転送時にエラーが発生した場合の対処方法について説明します。

この STEP では、次のことを学習します。

- ✓ エラー発生時の動作
- ✓ エラーを無視して実行する方法
- ✓ エラー情報をファイルへ書き込む方法
- ✓ スクリプト コンポーネント（C#）によるエラー メッセージの取得

5.1 作成するパッケージの概要

作成するパッケージの概要

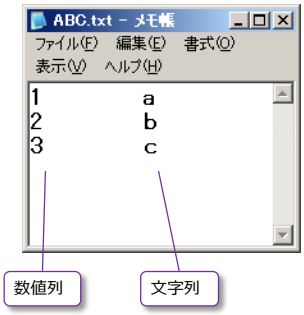
この STEP では、データ転送時に起こりがちな、転送元のデータにイレギュラーな値が入っていた場合に発生するエラーの対処方法について説明します。前の Step で作成したエラーの発生するパッケージ（**SSISoyo2** プロジェクト）を利用して、エラーの対処方法を試していきます。

SSISoyo2 プロジェクトの内容

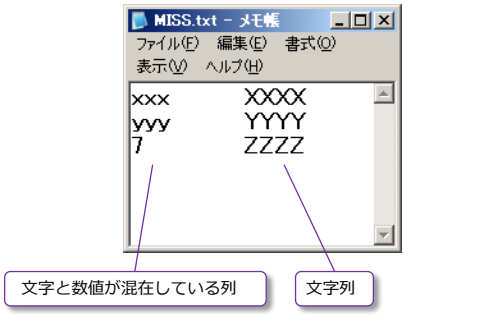
この Step の実習を行うには、事前に Step3 と Step4 を行い、**SSISoyo2** プロジェクトを作成しておく必要があります。**SSISoyo2** プロジェクトのパッケージは、次のようなテキスト ファイル（タブ区切り）のデータを SQL Server へ転送し、データ型のエラー（文字と数値が混在している場合のエラー）を発生させるものです。

転送するファイル

正常に転送されるファイル（ABC.txt）

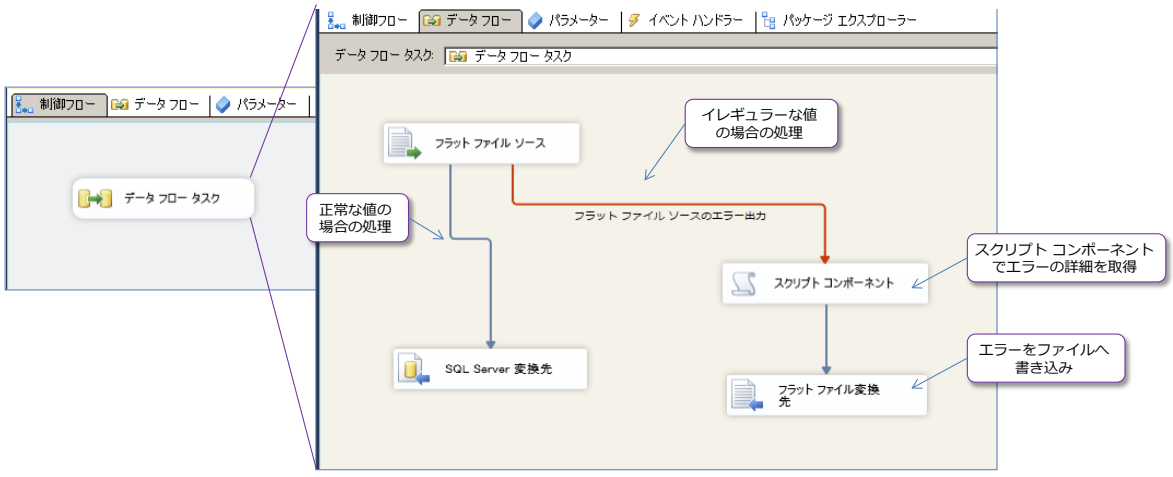


エラーが発生するファイル（MISS.txt）



作成するパッケージ

この Step では、**SSISoyo2** プロジェクトを、次のように変更していきます。



5.2 SSISoyo2 プロジェクトの動作確認

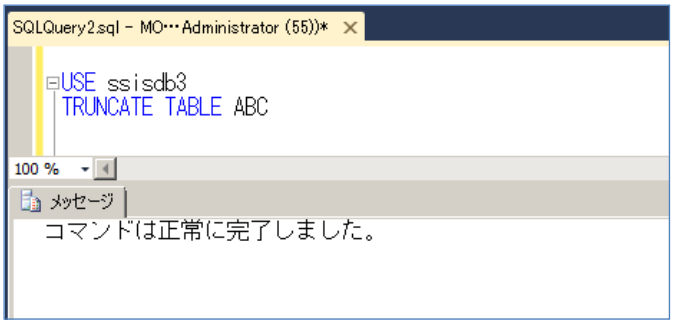
SSISoyo2 プロジェクトの動作確認（エラーが発生することの確認）

まずは、前の Step で作成した **SSISoyo2** プロジェクトのパッケージを実行して、動作確認をします。

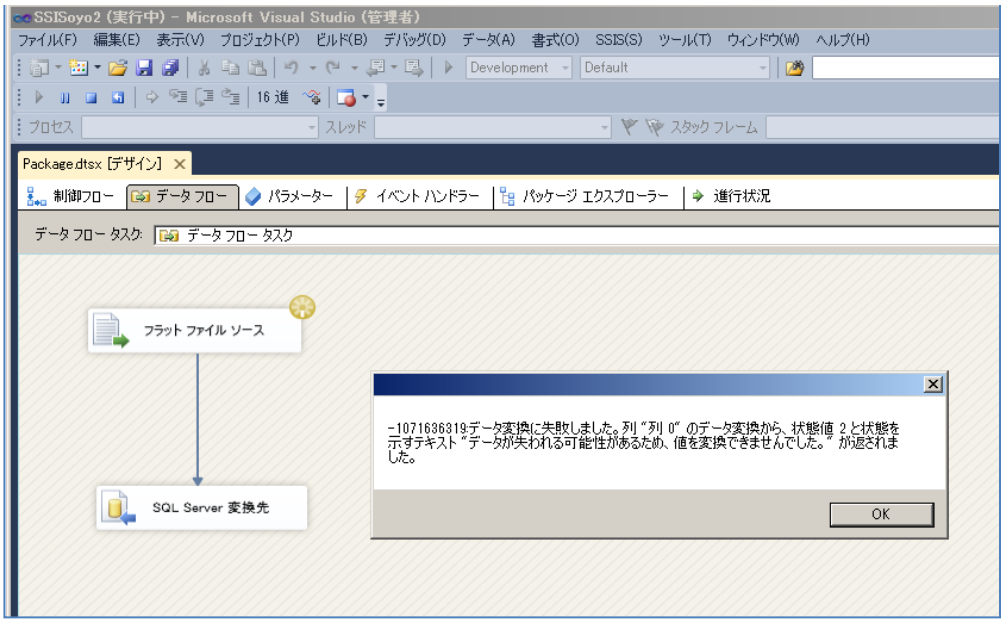
1. パッケージの実行結果を確認しやすくするために、転送先の **ssisdb3** データベース内にある **ABC** テーブルを一度空にしておきます。

   Management Studio を起動して、クエリ エディターから次のように **TRUNCATE TABLE** ステートメントを実行して、「**ABC**」テーブルのすべてのデータを削除しておきます。

```
USE ssisdb3
TRUNCATE TABLE ABC
```



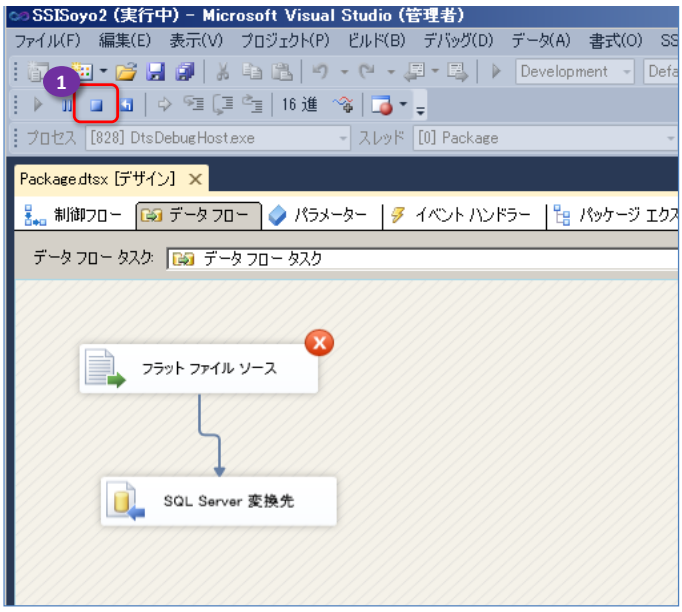
2. 次に、パッケージの［**データ フロー**］タブを開いてから、デバッグを実行します。



配置している［**フラット ファイル ソース**］が実行中に変わって、前の Step で作成した **OnError** イベント ハンドラーが実行されて、エラー番号とエラーの説明がメッセージ ボックスで表示されることを確認できます。

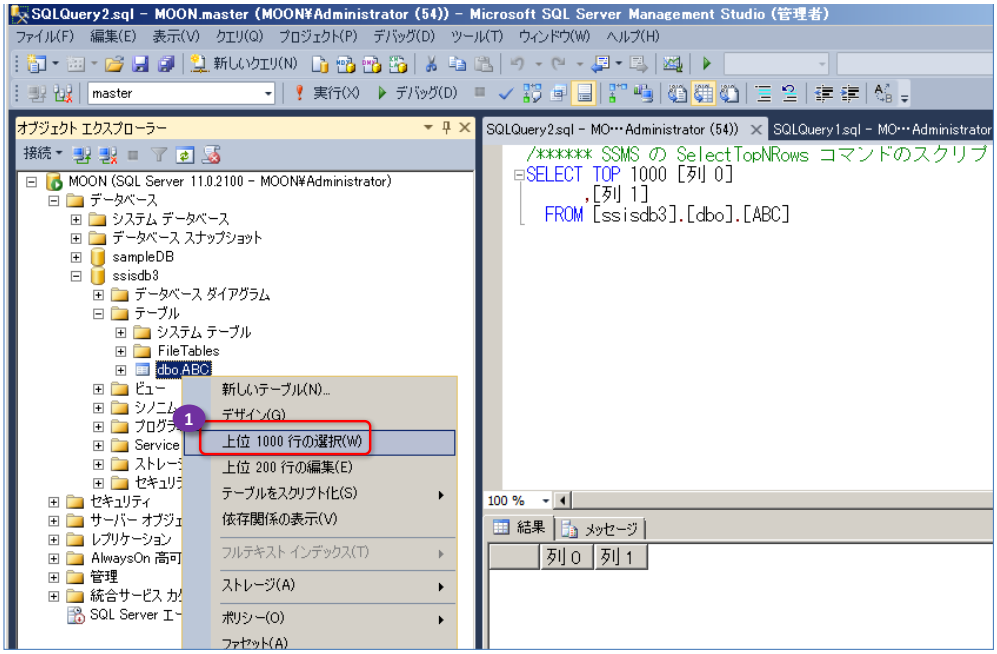
このメッセージ ボックスは、エラー（OnError イベント）が発生した数の分だけ表示されるので、その分［**OK**］ボタンをクリックします。

3. すべてのメッセージ ボックスが表示され、［**フラット ファイル ソース**］がエラーになって、転送が失敗したことを確認できます。



確認後、デバッグを終了します。

4. 次に、Management Studio から、［**ABC**］テーブルを右クリックして、［**上位 1000 行の選択**］をクリックし、データを確認します。



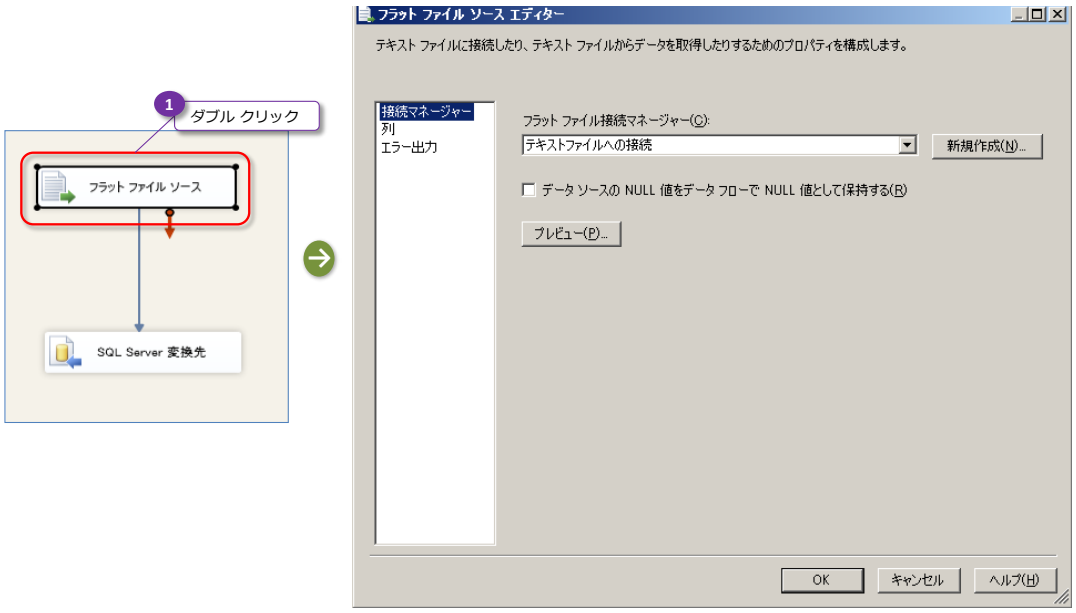
データ転送が失敗しているので、**MISS.txt** ファイルのデータは追加されていないことを確認できます。

5.3 エラーを無視する

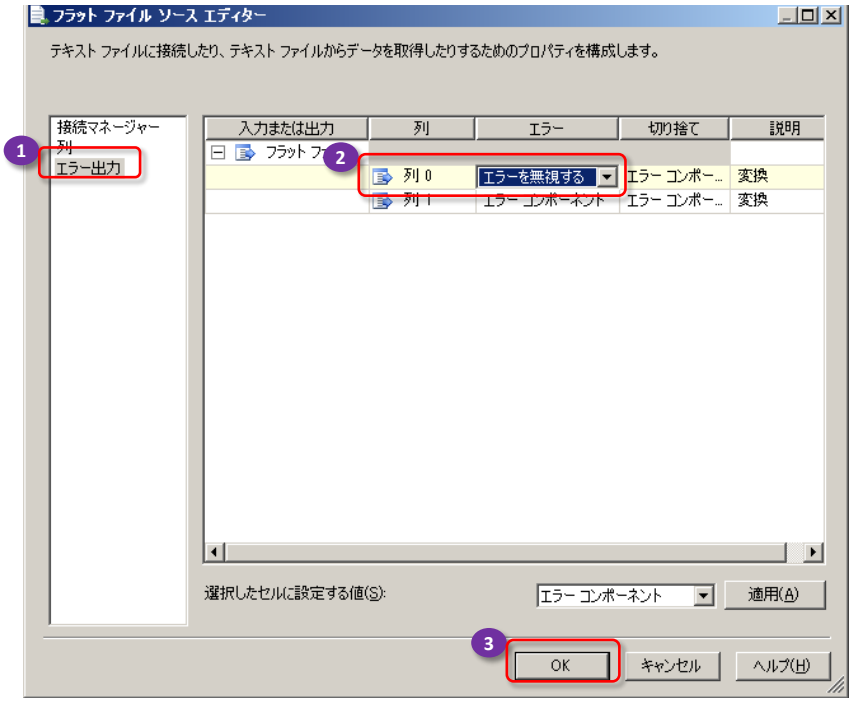
エラーを無視する

SSIS パッケージには、データ転送時にエラーが発生した場合でも、エラーを無視してデータ転送を続行させる機能があります。それでは、これを試してみましょう。

1. エラーを無視させるには、次のように配置した［**フラット ファイル ソース**］をダブル クリックして［**フラット ファイル ソース エディター**］ダイアログを表示します。

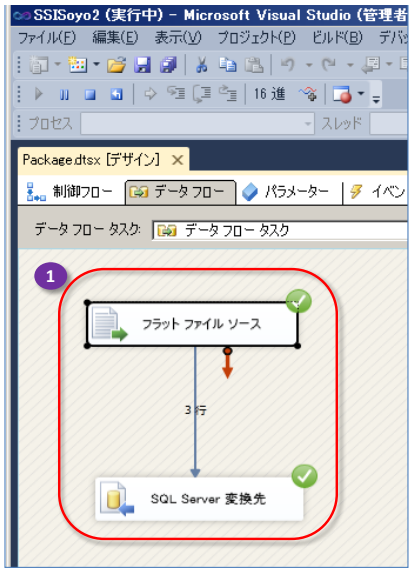


2. このダイアログでは、次のように［**エラー出力**］ページをクリックして、［**列 0**］の［**エラー**］ドロップダウン リストを「**エラーを無視する**」へ変更し、［**OK**］ボタンをクリックします。



転送の実行と結果

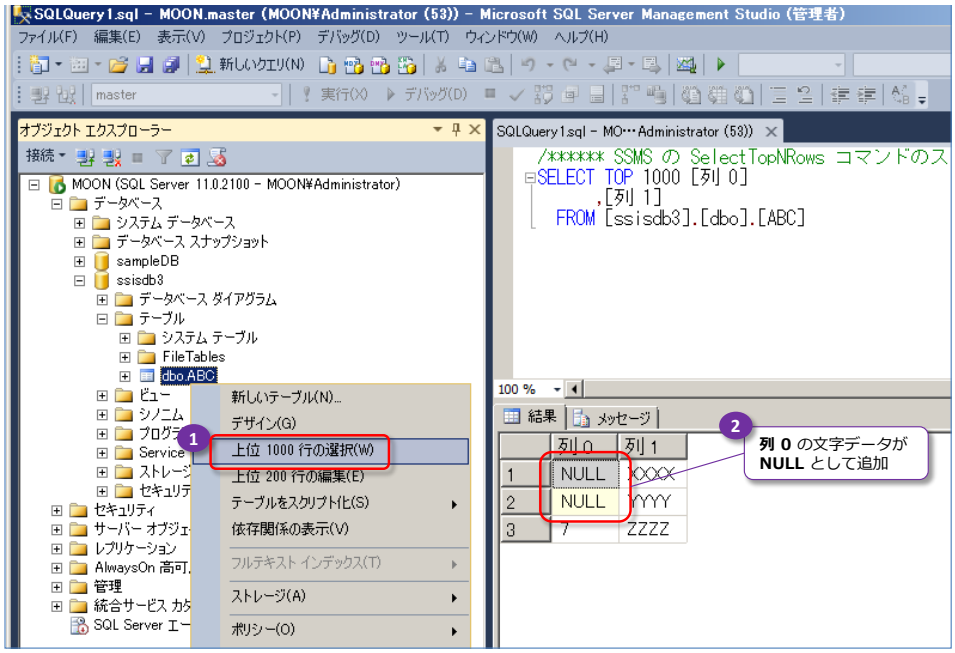
3. エラーが無視されることを確認するために、パッケージをデバッグ実行します。



今度は、すべてのコンポーネントに緑のチェックマークが付いて、データ転送が成功していることを確認できます。

4. 確認後、デバッグを終了します。

5. 転送されたデータを確認するために、Management Studio から、「**ABC**」テーブルを右クリックして、［**上位 1000 行の選択**］をクリックします。



MISS.txt のデータが追加されていることを確認できますが、「**列 0**」列の文字データは「**NULL**」として追加され、正しいデータ（**7, ZZZZ**）はきちんと転送されていることを確認できます。

Note： 列で NULL を許可していない場合

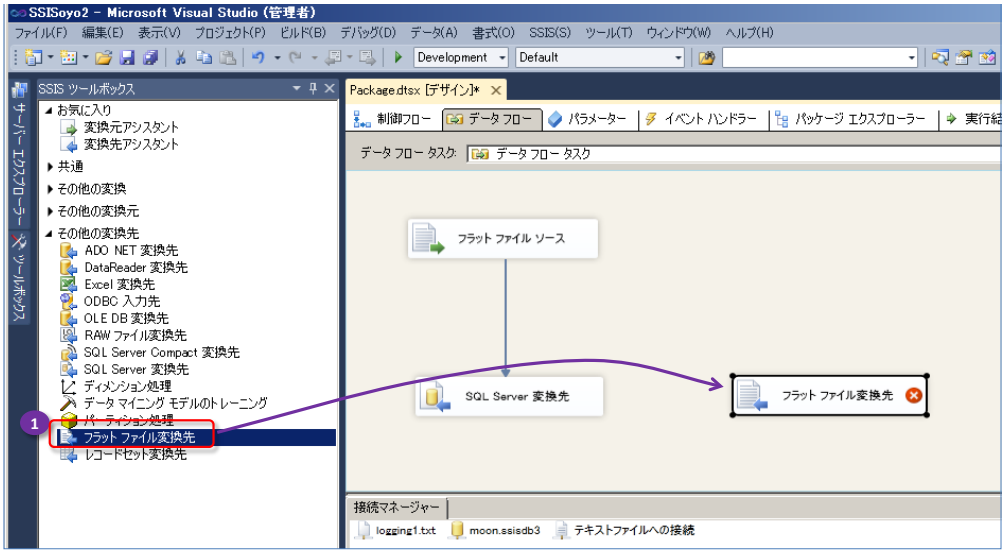
今回の例では、「**列 0**」列の文字データは NULL として挿入されていますが、「**列 0**」列で NULL を許可していない場合（**NOT NULL** の場合）はどうなると思いますか？ この場合は、エラーが発生したデータ行は追加されずに、正しいデータのみが追加されるようになります。

5.4 エラー情報をファイルへ書き込む

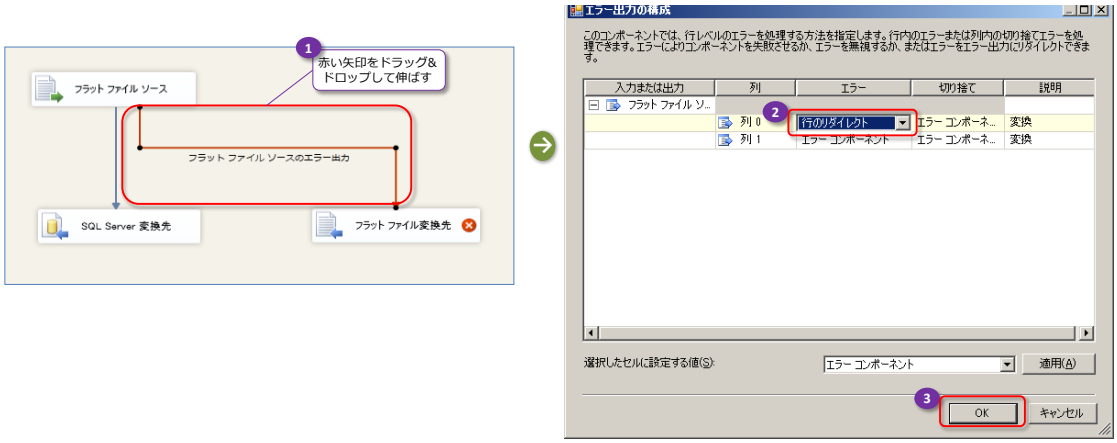
エラー情報をファイルへ書き込む

次に、「**列 0**」でエラーが発生したときのエラー情報（エラーが発生した行データとエラー情報）をファイルへ書き込むようにパッケージを変更してみましょう。

1. エラー情報をファイルへ書き込むようにするには、次のように SSIS ツールボックスから［**その他の変換先**］カテゴリにある［**フラット ファイル変換先**］をドラッグ＆ドロップして配置します。

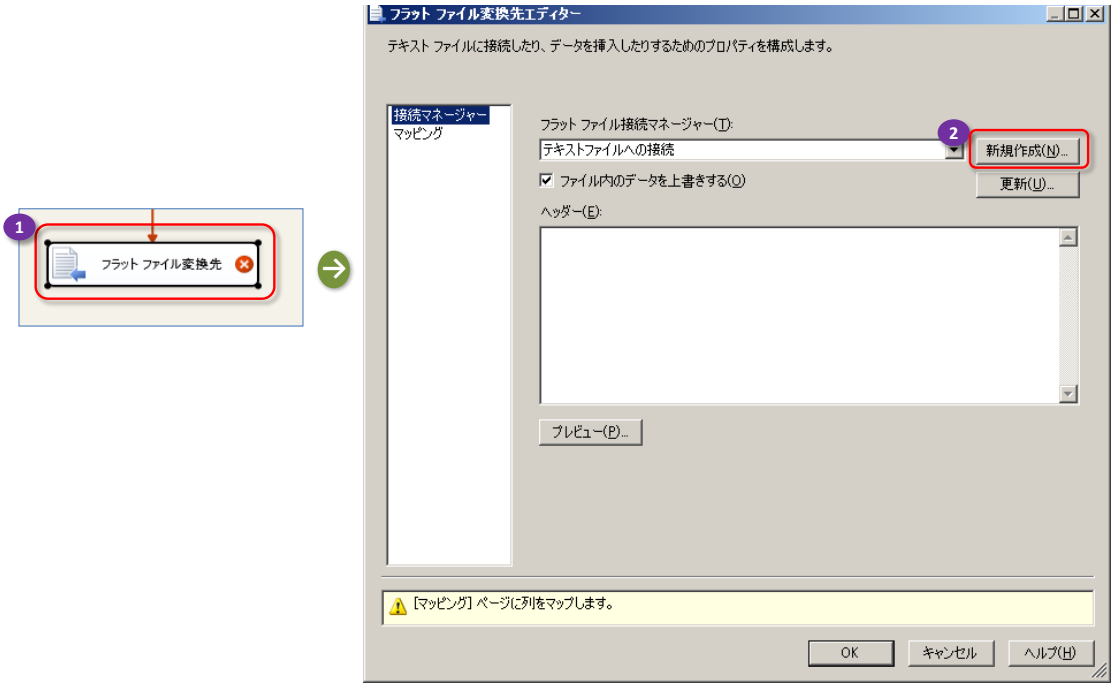


2. 次に、［**フラット ファイル ソース**］でエラーが発生した場合に、［**フラット ファイル変換先**］へ転送されるように、次のように［**フラット ファイル ソース**］をクリックして表示される "**赤い矢印**" を［**フラット ファイル変換先**］まで伸ばします。



