```
#region Help:  Introduction to the script task
/* The Script Task allows you to perform virtually any operation that can be accomplished in
 * a .Net application within the context of an Integration Services control flow.
 *
 * Expand the other regions which have "Help" prefixes for examples of specific ways to use
 * Integration Services features within this script task. */
#endregion



#region Namespaces
using System;
using System.Data;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
#endregion

namespace ST_95114369a7f64120921fb89754566827
{
    using System.IO;


    /// <summary>
    /// ScriptMain is the entry point class of the script.  Do not change the name, attributes,
    /// or parent of this class.
    /// </summary>
    [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
    public partial class ScriptMain :
Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
    {
```

```
#region Help:  Using Integration Services variables and parameters in a script
/* To use a variable in this script, first ensure that the variable has been added to
 * either the list contained in the ReadOnlyVariables property or the list contained in
 * the ReadWriteVariables property of this script task, according to whether or not your
 * code needs to write to the variable.  To add the variable, save this script, close this instance of
 * Visual Studio, and update the ReadOnlyVariables and
 * ReadWriteVariables properties in the Script Transformation Editor window.
 * To use a parameter in this script, follow the same steps. Parameters are always read-only.
 *
 * Example of reading from a variable:
 *  DateTime startTime = (DateTime) Dts.Variables["System::StartTime"].Value;
 *
 * Example of writing to a variable:
 *  Dts.Variables["User::myStringVariable"].Value = "new value";
 *
 * Example of reading from a package parameter:
 *  int batchId = (int) Dts.Variables["$Package::batchId"].Value;
 *
 * Example of reading from a project parameter:
 *  int batchId = (int) Dts.Variables["$Project::batchId"].Value;
 *
 * Example of reading from a sensitive project parameter:
 *  int batchId = (int) Dts.Variables["$Project::batchId"].GetSensitiveValue();
 * */

#endregion

#region Help:  Firing Integration Services events from a script
/* This script task can fire events for logging purposes.
```

```
 *
 * Example of firing an error event:
 *  Dts.Events.FireError(18, "Process Values", "Bad value", "", 0);
 *
 * Example of firing an information event:
 *  Dts.Events.FireInformation(3, "Process Values", "Processing has started", "", 0, ref fireAgain)
 *
 * Example of firing a warning event:
 *  Dts.Events.FireWarning(14, "Process Values", "No values received for input", "", 0);
 * */
#endregion

#region Help:  Using Integration Services connection managers in a script
/* Some types of connection managers can be used in this script task.  See the topic
 * "Working with Connection Managers Programatically" for details.
 *
 * Example of using an ADO.Net connection manager:
 *  object rawConnection = Dts.Connections["Sales DB"].AcquireConnection(Dts.Transaction);
 *  SqlConnection myADONETConnection = (SqlConnection)rawConnection;
 *  //Use the connection in some code here, then release the connection
 *  Dts.Connections["Sales DB"].ReleaseConnection(rawConnection);
 *
 * Example of using a File connection manager
 *  object rawConnection = Dts.Connections["Prices.zip"].AcquireConnection(Dts.Transaction);
 *  string filePath = (string)rawConnection;
 *  //Use the connection in some code here, then release the connection
 *  Dts.Connections["Prices.zip"].ReleaseConnection(rawConnection);
 * */
#endregion
```

```
    /// <summary>
    /// This method is called when this script task executes in the control flow.
    /// Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.
    /// To open Help, press F1.
