Installation Guide

Pre-requisite

- Powershell version 6 and above

Install Window Service

1. Install MessageService_v1.0.0.exe
2. Open installation directory and execute below powershell script to install the window service.
    - install.ps1
3. User can install multiple window service with different configuration file.
    1. Edit install.ps1
        - Update ServiceName, DisplayName with new names
        - Update Config file path
    2. Execute the powershell

Uninstall

1. Open installation directory and execute below powershell script to install the window service.
    - uninstall.ps1

Message Services

Multithread Window service help to connect different services like IBM MQ, RabbitMQ, AzureQueue and Kafka.

Configuration

IBM MQ

Configure IBM MQ to connect different incoming and outgoing queues. ### Common Properties * **Enabled**: true|false. It will Enabled/Disabled the service * **ProcessingTimeInMilliseconds**: It will process incoming and outgoing message by given processing time.

IncomingQueues:

Configure single or multiple Incoming Queues by giving unique key name. * **QueueName**: Name of defined MQ queue * **QueueManagerName**: Name of MQ Manager * **FileDirectory**: Output directory path where message will get download * **FileExtension**: Create the given file extension * **Enabled**: Enabled/Disabled the respective configuration

```
"IncomingQueues": {
          "IncomingQueue1": {
            "QueueName": "QUEUE.1",
            "QueueManagerName": "QM1",
            "FileDirectory": "\\source\\outputpath",
            "FileExtension": ".xml",
            "Enabled": "true"
          },
          "IncomingQueue2": {
            "QueueName": "QUEUE.2",
            "QueueManagerName": "QM1",
            "FileDirectory": "\\source\\outputpath1",
            "FileExtension": ".xml",
            "Enabled": "false"
          }
        }
```

OutgoingQueues:

Configure single or multiple Outgoing Queues by giving unique key name. * **QueueName**: Name of defined MQ queue * **QueueManagerName**: Name of MQ Manager * **FileDirectory**: Message get read and upload from the respective MQ Queue * **FileExtension**: Read only the given file extension * **CompletionDirectory**: Message get move to respective directory after processing * **Enabled**: Enabled/Disabled the respective configuration

```
"OutgoingQueues": {
         "OutgoingQueue1": {
           "QueueName": "QUEUE.1",
           "QueueManagerName": "QM1",
           "FileDirectory": "\\source\\incomingfile",
           "FileExtension": ".xml",
           "CompletionDirectory": "\\source\\processed",
           "Enabled": "true"
         },
         "OutgoingQueue2": {
           "QueueName": "QUEUE.2",
           "QueueManagerName": "QM1",
           "FileDirectory": "\\source\\incomingfile1",
           "FileExtension": ".xml",
           "CompletionDirectory": "\\source\\processed1",
           "Enabled": "false"
         }
       }
```

QueueManagers:

Configure single or multiple IBM MQ QueueManager configuration * **SET_MQCONN_PROPERTIES**: If set to false all the MQ configuration will get ignore and only the Environment Variable configuration will get used * **MQSERVER**: Configure MQ connection details * channel_name/protocol/servername(port) * **MQCNO_RECONNECT**: You can make your client reconnect automatically to a queue manager during an unexpected connection break. * **MESSAGE_TIMOUT_IN_MS**: Message read timeout if message not found * **USE_ENCODING_LEADING_BYTES**: remove message encoding leading bytes * **MESSAGE_ENCODING**: Message Encoding * **USE_MQCSP_AUTHENTICATION_PROPERTY**: Set to true if user has explicit username and password * **USER_ID_PROPERTY**: User Id for authentication. It only use if USE_MQCSP_AUTHENTICATION_PROPERTY is true * **PASSWORD_PROPERTY**: Password for authentication. It only use if USE_MQCSP_AUTHENTICATION_PROPERTY is true * **SET_MQCONN_SSL**: Set to true if MQ connect with SSL * **MQSSLKEYR**: MQSSLKEYR specifies the location of the key repository that holds the digital certificate belonging to the user, in stem format.

```
*USER: IBM MQ.NET accesses the current user's certificate store to retrieve the client certi
*SYSTEM": IBM MQ.NET accesses the local computer account to retrieve the certificates.
```

- **MQSSLPEERNAME**: The SSLPEERNAME attribute is used to check the Distinguished Name (DN) of the certificate from the peer queue manager.
- **MQCERTLABEL**: This attribute specifies the certificate label of the channel definition.