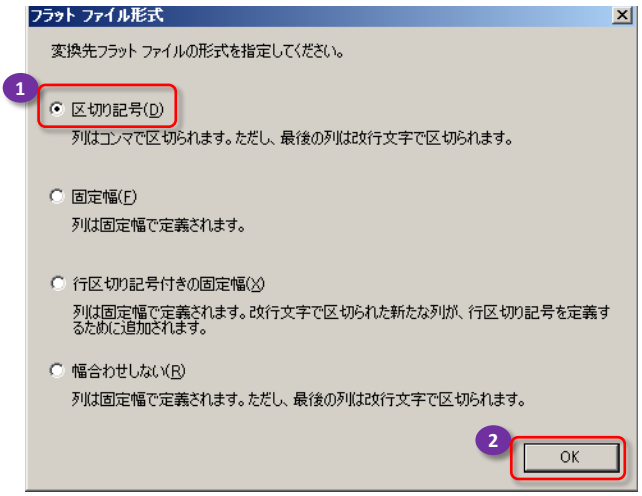
これにより、［**エラー出力の構成**］ダイアログが表示されるので、［**列 0**］の［**エラー**］列を「**エラーを無視する**」から「**行のリダイレクト**」へ変更します。これで、「**列 0**」列でエラーが発生した場合に、エラー情報を別のコンポーネントへリダイレクト（転送）できるようになります。

3. 次に、[**フラット ファイル変換先**] をダブル クリックして、リダイレクトされたエラー情報を書き込むファイルを設定します。

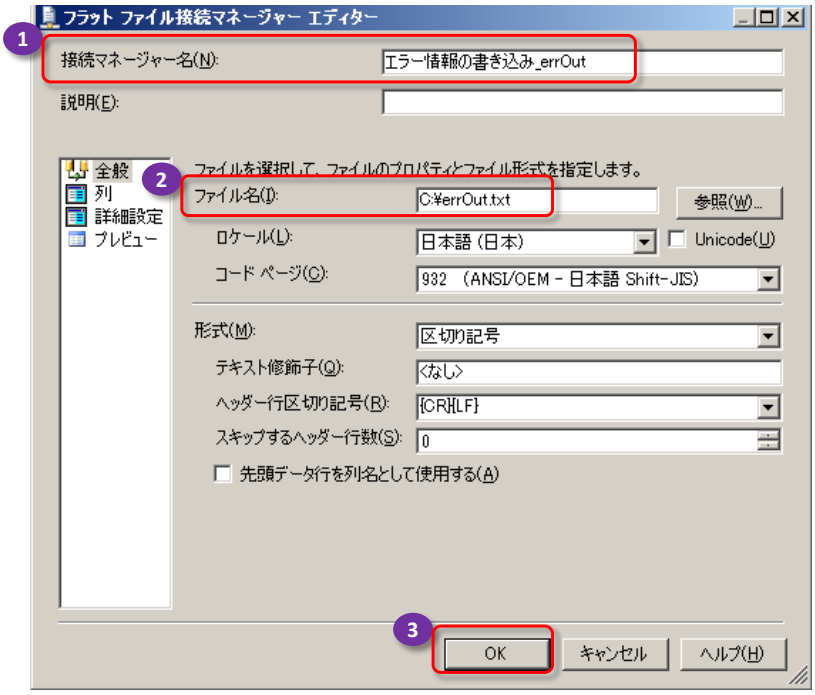


[**フラット ファイル変換先エディター**] ダイアログが表示されたら、[**新規作成**] ボタンをクリックします。

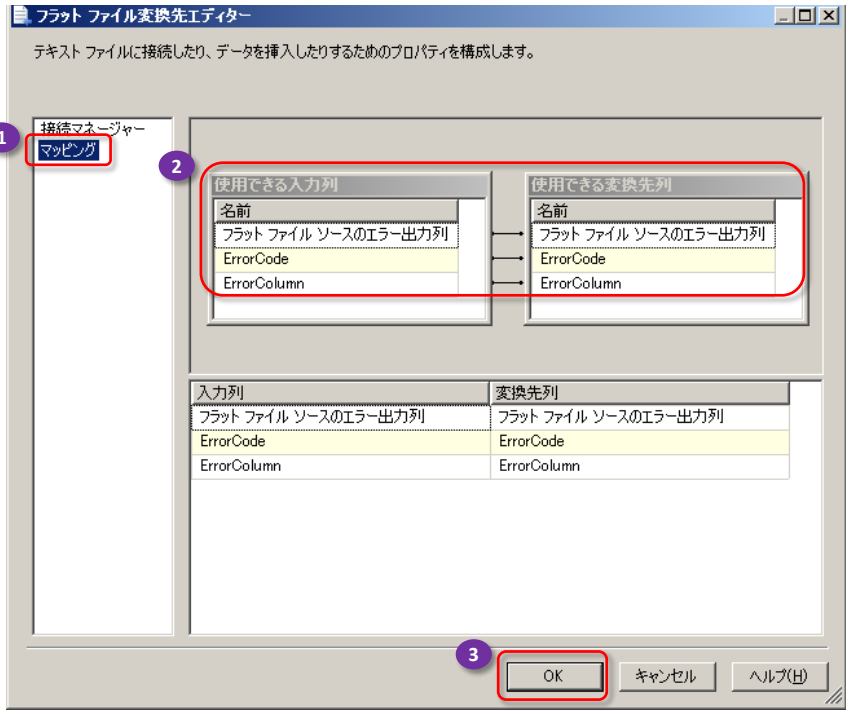
4. これにより、次のように [**フラット ファイル形式**] ダイアログが表示されて、書き込むファイルの書式を問われるので、「**区切り記号**」(カンマ区切り) が選択されていることを確認して、[**OK**] ボタンをクリックします。



5. 次に、[**フラット ファイル接続マネージャー エディター**] ダイアログが表示されるので、今回は、次のように [**接続マネージャー名**] へ「**エラー情報の書き込み_errOut**」、[**ファイル名**] へ「**C:¥errOut.txt**」と入力して、[**OK**] ボタンをクリックします。



6. ［**フラット ファイル変換先エディター**］ダイアログへ戻ったら、［**マッピング**］ページをクリックして、ファイルへ書き込むデータのマッピングを設定します。



受け取るデータ（使用できる入力列）には、［**フラット ファイル ソースのエラー出力列**］（エラーが発生した該当行）と、［**ErrorCode**］（エラー番号）、［**ErrorColumn**］（エラー列）があり、エラーに関する情報が転送されてくることを確認できます。この転送されたデータをそのままファイルへ書き込むように、各列が線で結ばれている（マッピングされている）ことを確認して、［**OK**］ボタンをクリックします。

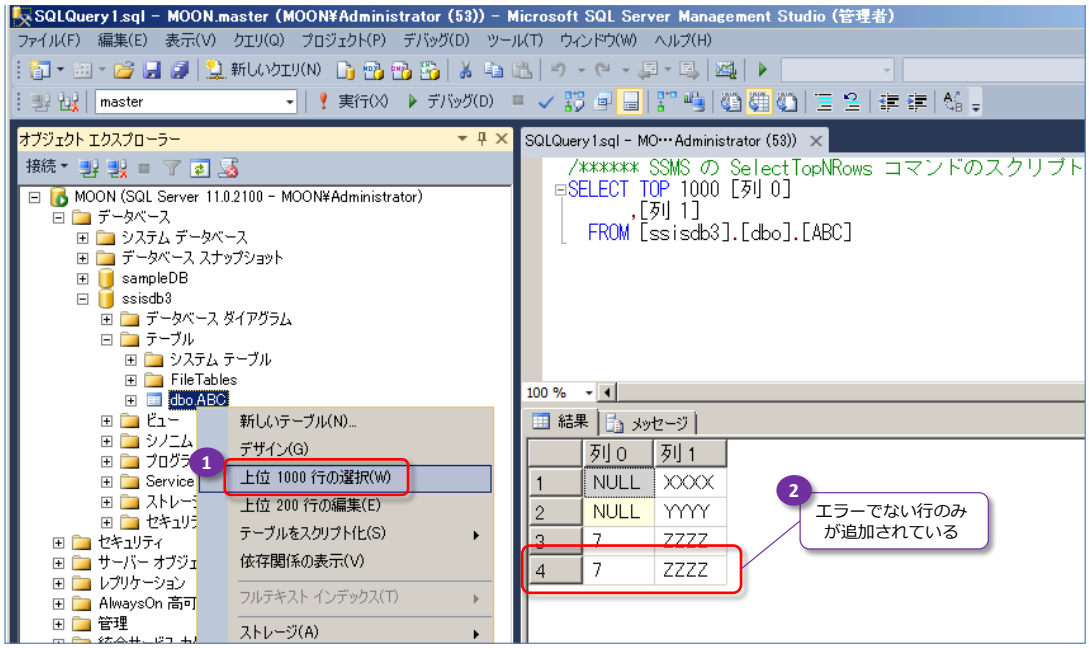
転送の実行と結果の確認

7. ここまでの設定を確認するために、パッケージをデバッグ実行します。

すべてのコンポーネントに緑のチェックマークが付いて、データ転送が成功していることを確認できます。また、**青色の矢印**（成功時の処理）には「**1 行**」、**赤色の矢印**（エラー時の処理）には「**2 行**」と表示されて、成功が 1 行、エラーが 2 行であったことを確認することができます。

8. 確認後、デバッグを終了します。

9. 転送されたデータを確認するには、Management Studio から、「**ABC**」テーブルを右クリックして、［**上位 1000 行の選択**］をクリックします。



今度は、正しいデータのみが追加され、エラーが発生した行は追加されていないことを確認できます。

10. 次に、**Windows エクスプローラー**から **C:¥errOut.txt** ファイルを開き、エラー情報を取得できたかどうかを確認します。

エラーが発生した該当行がそのまま格納され、カンマで区切られてエラー番号を取得できていることを確認できます。

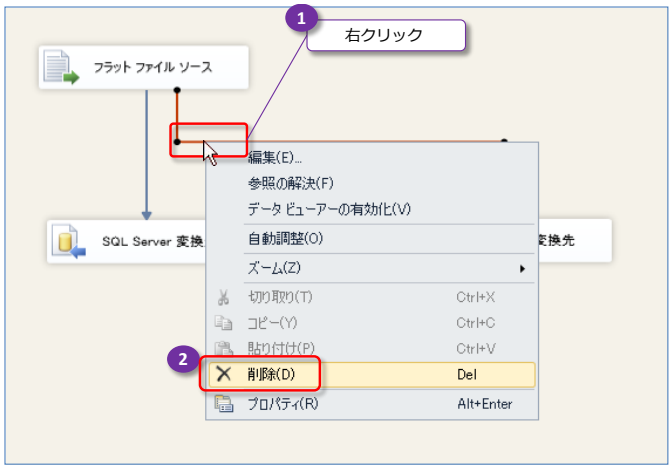
5.5 スクリプト コンポーネント（C#）によるエラー メッセージの取得

スクリプト コンポーネントによるエラー メッセージの取得

ここまでの手順では、「**列 0**」列のエラー発生時に「**エラー行**」と「**エラー番号**」をファイルへ書き込むことができましたが、エラー番号だけではどういったエラーなのかがよく分かりません。そこで、エラー番号をもとに**エラー メッセージ**を取得するようにパッケージを変更して、エラーの内容を分かりやすくしてみましょう。これを行うには、**スクリプト コンポーネント**を利用します。

それでは、これを試してみましょう。

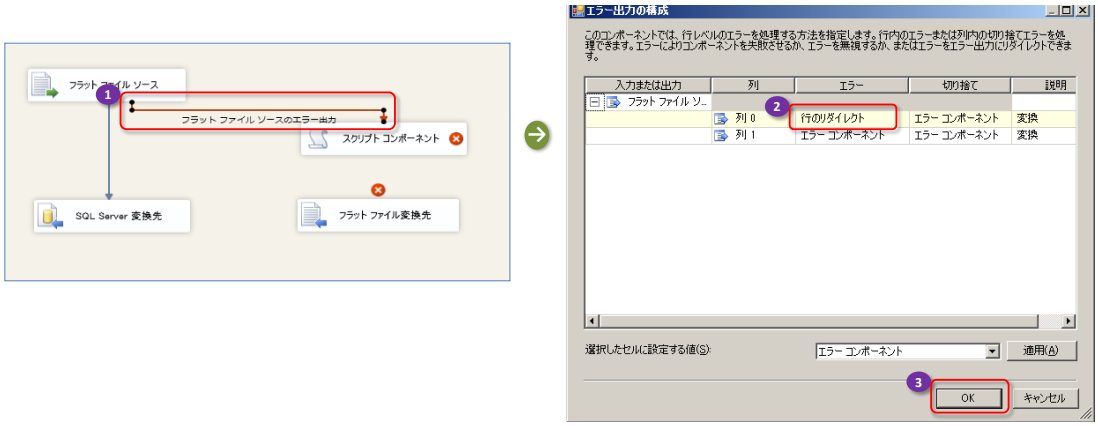
1. まずは、スクリプト コンポーネントを配置する前の作業として［**フラット ファイル ソース**］から［**フラット ファイル変換先**］へ伸びている「**赤い矢印**」を右クリックして、［**削除**］をクリックし、削除しておきます。



2. 次に、SSIS ツールボックスから［**共通**］カテゴリの［**スクリプト コンポーネント**］をドラッグ＆ドロップして［**フラット ファイル変換先**］の上部へ配置します。

これにより、［**スクリプト コンポーネントの種類を選択**］ダイアログが表示されるので、［**変換**］が選択されていることを確認して、［**OK**］ボタンをクリックします。

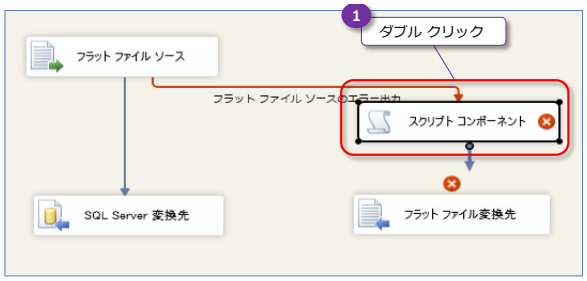
3. 次に、［**フラット ファイル ソース**］をクリックして表示される**赤い矢印**を［**スクリプト コンポーネント**］まで伸ばします。



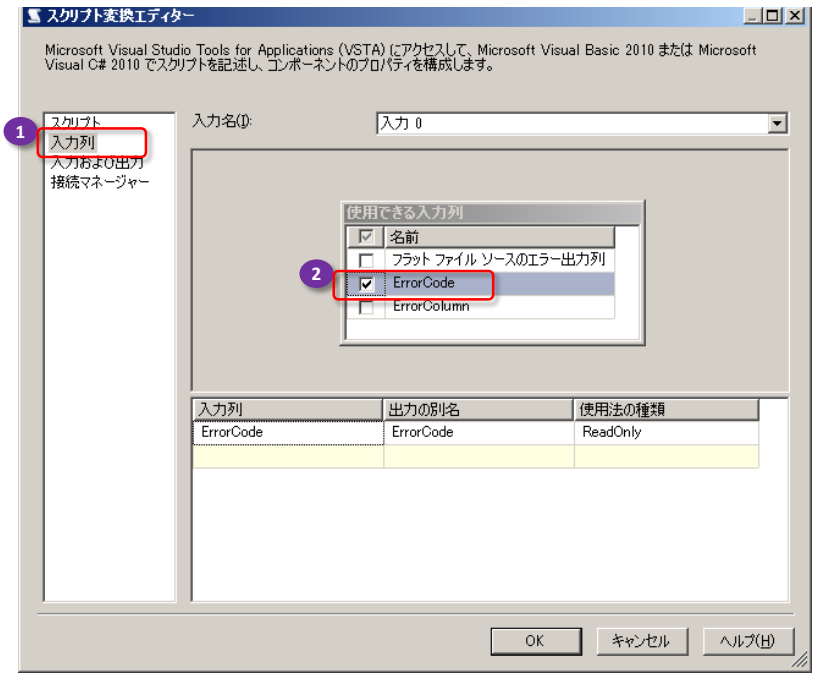
これにより、[**エラー出力の構成**] ダイアログが表示されるので、「**列 0**」列のエラーの情報を取得できるようにするために、「**列 0**」の [**エラー**] 列が「**行のリダイレクト**」になっていることを確認して、[**OK**] ボタンをクリックします。

スクリプトの作成（C# コードの記述）

4. 次に、[**スクリプト コンポーネント**] をダブル クリックします。

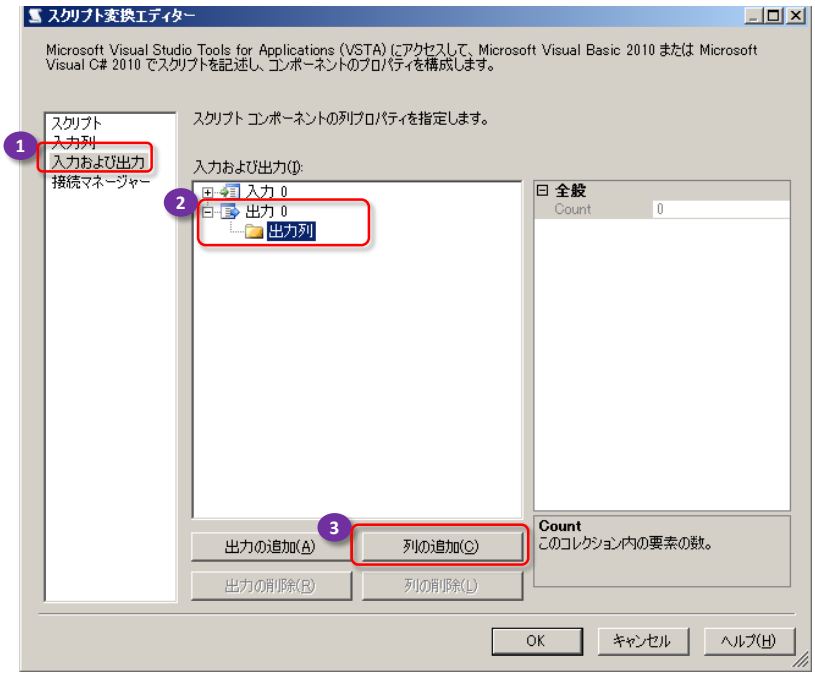


5. [**スクリプト変換エディター**] ダイアログが表示されたら、[**入力列**] ページをクリックして開き、[**ErrorCode**] にチェック マークをつけます。

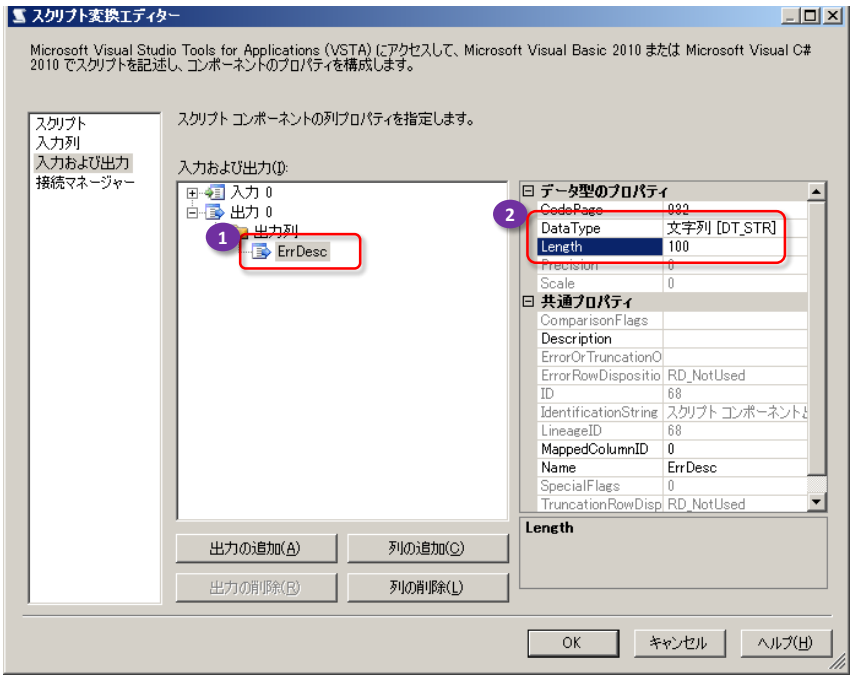


これにより、エラー発生時に［**フラット ファイル ソース**］からエラー番号を取得できるようになります。

6. 続いて、取得したエラー番号からエラー メッセージを取得するための列を追加します。［**入力および出力**］ページをクリックし、［**出力 0**］を展開して［**出力列**］フォルダーを選択した状態で、［**列の追加**］ボタンをクリックします。

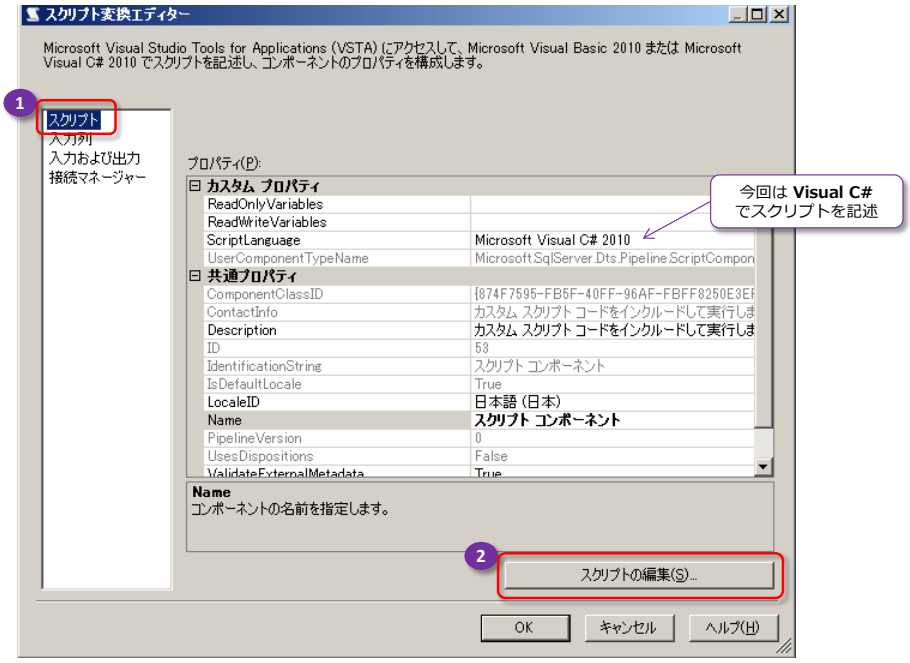


7. すると、次のように追加する列の名前を入力できるようになるので、今回は「**ErrDesc**」と入力します。



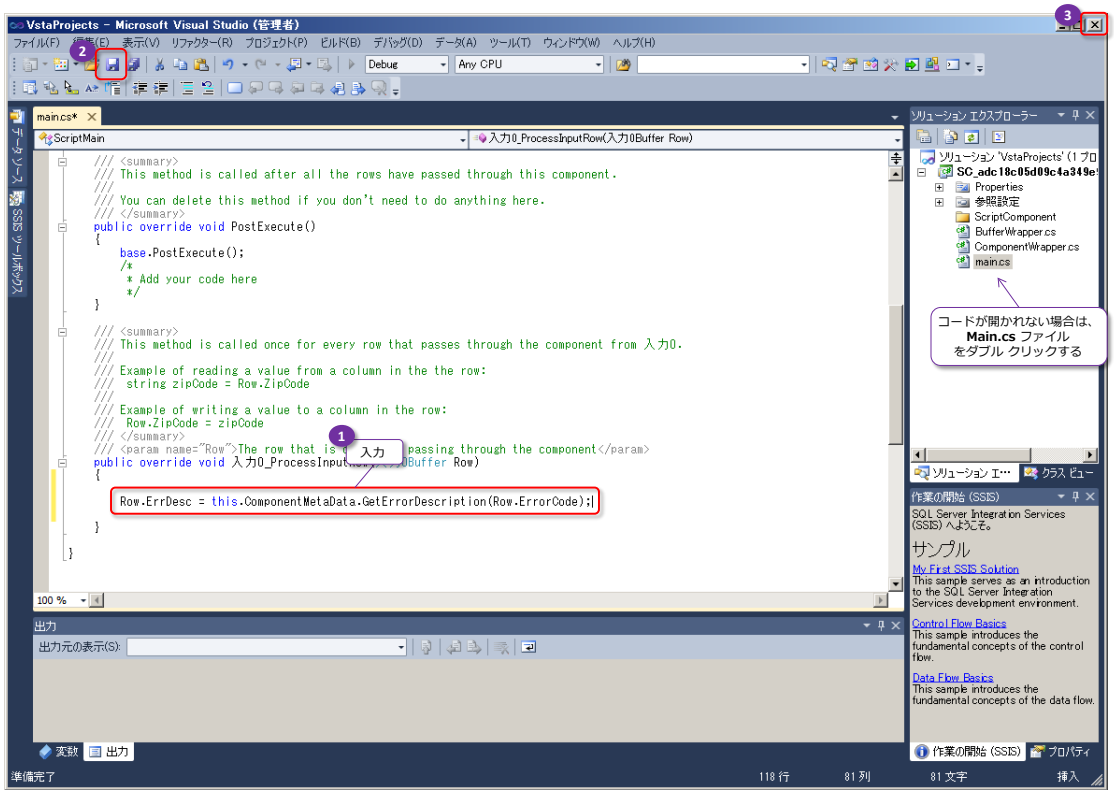
右側のプロパティ画面では、［**DataType**］を「**文字列 [DT_STR]**」、［**Length**］を「**100**」へ変更します。

8. 続いて、エラーの説明を取り出すスクリプトを記述するために、次のように［**スクリプト**］ページをクリックして、［**スクリプトの編集**］ボタンをクリックします。



9. これにより、「**スクリプト エディター**」が起動されるので、「**public override void 入力0_ProcessInputRow(～)**」内に、次のコード（**C#**）を入力します。

```
Row.ErrDesc = this.ComponentMetaData.GetErrorDescription(Row.ErrorCode);
```

