http://www.sqlservercentral.com/articles/SSIS/61360/
Printed 2007/11/15 07:51AM

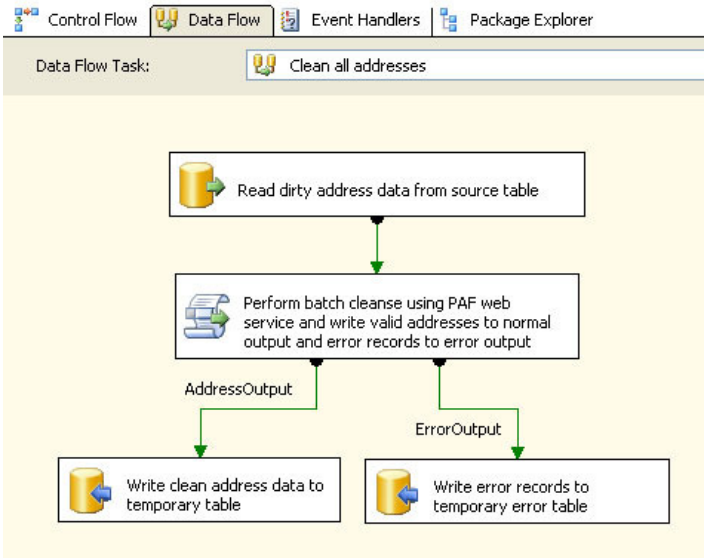
Cleaning Address Data with SSIS Using a Web Service

By Dr. John Tunnicliffe, 2007/10/19

Cleaning up address data from legacy systems is not always easy to achieve. Moreover, calling a web service to process batches of addresses from within an SSIS data flow is not for the uninitiated! This article not only outlines an easy way to call a web service from within the data flow, but also how to configure a Script Component for asynchronous batch processing of 1,000 addresses at a time.

The Data Flow

The screenshot below depicts our Data Flow in the Visual Studio designer.



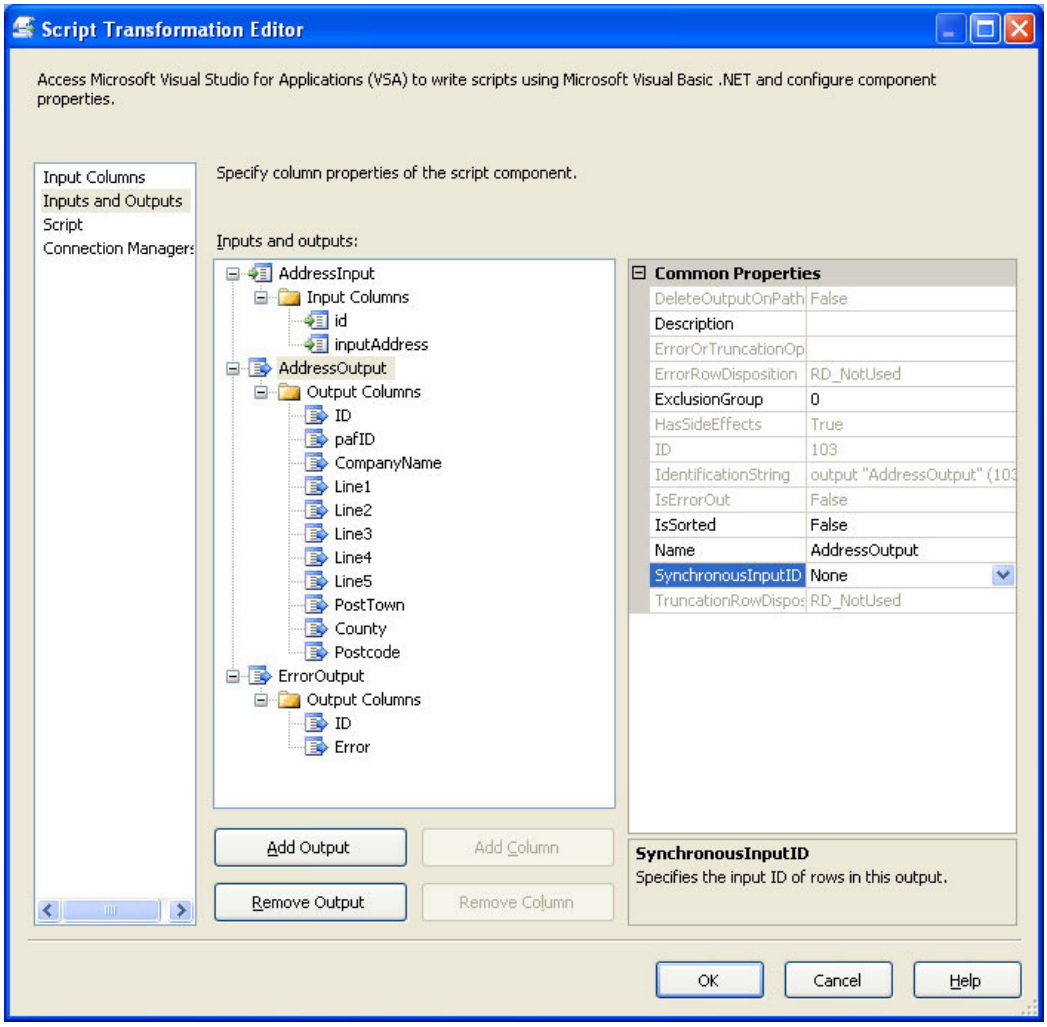
The data flow looks pretty straight forward. However, when you look under the covers of the Script Component, things are a little more intricate.