- **MQSSLCIPHERSPEC**: The CipherSpec settings for an application are used during the handshake with the MQ Manager server.

`If the CipherSpec value supplied by the application is not a CipherSpec known to IBM MQ, th`

- **MQSSLRESET**: It represents the number of unencrypted bytes sent and received on a TLS channel before the secret key is renegotiated. It is optional
- **MQSSLCERTREVOCATIONCHECK**: The SSLStream class supports certificate revocation checking. It is optional

```
"QueueManagers": {
          "QM1": {
            "SET_MQCONN_PROPERTIES": "true",
            "MQSERVER": "DEV.ADMIN.SVRCONN/TCP/localhost(1414)",
            "MQCNO_RECONNECT": "false",
            "MESSAGE_TIMOUT_IN_MS": "1000", // read timeout if message not found
            "USE_ENCODING_LEADING_BYTES": "true",
            "MESSAGE_ENCODING": "UTF16",
            "USE_MQCSP_AUTHENTICATION_PROPERTY": "false",
            "USER_ID_PROPERTY": "admin", // it only use if USE_MQCSP_AUTHENTICATION_PROPERTY
            "PASSWORD_PROPERTY": "passw0rd",
            "SET_MQCONN_SSL": "false",
            "MQSSLKEYR": "*USER",
            //*USER: IBM  MQ.NET accesses the current user's certificate store to retrieve t
            //*SYSTEM": IBM MQ.NET accesses the local computer account to retrieve the certi
            "MQSSLPEERNAME": "PEERNAME",
            "MQCERTLABEL": "ibmwebspheremqlogonuserID",
            "MQSSLCIPHERSPEC": "TLS_RSA_WITH_AES_128_CBC_SHA",
            "MQSSLRESET": "500000",
            "MQSSLCERTREVOCATIONCHECK": "false"
          }
        }
```

RabbitMQ

Configure Rabbit MQ to connect different consumer and publisher. ### Common Properties * **Enabled**: true|false. It will Enabled/Disabled the service ### Consumers: Configure single or multiple Incoming Queues by giving unique key name. * **QueueManagerName**: Name of RabbitMQ Manager which has configuration for respective RabbitMQ server configuration * **QueueName**: Name of the RabbitMQ Queue * **ExchangeName**: Exchange name of Rabbit MQ * **RoutingKey**: Name of route which consumer used to bind with Rabbit MQ server * **ConsumerTag**: Name of consumer tag used by client libraries to determine what handler to invoke for a given delivery * **AutoAcknowledgment**: if set to true than it will acknowledge automatically. Default is false * **FileDirectory**: Output directory path where message will get

download * **FileExtension**: Create the given file extension * **Enabled**: Enabled/Disabled the respective configuration

```
"Consumers": {
          "Consumer1": {
            "QueueManager": "QueueManager1",
            "QueueName": "queue1",
            "ExchangeName": "amq.fanout",
            "RoutingKey": "",
            "ConsumerTag": "queue1-consumer",
            "AutoAcknowledgment": "false",
            "FileDirectory": "\\source\\output\\rabbitmq",
            "FileExtension": ".xml",
            "Enabled": "true"
          }
        }
```

Publishers:

Configure single or multiple publisher Queues by giving unique key name. * **QueueManagerName**: Name of RabbitMQ Manager which has configuration for respective RabbitMQ server configuration * **QueueName**: Name of the RabbitMQ Queue * **ExchangeName**: Exchange name of Rabbit MQ * **RoutingKey**: Name of route which consumer used to bind with Rabbit MQ server * **ConfirmPublish**: if set to true than it will enable publisher acknowledgement automatically. Default is false * **FileDirectory**: Message get read and upload from the respective Queue * **FileExtension**: Read only the given file extension * **CompletionDirectory**: Message get move to respective directory after processing * **Enabled**: Enabled/Disabled the respective configuration

```
"Publisher1": {
            "QueueManager": "QueueManager1",
            "QueueName": "queue1",
            "ExchangeName": "amq.fanout",
            "RoutingKey": "",
            "Mandatory": "false",
            "ConfirmPublish": "false",
            "FileDirectory": "\\source\\incoming\\rabbitmq",
            "CompletionDirectory": "\\source\\completion\\rabbitmq",
            "FileExtension": ".txt",
            "Enabled": "true"
          }
```