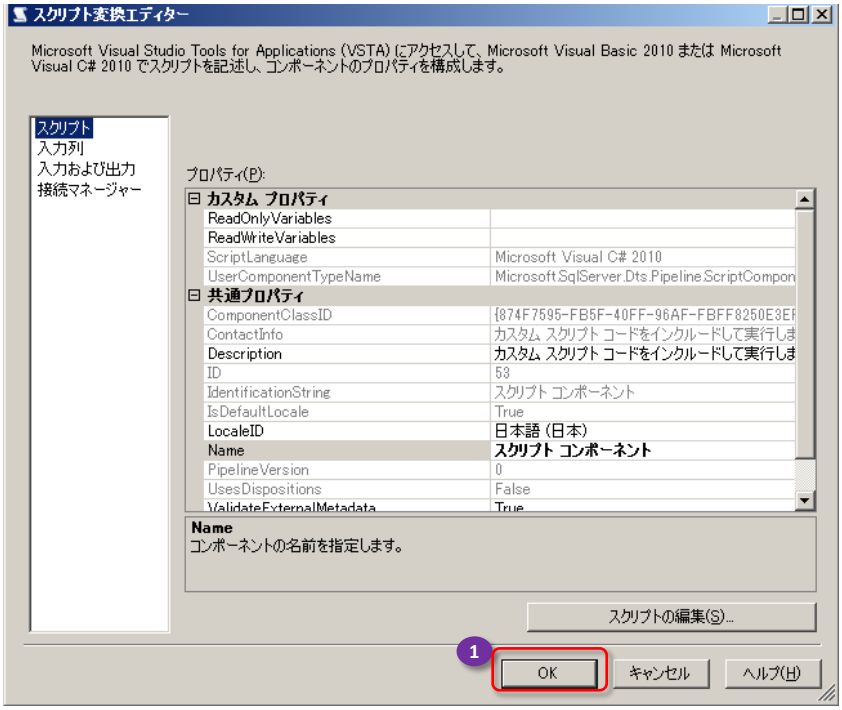


このコード内の **Row** は、今まさに処理をしている該当行を意味しています。「**Row.Error**

Code」と指定することで、前の手順で受け取った **ErrorCode** 列のデータ（エラー発生時のエラー番号）を取得でき、「**GetErrorDescription**」メソッドへこの番号を与えることで、エラー番号に対応したエラーの説明（エラー メッセージ）を取得できるようになります。また、「**Row.ErrDesc**」は、前の手順で作成した「**ErrDesc**」列で、この列へエラー メッセージを代入するという意味です。なお、Visual Basic でコードを記述する場合は、「**this**」を「**Me**」へ変更して、最後のセミコロン（**;**）を削除します。

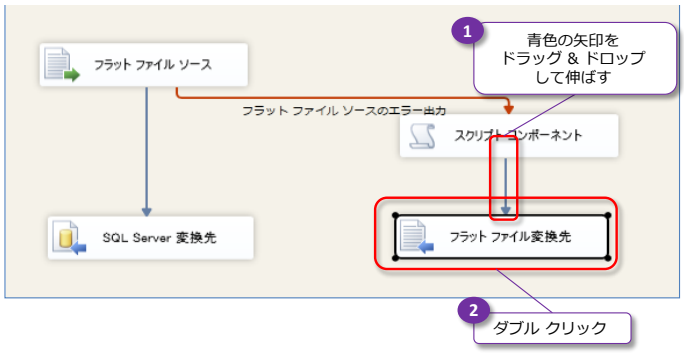
入力後、保存してスクリプト エディターを閉じます。

10. ［**スクリプト変換エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックして閉じます。



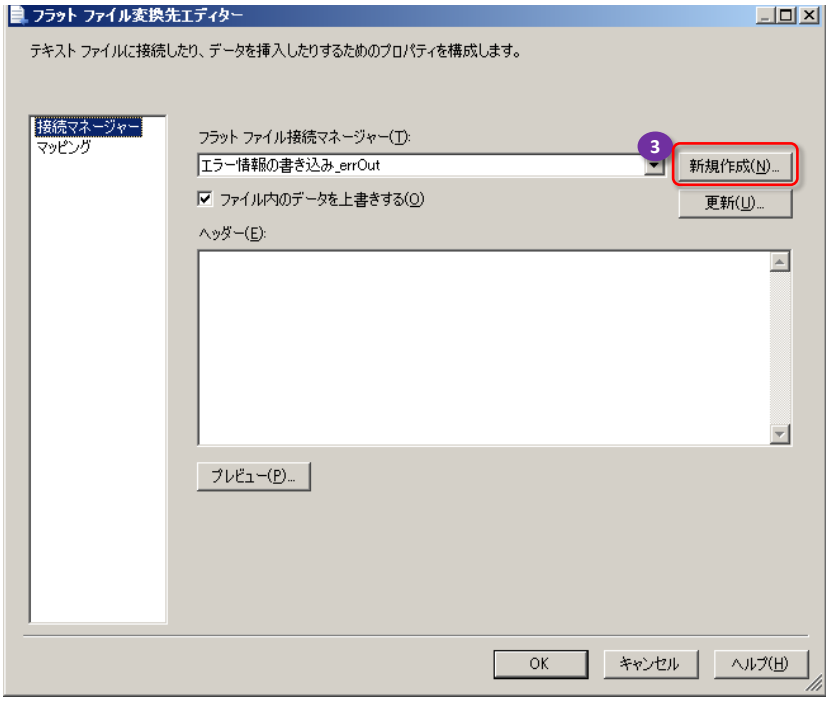
エラーの説明をファイルへ書き込む

11. スクリプト コンポーネントで取得したエラーの説明をファイルへ書き込むようにするために、次のように［**スクリプト コンポーネント**］の青色の矢印を［**フラット ファイル変換先**］まで伸ばします。続いて、［**フラット ファイル変換先**］をダブル クリックします。

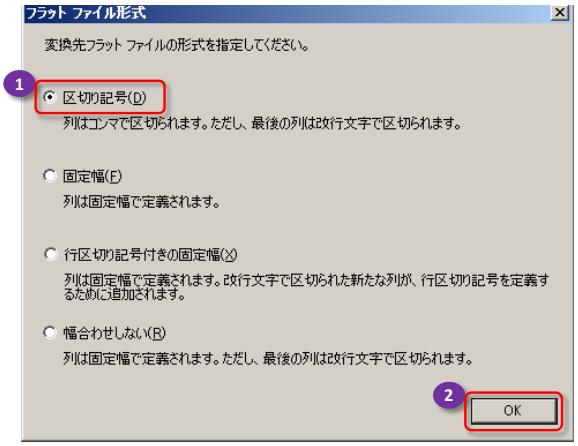


これにより、［**フラット ファイル変換先エディター**］ダイアログが表示されるので、［**新規作**

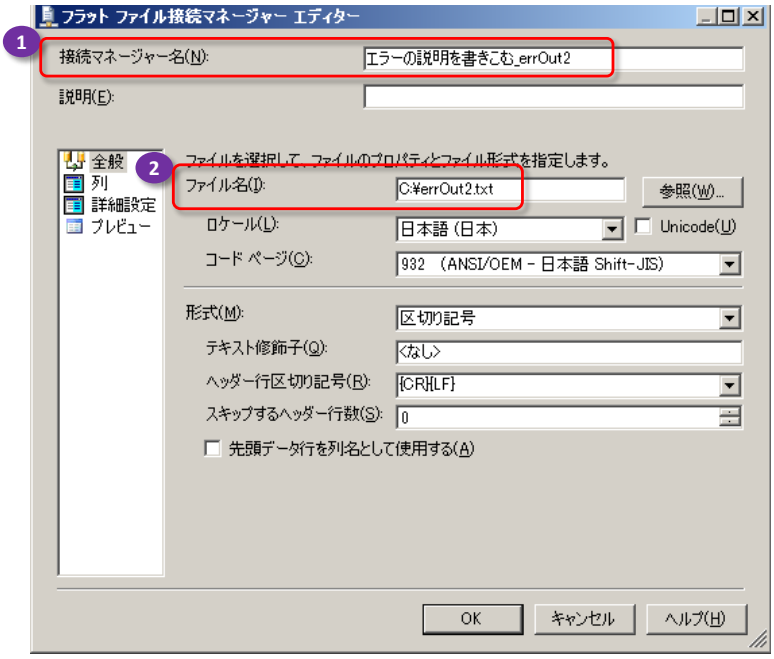
成］ボタンをクリックします。



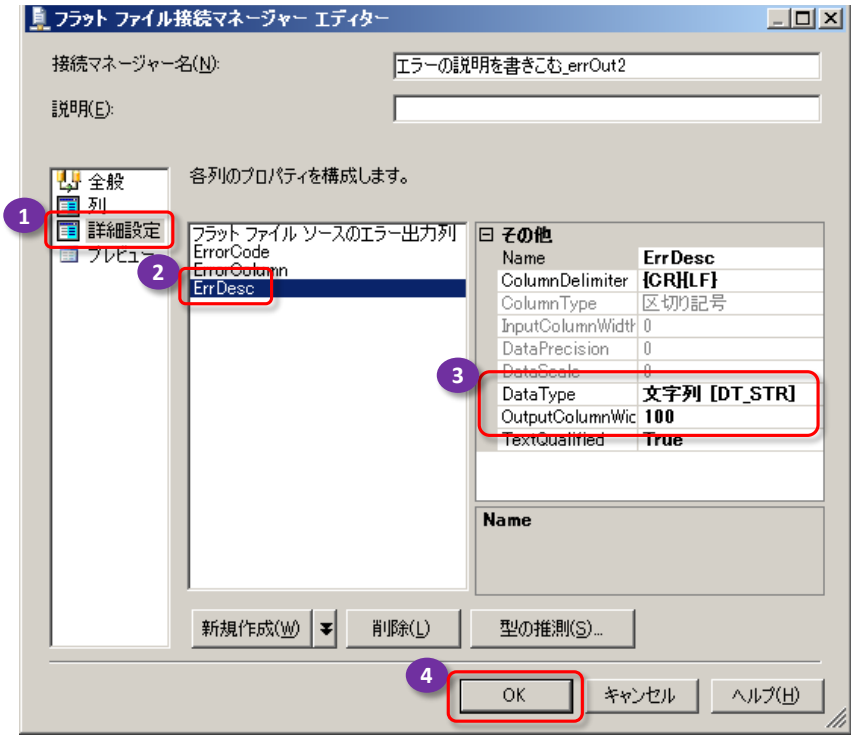
12. ［**フラット ファイル形式**］ダイアログが表示されたら、［**区切り記号**］（カンマ区切り）が選択されていることを確認して、［**OK**］ボタンをクリックします。



13. 次に、［**フラット ファイル接続マネージャー エディター**］ダイアログが表示されるので、今回は［**接続マネージャー名**］へ「**エラーの説明を書き込む_errOut2**」とし、［**ファイル名**］へ「**C:¥errOut2.txt**」と入力します。

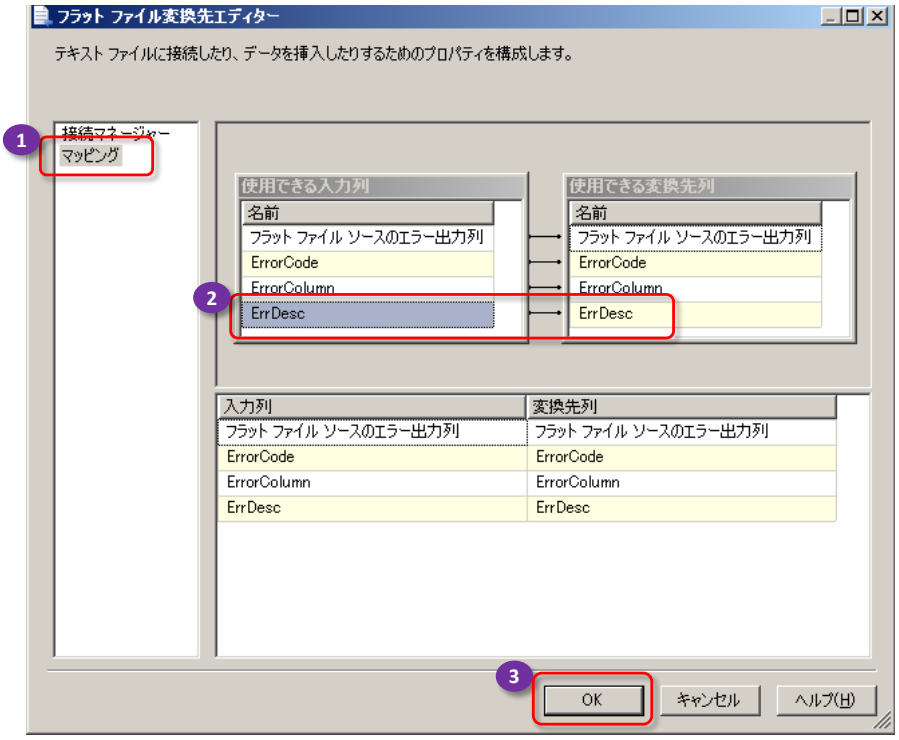


14. 続いて、［**詳細設定**］ページをクリックして、列のプロパティを確認します。



［**ErrDesc**］列を選択して、前の手順で設定したように、［**DataType**］が「**文字列**」で、［**OutputColumnWidth**］が「**100**」となっていることを確認し、［**OK**］ボタンをクリックします。

15. ［**フラット ファイル変換先エディター**］ダイアログへ戻ったら、［**マッピング**］ページをクリックして、ファイルへ書き込むデータのマッピングを設定します。



［**使用できる入力列**］の「**ErrDesc**」列を、［**使用できる変換先列**］の「**ErrDesc**」列へドラッグ＆ドロップして、スクリプト コンポーネントで取得したエラー メッセージをファイルへ書き込むようにします。

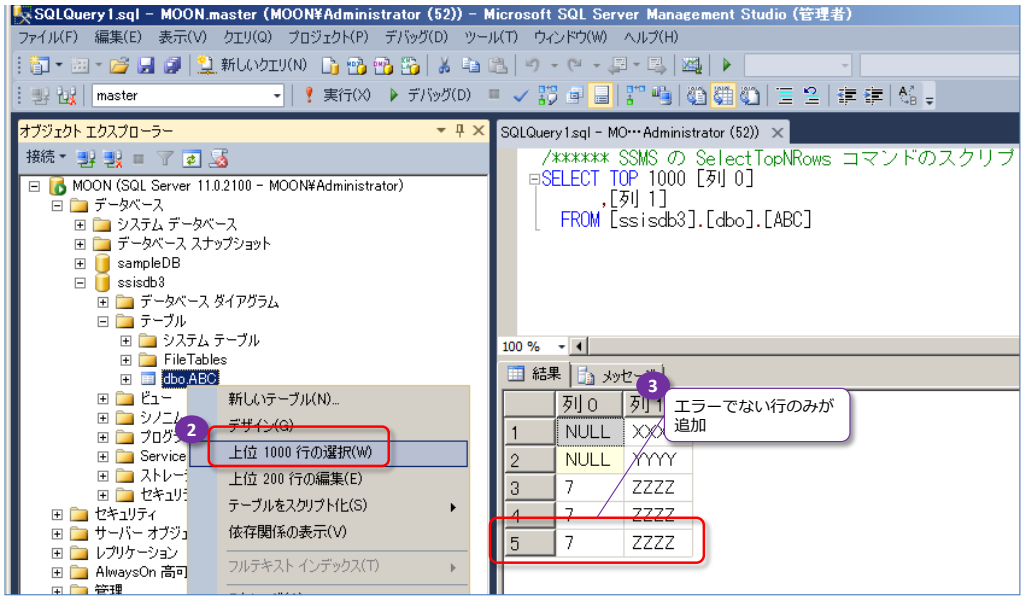
転送の実行と結果の確認

16. パッケージをデバッグ実行して、ここまでの設定を確認します。

すべてのコンポーネントに緑のチェックマークが付いて、データ転送が成功したことを確認できます。

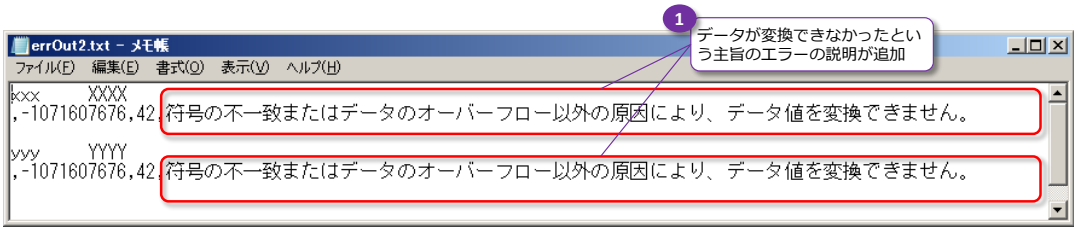
17. 確認後、デバッグを終了します。

18. 転送されたデータを確認するために、Management Studio から、［**ABC**］テーブルを右クリックして、［**上位 1000 行の選択**］をクリックします。



正しいデータのみが追加され、エラーが発生した行は追加されていないことを確認できます。

19. 次に、エクスプローラーから **C:¥errOut2.txt** ファイルを開いて、内容を確認します。



今度は、エラー番号に加えて、エラー メッセージ（エラーの説明）も取得できていることを確認できます。このように、スクリプト コンポーネントを利用すると、エラーが発生した該当行を取得して、かつエラーの内容を確認できるようになるので、大変便利です。

20. 最後に、[**ファイル**] メニューの [**すべてを保存**] をクリックして、**SSISoyo2** プロジェクトを保存しておきます。このプロジェクトは、次の Step でも引き続き使用します。

STEP 6. Foreach ループと ブレークポイント

この STEP では、複数ファイルを読み込む際に便利な Foreach ループ コンテナーや、デバッグ機能のブレークポイントの設定方法などを説明します。

この STEP では、次のことを学習します。

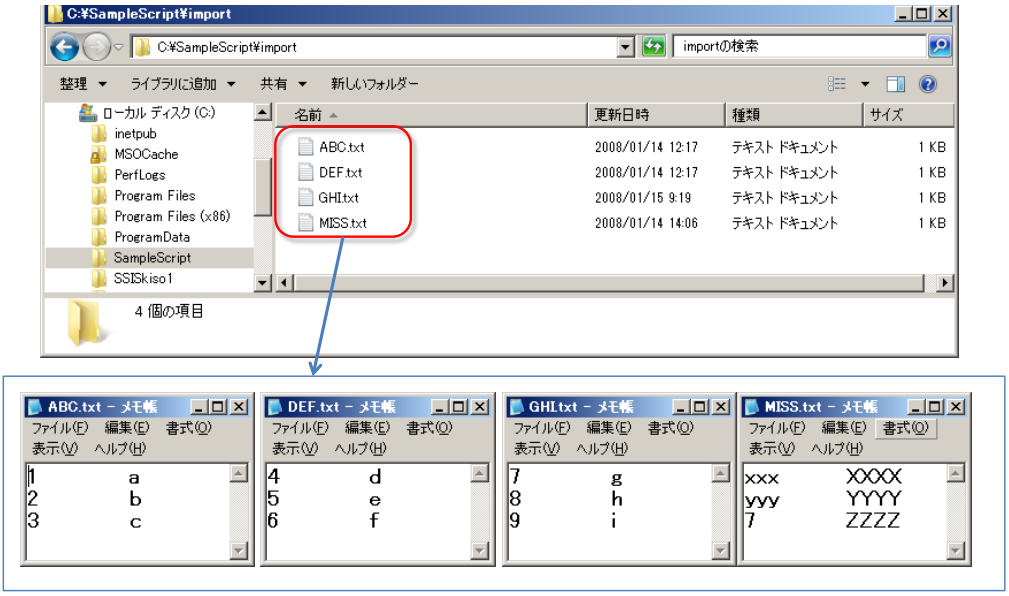
- ✓ Foreach ループ コンテナーによる複数ファイルの読み取り
- ✓ ブレークポイントの設定

6.1 Foreach Loop コンテナーによる複数ファイルの読み取り

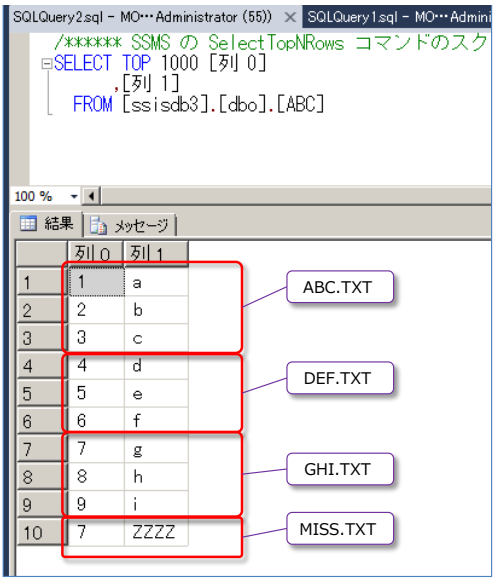
Foreach Loop コンテナーによる複数ファイルの読み取り

Foreach Loop コンテナーを利用すると、同じフォルダー内の複数のファイルのデータをまとめて読み取って、転送させることができるようになります。この Step では、前の Step で作成した「**SSISoyo2**」プロジェクトへ Foreach Loop コンテナーを追加して、サンプル スクリプトにある「**import**」フォルダー内のテキスト ファイルをすべて取り込む手順を説明します。

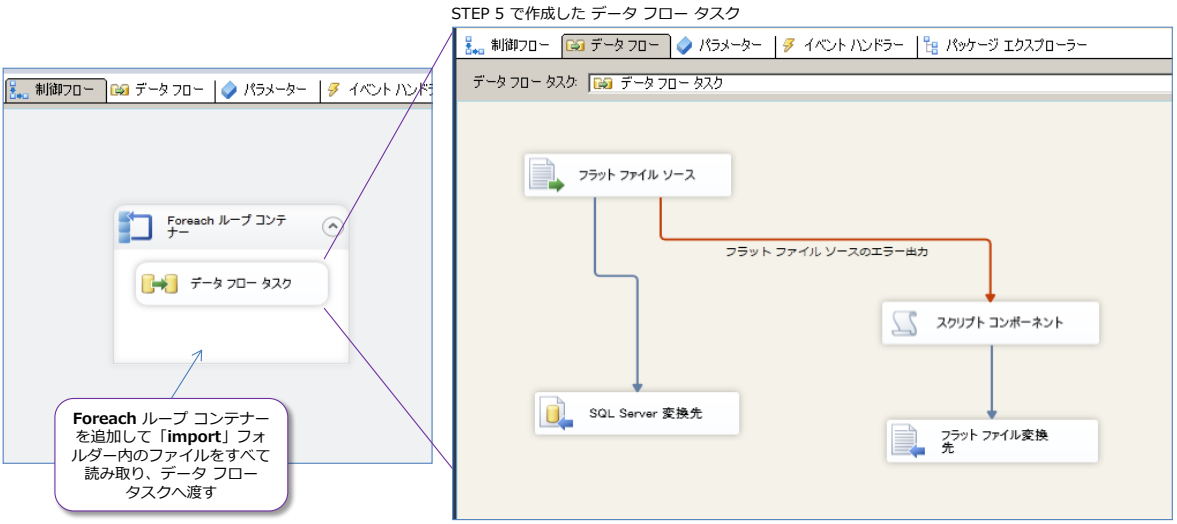
転送するデータ



転送後のデータ

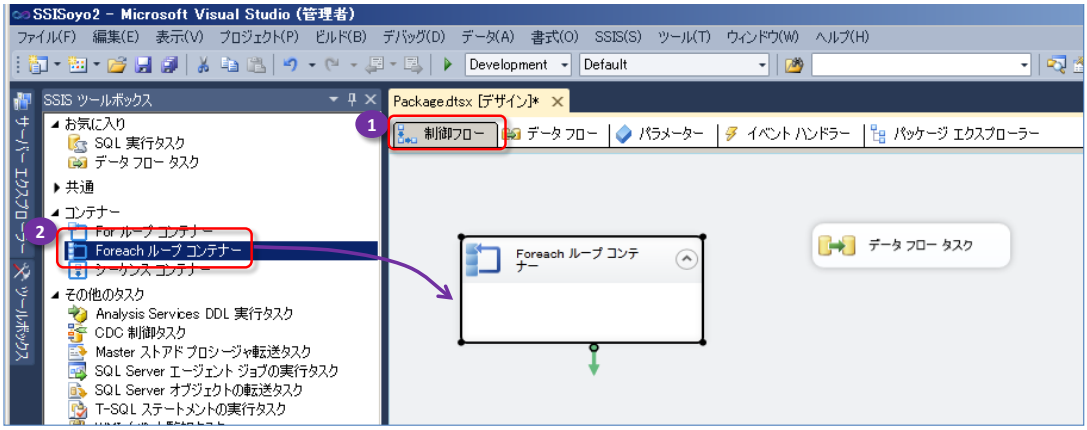


作成するパッケージ

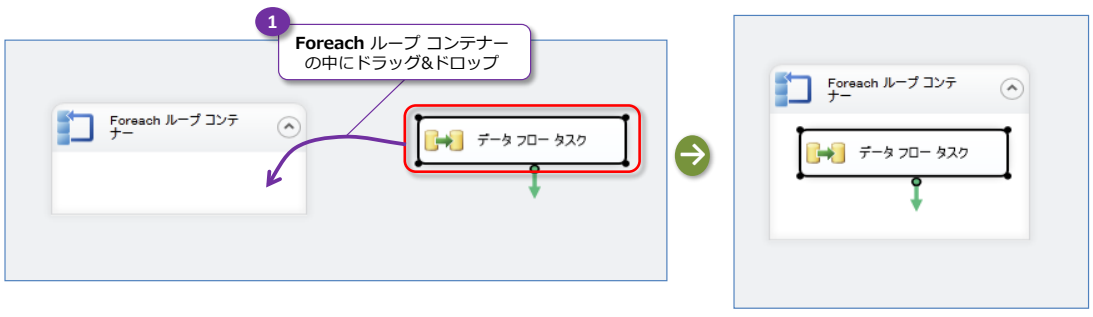


Foreach Loop コンテナーの配置

1. Foreach ループ コンテナーを利用するには、次のように［**制御フロー**］タブで、SSIS ツールボックスの［**コンテナー**］カテゴリの中から［**Foreach ループ コンテナー**］をドラッグ＆ドロップして配置します。