    /// </summary>
    public void Main()
    {
        //TESTING CODE
        // MessageBox.Show("FileName: " + Dts.Variables["FileName"].Value.ToString() + " MonthDigit: "
+ Dts.Variables["MonthDigit"].Value.ToString() +
        //  " YearDigit: " + Dts.Variables["YearDigit"].Value.ToString() + " DestinationFileFolder: " +
Dts.Variables["DestinationFileFolder"].Value.ToString());
        //
        try
        {
            //SET VARIABLES FOR SOURCE AND DESTINATION
            var SourceFileFolder = Dts.Variables["CorrespondenceSourceFileFolder"].Value.ToString();
            var SourceFileNameNoExtension = Dts.Variables["FID"].Value.ToString();
            var DestinationFileFolder =
Dts.Variables["CorrespondenceDestinationFileFolder"].Value.ToString() +
Dts.Variables["YearDigit"].Value.ToString() + "\\" + Dts.Variables["MonthDigit"].Value.ToString() + "\\";
            var DestinationFilePath = "";
            var SourceFilePathFull = "";
            string[] destinationFiles = new string[0];
            //Vars for Error Logging
            Dts.Variables["FIDError"].Value = Dts.Variables["FID"].Value.ToString();
            Dts.Variables["InternalIDError"].Value = Dts.Variables["InternalID"].Value.ToString();
```

```
        Dts.Variables["DateCorrespondenceError"].Value =
Dts.Variables["DateCorrespondence"].Value.ToString();



        //LOOPING ROWS AND LOGGING TEST
        //if (SourceFileNameNoExtension== "36121")
        //{
        //throw new IndexOutOfRangeException();
        //}
        //MessageBox.Show("InternalID: " + Dts.Variables["InternalID"].Value.ToString());
        //


        //IF DESTINATION FOLDER DOES NOT EXIST, CREATE IT
        var sourcefiles = Directory.GetFiles(SourceFileFolder, SourceFileNameNoExtension + ".*");
        if (Directory.Exists(DestinationFileFolder))
        {
           destinationFiles = Directory.GetFiles(DestinationFileFolder, SourceFileNameNoExtension +
".*");
        }


        //IF DESTINATION FILE DOES NOT EXIST
        if (destinationFiles.Length == 0)
        {
           //CHECK IF SOURCE FILE EXISTS
           if (sourcefiles.Length > 0)
           {
              DestinationFilePath = DestinationFileFolder + Path.GetFileName(sourcefiles[0]);
              SourceFilePathFull = sourcefiles[0];
              //IF DESTINATION FOLDER DOES NOT EXIST YET, CREATE IT
```

```
                if (!Directory.Exists(DestinationFileFolder))
                {
                    Directory.CreateDirectory(DestinationFileFolder);
                }
                //IF DESTINATION FILE DOES NOT EXIST YET COPY IT AND DELETE IT FROM SOURCE
                if (!File.Exists(DestinationFilePath))
                {
                    File.Copy(SourceFilePathFull, DestinationFilePath);
                    File.Delete(SourceFilePathFull);

                }
            }
            //FILE IN DATATABLE DOES NOT EXIST AT SOURCE, ADD MESSAGE IN DESTINATION FILEPATH COLUMN
            else
            {
                DestinationFilePath = "File does not exist at source";
            }
            Dts.Variables["DestinationFilePath"].Value = DestinationFilePath;
        }
        else
        {
            Dts.Variables["DestinationFilePath"].Value = DestinationFileFolder + Path.GetFileName(destinationFiles[0]);
        }

        //LOOPING AND LOGGING TEST
        //throw new  IndexOutOfRangeException();
```

```
        }
        //RUNTIME ERROR LOGGED.  THIS ERROR WILL BE PROPAGATED AS A NEW FIREINFORMATION EVENT AT THE EVENTS LEVEL, DEBUGGING INFO WILL BE LOGGED AND THE LOOP WILL CONTINUE
        catch (Exception ex)
        {
            bool fireAgain = false;
            Dts.Events.FireInformation(0, "Script Task Information Error - CREATE FOLDER AND MOVE FILE", "FID: " + Dts.Variables["FID"].Value.ToString() + " InternalID: " + Dts.Variables["InternalID"].Value.ToString() + "\r ERROR: " + ex.Message + "\r" + ex.StackTrace, String.Empty, 0, ref fireAgain);
            Dts.Events.FireError(0, "Script Task Error Error - CREATE FOLDER AND MOVE FILE", "FID: " + Dts.Variables["FID"].Value.ToString() + " InternalID: " + Dts.Variables["InternalID"].Value.ToString() + "\r ERROR: " + ex.Message + "\r" + ex.StackTrace, String.Empty, 0);
            Dts.Variables["ErrorCount"].Value = (int)Dts.Variables["ErrorCount"].Value + 1;
        }
        finally
        {

            if ((int)Dts.Variables["ErrorCount"].Value > 10)
            {
                Dts.TaskResult = (int)ScriptResults.Failure;
            }
            else
            {
                Dts.TaskResult = (int)ScriptResults.Success;
            }
        }

    }
```

```
        #region ScriptResults declaration
        /// <summary>
        /// This enum provides a convenient shorthand within the scope of this class for setting the
        /// result of the script.
        ///
        /// This code was generated automatically.
        /// </summary>
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion

    }
}
```