The Web Service

The guys at Postcode Anywhere provide a flexible web service that validates and cleans international address data. They also do stuff like lifestyle profiling, route planning and credit card validation. Their UK service is based on the Royal Mail's Postcode Address File (PAF) which contains over 28 million addresses. A quick look at the developer documentation reveals the Cleanse method that validates and cleans a single address at a time. Clearly we could call the Cleanse method once for each row in our data flow. That would be ok for one or two rows, but if we want to validate over a hundred thousand records, then we need something far more efficient. Enter BatchCleanse which validates a thousand addresses at a time. Much more efficient! But how do we get a row-based SSIS transform to call a batch-orientated web service?

Building an Asynchronous Batch-Orientated Script Component

Anyone familiar with the Script Component will know that it can be used in three forms: as a Source, a Destination and as a Transformation. In its default form, the Transformation is synchronous (i.e. one line of output is written for every line of input). As we want to perform batch processing, the first thing we need to do is to disassociate the input from the output (i.e. make it asynchronous). This can be done by changing the SynchronousInputID of the output to "None" (see screenshot below).



The Postcode Anywhere web service requires the address to be comma delimitated. So if your data is not in this format, then simply add another script component to transform the data.

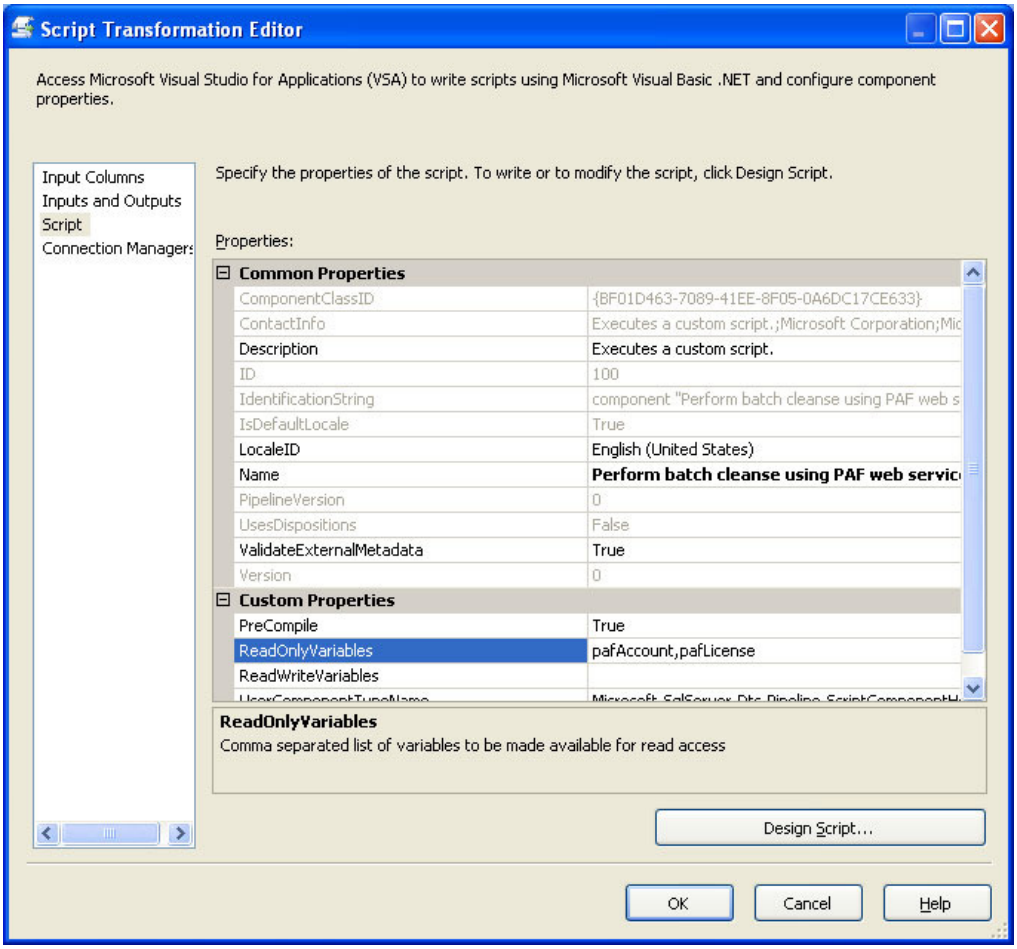
Setting up Inputs and Outputs

Before writing any code, it is important to setup your inputs and outputs correctly on your new Script Component as SSIS uses this information to automatically generate wrappers around your input and output data flow buffers.

The inputs are easy as they come automatically from your data source. The data outputs are more time consuming to set up, but you are better off doing this now before attempting to write any script as, without an output buffer, your script will not be able to achieve anything!

Passing in License Keys

The Postcode Anywhere web service requires the account and license key details to be passed in every call. In order to make this easy to configure, I have created two variables: pafAccount and pafLicense. The value of these variables can then be set in the package configuration file rather than editing the code directly.



Calling a Web Service from a Script Component

SQL Server Integration Services (SSIS) is a very flexible ETL and data cleansing tool. However, in its current incarnation, SSIS does not make it easy to call a web service within the Data Flow task as Visual Studio for Applications (VSA) does not have the Add Web Reference command that is familiar to Visual Studio developers. Indeed there is precious little documentation on the topic.