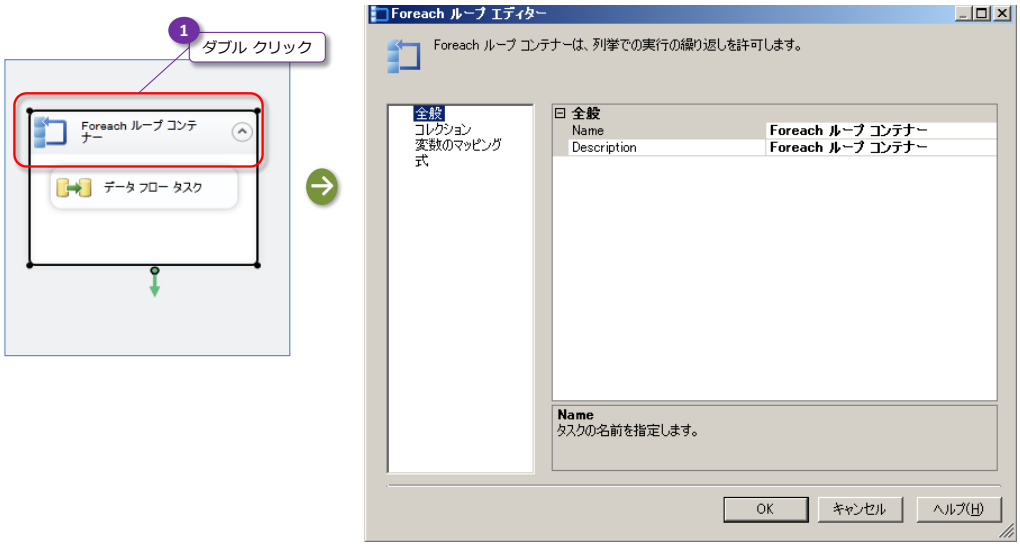


2. 次に、［**データ フロー タスク**］を［**Foreach ループ コンテナー**］の枠の中へドラッグ＆ドロップして移動します。



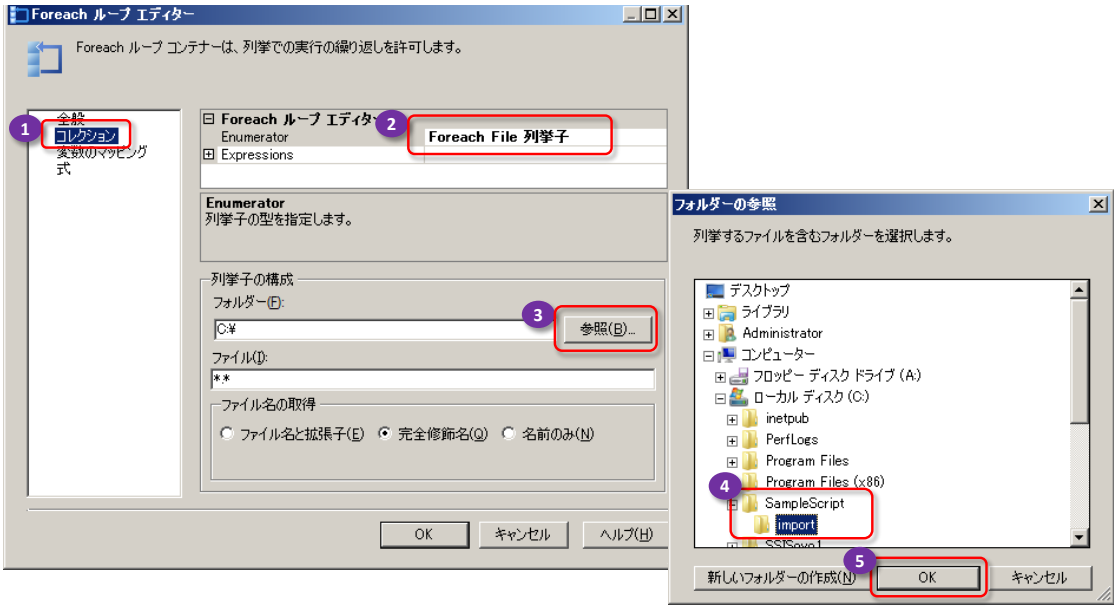
Foreach ループ コンテナーの対象の設定

3. 次に、Foreach ループ コンテナーの対象（繰り返し読み取るデータ）を設定するために［**Foreach ループ コンテナー**］をダブル クリックして、［**Foreach ループ エディター**］ダイアログを開きます。



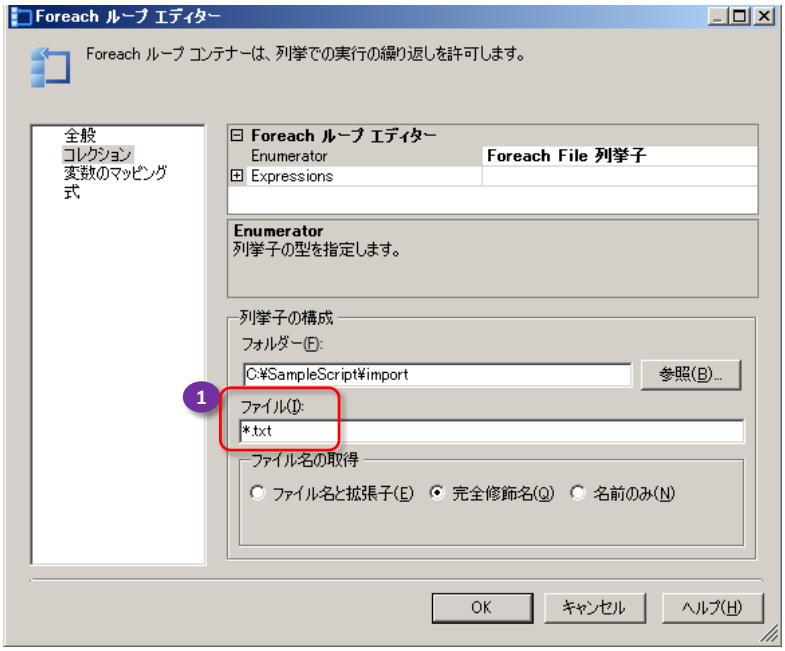
4. このダイアログでは、［**コレクション**］ページをクリックします。

今回はサンプル スクリプトの中にある「**import**」フォルダーのテキスト ファイルを対象とするので、次のように［**Enumerator**］（列挙子）で「**Foreach File 列挙子**」を選択し、［**フォルダー**］で［**参照**］ボタンをクリックます。



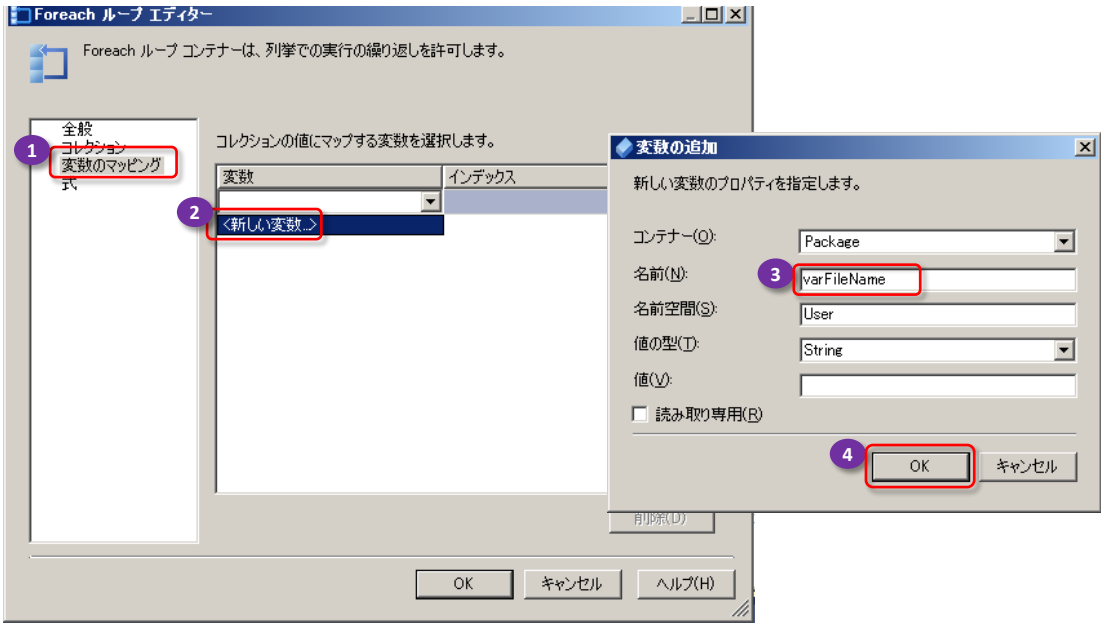
［**フォルダーの参照**］ダイアログが表示されたら、サンプル スクリプト内にある「**import**」フォルダーを選択して、［**OK**］ボタンをクリックします。

5. ［**Foreach ループ エディター**］へ戻ったら、次のように［**ファイル**］へ「***.txt**」と入力します。



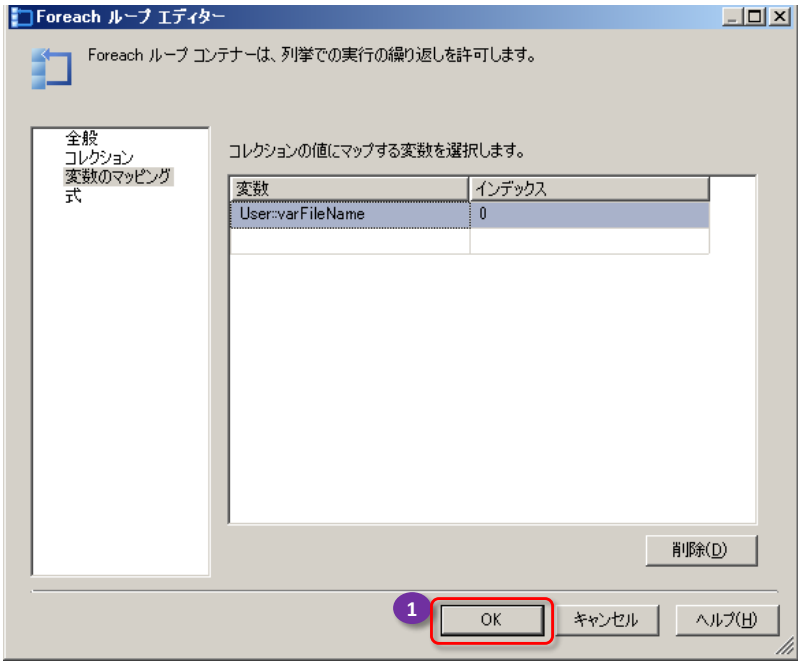
これで、**import** フォルダー内の拡張子が「**.txt**」のファイルをすべて読み取れるようになります。

6. 次に、読み取ったファイルの名前（パス）を変数へ割り当てるために、次のように［**変数のマッピング**］ページをクリックして開き、［**変数**］で「**新しい変数**」を選択します。



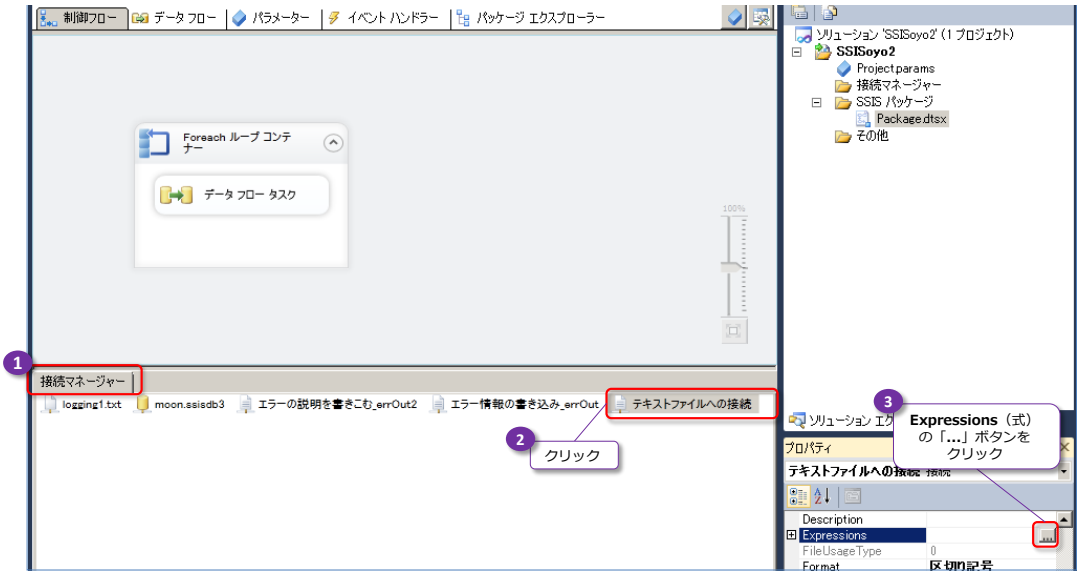
これにより、［**変数の追加**］ダイアログが表示されるので、［**名前**］へ任意の変数名（**varFileName** など）を入力して、［**OK**］ボタンをクリックします。

7. ［**Foreach ループ エディター**］へ戻ったら、［**OK**］ボタンをクリックして閉じます。



変数をファイルの接続情報に割り当てる

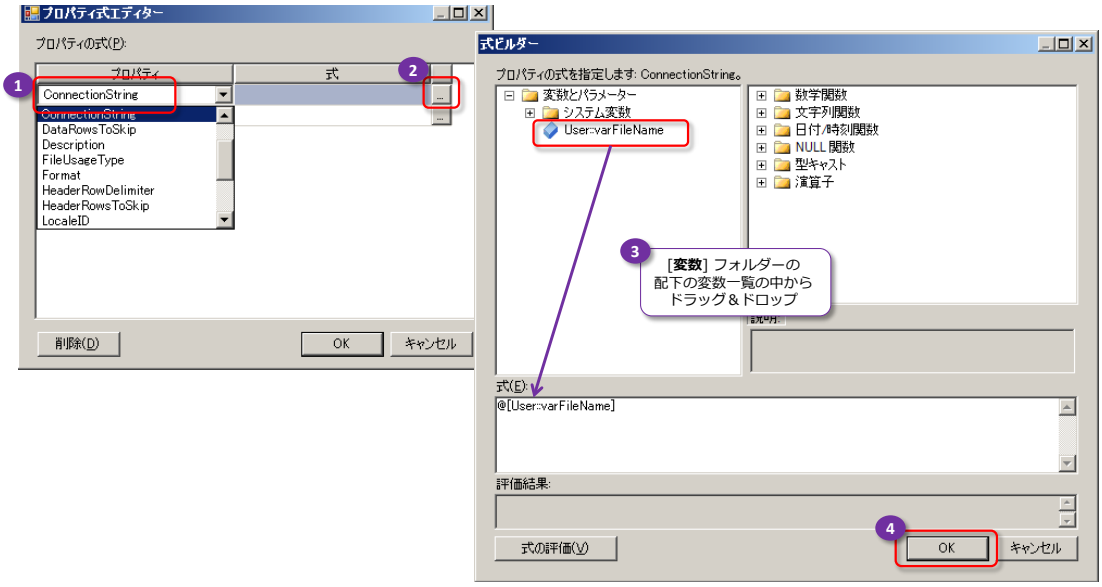
8. 次に、設定した変数（Foreach ループ タスクで読み取ったファイルのパス）を転送元のファイルとして指定するために、次のように画面下部の［**接続マネージャー**］タブで「**テキストファイルへの接続**」をクリックします。



この「**テキストファイルへ接続**」は、「**フラット ファイル ソース**」で利用していた「**MISS.txt**」ファイルへの接続に利用していたものです。

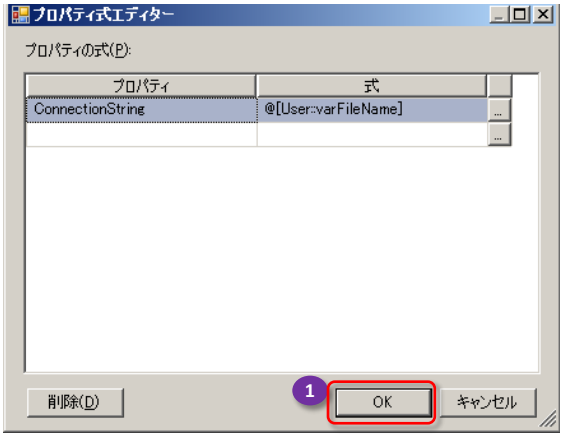
続いて、［**プロパティ**］ウィンドウで、［**Expressions**］を選択して、「**...**」ボタンをクリックします。

9. これにより、［**プロパティ式エディター**］ダイアログが表示されるので、次のように［**プロパティ**］で「**ConnectionString**」を選択して、「**...**」ボタンをクリックします。



［**式ビルダー**］ダイアログが表示されたら、［**変数**］フォルダーを展開して表示される変数の一覧の中から「**User::varFileName**」を［**式**］へドラッグ＆ドロップして、［**OK**］ボタンをクリックします。

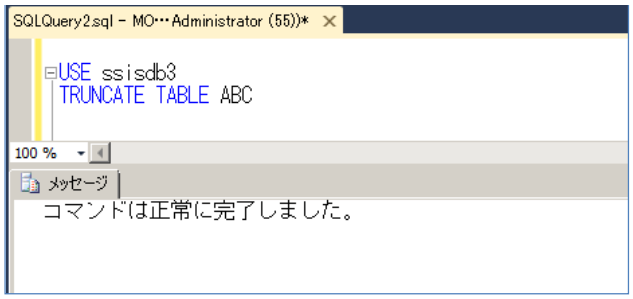
10. ［**プロパティ式エディター**］へ戻ったら、［**OK**］ボタンをクリックして閉じます。



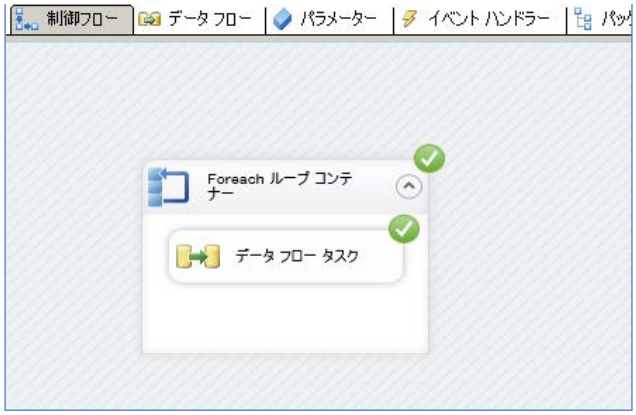
これにより、フラット ファイル ソースの接続先（**ConnectionString**）を変数（Foreach ループ タスクで読み取ったファイルのパス）へ割り当てることができます。

転送の実行と結果の確認

11. データの転送の結果を確認しやすくするために、まずは、次のように **Management Studio** の**クエリ エディター**から「**ssisdb3**」データベースの「**ABC**」テーブルに対して、**TRUNCATE TABLE** ステートメントを実行して、全行を削除しておきます。



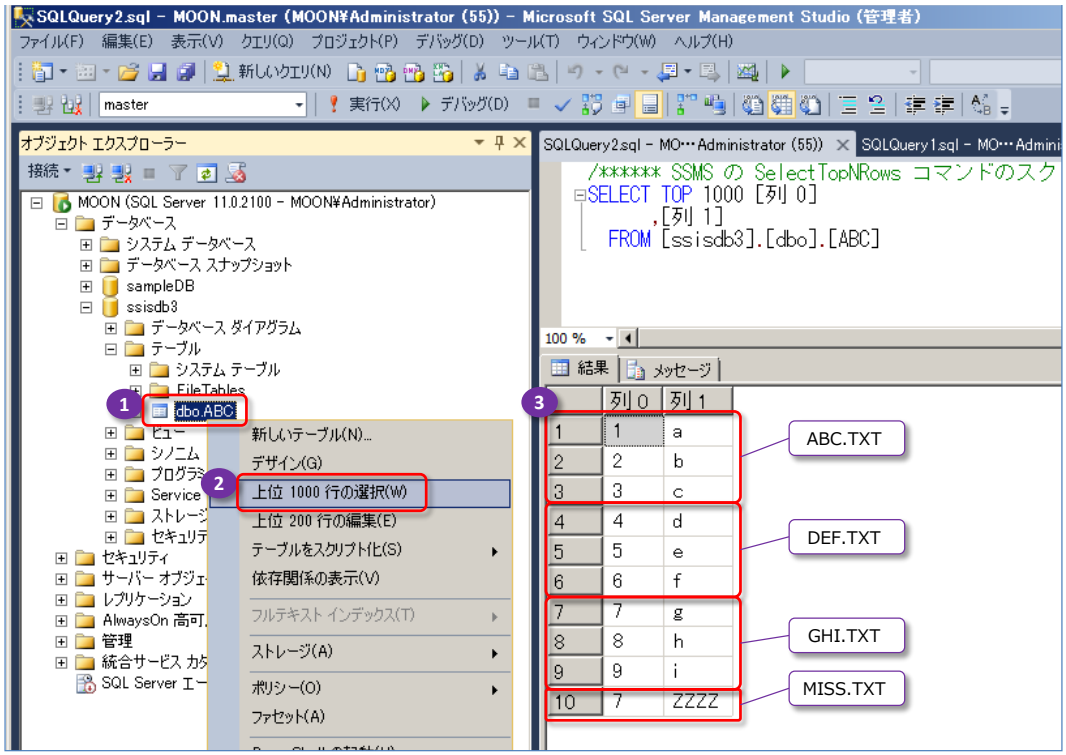
12. **SSIS デザイナー**へ戻り、パッケージをデバッグ実行します。



すべてのタスクに緑のチェックマークが付いて、データの転送が成功していることを確認できます。

13. 確認後、デバッグを終了します。

14. 転送されたデータを確認するために、Management Studio から、[**ABC**] テーブルを右クリックして、[**上位 1000 行の選択**] をクリックします。



4 つのファイルのデータがすべて追加されていることを確認できます。このように、Foreach ループ コンテナーを利用すると、複数のファイルをまとめて転送することができるので、大変便利です。

15. 最後に、[**ファイル**] メニューの [**すべてを保存**] をクリックして、**SSISoyo2** プロジェクトを保存しておきます。

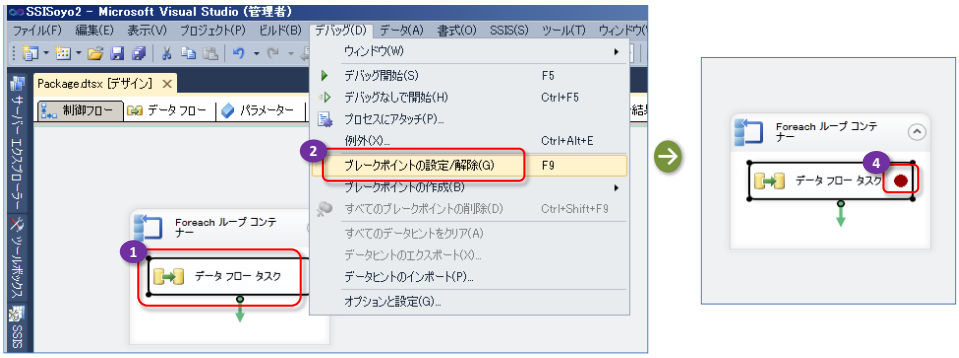
6.2 ブレークポイントを利用したステップ実行

ブレークポイントを利用したステップ実行

SSIS デザイナーでは、**ブレークポイント**を設定して、タスクを 1 つ 1 つステップ実行することも可能です。これを利用すると、Foreach ループ コンテナーが読み取ったファイル名（変数）などを確認することができるので、大変便利です。

それでは、これを試してみましょう。。

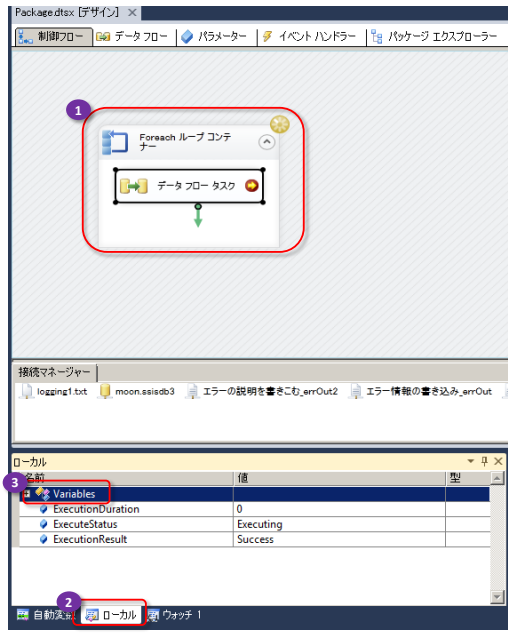
1. ブレークポイントを設定するには、次のように［**データ フロー タスク**］を選択し、［**デバッグ**］メニューの［**ブレークポイントの設定/解除**］をクリックします。



これにより、［**データ フロー タスク**］へ **"赤丸"** のアイコンが追加されます。

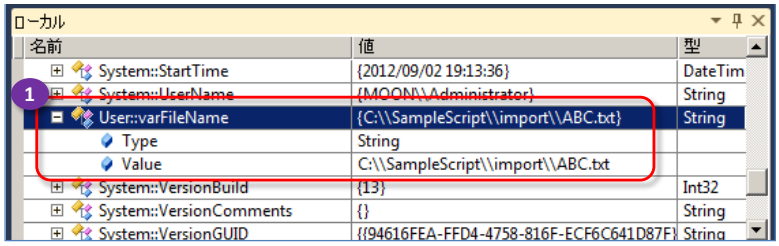
変数の値の確認

2. ブレークポイントの効果を確認するために、パッケージをデバッグ実行します。
すると、次のように［**Foreach ループ コンテナー**］が処理中に変わって、データ転送が一時的に中断されます。



