http://www.sqlservercentral.com/articles/Integration+Services+(SSIS)/69768/
Printed 2010/04/29 07:59AM

Capturing Real-Time Currency Conversions in SSIS

By Jeff Singleton, 2010/03/10

Overview

With international commerce growing every day, the means by which funds are exchanged grows as well. From exchange rate calculations to international banking/trading and companies that do business internationally the need to convert currencies in real time, is becoming a necessity in the global marketplace. In a typical foreign exchange transaction, a party purchases a quantity of one currency by paying a quantity of another currency using floating exchange rates.

In this article, I will demonstrate a data flow solution for converting Canadian currency using real time floating exchange rates through the use of a web service task, a record set result, and parameter driven dynamic SQL.

Implementation

First let's take a look at the data to be converted. Here we have a SQL table with an amount and a currency code. The plan is to convert the Canadian dollar (CAD) amount into US dollars (USD). To capture the real time conversion rate, we will first need to call on a Web Service Task.

Once you have imported the Web Service Task into the control flow, you will need to configure it. I am going to use a handy site (http://www.webservicex.net) that provides free access to web services. Using WSDL (Web Services Description Language) which is an XML-based language that provides a model for describing Web services, we are able to call on this web service to get the floating exchange rate.

When configuring the web service task, click on HttpConnection and 'New Connection'. At the Http Connection Manager screen, copy and paste the full path to the web service WSDL. Also copy and paste that same link into a browser window and click file and 'Save As' to save the physical WSDL file to your local machine. Back at the Web Service Task Editor, Import the WSDL file that you just saved from the browser window to your local machine as the WSDLFile on the Web Service Task Editor.

On the Input tab, you will see a service drop down; choose CurrencyConvertor. Under Method choose ConversionRate. One the pane below that, you will be able to select the currency that you want to convert from to the currency that you want to convert to. In this example, I am converting from CAD to USD.

Next, create a Data Flow Task. We will use this data flow task to dynamically parse the exchange rate and store it in a Record Set destination. In the Data Flow Task, drag and drop a Flat File Source and choose a new Flat File Connection Manager. When configuring the Flat File Connection Manager, choose the XML file that you are using to hold the output of the currency conversion. Normally, you could use an XML source as the file but to show some more complex ETL, I am going to read this data in as a flat file and parse the exchange rate in a Derived Column Transformation using logic that will dynamically determine the length of the conversion rate. Under file format, choose ragged right and also select 1 header row to skip so that we are only going to read the data in the file and not the XML header. Since there is only 1 record and 1 column, you will only have 1 column in the advanced tab that we will call 'ConversionRate'.

Now, drag and drop a Derived Column transformation and connection it to the flat file source. Here, you will have a single Derived column to capture only the conversion rate. Using substring and findstring, you can dynamically parse the field to take everything from the 9th byte (where the field starts) to the character "</" which denotes the end of the string. This way, if the field is 10 characters long or 40 characters long, we will only capture the conversion rate and no other data.

On the Output tab, choose File Connection and create a simple XML file to hold the output of the **conversion.**

Next, connect a Data Conversion Transformation to your data flow and convert the ConversionRate field to Output as ConversionNumeric as a data type DT_NUMERIC with a precision of 18 and a scale of 6.

As the last part of the Data Flow, we are going to store the conversion rate in a Record Set Destination. A Record Set Destination creates and populates and in-memory ADO recordset. We can then utilize this recordset later in the package. Under the Recordset Destination Editor, create a new variable called ExecutedConversion at the package scope.

Your only input column will be the ConversionNumeric field that you created in the data conversion transform.

Now, create a Foreach Loop in your control flow connected to your data flow task. Under the connection tab, change the Enumerator to a 'Foreach ADO Enumerator'. Choose the Recordset variable 'User::ExecutedConversion' as the ADO object source variable and 'Rows in the first table' as your Enumeration mode.

On the Variable Mappings tab, again select the variable 'User::ExecutedConversion' and set the Index value equal to 0. These two steps will capture the conversion rate and enumerate (or sequentially list) and pass the values captured into the Foreach Loop.

Once the Foreach Loop has been configured, drag and drop an Execute SQL task into the Foreach Loop. Set your connection to the database that has the table with the amounts to be converted, set BypassPrepare equal to true and use the below code as your SQLStatement:

```
UPDATE CurrencyConversion

SET USD_Amount = Amount * ?,

ConvertedDate = getdate()

WHERE USD_Amount IS NULL
```

The question mark in the SQL statement will use a value passed to the Parameter Mapping of the Execute SQL Task. Select 'User::ExecutedConversion' as the variable name with a data type of FLOAT , a Parameter Name of 0 and a Parameter Size as -1.

Now, execute your package and check your table to verify that the currency conversion has worked properly.

Conclusion

In this article, I've demonstrated how to use SQL Server Integration Services (SSIS) to create a process that can capture real-time currency conversion rates, parse the conversion rate dynamically with the use of a derived column transformation and pass the conversion rate to a record set to be used in a SQL task.