Locally hosted Azure queue consumer using singleton task schedulers for D365 CRM

This framework was created to work with D365 CRM outbound interfaces. Creates and updates for several different CRM entities were piped out to an Azure service bus queue. The queue consumer pulls down the queue message and determines which type of concrete class to build and creates a new task out of the run method of that class. It was necessary that some messages are processed in order while others can be processed in parallel. This framework decouples the message processing from the message reception.

Azure Service Bus Queue

CRM Server

Queue Consumer

ServiceClass

1. The service class will call a factory class to dynamically build the appropriate concrete class based on the queue message and call the run() of that class

<<Interface>>
CommonInterface

run()

FactoryClass

2. The concrete class knows whether or not it is performing a function for which the order is important. When order must be preserved, it throws the function into a task scheduler that will process sequentially. And if order does matter then it will throw it into one where tasks can be performed in parallel.

ConcreteClass

run()

3. After the function is placed in the task scheduler, the queue consumer is free to grab the next message from the Azure queue without having to wait for lengthy process of interacting with the interfacing applications through their respective SDKs.

The consumer is now free to get the next message from the Azure queue

Singletons in memory

Parallel Task Scheduler

4. The individual tasks make web service calls/ SDKs to exchange information

Sequential Task Scheduler

Interfacing Applications

MZP, Epic, Calyx, etc