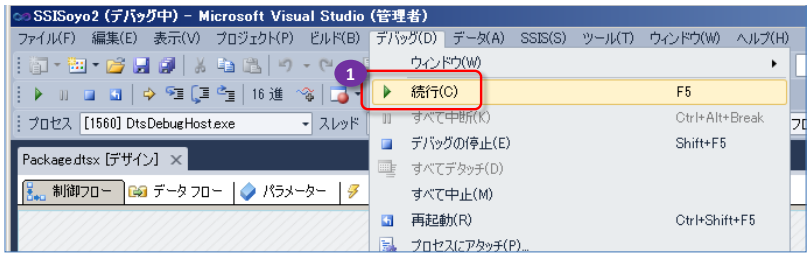
同時にブレークポイントの赤丸アイコンに**黄色の矢印**が追加されて、ここで停止していることが分かります。変数の値を確認するには、この状態で、画面左下の［**ローカル**］タブをクリックして、［**ローカル**］ウィンドウを開き、［**Variables**］（変数）を展開します。［**ローカル**］タブが表示されない場合は、［**デバッグ**］メニューの［**ウィンドウ**］から［**ローカル**］をクリックします。

続いて、[**Variables**]に表示される一覧の中から、[**User::varFileName**]を展開して[**Value**]の値を確認します。

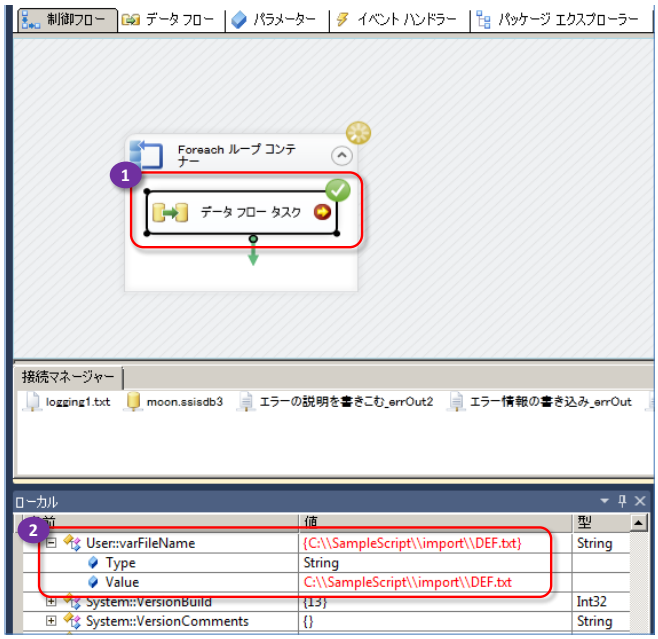


すると、「**C:¥¥SampleScript¥¥import¥¥ABC.txt**」のように表示され、サンプル スクリプトの **import** フォルダーから「**ABC.txt**」ファイルを読み取っていること（現在の「**User::varFileName**」変数の値）が分かります。

3. 次に、データの転送を続行するために［**デバッグ**］メニューの［**続行**］をクリックします。

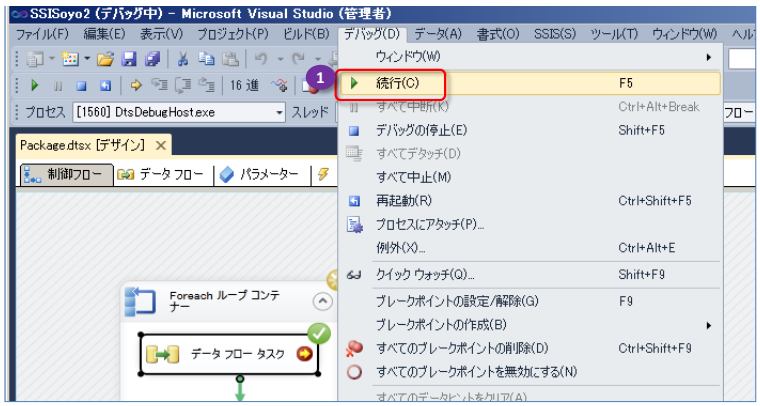


4. すると、［**データ フロー タスク**］には緑のチェックマークが付きますが、再びブレークポイントで、動作が一時的に中断します。

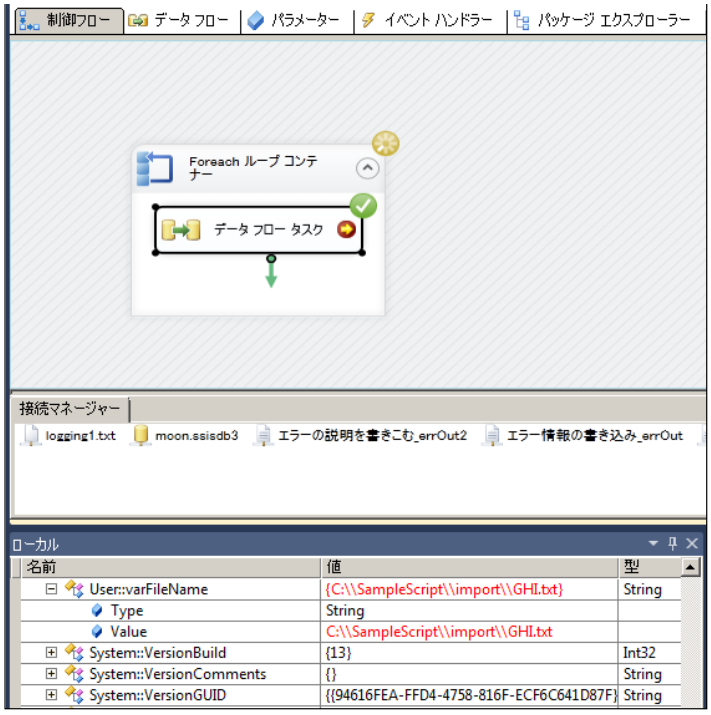


［**ローカル**］ウィンドウの［**User::varFileName**］の［**Value**］値を確認すると、今度は、「**C:¥¥SampleScript¥¥import¥¥DEF.txt**」のように表示されて、2 つ目のファイルが読み取られていることを確認できます。

5. 引き続き、データの転送を続行するために［**デバッグ**］メニューの［**続行**］をクリックします。

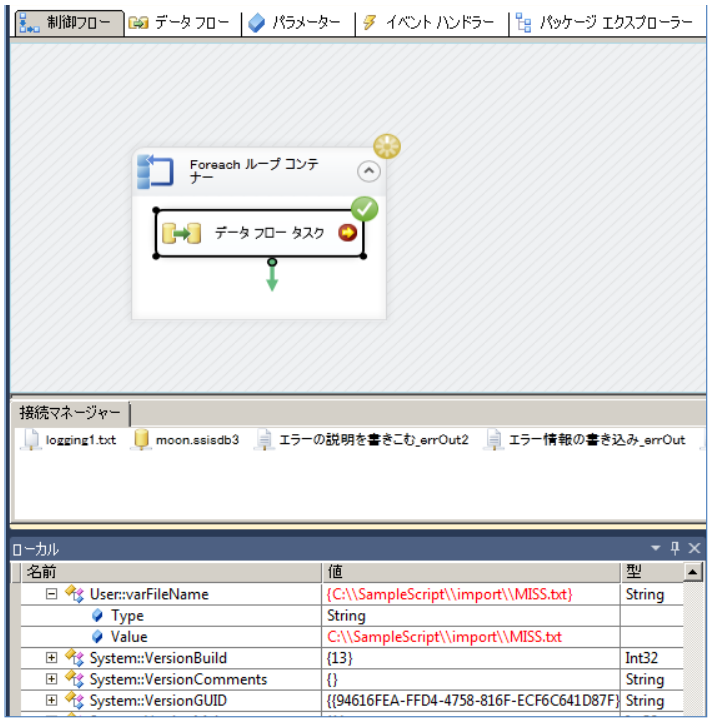


今度は、3 つ目のファイル「**GHI.txt**」を読み取っているところで止まります。

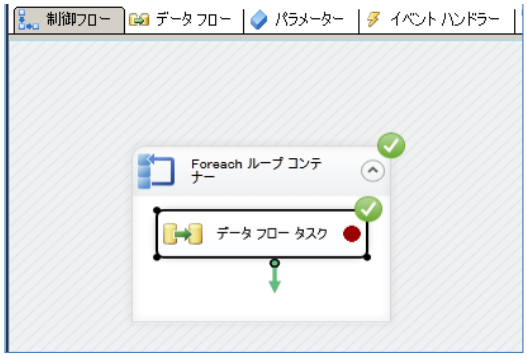


6. もう一度、データの転送を続行するために［**デバッグ**］メニューの［**続行**］をクリックします。

今度は、4 つ目のファイル「**MISS.txt**」を読み取っているところで止まります。



7. さらに、[**デバッグ**]メニューの［**続行**］をクリックします。



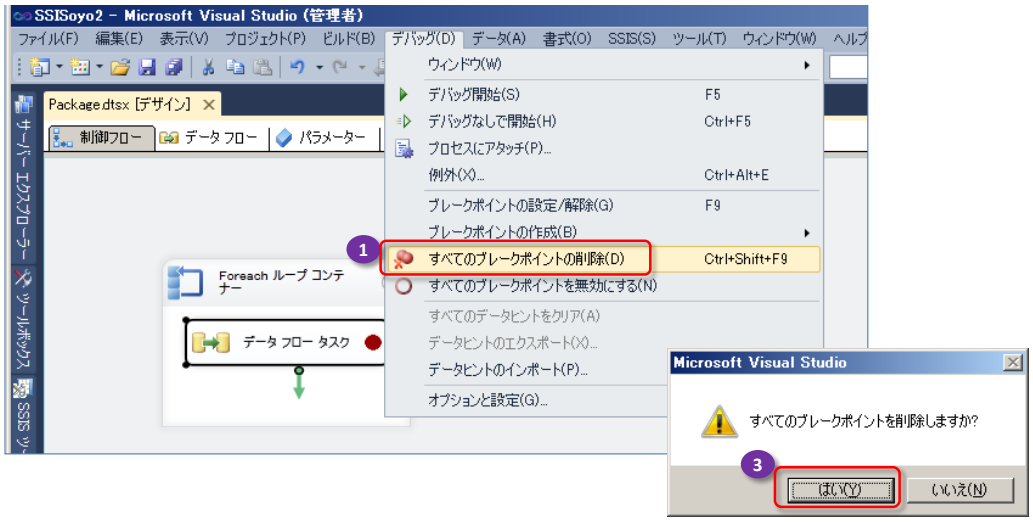
すべてのタスクに緑のチェックマークが付くと、データ転送がすべて完了です。

8. すべての動作を確認後、デバッグを終了します。

以上のように、ブレークポイントを利用すると、タスクを 1 つ 1 つステップ実行できるようになって、また変数の値も確認できるので、大変便利です。

ブレークポイントの削除

1. 設定したブレークポイントを削除したい場合は、次のように［**デバッグ**］メニューの［**すべてのブレークポイントの削除**］をクリックします。



2. 最後に、［**ファイル**］メニューの［**すべてを保存**］をクリックして、**SSISoyo2** プロジェクトを保存しておきます。このプロジェクトは、次の Step でも引き続き使用します。

STEP 7. パッケージの配置

この STEP では、Integration Services サーバーへのパッケージの配置やパラメーター化、別マシンへのパッケージの配置方法などを説明します。

この STEP では、次のことを学習します。

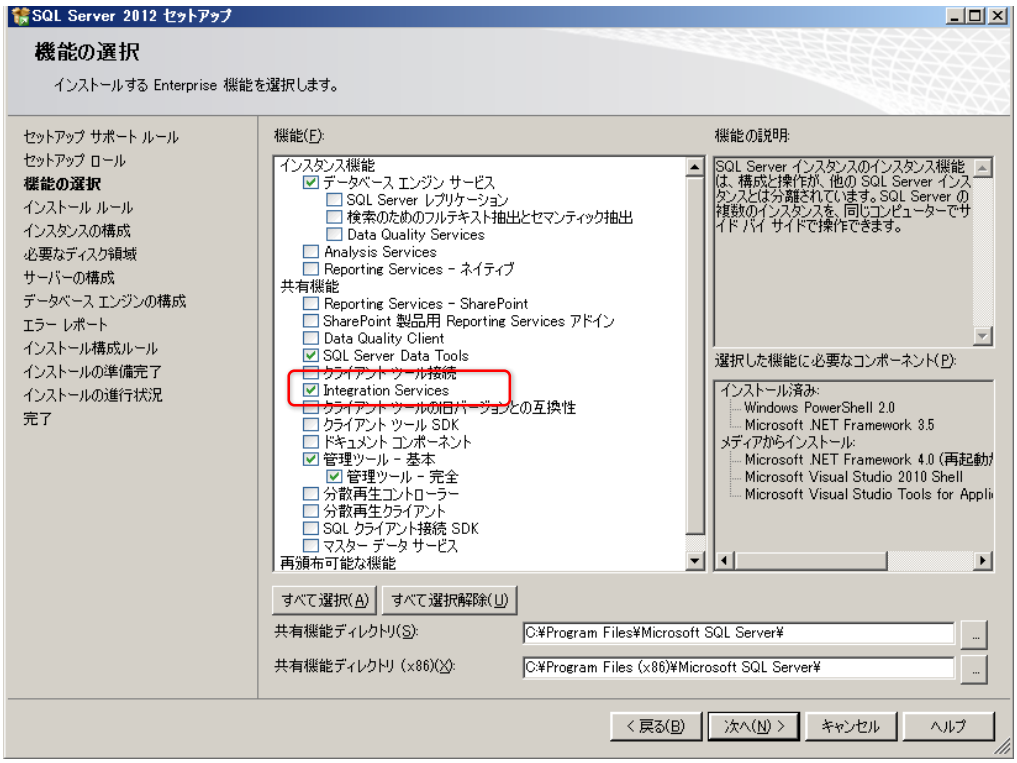
- ✓ Integration Services サーバーとは
- ✓ パラメーター化
- ✓ SSISDB カタログの作成
- ✓ Integration Services サーバーへのパッケージの配置
- ✓ 別マシンへのパッケージの配置

7.1 Integration Services サーバーとは

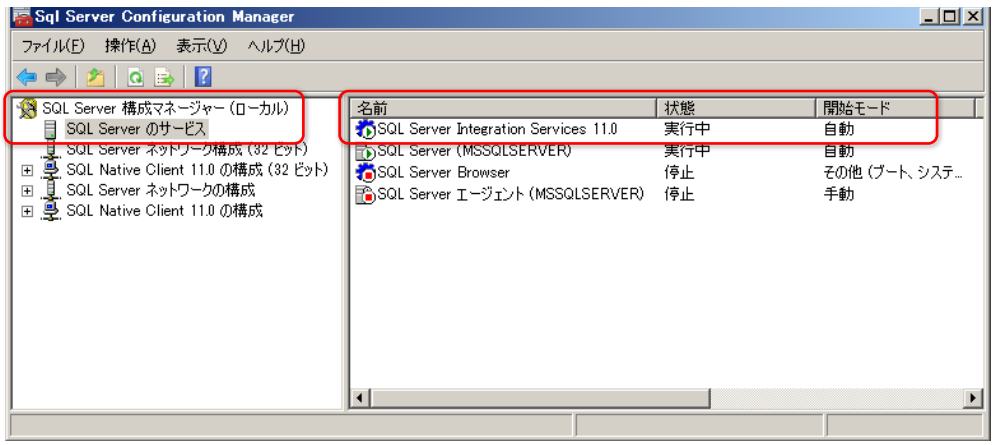
Integration Services サーバーとは

作成した SSIS パッケージは、「**Integration Services サーバー**」へ配置することで、パッケージをサーバーで管理したり、実行したりできるようになります。

Integration Services サーバーは、SQL Server 2012 のインストール時に、次のように［**Integration Services**］を選択したサーバーです。



［**Integration Services**］をインストールすると、「**SQL Server Integration Services 11.0**」という名前サービスが追加されて、自動実行されます。これは、「**SQL Server 構成マネージャー**」ツールで次のように確認できます。



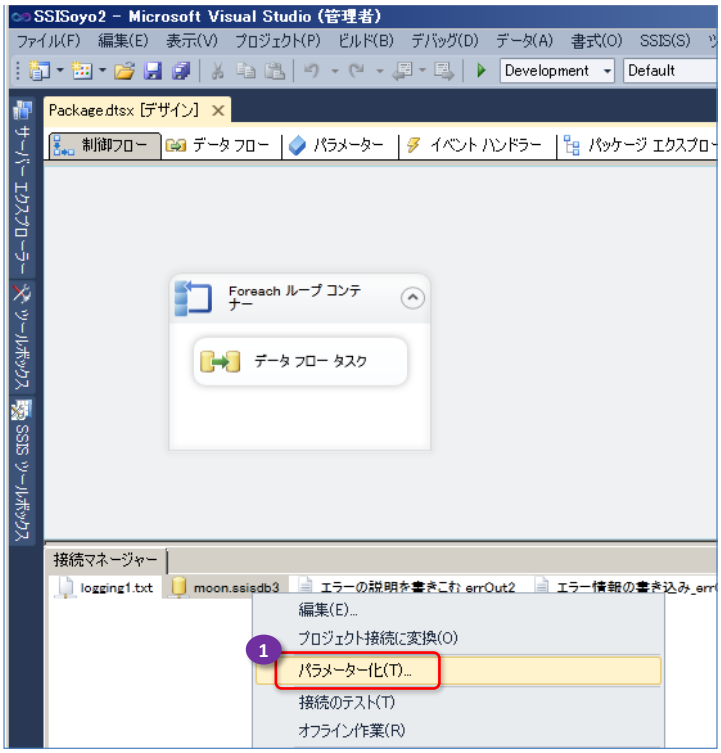
このサービスが起動しているマシンが「**Integration Services サーバー**」です。

7.2 パラメーター化機能

パラメーター化機能（サーバー名とパスのパラメーター化）

Integration Services サーバーへの配置を行う前に、作成したプロジェクトを、本番機など別のマシンへ配置する際に便利な「**パラメーター化**」機能を利用する手順を説明します。開発機と本番機では、接続する SQL Server の名前が違ったり、ファイルの配置してあるパスが違ったりするので、そういった場合にパラメーター化機能を利用すると大変便利です。

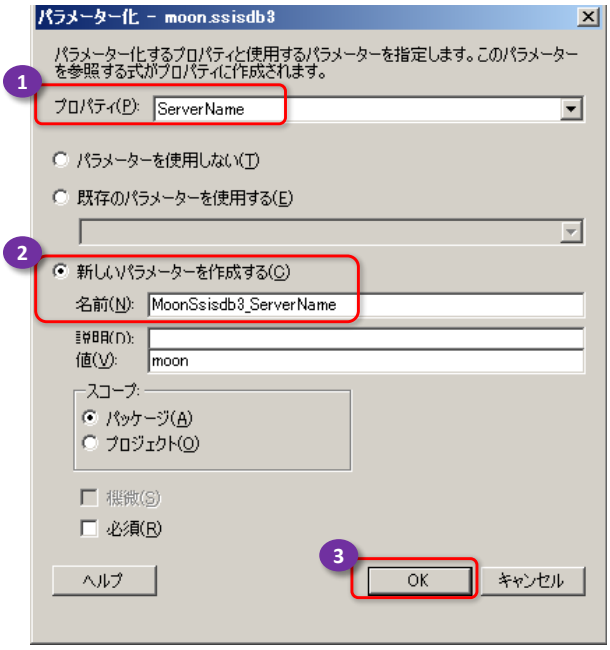
1. まずは、データの転送先となる SQL Server の名前のパラメーター化を行ってみましょう。接続する SQL Server の名前を**パラメーター化**するには、次のように［**接続マネージャー**］の「**サーバー名.ssisdb3**」を右クリックして、［**パラメーター化**］をクリックします。



Note： パラメーター化は SQL Server 2012 からの新機能

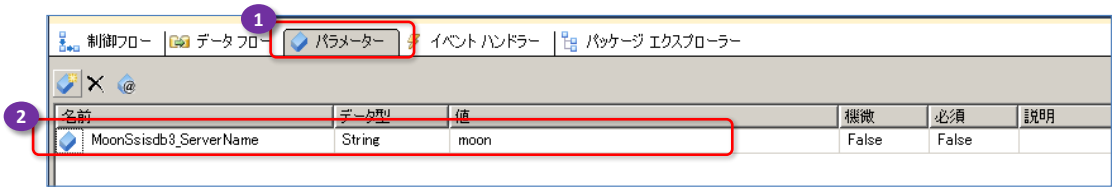
パラメーター化は、SQL Server 2012 から提供された新機能です。SQL Server 2008 R2 以前のバージョンを利用している場合は、「**パッケージ構成ファイル**」（**.dtsConfig**）という機能を利用することで、パラメーター化と同じようなことを実現することができます。

2. ［**パラメーター化**］をクリックすると、次のように［**パラメーター化**］ダイアログが表示されるので、次のように［**プロパティ**］で「**ServerName**」を選択します。



［**新しいパラメーターを作成する**］が選択されていることを確認して、［**名前**］へ任意のパラメーター名（既定値は **サーバー名 Ssisdb3_ServerName**）を入力し、［**OK**］ボタンをクリックします。

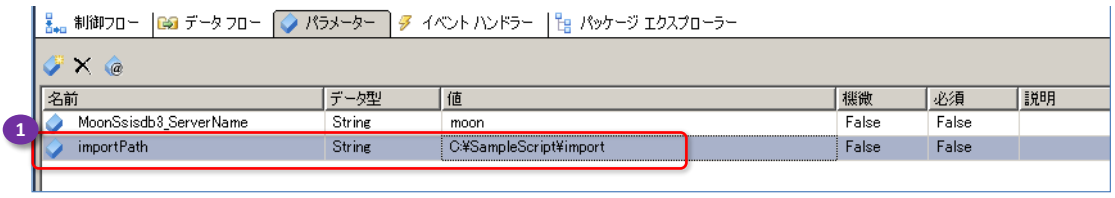
3. ［**パラメーター**］タブを開くと、作成したパラメーターが表示されていることを確認できます。



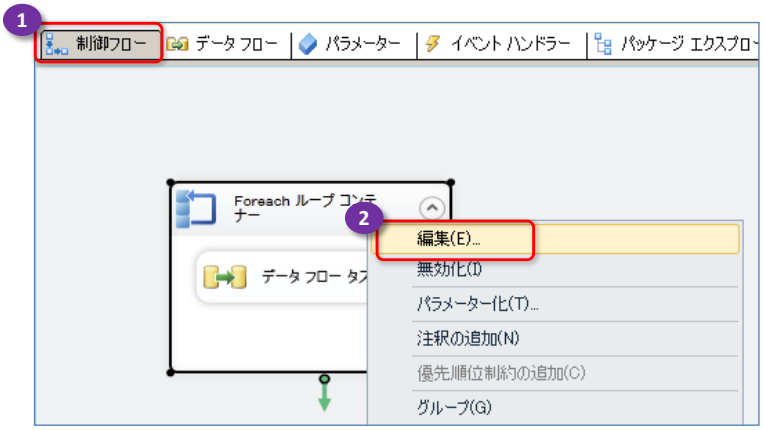
これで、データ転送先となる SQL Server の名前のパラメーター化が完了です。

4. 次に、Foreach ループ コンテナーで指定した読み取り対象のフォルダーをパラメーター化してみます。次のように、［**パラメーター**］タブで、［**新規作成**］ボタンをクリックします。

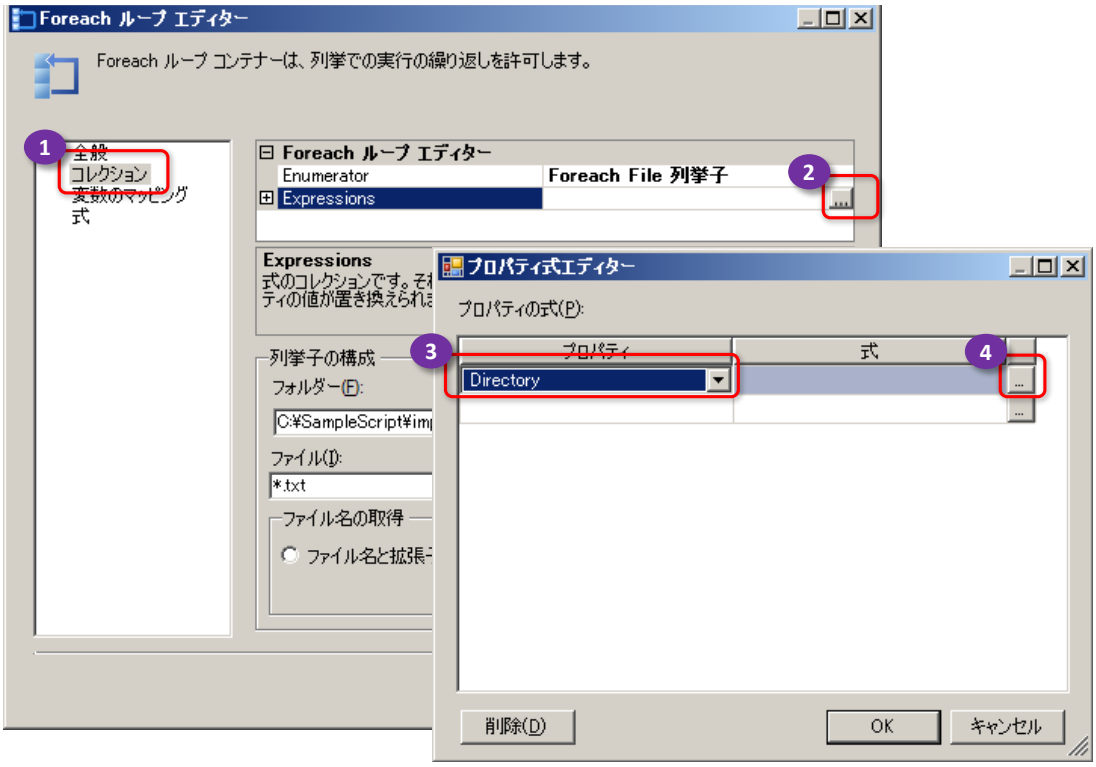
5. これにより、新しくパラメーターが作成されるので、次のように［**名前**］へ任意のパラメーター名（画面は **importPath**）を入力して、［**データ型**］へは「**String**」を選択し、既定値となる［**値**］へはサンプル スクリプトの **import** フォルダーへのパスを入力します。



6. 続いて、[**制御フロー**] タブを開いて、[**Foreach ループ コンテナー**] をダブル クリックもしくは右クリックして [**編集**] を選択します。

