

**Qualcomm Car-to-Cloud Platform**

**Usage of queue interface**

**Version No.2.0**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** |
| **Name** |  |  |  |
| **Role** |  |  |  |
| **Signature** |  |  |  |
| **Date** |  |  |  |

Table of Contents

[1. MESSAGE QUEUE INTERFACE 2](#_Toc72162280)

[2. APPLICATION 2](#_Toc72162281)

[2.1. c2c\_base\_queue\_intf 3](#_Toc72162282)

[2.2. c2c\_base\_queue\_sqs\_impl 5](#_Toc72162283)

May 20, 2021

Cognizant

# MESSAGE QUEUE INTERFACE

An interface contains variables and signatures of function. These signatures can be implemented by other classes that implements the interface. This functionality improves the flexibility and stability of programs and applications.

A significant way to utilize this functionality is through separating the interface and its implementation by converting the interface as jar and adding it to its implementation as a dependency. This separation of interface enables the possibility of having multiple implementations for the same interface.

**FEATURES**

* An interface project that outlines the signatures of basic functionalities
* An implementation project that implements the signatures in the interface project
* A base class whose main purpose is to call the implemented methods and determine the flow of the application
* An ability to change the implementation by changing the jar file in the application without changing or affecting the flow of the base program.

The idea is to use this concept to provide a common Cloud Agnostic Method.

# APPLICATION

The message queue application is meant to publish, receive, listen and delete messages with respect to a queue via a cloud services. Different cloud services provide this functionality, for example AWS SQS or Azure Service Bus. For this application AWS SQS has been chosen.

The project can be divided into two projects –

* **c2c\_base\_queue\_intf** – Message Queue Interface provides the following functions :

Publish – To publish messages to the queue.

Receive– To receive messages from the queue.

Listen– To listen to the messages from the queue.

Delete– To delete messages from the queue.

Enable Listener– To enable the listener to start receiving messages from the queue.

Disable Listener– To stop the listener from receiving any message from the queue.

Reconnect– To reconnect the client for connection establishment.

This is converted as a JAR file.

* **c2c\_base\_queue\_sqs\_impl** – This project provides AWS specific implementation for the queue interface methods. AWS SQS provides the message queuing service. It uses the interface’s JAR file as a dependency.

## c2c\_base\_queue\_intf

* ***IC2CServiceQueueClient*** an interface which contains:
* *public void publish(C2CQueueMessage c2cQueueMessage) throws QueueApplicationException, InvalidQueueException* –

The publish method accepts the *c2cQueueMessage* which is serialized as aparameter to publish the messages to the queue.

The size of the queue is limited to 1MB to maintain cloud agnostic behavior.

It throws two custom exceptions - QueueApplicationException, InvalidQueueException.

* *public List<C2CQueueMessage> receive() throws QueueApplicationException, InvalidQueueException* –

The receive method is used for polling or retrieving messages of the form C2CQueueMessage from the queue in fixed intervals.

It throws two custom exceptions - QueueApplicationException, InvalidQueueException.

* *public void listen(MessageListener listener) throws JMSException–*

The listen method acts as listener to the queue where the messages are received asynchronously.

The consuming application should implement a JMS listener. The onMessage method of the MessageListener interface is called when messages are received. The messages received will be of JMS message type.

It throws a *JMSException*.

* *public void deleteMessage(String messageReceiptId) throws InvalidQueueException, QueueApplicationException*

The delete method can delete a message using the messageReceiptId as parameter.

It throws two custom exceptions - QueueApplicationException, InvalidQueueException.

* *public void enableListener() throws JMSException–*

The *enableListener* method enables the Listener to consume messages from the queue.

It throws a *JMSException.*

* *public void disableListener() throws JMSException–*

The *enableListener* method disables the Listener to consume messages from the queue.

It throws a *JMSException.*

* *public void reconnect() throws JMSException–*

The reconnect method reconnects the client for connection establishment.

It throws a *JMSException.*

It contains model classes:

* ***C2CQueueMessage*** – A model class that contains an object Message and String messageReceiptId with their corresponding getters and setters. messageReceiptId is set by the caller application once the message is received from the queue. This ensures that the particular message is deleted from the queue to avoid duplication. It also contains three constructors – one that accept no arguments, one that accepts a single argument - message and one that accepts two arguments - message and messageReceiptId.
* ***C2CQueueConnectionConfiguration*** – A model class that contains variables of possible parameters such as region, accessKey, secretKey, queue name, wait time, receive message count, enableRequestHandling and extendedMessageStorageUrl, which are needed for configuration in the builder class. It also includes its getters and setters. It also contains the Builder class with parameters such as region, accessKey, secretKey, queue name, wait time, receive message count, enableRequestHandling and extendedMessageStorageUrl. The Builder class has a build method to create the client. It also includes the getters and setters.

*c2c\_base\_queue\_intf* also contains custom exceptions:

* ***QueueApplicationException*** – A custom exception which extends *RuntimeException* that is thrown for *IllegalArgumentException* and *SdkClientException*. *IllegalArgumentException* is thrown when the region name is incorrect or incomplete. *SdkClientException* is thrown when the region is invalid or blank.
* ***InvalidQueueException*** – A custom exception which extends *RuntimeException* thrown when the queue cannot be found or cannot be initialized using the details provided.
* ***QueueValidationException*** – A custom exception which extends *RuntimeException* thrown for errors related to input validation. This exception is thrown when message size is greater than 1 MB and also when the c2cQueueMessage object is not serializable.

## c2c\_base\_queue\_sqs\_impl

*c2c\_base\_queue\_intf* can be implemented to publish, receive, listen and delete messages in a queue of different cloud service. *c2c\_base\_queue\_sqs\_impl* is the AWS specific implementation of Message Broker Queue for c2c\_base\_queue\_intf’s interfaces. It contains the body for each of the interface’s function signatures.

* ***C2CServiceQueueClientImpl*** -
* Contains a constructor that is responsible for configuration with values provided by caller class of *C2CQueueConnectionConfiguration* type.
* Contains function SQSConnection to establish connection with AWS SQS
* Contains implementations for publish, receive, listen, delete, enable listener, disable listener and reconnect function signatures from interface project’s *C2CServiceQueueClient* class. In the receive function of the implementation project the message count and the wait time are configurable. **The message count cannot be more than 10**.

*c2c\_base\_queue\_intf* will be converted to a jar file and added to *c2c\_base\_queue\_sqs\_impl* as a dependency.