DATABASE JOURNAL
The Knowledge Center for Database Professionals

SQL Server 2008 and 2008 R2 Integration Services - Using Script Task To Implement FTP Functionality

July 1, 2011

As we recently demonstrated, SQL Server 2008 R2 Integration Services (SSIS) includes a specialized task that allows you to incorporate FTP transfers (http://www.databasejournal.com/features/mssql/sql-server-integration-services-ftp-task.html) (as well as the most basic file system based operations) into SSIS packages. While having such a task available makes sending and receiving files via FTP protocol during package execution very straightforward, this simplicity comes at the price of limited flexibility. In particular, such an approach does not provide the ability to designate multiple arbitrarily named files to be transferred (even though this can be somewhat mitigated by leveraging wildcards). Fortunately, you can eliminate this handicap by implementing equivalent functionality using Script Task.

In order to accomplish our objective, we will take advantage of SqlServer.Dts.Runtime.ConnectionManager class (which provides access to both generic and FTP Connection Manager-specific properties and methods) in combination with SqlServer.Dts.Runtime.FtpClientConnection (http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.dts.runtime.ftpclientconnection.aspx) class (which facilitates uploads and downloads via SendFiles (http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.dts.runtime.ftpclientconnection.sendfiles.aspx) and ReceiveFiles (http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.dts.runtime.ftpclientconnection.receivefiles.aspx) methods, as well as management of source and target file system).

To implement sample code illustrating their use, launch Business Intelligence Development Studio and create a new project based on the Integration Services template. Drag the Script Task icon from the Toolbar and drop it onto the Designer interface. With the newly generated task highlighted, open the Variables window and define variables of String data type and Script Task scope that will determine characteristics of the file transfer (and, for the most part, map to respective properties of the FTP Connection Manager, which we will generate dynamically in our code):

- FPTServer - name or IP address of the FTP server operating as either the source or destination of the transfer (corresponding to the ServerName FTP Connection Manager property)
- FTPUser - name of the user that can be successfully authenticated and authorized to access the FTP server (corresponding to the ServerUserName FTP Connection Manager property)
- FTPPassword - password of the user (corresponding to the ServerPassword FTP Connection Manager property). Note that the authentication mechanism is limited to either basic or anonymous authentication. If you are concerned about storing passwords in clear text in your package, you might want to consider developing a custom mechanism that encrypts it (although keep in mind that as part of FTP session, its content will be transmitted in clear text anyway).
- FileToSend1 and FileToSend2 - full local paths of files to be transferred to the FTP server (we decided to use two for the sake of simplicity, but in general, their number is arbitrary)
- FileToReceive1 and FileToReceive2 - paths of files to be transferred from the FTP server (starting from the FTP root folder). Just as with files being sent out, you can easily add others.
- LocalPath - path designating source or destination folder in the local file system
- RemotePath - path designating source or destination folder on the FTP server

Note that we could also define in the same manner other properties of the FTP Connection Manager, such as ServerPort (designating port utilized by the FTP server and set by default to 21), TimeOut (representing maximum amount of time in seconds before the FTP transfer is abandoned if the target server is not responding), ChunkSize (determining amount of data in KB transported in individual FTP packets), Retries (identifying maximum number of connection attempts), or UsePassiveMode (allowing

to switch between passive and active mode), but in our case, we set their values directly within the Script Task.

Next, use the context-sensitive menu of the Script Task to display its Editor dialog box. Designate Visual Basic .NET 2008 as the ScriptLanguage and specify User::FiletoReceive1, User::FiletoReceive2, User::FiletoSend1, User::FiletoSend2, User::FTPPassword, User::FTPServer, User::FTPUser, User::LocalPath, and User::RemotePath as ReadOnlyVariables. Finally, click on Edit Script... command button to reveal the Visual Studio Tools for Applications 2.0 interface, where you enter the following code:

```
Public Sub Main()


Const FTPPort As String = "21"

Const FTPTimeOut As String = "60"

Const FTPChunkSize As String = "2"

Const FTPRetries As String = "5"


Dim propList As String = ""

Dim overwrite As Boolean = True

Dim isTransferAsci As Boolean = True

Dim filesToSend() As String = {Dts.Variables("FileToSend1").Value.ToString, _

Dts.Variables("FileToSend2").Value.ToString}

Dim filesToReceive() As String = {Dts.Variables("FileToReceive1").Value.ToString, _

Dts.Variables("FileToReceive2").Value.ToString}


  Try


    Dim ftpCM As ConnectionManager = Dts.Connections.Add("FTP")


    ftpCM.Properties("ServerName").SetValue(ftpCM, Dts.Variables("FTPServer").Value.ToString)

    ftpCM.Properties("ServerUserName").SetValue(ftpCM, Dts.Variables("FTPUser").Value.ToString)

    ftpCM.Properties("ServerPassword").SetValue(ftpCM, Dts.Variables("FTPPassword").Value.ToString)

    ftpCM.Properties("ServerPort").SetValue(ftpCM, FTPPort)

    ftpCM.Properties("Timeout").SetValue(ftpCM, FTPTimeOut)

    ftpCM.Properties("ChunkSize").SetValue(ftpCM, FTPChunkSize)

    ftpCM.Properties("Retries").SetValue(ftpCM, FTPRetries)

    ftpCM.Properties("UsePassiveMode").SetValue(ftpCM, False)


    Dim ftpCC As FtpClientConnection = New FtpClientConnection(ftpCM.AcquireConnection(Nothing))
```

```
    ftpCC.Connect()

    ftpCC.SendFiles(filesToSend, Dts.Variables("RemotePath").Value.ToString, overwrite, isTransferAsci)

    ftpCC.ReceiveFiles(filesToReceive, Dts.Variables("LocalPath").Value.ToString, overwrite,
isTransferAsci)

    ftpCC.Close()


    Dts.TaskResult = ScriptResults.Success


  Catch ex As Exception


    MessageBox.Show(ex.Message.ToString, "Exception")

    Dts.TaskResult = ScriptResults.Failure


  End Try


End Sub
```

See all articles by Marcin Policht (http://www.databasejournal.com/article.php/1503191/Marcin-Policht.htm)