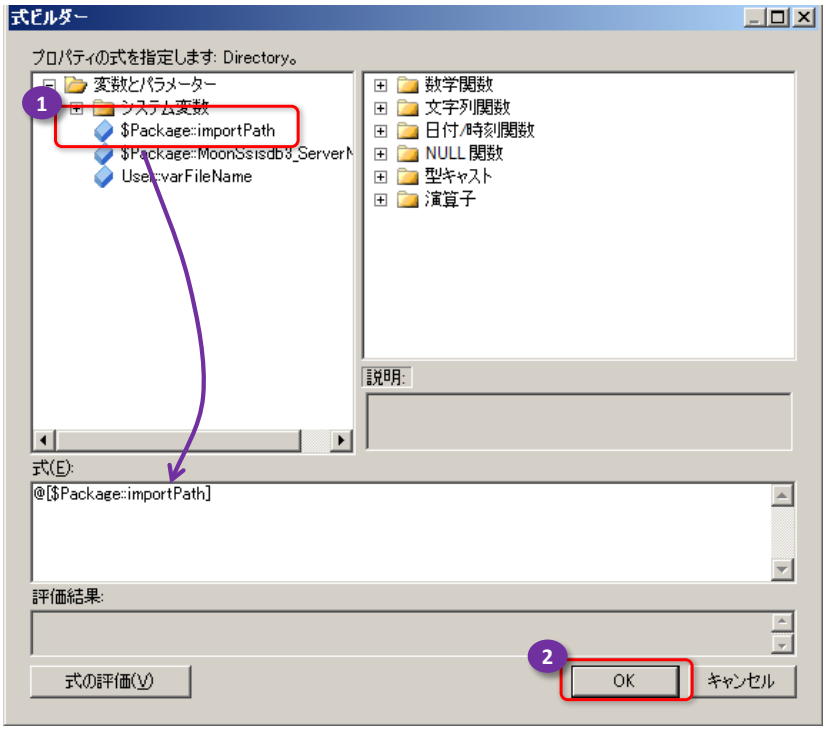


7. [**Foreach ループ エディター**] ダイアログが開いたら、[**コレクション**] ページを開いて [**Expressions**] プロパティの [**...**] ボタンをクリックします。



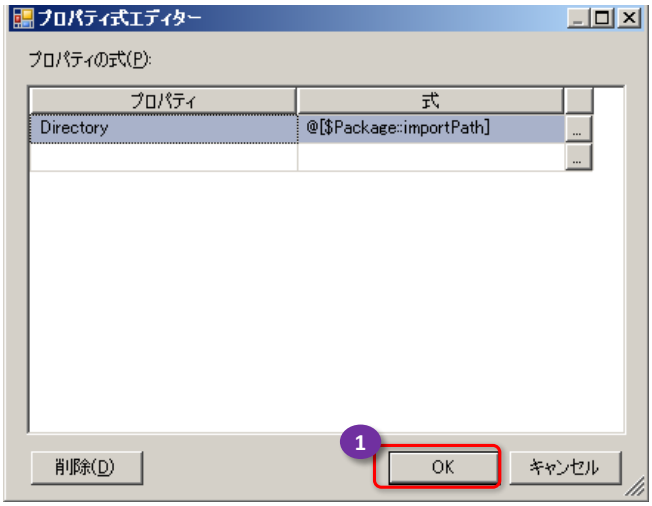
[**プロパティ式エディター**] ダイアログで [**Directory**] プロパティを選択して [**...**] ボタンをクリックします。

8. [**式ビルダー**] ダイアログが開いたら、[**変数とパラメーター**] フォルダーを展開して表示されるパラメーターの一覧の中から、前の手順で作成した「**$Package::importPath**」パラメーターを [**式**] へドラッグ＆ドロップします。



［**OK**］ボタンをクリックします。

9. ［**プロパティ式エディター**］ダイアログへ戻ったら、式が設定されていることを確認して、［**OK**］ボタンをクリックします。



［**Foreach ループ エディター**］ダイアログへ戻ったら、［**OK**］ボタンをクリックします。

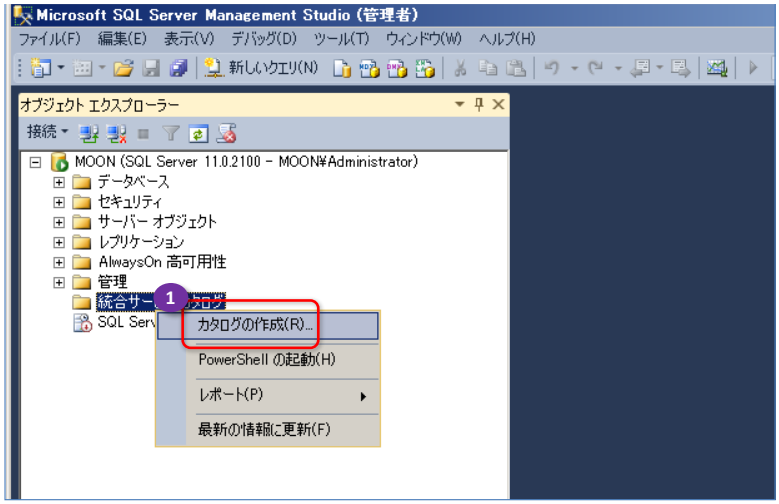
これで、Foreach ループ コンテナーの読み取り対象のフォルダーのパラメーター化が完了です。

7.3 SSISDB カタログの作成

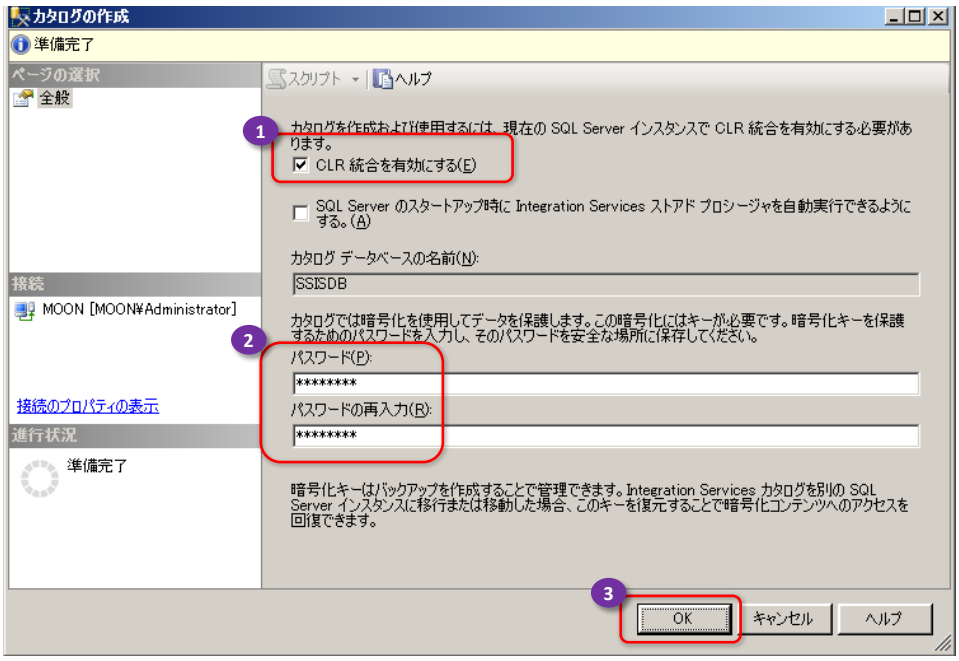
SSIDB カタログの作成

Integration Services サーバーへパッケージを配置するには、事前に「**SSIDB カタログ**」（**SSISDB データベース**）を作成しておく必要があります。

1. **SSIDB カタログ**を作成するには、Integration Services サーバー上で Management Studio を起動して、[**オブジェクト エクスプローラー**] の [**統合サービス カタログ**] フォルダーを右クリックし、[**カタログの作成**] をクリックします。



2. [**カタログの作成**] ダイアログが表示されたら、[**CLR 統合を有効にする**] をチェックして、[**パスワード**] と [**パスワードの再入力**] へ暗号化のための任意のパスワードを入力して、[**OK**] ボタンをクリックします。



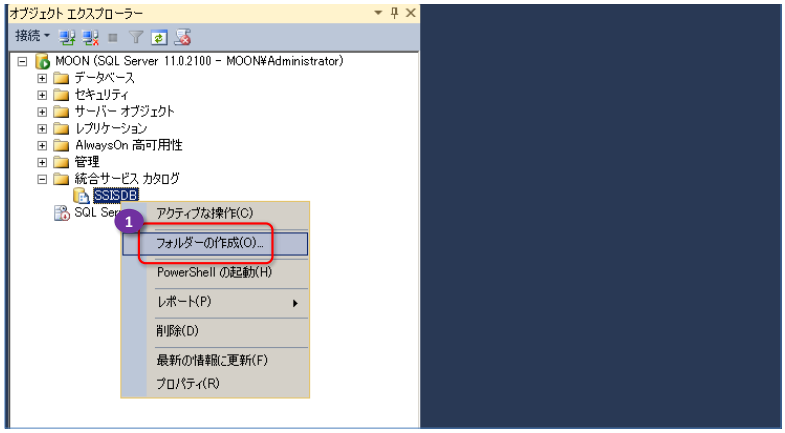
これにより、**SSISDB カタログ**が作成されます（**SSISDB** という名前のシステム データベ

ースが自動作成されます)。

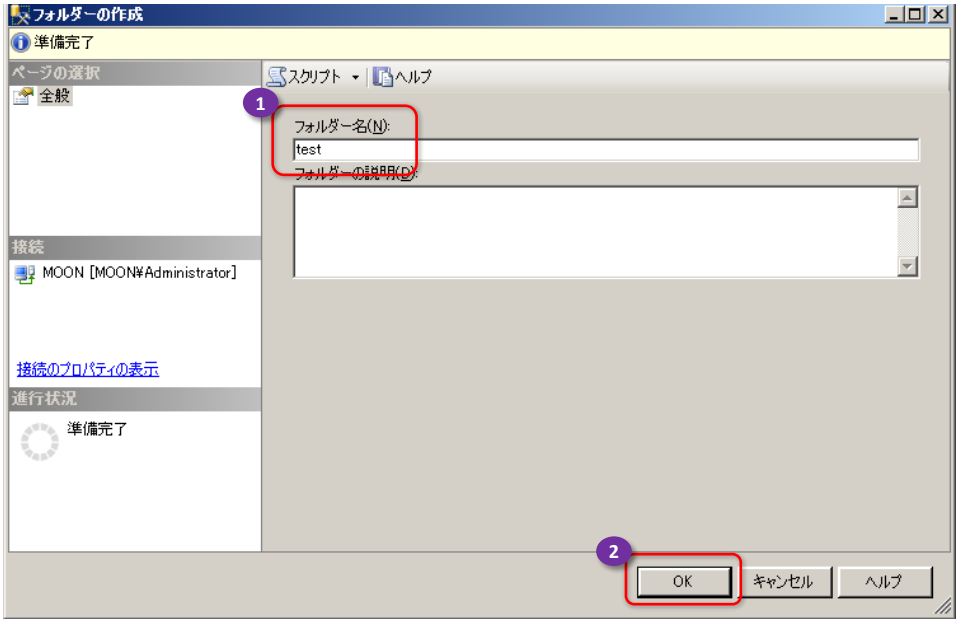
SSIS パッケージ格納用のフォルダーの作成

次に、SSIS パッケージを格納するための**フォルダー**を、**SSISDB カタログ**内に作成します。

1. フォルダーを作成するには、次のように［**統合サービス カタログ**］を展開して、［**SSISDB カタログ**］を右クリックし、［**フォルダーの作成**］をクリックします。



2. ［**フォルダーの作成**］ダイアログが表示されたら、［**フォルダー名**］へ任意のフォルダー名（画面は **test**）を入力して、［**OK**］ボタンをクリックします。

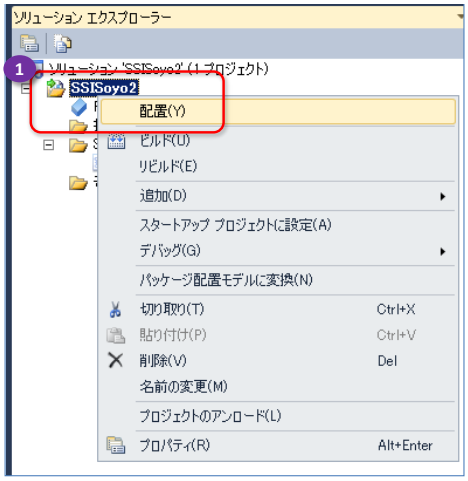


7.4 Integration Services サーバーへのパッケージの配置

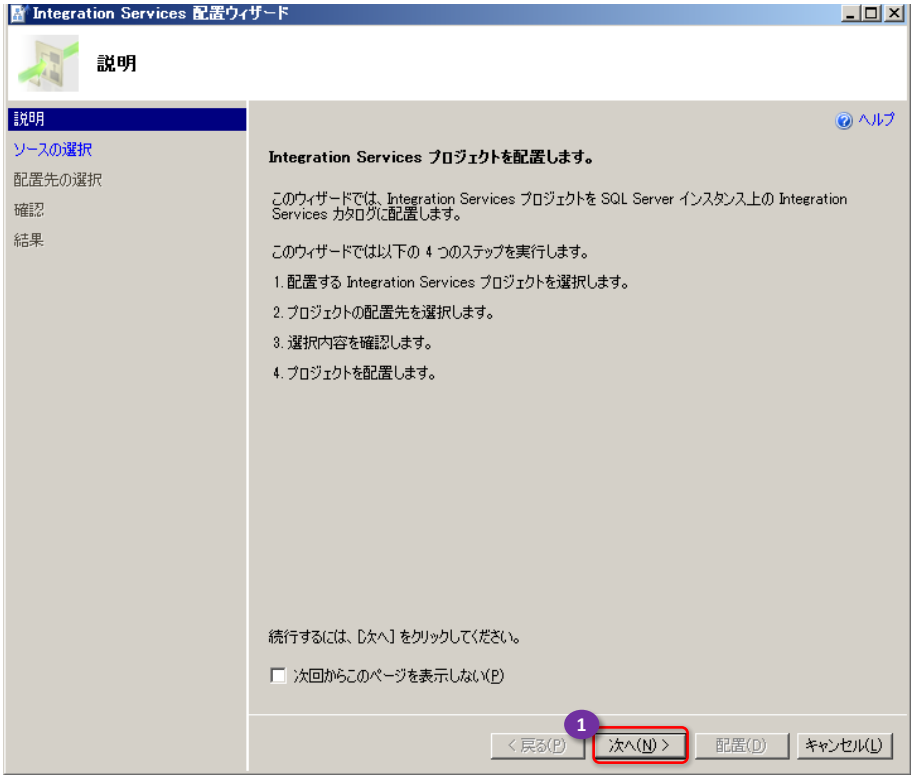
Integration Services サーバーへのパッケージの配置

次に、SSIS パッケージを Integration Services サーバーへ配置します。配置には、**SQL Server Data Tools** を利用します。

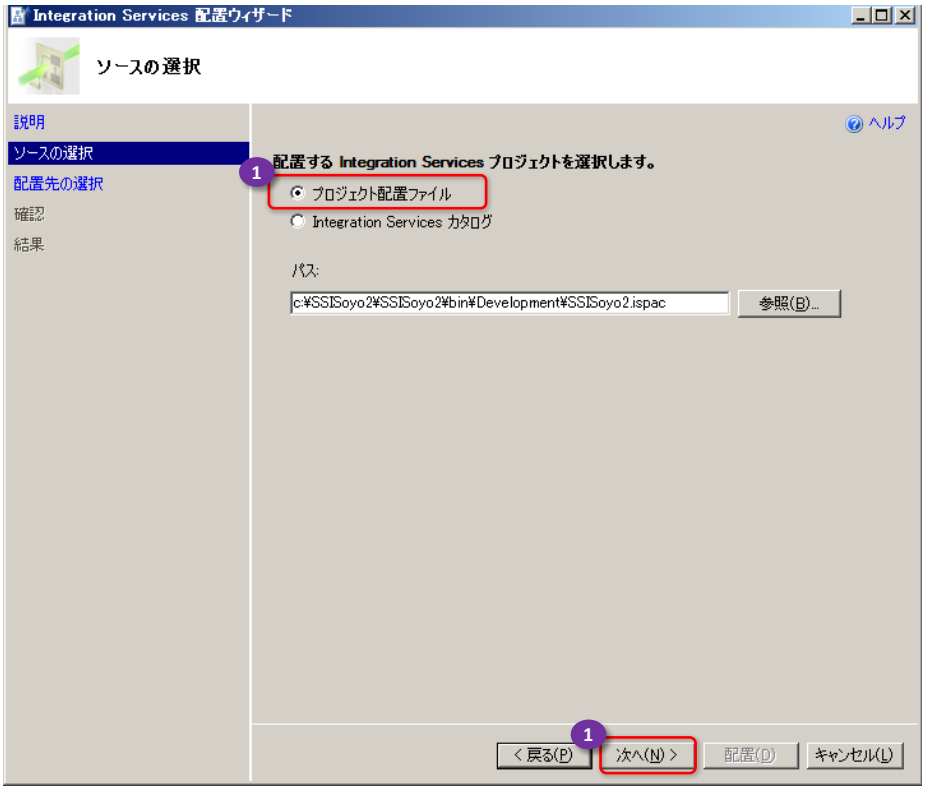
1. **SQL Server Data Tools** でプロジェクト（**SSISoyo2**）を開き、次のように［**ソリューション エクスプローラー**］で、［**プロジェクト名**］（**SSISoyo2**）を右クリックして、［**配置**］をクリックします。



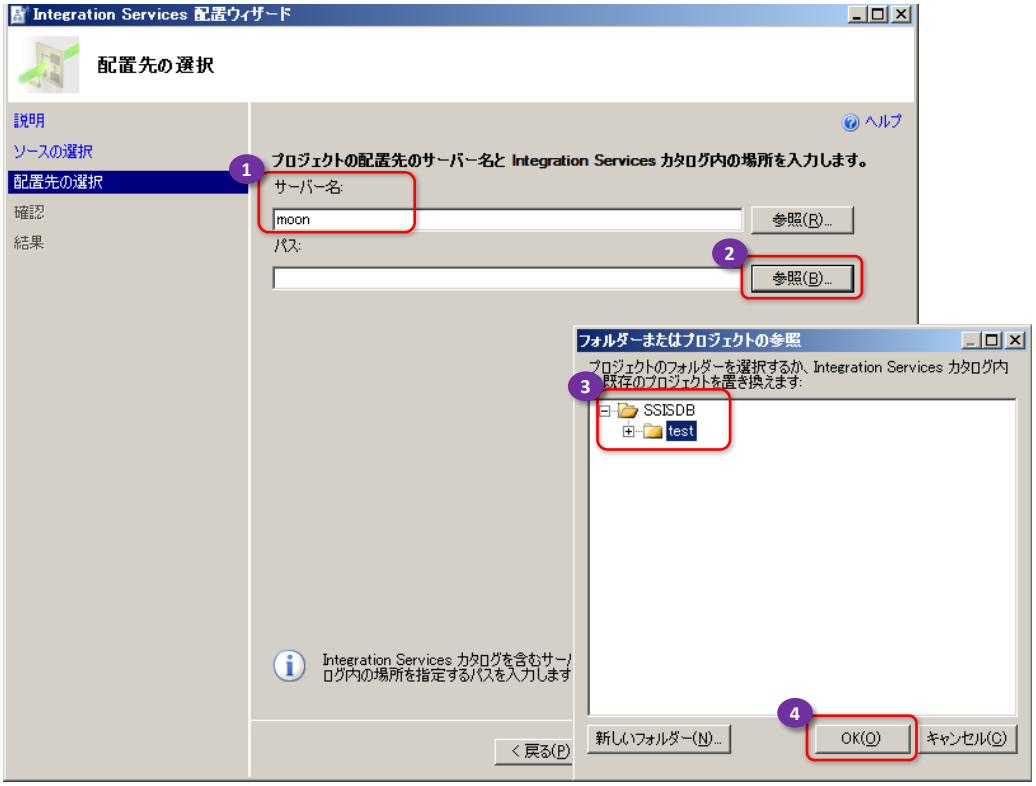
2. これにより、［**Integration Services 配置ウィザード**］が起動するので、［**次へ**］ボタンをクリックします。



3. 次の［**ソースの選択**］ページでは、［**プロジェクト配置ファイル**］が選択されていることを確認して、［**次へ**］ボタンをクリックします。



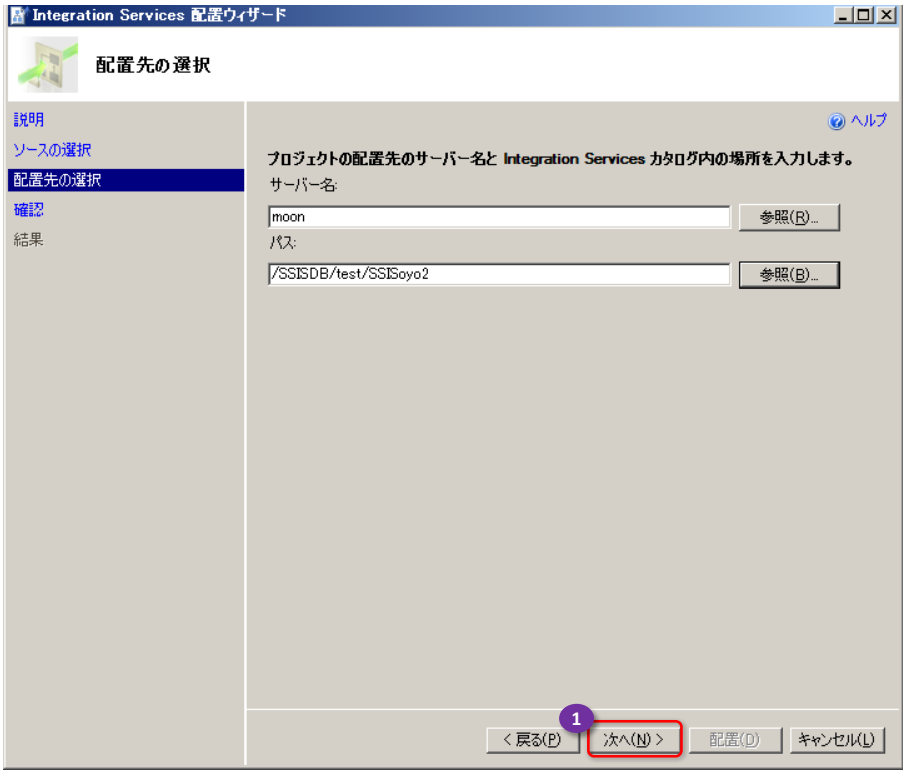
4. 次の［**配置先の選択**］ページでは、［**サーバー名**］へ配置先のサーバー名（Integration Services サーバーの名前）を入力します。



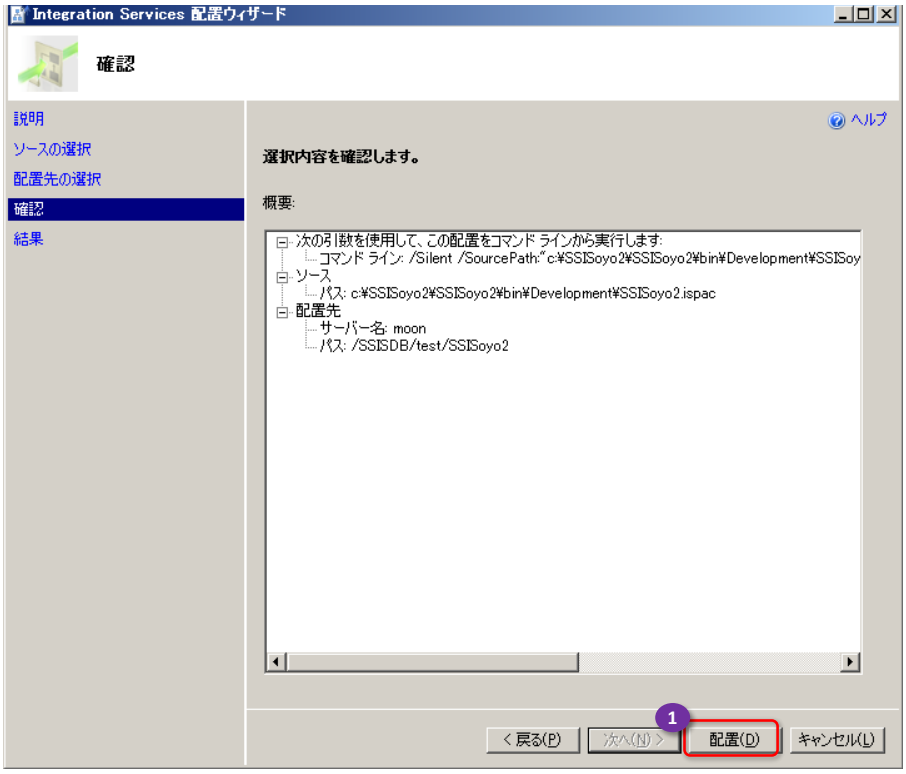
続いて、［**パス**］の［**参照**］ボタンをクリックして、［**フォルダーまたはプロジェクトの参照**］

ダイアログで、プロジェクトの配置先となるフォルダーとして、前の手順で作成したフォルダー（今回は、**test**）を選択して、［**OK**］ボタンをクリックします。