So, to call a web service from within an SSIS script component you have two options:

1. Create a proxy class, in any .NET language, using the command prompt utility wsdl.exe and compile this into a strong-named assembly which is then registered in the GAC.
2. Alternatively create a Visual Basic proxy class using wsdl.exe and import the new class directly into your VSA project.

The advantage of the first approach is that several script components can reference the same assembly. However, it has some serious disadvantages. Firstly, your new assembly will not appear on the Add Reference dialog box within VSA unless you place a copy in either the %windir%\Microsoft.NET\Framework\v2.0.xxxxx or the %ProgramFiles%\Microsoft SQL Server\90\SDK\Assemblies folder. Secondly, DBAs are often reluctant to place anything in the GAC on production servers!

In practice, we found that option 2) provided a much better approach because the class is easy to import into the VSA project and thereafter the code is stored inside the SSIS package, which simplifies deployment and, of course, simultaneously pleases our DBAs!

Much to the annoyance of all serious developers, Visual Basic.NET is the only language currently supported by SSIS. However, this is something that will be resolved in the forthcoming Katmai release of SQL Server. So to create the Visual Basic proxy class for the Postcode Anywhere service, use the following command-line:

```
wsdl /language:VB http://services.postcodeanywhere.co.uk/uk/lookup.asmx?wsdl /out:c:\PostcodeAnywhere.vb
```

Now click the Design Script button on your Script Component (having already defined all the inputs and outputs!) and add the new proxy class by selecting "Add Existing Item..." from the Project menu. You may need to delete the first few "garbage" characters, which are the Unicode byte order mark. You will now need to add a reference to the following .NET assemblies: System.Xml and System.Web.Services. Without these references, the proxy class will not compile.

The Guts of the Code

The AddressInput_ProcessInputRow subroutine is called by SSIS for every row of data in our AddressInput buffer. In order to batch up the addresses, we add them to a Dictionary object for later processing as follows:

```
Public Class AddressKeyValuePair
  Public DatabaseId As Integer  ' record id from the database
  Public wholeAddress As String ' comma delimated address

  Public Sub New(ByVal id As Integer, ByVal sInputAddress As String)
```

```
    DatabaseId = id
    wholeAddress = sInputAddress
  End Sub
End Class

Public Class ScriptMain
...
Public Overrides Sub AddressInput_ProcessInputRow(ByVal Row As AddressInputBuffer)
  ' add address to our dictionary object
  BatchOfAddresses.Add(mRow, New AddressKeyValuePair(Row.id, Row.inputAddress))
  ' increment row counter
  mRow = mRow + 1
  ' max batch size for Postcode Anywhere BatchCleanse is 1000
  If BatchOfAddresses.Count >= 1000 Then
        ProcessBatchOfAddresses()
        mRow = 0
  End If
End Sub ...
```

