Hi,

The attached zip contains a c# solution to build 3 .Net Core 3.1 applications:

* MessageControllerWebApi
* MessageListenerClient
* MessagePublisherClient

Please run them in the above order.

Messages are read by the publisher app from the excel files arriving in the IncomingRequests folder. I have created a backup sub folder (.\IncomingRequests\FileSource) from where to source other files. The messages once read are then delivered to the Azure storage queue via MessageControllerWebApi.

Note - I have also created a test method (shouldn’t really be a test method) to bulk publish random messages – TestPublishRandomMessages()

The listener process (subscriber) reads the messages from the Azure storage queue also via MessageControllerWebApi.

The rest service, MessageControllerWebApi, can be installed as a Windows service. It has two functions – to publish messages and dequeue the last message.

The queue name is configurable in appsettings.json. I used the name **testqueuecox1.**

If the 3 apps are run in the above order the publisher should instantly publish 10 messages which are then read by the listener. All 3 apps should write to the console. Until further messages are published all three apps should then idle.

In response to Question 5 in the Readme doc you provided, MessageControllerWebApi uses dependency injection for interface IMessageQueueRepo. See Startup.cs:

**services.AddSingleton<IMessageQueueRepo, AzureStorageQueueRepo>();**

As long other queuing providers share a similar spec it could be as easy as creating a new concrete class, similar to that of AzureStorageQueueRepo, using the same interface and amending this line.

Hope this makes sense. Enjoy!

Best Rgds,

Andrew Cox