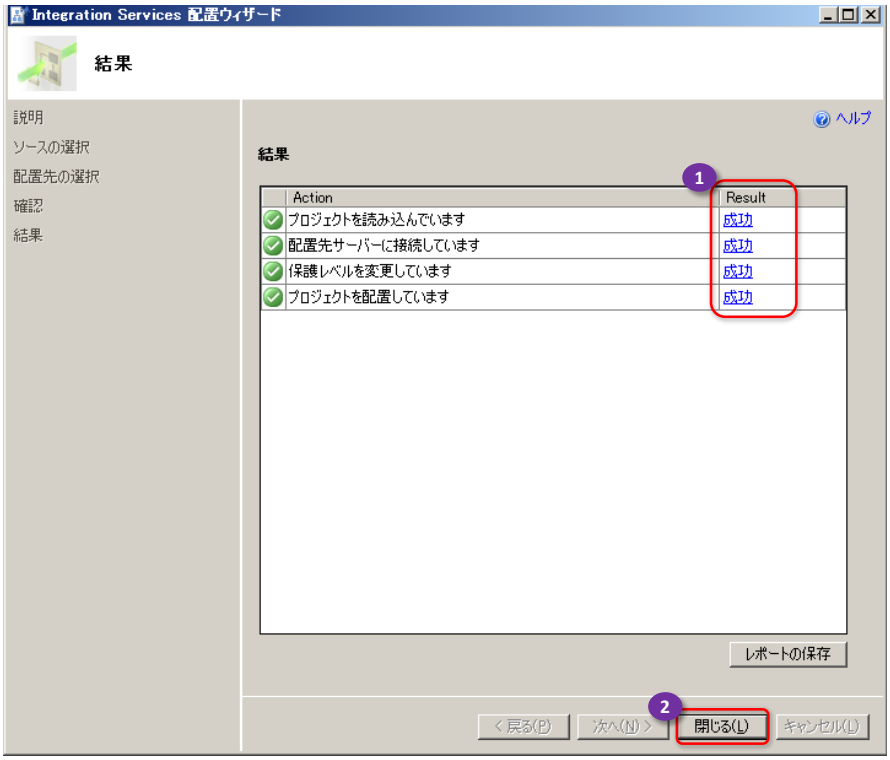
5. ［**配置先の選択**］ページへ戻ったら、［**次へ**］ボタンをクリックします。



6. 次の［**確認**］ページでは、これまで設定してきた内容が表示されます。確認後、［**配置**］ボタンをクリックして、配置を実行します。



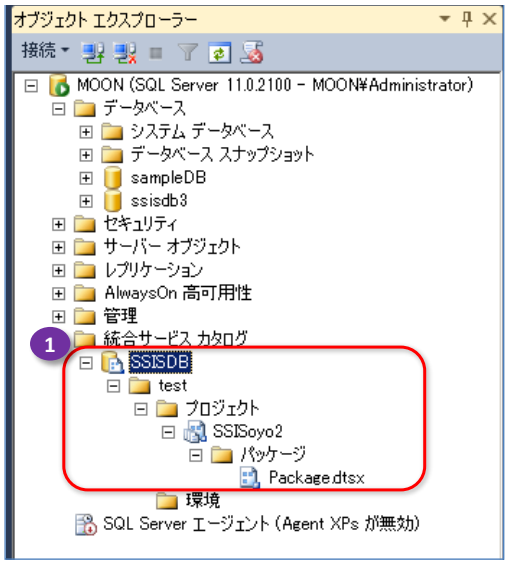
7. ［**結果**］ページでは、配置が完了し、すべての［**Result**］が「**成功**」になったことを確認して、［**閉じる**］ボタンをクリックします。



これで、サーバーへの配置が完了です。

配置したパッケージの確認

1. 配置したパッケージを確認するには、Management Studio のオブジェクト エクスプローラーで **SSISDB カタログ**を展開して、「**test**」フォルダーを展開します。

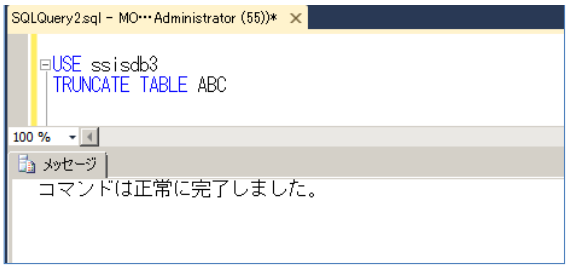


「**プロジェクト**」フォルダー内に **SSISoyo2** プロジェクトが配置され、「**パッケージ**」フォルダーには、パッケージ（**Package.dtsx**）が表示されていることを確認できます。

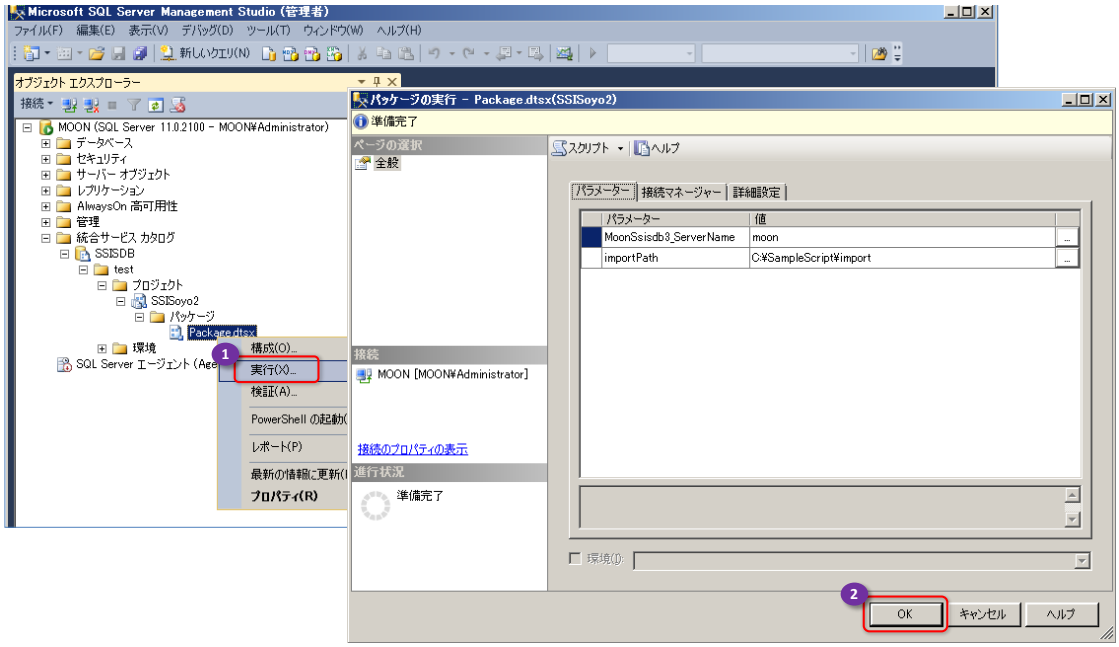
Management Studio からの実行

パッケージが正しく配置されたことを確認するため、Management Studio から実行してみましょう。

1. データの転送の結果を確認しやすくするために、まずは、次のように **Management Studio** の**クエリ エディター**から「**ssisdb3**」データベースの「**ABC**」テーブルに対して、**TRUNCATE TABLE** ステートメントを実行して、全行を削除しておきます。



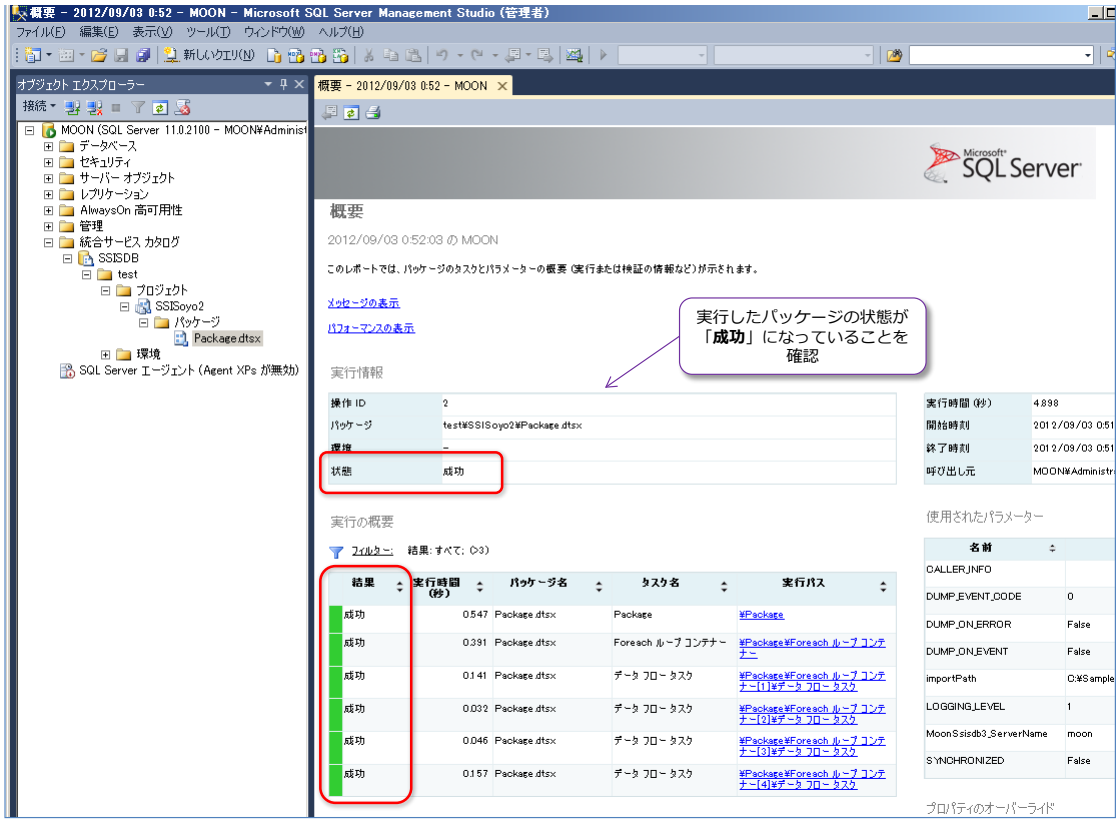
2. 次に、オブジェクト エクスプローラーで [**SSISDB カタログ**] 内の **SSISoyo2** プロジェクトを展開して、パッケージ名（**Package.dtsx**）を右クリックし、[**実行**] をクリックします。



[**パッケージ実行**] ダイアログが表示されたら、[**OK**] ボタンをクリックします。これにより、パッケージが実行されます。

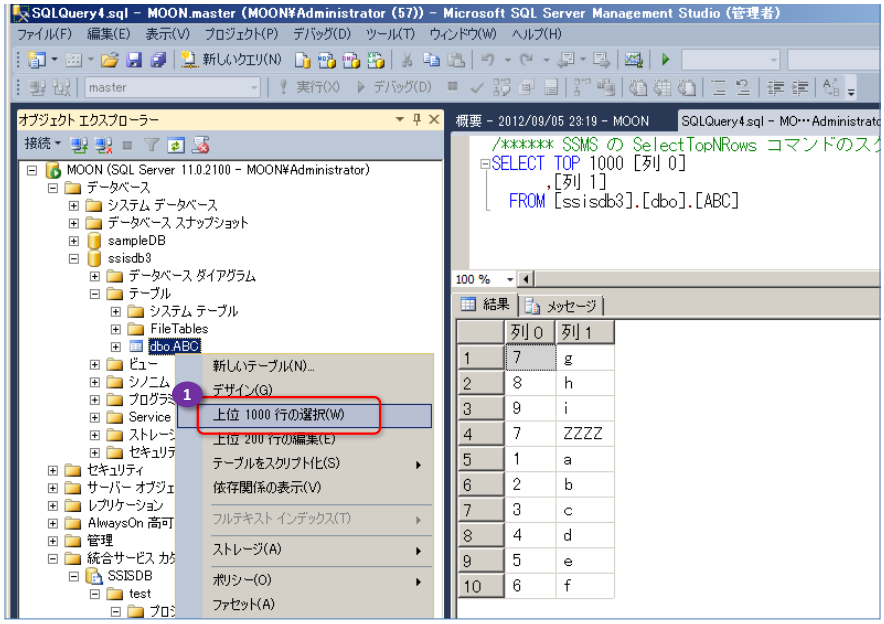
3. 実行後、「**概要レポートを今すぐ開きますか？**」というメッセージが表示されるので、[**はい**] ボタンをクリックします。

4. これによって、実行したパッケージの概要レポートを表示することができます。



［**状態**］が「**成功**」になっていれば、実行が成功です。

5. 転送されたデータを確認するために、オブジェクト エクスプローラーで［**ABC**］テーブルを右クリックして、［**上位 1000 行の選択**］をクリックします。



7.5 別マシンへのパッケージの配置

別マシンへのパッケージの配置

次に、Management Studio を利用して、パッケージを別のマシンへ配置します。ここからの手順は別のマシンが必要になりますが、同一マシンで試している方は、別マシンの部分を同一マシンへ置き換えて読み進めてください。また、同一マシンで試している場合は、手順 2～5 の作業を省略してください。

1. まずは、パッケージが利用するサンプル スクリプト内の「**import**」フォルダー（Foreach ループ コンテナーの読み取り対象のフォルダー）配下のファイルをすべて、別マシン上の任意のフォルダー（**C:¥import2** など）へコピーします。

2. 次に、ファイルの転送先となるデータベース（**ssisdb3**）とテーブル（**ABC**）を別マシン上で作成しておきます。Management Studio のクエリ エディターを開いて、次のようにステートメントを記述し、実行しておきます。

```
CREATE DATABASE ssisdb3
go
USE ssisdb3
CREATE TABLE ABC ( [列 0] int, [列 1] varchar(50) )
```

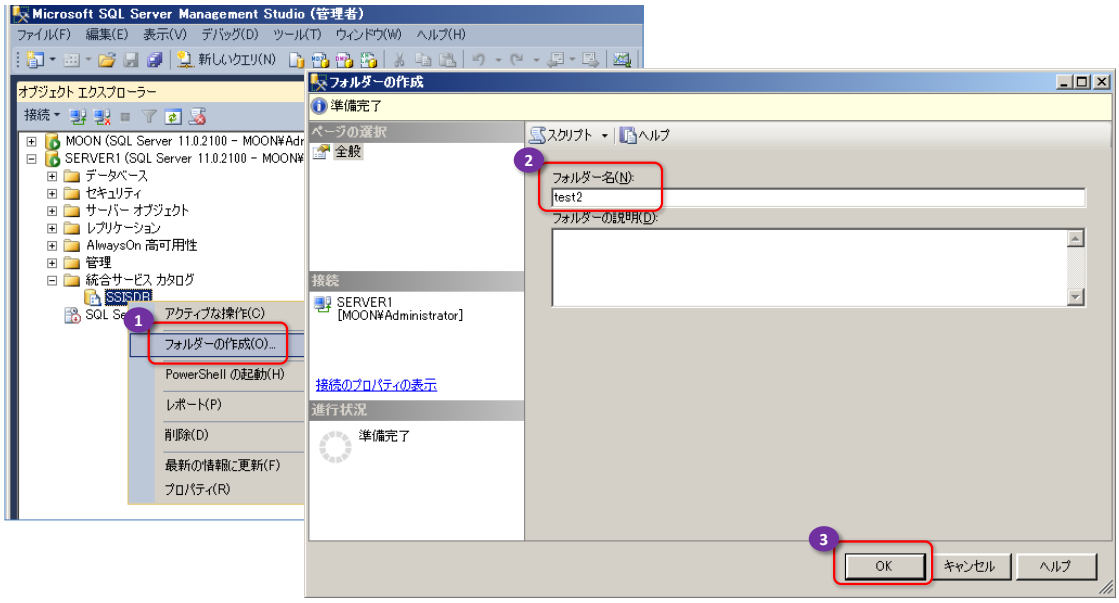
なお、この手順は、同一マシンで試している場合は、省略してください。

3. 次に、配置先となる別マシン上にも **SSISDB カタログ**を作成します。Management Studio の［**オブジェクト エクスプローラー**］で、配置先マシンの［**統合サービス カタログ**］フォルダーを右クリックして、［**カタログの作成**］をクリックします。

［**カタログの作成**］ダイアログが開いたら、［**CLR 統合を有効にする**］をチェックして、［**パスワード**］と［**パスワードの再入力**］へ暗号化のためのパスワードを入力し、［**OK**］ボタンをクリックします。

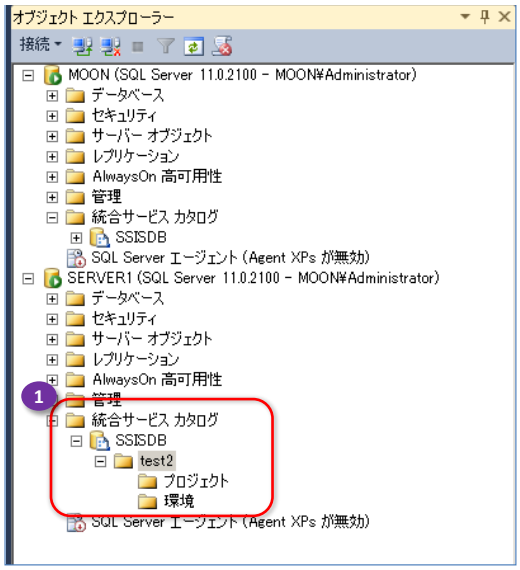
これにより、配置先の別マシンにも **SSISDB カタログ**が作成されます。

4. 次に、［**統合サービス カタログ**］展開して、**SSISDB カタログ**を右クリックし、［**フォルダーの作成**］をクリックします。



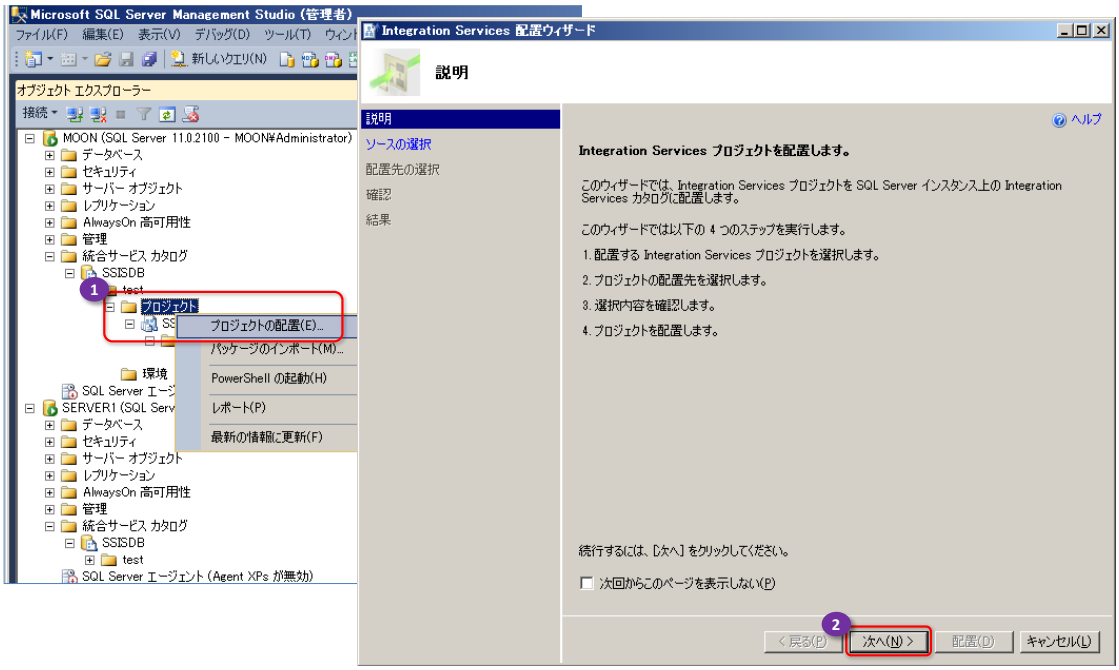
［**フォルダーの作成**］ダイアログが表示されたら、［**フォルダー名**］へ任意のフォルダー名を入力して（画面は **test2**）、［**OK**］ボタンをクリックします。

5. これで、配置先の別マシン（画面は **SERVER1**）にもパッケージを配置するためのフォルダーを作成することができました。



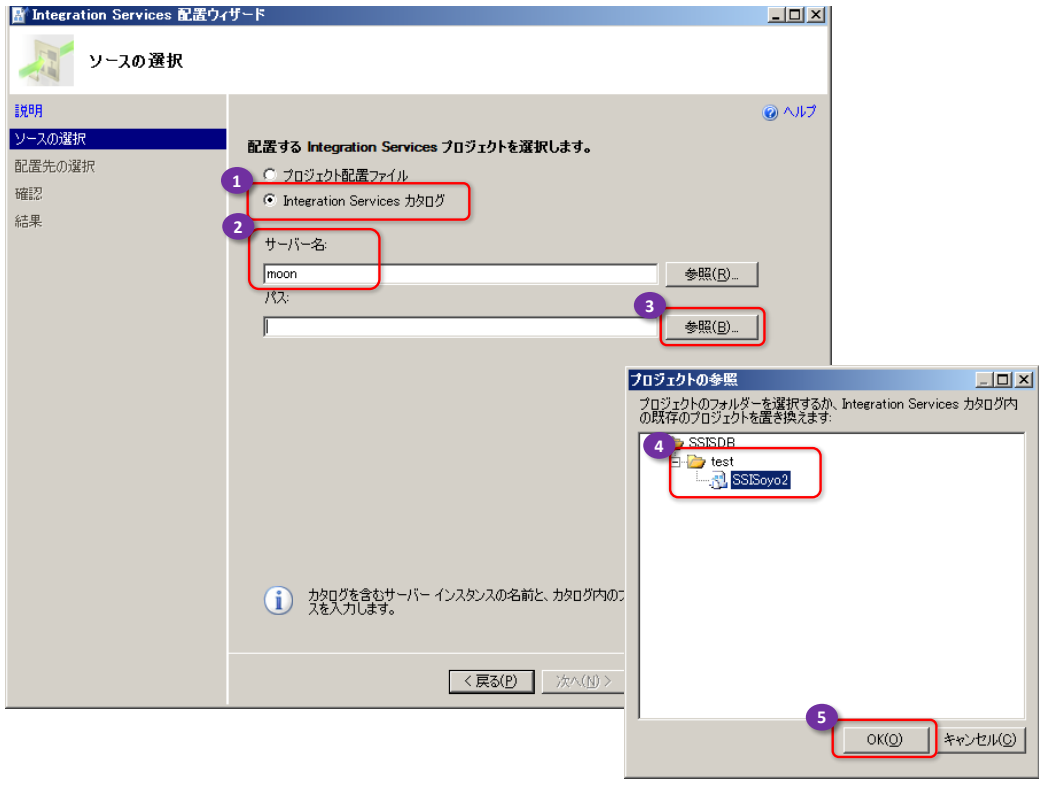
別マシンへの配置

6. 次に、Management Studio のオブジェクト エクスプローラーで、元のマシンに配置してあるパッケージの［**プロジェクト**］フォルダーを右クリックして、［**プロジェクトの配置**］をクリックします。



これにより、[**Integration Services 配置ウィザード**] が起動します。

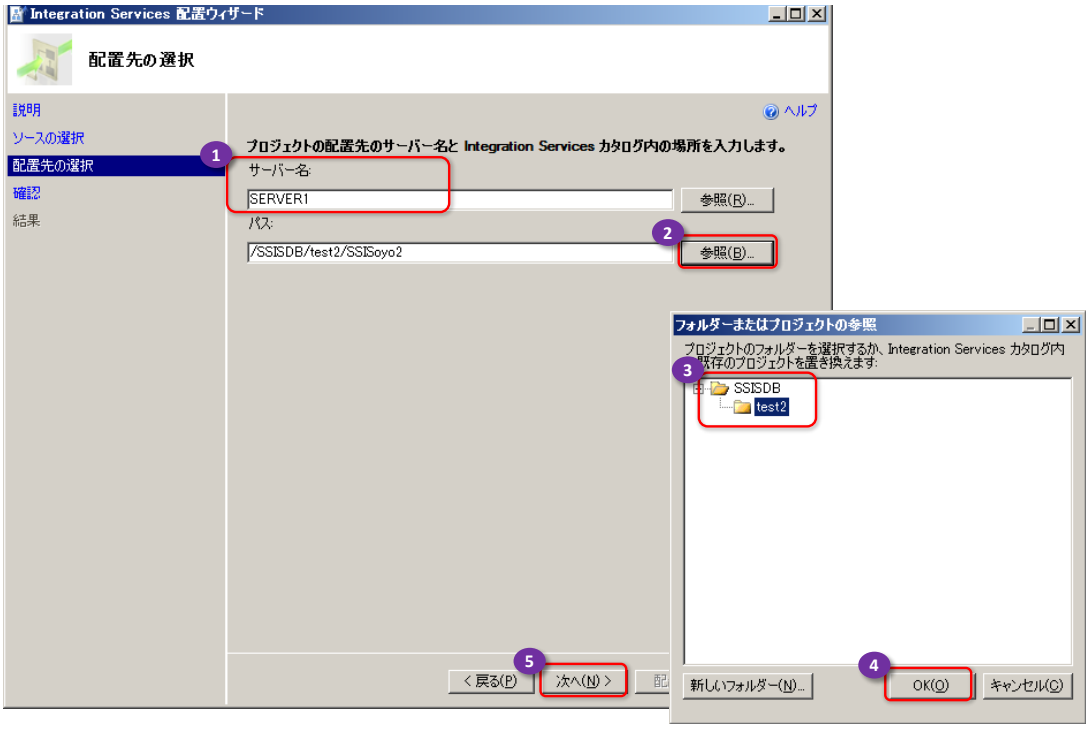
7. 次の [**ソースの選択**] ページでは、配布対象となるパッケージの場所を指定します。ここでは、[**Integration Services カタログ**] をチェックして、[**サーバー名**] へパッケージが配置されているサーバーの名前（画面は **moon**）を入力し、[**パス**] で [**参照**] ボタンをクリックします。



[**プロジェクトの参照**] ダイアログが表示されたら、**test** フォルダーの **SSISoyo2** プロジェクトを選択して、[**OK**] ボタンをクリックします。