It is the call to ProcessBatchOfAddresses that does the real work of calling the Postcode Anywhere web service. The code for this is shown below along with the code for a helper function called GetReferenceTo¬PostcodeAnywhere-Service which does exactly what it says on the tin!

```
Private Sub ProcessBatchOfAddresses()
  Dim pafResults As AddressResults
  Dim iRow As Integer
  Dim pafWebService As LookupUK ' reference to the paf web service
  If mPafErrorCode = 0 Then
    If BatchOfAddresses.Count > 0 Then
      ' now get a reference to the Postcode Anywhere web service
      pafWebService = GetReferenceToPostcodeAnywhereService()
      If pafWebService Is Nothing Then
        Err.Raise(604, Nothing, "ProcessBatchOfAddresses: Failed to create instance of PostcodeAnywhere web service")
      End If
      Try
        ' call Postcode Anywhere web service
        Dim batch As List(Of String) = New List(Of String)
        For Each addressToProcess As AddressKeyValuePair In BatchOfAddresses.Values
          batch.Add(addressToProcess.wholeAddress)
        Next
        pafResults = pafWebService.BatchCleanse(batch.ToArray(), enLanguage.enLanguageEnglish, enContentType.enContentSt
        Catch ex As Exception
          ' this error is only raised if you have no internet connectivity
          Err.Raise(605, Nothing, "ProcessBatchOfAddresses: Error calling Postcode Anywhere service: " & ex.Message)
        End Try
        Try
          If pafResults.IsError Then
            ' if Postcode Anywhere web service returns an error, it is usually
            ' due to your account having no credit left, so record error and
            ' do not bother calling web service again
            mPafErrorCode = pafResults.ErrorNumber
            Err.Raise(606, "ProcessBatchOfAddresses: pafResults.ErrorMessage " & pafResults.ErrorMessage, pafResults.Err
          Else
           ' process results
           iRow = 0
           For Each outputAddress As Address In pafResults.Results
             If outputAddress.Postcode = "" Then
                With Me.ErrorOutputBuffer
                  .AddRow()
                  .ID = BatchOfAddresses.Item(iRow).DatabaseId
                  .Error = "Failed paf Address lookup"
                End With
             Else
                With Me.AddressOutputBuffer
                  .AddRow()
                  .pafID = outputAddress.Id
                  .ID = BatchOfAddresses.Item(iRow).DatabaseId
                  .OrganisationName = outputAddress.OrganisationName
                  .Line1 = outputAddress.Line1
                  .Line2 = outputAddress.Line2
                  .Line3 = outputAddress.Line3
                  .Line4 = outputAddress.Line4
                  .Line5 = outputAddress.Line5
                  .PostTown = outputAddress.PostTown
                  .Postcode = outputAddress.Postcode
                  .County = outputAddress.County
                End With
             End If
             iRow = iRow + 1
        Next
        End If
        Catch ex As Exception
                Err.Raise(607, Nothing, "ProcessBatchOfAddresses: Error processing results: " & ex.Message)
        End Try
     End If
     End If
    BatchOfAddresses.Clear()
End Sub

Public Function GetReferenceToPostcodeAnywhereService() As LookupUK
  ' we cannot rely on placing this code in AquireConnections
  If mPafWebService Is Nothing Then
    ' create new instance of the Postcode Anywhere web service
    mPafWebService = New LookupUK()
    mPafWebService.Timeout = 30000
    ' must provide proxy settings to allow the call to the web service
    ' to get through firewall
    Dim wp As WebProxy = CType(GlobalProxySelection.Select, WebProxy)
```

```
    wp.Credentials = CredentialCache.DefaultCredentials
    mPafWebService.Proxy = wp
    GetReferenceToPostcodAnywhereService = mPafWebService
  End If
End Function
```

If you are familiar with coding the Script Component, you will know that you are supposed to override the AcquireConnections method and instantiate connections there. However, with asynchronous script components the AcquireConnections method is not fired in the order you would expect, so ProcessBatchOfAddresses chokes when it attempts to process the first batch. To overcome this, the GetReferenceTo¬PostcodeAnywhere¬Service helper function ensures the web service is instantiated correctly before any web methods are invoked.

The sample code provides a fully operational package plus SQL scripts to create the database tables and populate the source table with a little test data. See the notes below for guidance on how to configure everything.

Happy coding

Dr John
dr.john.tunnicliffe*btinternet.com

Using the Sample Code

You may download the sample code from here. Once downloaded, you will need to alter/edit the following things to get it to work on your system:

Using the sample code

Item to configure	Configuration Notes
Database tables	Run the create_paf_sample_tables.sql script against your database to create the tables. Run address_data.sql to populate the input_dirty_address_data table with some data.
pafAccount and pafLicense variables	Visit www.postcodeanywhere.co.uk and obtain your own account and license codes. Then edit the value of the two variables in the pafExample package.
paf_sample_db connection manager	Edit the connection manager in the pafExample package and enter your own database connection details.