QueueManagers

Configure single or multiple RabbitMQ queue manager by giving unique key name. * **ClientProvidedName**: It is simply a human friendly name for a

connection. * **HostName**: Connect to a RabbitMQ node using a hostname (rabbit mq server name) * **VirtualHostName**: RabbitMQ virtual hosts are logical groups of entities name * **username**: RabbitMQ Username * **Password**: RabbitMQ password * **AutomaticRecoveryEnabled**: If set to true it will enable automatic recovery mode, Default true * **TopologyRecoveryEnabled**: If set to true it will enable topological recovery mode Default true * **SslEnabled**: If set true than it will enable to configuration for SSL * **ServerName**:SSL Server name * **CertPath**: SSL Certification path * **CertPassphrase**: SSL Certificate pass phrase

```
"QueueManager1": {
            "ClientProvidedName": "Pro1",
            "HostName": "localhost",
            "VirtualHostName": "/",
            "Username": "guest",
          *  "Password": "guest",
            "AutomaticRecoveryEnabled": true,
            "TopologyRecoveryEnabled": true,
            "SslEnabled": false,
            "ServerName": "",
            "CertPath": "",
            "CertPassphrase": ""
          }
```

Kafka

Configure Kafka to connect different consumer and publisher. ### Common Properties * **Enabled**: true|false. It will Enabled/Disabled the service ### Consumers Configure single or multiple Consumers by giving unique key name. * **GroupId**: Unique Kafka Group Id * **Topics**: Comma separated names of topic * **KafkaManager**: Kafka manager name * **EnableAutoCommit**: Commit each develivered message automatically * **AutoCommitIntervalMs**: Milliseconds in which Kafka check to commit message automatically * **AutoOffsetReset**: * Earliest: It automatically reset the offset to the earliest offset * Latest: It automatically reset the offset to the latest offset * None: Throw exception to the consumer if no previous offset is found for the consumer's group * **ManualCommit**: If true it will commit the message manually * **FileDirectory**: Output directory path where message will get download * **FileExtension**: Create the given file extension * **Enabled**: Enabled/Disabled the respective configuration

```
"Consumer1": {
            "GroupId": "mygroup",
            "Topics": "MyTopic1",
            "KafkaManager": "KafkaManagerPlaintext",
            "EnableAutoCommit": "false",
            "AutoCommitIntervalMs": "600000",
            "EnableAutoOffsetStore": "true",
```

```
            "AutoOffsetReset": "Earliest",
            "ManualCommit": "false",
            "FileDirectory": "C:\\source\\data\\incomingfile",
            "FileExtension": ".xml",
            "Enabled": "true"
        }
```

Publishers

- **KafkaManager**: Kafka Manager name
- **Topics**: Comma separated names of topic
- **Acks**:
  - All or 1: This will mean the leader will write the record to its local log but will respond without awaiting full acknowledgement from all followers.
  - 0: In Fire and Forget Scenario kafka are not wait for any response and there is no any retries.
- MessageSendMaxRetries:How many times to retry sending a failing Message.