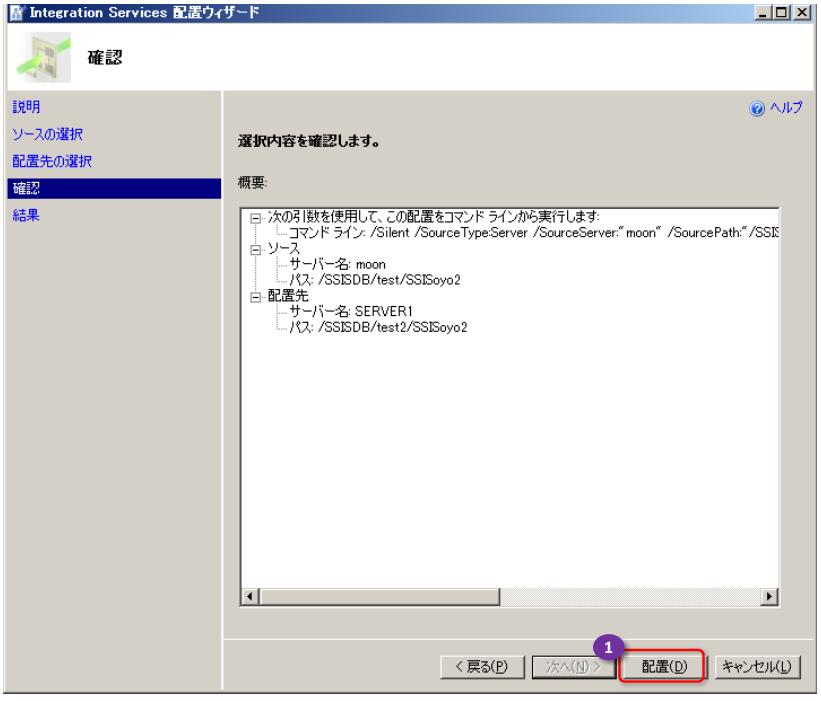
［**ソースの選択**］ページへ戻ったら、［**次へ**］ボタンをクリックします。

8. 次の［**配置先の選択**］ページでは、配置先となるサーバー名とフォルダーを指定します。

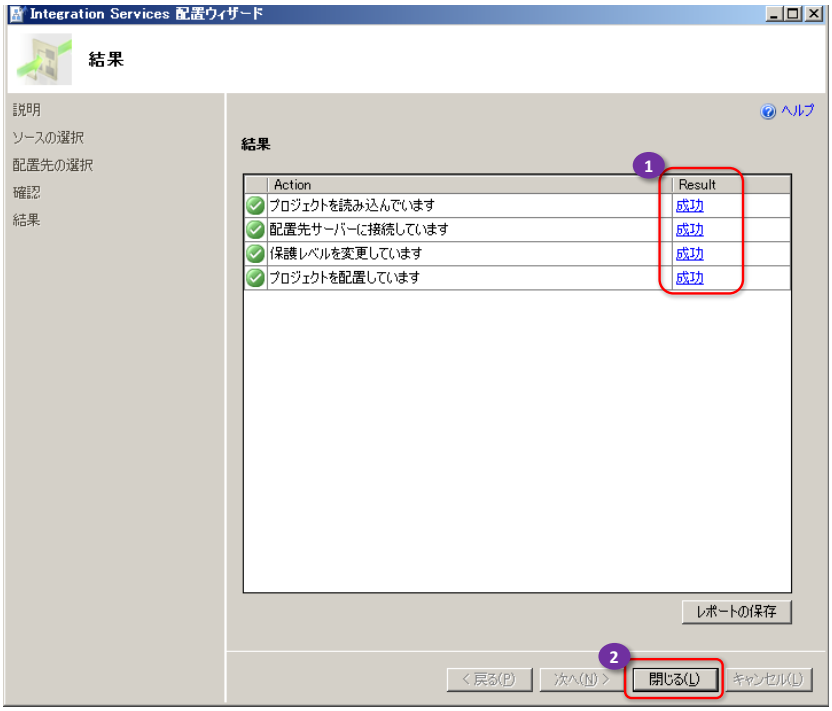


［**サーバー名**］へ配置先となる別マシン名（画面は **SERVER1**）を入力して、［**パス**］の［**参照**］ボタンをクリックし、作成したフォルダー名（画面は **test2**）を指定して、［**次へ**］ボタンをクリックします。

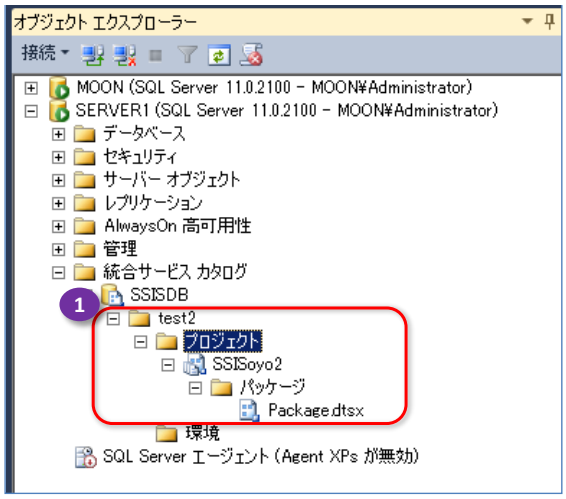
9. 次の［**確認**］ページでは、これまで設定してきた内容を確認して、［**配置**］ボタンをクリックし、配置を実行します。



10. 次の［**結果**］ページでは、すべての［**Result**］が「**成功**」となっていることを確認して［**閉じる**］ボタンをクリックします。

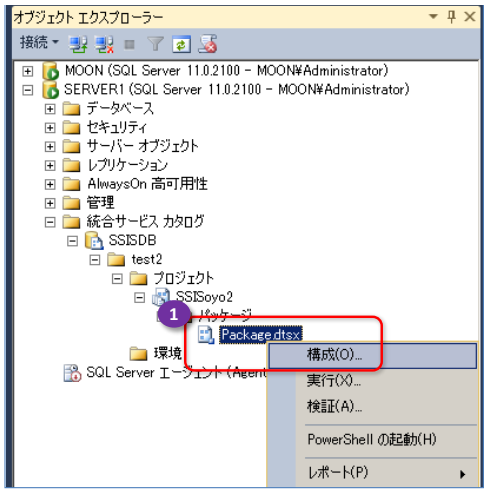


11. 次に、オブジェクト エクスプローラーで、配置先のマシンの［**SSISDB カタログ**］内の［**test2**］フォルダーを右クリックして、［**最新の情報に更新**］をクリックし、**test2** フォルダー内に **SSISoyo2** プロジェクトが配置されていることを確認します。

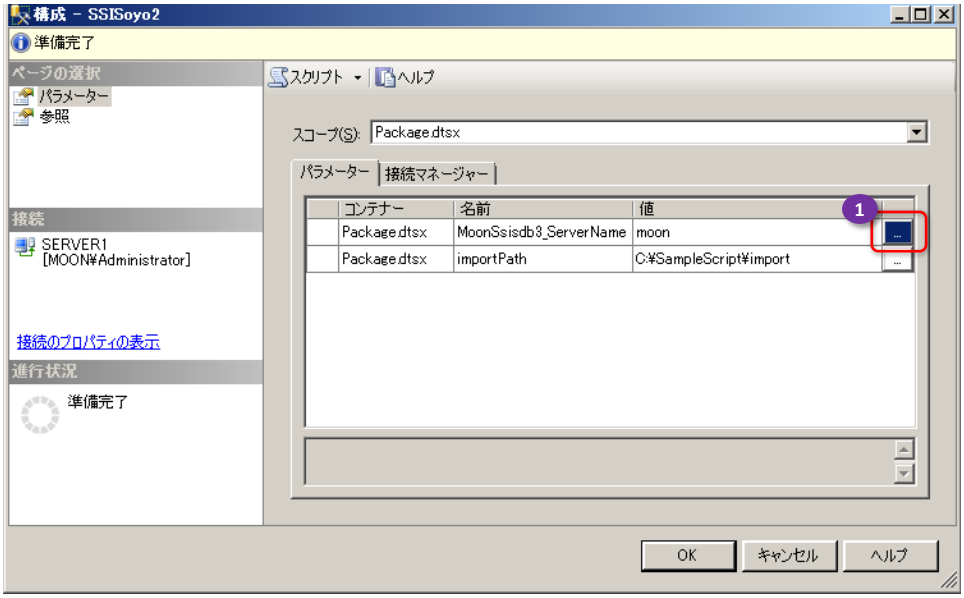


構成の編集（パラメーター値の編集）

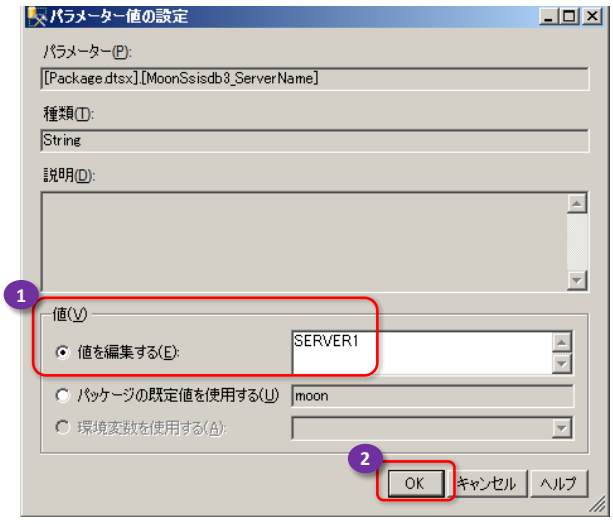
12. 次に、新しく配置した「**SSISoyo2**」プロジェクト内の「**Package.dtsx**」を右クリックして、［**構成**］をクリックし、構成の編集（パラメーター値の編集）を行います。



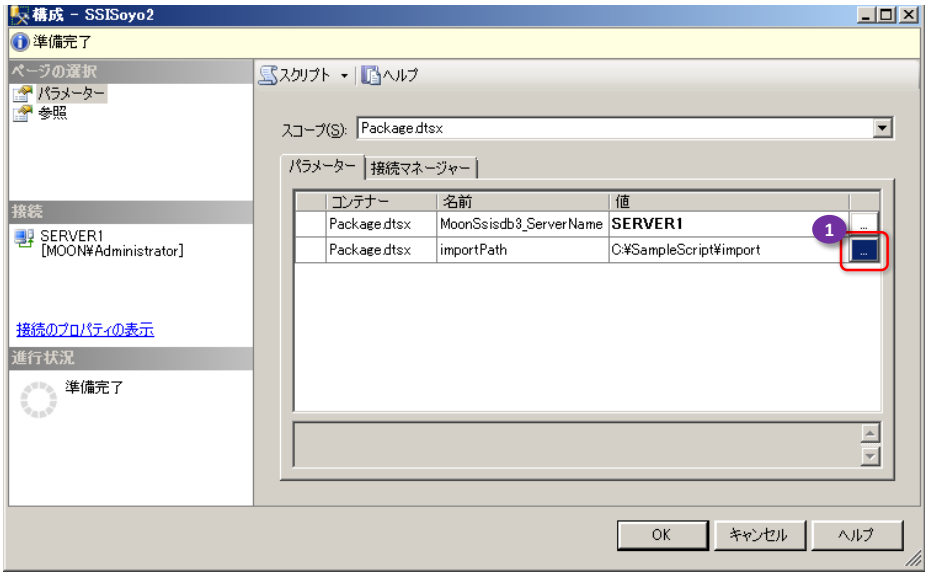
13. ［**構成**］ダイアログが表示されたら、「**サーバー名 Ssidb3_ServerName**」パラメーターの［**...**］ボタンをクリックします。



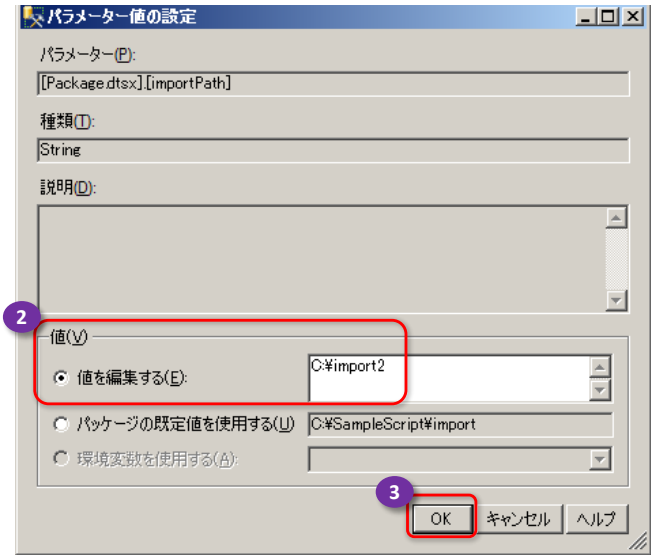
14. ［**パラメーター値の設定**］ダイアログが表示されたら、［**値**］で［**値を編集する**］を選択して、別マシンのサーバー名（画面は **SERVER1**）を入力し、［**OK**］ボタンをクリックします。



15. 次に、[**構成**] ダイアログへ戻ったら、「**importPath**」パラメーターの [**...**] ボタンをクリックします。



16. [**パラメーター値の設定**] ダイアログが表示されたら、[**値**] で [**値を編集する**] を選択して、「**import**」フォルダーをコピーしたローカル パス (画面は **C:¥import2**) へ変更し、[**OK**] ボタンをクリックします。

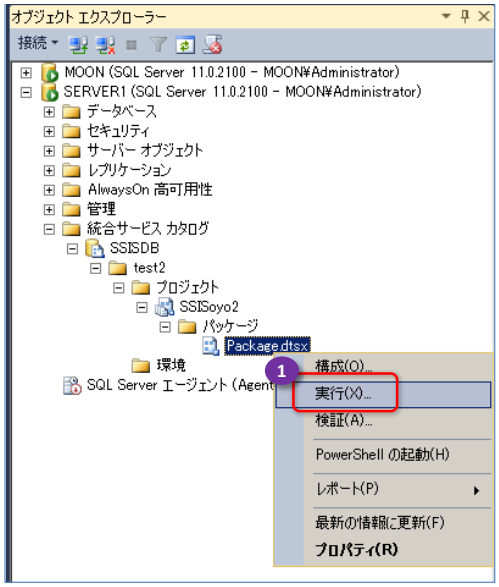


17. [**構成**] ダイアログへ戻ったら、[**OK**] ボタンをクリックして閉じます。

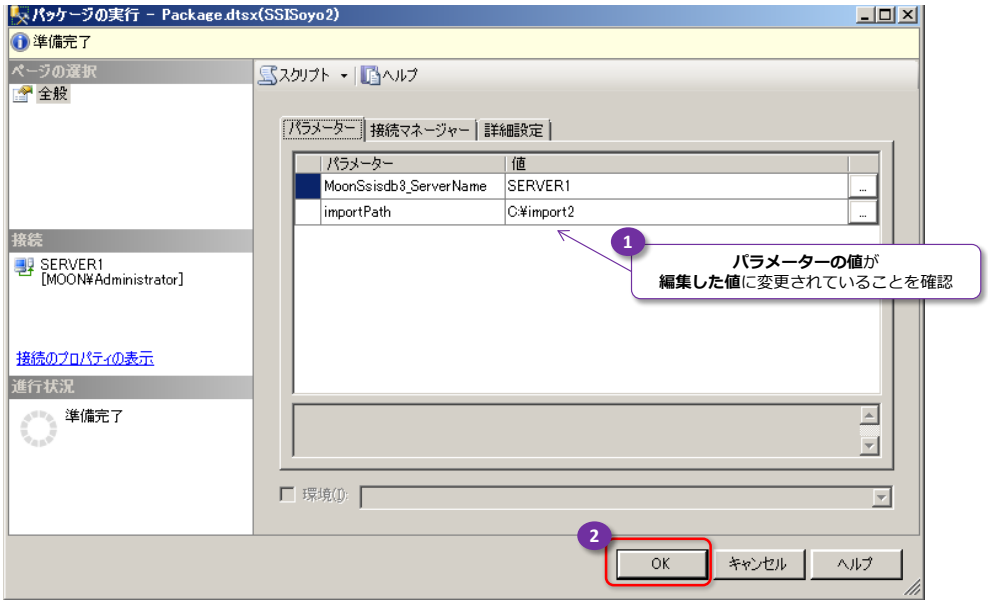
パッケージの実行

最後に、パッケージが正しく実行できるかどうかを試してみましょう。

18. オブジェクト エクスプローラーで、別マシン（画面は **SERVER1**）へ配置した **SSISoyo2** プロジェクト内の **Package.dtsx** パッケージを右クリックして、［**実行**］をクリックします。



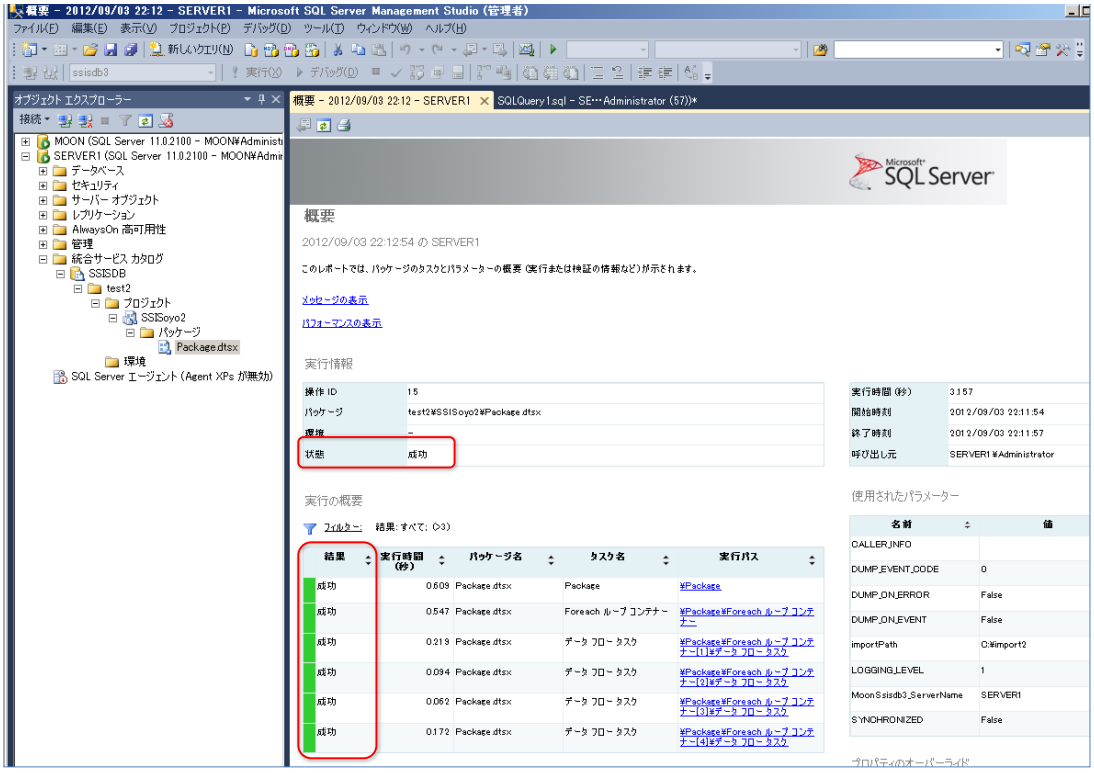
19. ［**パッケージの実行**］ダイアログが表示されたら、**パラメーターの値**が、前の手順で編集した値へ変更されていることを確認して、［**OK**］ボタンをクリックします。



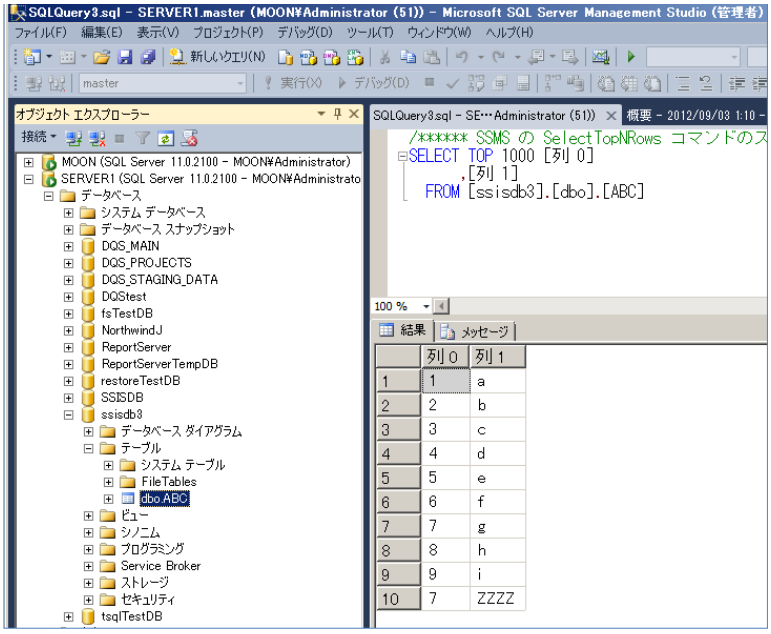
これにより、パッケージが実行されます。

20. 実行後、「**概要レポートを今すぐ開きますか？**」メッセージが表示されたら、［**はい**］ボタンをクリックします。

21. これにより、パッケージの概要レポートが表示されて、実行状態が「**成功**」であることを確認できれば、実行が成功です。



22. 完了後、Management Studio から「**ssisdb3**」データベースの「**ABC**」テーブルを開き、データが追加されていることを確認しておきます。



以上で、パッケージを別マシンへ配置する手順が完了です。このように、パラメーター化機能を利用することで、開発機と本番機でマシン構成が異なる場合でも容易に構成を変更でき、パッケージを実行できるようになるので、大変便利です。

おわりに

最後まで試された皆さま、いかがでしたでしょうか。Integration Services は、業務で大変役立つツールですので、ぜひ活用してみていただければと思います。Integration Services には、条件分割や文字マップ、ADO.NET 変換元／変換先、トランザクション、チェックポイント、Integration Services サービスへの接続など、まだまだたくさんの機能が提供されています。これらの機能については、オンライン ブックを参考にチャレンジしてみてください。

執筆者プロフィール

有限会社エスキューエル・クオリティ（http://www.sqlquality.com/）

SQLQuality（エスキューエル・クオリティ）は、**日本で唯一の SQL Server 専門の独立系コンサルティング会社**です。過去のバージョンから最新バージョンまでの SQL Server を知りつくし、多数の実績と豊富な経験を持つ、OS や .NET にも詳しい **SQL Server の専門家（キャリア 17 年以上）がすべての案件に対応します**。人気メニューの「**パフォーマンス チューニング サービス**」は、100%の成果を上げ、過去すべてのお客様環境で驚異的な性能向上を実現。チューニング スキルは**世界トップレベル**を自負、検索エンジンでは（英語情報を含めて）ヒットしないノウハウを多数保持。ここ数年は **BI／DWH システム構築支援**のご依頼が多い。

主なコンサルティング実績

- 大手映像制作会社の **BI システム構築支援**（会計／業務システムにおける予実管理／原価管理など）
- 大手流通系の **DWH／BI システム構築支援**（POS データ／在庫データ分析）
  大規模テラバイト級データ ウェアハウスの物理・論理設計支援および運用管理設計支援
- 大手アミューズメント企業の **BI システム構築支援**（人事システムにおける人材パフォーマンス管理）
- 外資系医療メーカーの Analysis Services による「**販売分析**」システムの構築支援（売上／顧客データ分析）
- **9 TB** データベースの物理・論理設計支援（パーティショニング対応など）
- ハードウェア リプレイス時の**ハードウェア選定**（最適なサーバー、ストレージの選定）、**高可用性環境**の構築
- **SQL Server 2000**（32 ビット）から **SQL Server 2008**（x64）への移行／アップグレード支援
- 複数台の SQL Server の **Hyper-V 仮想環境**への移行支援（サーバー統合支援）
- **2 時間**かかっていた日中バッチ実行時間を、わずか **5 分**へ短縮（**95.8%** の性能向上）
- ピーク時の CPU 利用率 **100%** のシステムを、わずか **10%** にまで軽減し、大幅性能向上
- 平均 **185.3ms** かかっていた処理を、わずか **39.2ms** へ短縮（**78.8%** の性能向上）
- **Java 環境**（Tomcat、Seasar2、S2Dao）の SQL Server パフォーマンス チューニング etc

コンサルティング時の作業例（パフォーマンス チューニングの場合）

- アプリケーション コード（VB、C#、Java、ASP、VBScript、VBA）の解析／改修支援
- ストアド プロシージャ／ユーザー定義関数／トリガー（Transact-SQL）の解析／改修支援
- インデックス チューニング／SQL チューニング／ロック処理の見直し
- 現状のハードウェアで将来のアクセス増にどこまで耐えられるかを測定する高負荷テストの実施
- IIS ログの解析／アプリケーション ログ（log4net／log4j）の解析
- ボトルネック ハードウェアの発見／ボトルネック SQL の発見／ボトルネック アプリケーションの発見
- SQL Server の構成オプション／データベース設定の分析／使用状況（CPU, メモリ, ディスク, Wait）解析
- 定期メンテナンス支援（インデックスの再構築／断片化解消のタイミングや断片化の事前防止策など）etc

松本美穂（まつもと・みほ）

有限会社エスキューエル・クオリティ 代表取締役
Microsoft MVP for SQL Server（2004 年 4 月～）
経産省認定データベース スペシャリスト／MCDBA／MCSD for .NET／MCITP Database Administrator
SQL Server の日本における最初のバージョンである「SQL Server 4.21a」から SQL Server に携わり、現在、SQL Server を中心とするコンサルティングを行っている。得意分野はパフォーマンス チューニングと Reporting Services。コンサルティング業務の傍ら、講演や執筆も行い、マイクロソフト主催の最大イベント Tech・Ed などでスピーカーとしても活躍中。SE や ITPro としての経験はもちろん、記名／無記名含めて多くの執筆実績も持ち、様々な角度から SQL Server に携わってきている。著書の『SQL Server 2000 でいってみよう』と『ASP.NET でいってみよう』（いずれも翔泳社刊）は、トップ セラー（前者は 28,500 部、後者は 16,500 部発行）。近刊に『SQL Server 2012 の教科書』（ソシム刊）がある。

松本崇博（まつもと・たかひろ）

有限会社エスキューエル・クオリティ 取締役
Microsoft MVP for SQL Server（2004 年 4 月～）
経産省認定データベース スペシャリスト／MCDBA／MCSD for .NET／MCITP Database Administrator
SQL Server の BI システムとパフォーマンス チューニングを得意とするコンサルタント。過去には、約 3,000 本のストアド プロシージャのチューニングや、テラバイト級データベースの論理・物理設計、運用管理設計、高可用性設計、BI・DWH システム設計支援などを行う。アプリケーション開発（ASP／ASP.NET、C#、VB 6.0、Java、Access VBA など）やシステム管理者（IT Pro）経験もあり、SQL Server だけでなく、アプリケーションや OS、Web サーバーを絡めた、総合的なコンサルティングが行えるのが強み。Analysis Services と Excel による BI システムも得意とする。マイクロソフト認定トレーナー時代の 1998 年度には、Microsoft CPLS トレーナー アワード（Trainer of the Year）を受賞。