Appenate Data integration.

Technical Specification for AppenateData application to be created to fulfill TFS task number 4611

Author: Peter Knudson

Date: December 26, 2019

Problem Description

This specification was created to describe an application to fulfill the needs described in TFS task number 4611. It involves a request from Mark Eaton to create data from ViewPoint which will be placed in a location that Appenate can retrieve the data file.

Solution Short Description

A new C# console application will be created that retrieves data from the ViewPoint database via a TSQL query. The data is then, SFTP’d to the McKinstry SFTP site. This will be done hourly, 7x24. The file can then be retrieved from the McKinstry SFTP folder by Appenate.

Solution Description full description

1. A query will retrieve data from the ViewPoint database that has the information needed to pre-populate fields in the Appenate mobile application subscription.
   1. This query will be executed and the resultant select set will be transformed into a .CSV file named **AppenateData.CSV**.
2. The .CSV file will be SFTP’d to the McKinstry SFTP server, using the following credentials:
   1. Host: SFTP.McKinstry.Com
   2. Port: 22
   3. Username: AppenateSFTP
   4. Password: In PasswordState
3. The application overwrites the previous file with the new file during each execution.
   1. Thus, only one file will reside on the SFTP server at any time.
4. AppenateData application will be launched automatically via a Windows Task Scheduler scheduled task.
   1. The application will run once per hour, 7x24.
   2. A suitable Service Account will be chosen for the application to run under.
5. Simple logging will be implemented.
   1. Each run of the application will append to a log file the run date and time, success or failure, and any additional failure messages thrown by any exceptions.
6. Emailing will be incorporated into the application. However, since the application is set to run every hour, it makes no sense to email success emails (one per hour, 24 per day, 7 days each week).
   1. Emailing will be implemented in a fashion to notify the author (PAK) to any run failures.
   2. The new McKinstry application emailing methodology will be incorporated versus anonymous SMTP or any other method. This uses an email account setup specifically for this purpose and is expected to be the mandated method sometime in 2020. The new method has been deemed more secure than other methods. The author has another application deployed to production using this email methodology and it has been running without error for two months at the time of this writing. Thus, it makes sense to use it for this application as well, even though the mandate has not yet been made.

AppenateData application

1. AppenateData is a C# console application.
   1. The application was created using Visual Studio 2019 and .Net framework, 4.7.2 was targeted.
2. The, Renci.SshNet NuGet package is used for the SFTP file moving.
3. A number of settings for the application are located in the app.config file.
   1. This allows for any changes needed to these settings to be made without a recompile and redeployment of the application.
4. Email capabilities are used by implementing the account that McKinstry I.T. as created for the purpose of sending emails from applications.

Additional Comments

As with many things, there is more than one way to accomplish this task (4611). A C# console application was chosen. The author believes it has advantages over other optional methods. For example, a PowerShell script could be created, however the readability and maintainability of the C# application is preferred. It is true that a PowerShell script could accomplish the SFTP’ing requirements via COM Interop or some other method. However, within a C# application the SFTP NuGet package being used seamlessly integrates with the C# console application by simply including it into the project. This NuGet package (Renci.SshNet) is written in C#, the same language as the parent application. In other words, most likely, any SFTP library used by a PowerShell script is ultimately a C# library.

A SSIS package is another option to accomplish the task. However there are greater resource and licensing costs required for the SSIS option compared with the zero cost of the console application. There is also the licensing costs of any 3rd party SSIS controls. If, they were to be used by a SSIS package to accomplish tasks such as creating the .CSV file, and the SFTP’ing of the .CSV file. The readability of a well written C# program far exceeds the understandability of a SSIS package; is the held belief of the author of this document.

Outstanding Issues

1. The selection or creation of a suitable Service Account for the application to be launched via the scheduled task.
2. The selection of an acceptable server to deploy the application on. No special requirements, other than the application via the server will need to be able to access the Viewpoint database in order to execute the query for the data. The application is not expected to need much in terms of horsepower, disk space, or any other server requirements other than very rudimentary server requirements. The server will need to have .Net framework 4.7.2 installed on it (which it may likely have by default if it is running a current server OS.