`Note: retrying may cause reordering unless enable.idempotence is set to true.`

- **RetryBackoffMs**: The backoff time in milliseconds before retrying a protocol request.
- EnableIdempotence: When set to true, the producer will ensure that messages are successfully produced exactly once and in the original produce order. The following configuration properties are adjusted automatically (if not modified by the user) when idempotence is enabled: max.in.flight.requests.per.connection=5 (must be less than or equal to 5), retries=INT32_MAX (must be greater than 0), acks=all, queuing.strategy=fifo. Producer instantation will fail if user-supplied configuration is incompatible.default: false importance: high
- **FileDirectory**: Message get read and upload from the respective Queue
- **FileExtension**: Read only the given file extension
- **CompletionDirectory**: Message get move to respective directory after processing Configure single or multiple publishers by giving unique key name.

```
"Publisher1": {
            "KafkaManager": "KafkaManagerPlaintext",
            "Topics": "MyTopic1",
            "Acks": "All",
            "MessageSendMaxRetries": "4",
            "RetryBackoffMs": "60000",
            "EnableIdempotence": "true",
            "FileDirectory": "\\source\\data\\outgoing",
            "FileExtension": ".xml",
```

```
            "CompletionDirectory": "\\source\\data\\processed",
            "Enabled": "true"
          }
```

KafkaManagers

Configure single or multiple Kafka manager by giving unique key name. * **BootstrapServers**: Kafka server name * **SecurityProtocol**: PlainText|Ssl * **SslCaLocation**: Security certificate authority location * **SslCertificateLocation**: Security certificate path location * **SslKeyLocation**: Security key path location

```
"KafkaManagers": {
          "KafkaManagerPlaintext": {
            "BootstrapServers": "localhost:9092",
            "SecurityProtocol": "PlainText" //PlainText, Ssl
          },
          "KafkaManagerSsl": {
            "BootstrapServers": "localhost:19093",
            "SecurityProtocol": "Ssl", //PlainText, Ssl
            "SslCaLocation": "\\ssl\\secrets\\root-ca.crt",
            "SslCertificateLocation": "\\ssl\\secrets\\kafka_client.crt",
            "SslKeyLocation": "\\ssl\\secrets\\kafka_client.key"
          }
        }
```

AzureQueue

Configure single or multiple Azure queue by giving unique key name. ### Common Properties * **Enabled**: true|false. It will Enabled/Disabled the service * **ProcessingTimeInMilliseconds**: It will process incoming and outgoing message by given processing time.

IncomingQueues

- **QueueName**: Queue Name
- **QueueManager**: Azure Queue Manager Name
- **FileDirectory**: Output directory path where message will get download
- **FileExtension**: Create the given file extension
- **Enabled**: Enable or Disabled the service

```
"IncomingQueues": {
          "Queue1": {
            "QueueName": "immqiso",
            "QueueManager": "QueueManager1",
            "FileDirectory": "C:\\source\\data\\incomingfile",
            "FileExtension": ".xml",
            "Enabled": "true"
```

```
        }
      }
```

OutgoingQueues

- **QueueName**: Queue Name
- **QueueManager**: Azure Queue Manager Name
- **FileDirectory**: Message get read and upload from the respective Queue
- **FileExtension**: Read only the given file extension
- **CompletionDirectory**: Message get move to respective directory after processing
- **Enabled**: Enable or Disabled the service

```
"IncomingQueues": {
        "Queue1": {
          "QueueName": "immqiso",
          "QueueManager": "QueueManager1",
          "FileDirectory": "C:\\source\\data\\incomingfile",
          "FileExtension": ".xml",
          "Enabled": "true"
        }
      }
```

QueueManagers

- **ConnectionString**: Azure Queue connection string
- **RetryPolicy**: Azure Queue Manager Name
  - DelayInSeconds: The delay between retry attempts for a fixed approach or the delay on which to base calculations for a backoff-based approach.
  - MaxDelayInSeconds: The maximum permissible delay between retry attempts when the service does not provide a Retry-After response header
  - MaxRetries: The maximum number of retry attempts before giving up.
  - Mode: The approach to use for calculating retry delays Fixed | Exponential.
  - NetworkTimeoutInSeconds: The timeout applied to an individual network operations.

```
"QueueManagers": {
        "QueueManager1": {
          "ConnectionString": "ConnectionString",
          "RetryPolicy": {
            "DelayInSeconds": 60000,
            "MaxDelayInSeconds": 1200000,
            "MaxRetries": 4,
            "Mode": "Exponential", //Exponential or Fixed
```

```
      "NetworkTimeoutInSeconds": 2000000
    }
  }
}
```