# Background

US Customs and Border Protection processes billions of messages each month and as such is one of the largest message users in the world. The OASIS Advanced Message Queuing Protocol (AMQP) Technical Committee (TC) advances a vendor-neutral and platform-agnostic protocol that offers organizations an easier, highly reliable, scalable, available, high performing and more secure approach to enterprise messaging.

The goal of this document is to provide real-world messaging requirements and use cases to OASIS on behalf of CBP to ensure that these uses cases are accounted for in the future advancement of AMQP protocol.

# Messaging Criteria

## Performance and Scalability

| **Serial ID** | **Criteria** | **Description** |
| --- | --- | --- |
|  | Bandwidth Throughput | AMQP server must have an available and very large queue. Queue size should be large enough to support the large message sizes without failing. The formula to calculate the maximum bandwidth supported is *Messages \* Size / Seconds*. |
|  | High Availability | Must achieve failover by sending messages to a broker cluster. If an individual broker in a cluster is brought down, the messages are still delivered by the other brokers in the cluster. |
|  | Latency | Message delivery latency must meet organization expectations. Calculate the average latency time it takes for a message to enter and leave a queue: (Message/Second) / Messages = Second (average response time per message). |
|  | Load Balancing | Load Balancing must be accomplish through a single queue's message load being balanced between multiple brokers. Message production would be targeted on a single broker while consumption would be done on multiple brokers. |
|  | Volume Throughput | Volume throughput must meet organization expectations. Calculate the maximum number of messages that can be processed: Messages / (Total Seconds in 24 Hours) = Messages/Second. |

## Security

| **Serial ID** | **Criteria** | **Description** |
| --- | --- | --- |
|  | Backup | Must be able to back up broker objects. The broker objects are backed up for recoverability. The backed up objects can be applied to another broker instance as if it existed before. |
|  | Authentication: Object | The AMQP server must be able to authenticate at an object level to allow access based on credentials. Only users with access to a queue can send and receive messages to that queue. |
|  | SSL Encryption | AMQP must support the ability for clients to use SSL Encryption when producing and consuming messages. |
|  | Authentication: Channel Exit | Must be able to authenticate credentials at the route level to allow access to the broker. |
|  | TLS Encryption with AES 256 Support | Must support the ability for clients to use TLS Encryption with AES 256 support when producing and consuming messages. |
|  | SSL integrity | Must support the ability for clients to use SSL Encryption when producing and consuming messages and the integrity of the SSL encrypted message is verified. |

## Messaging

| **Serial ID** | **Criteria** | **Description** |
| --- | --- | --- |
|  | Fanout Exchange | A message sent to a fanout exchange is broadcasted to all listening clients.  |
|  | Fire and Forget | The pattern presents asynchronous design by producing and consuming messages on a queue destination without waiting for a response. |
|  | Message Expiry | If Message Expiry is turned on, messages should expire after a certain amount of time.  |
|  | Persistent Messages | Messages on a queue persist through a broker failure or shutdown.  |
|  | Publish-Subscribe | Several queues and one topic exchange are created. The queues are subscribed to a topic. When a topic message is sent to the topic exchange, all queues subscribed to the topic should receive the message.  |
|  | Single-Phase Commit | Commit a message in a single-phase, regardless of any failures in the surrounding multi-resource distributed transaction.  |
|  | Two-Phase Commit | The AMQP server provides a two-phase commit .A transaction manager manages the transactions using a two-phase commit..  |
|  | Request Response | A client application will send a request to place a message onto a queue. Once successfully placed onto the queue. AMQP will send a response back to the client application that it has received the message.  |
|  | Priority Queuing | Messages with a higher priority are consumed before messages with a lower priority.  |
|  | Message Guarantee | The ability for the Broker to send acknowledgement ensuring that messages are sent and received. Messages with a higher priority are consumed before messages with a lower priority. |
|  | Message Redelivery | Must redeliver message automatically every time the consumer application rollback. Once the message delivery is committed by consumer i.e. message is consumed successfully, the redelivery is stopped. |
|  | Triggering | Ability to automatically trigger a transaction upon arrival of a message within a queue.  |
|  | Correlated Messaging and Indexing | Ability to correlate messages to some header attributes such that messages can be tied to a unit of work for continued processing. |
|  | Exits | Exits are the ability to take action in ad hoc fashion and at channel level on each message. |
|  | Grouping | Message grouping consists of messages are correlated and grouped by some header attribute. |
|  | Logging and Reporting | Ability to log and report into a specified location. |
|  | Message Segmentation | Message segmentation occurs when messages are too large for a queue and need be segmented into different messages to be processed by a consumer. |
|  | Message Expiry Report | The ability to generate a report when a message is expired.  |
|  | Content Based Routing | Routes message based on what is contained in the message. AMQP will examine a message and see if it meets the criteria based of a specific route. |

## Message Brokers

|  |  |  |
| --- | --- | --- |
| **ID** | **Criteria** | **Description** |
|  | CLI Management Tool | A CLI management tool is made available to view statuses and manipulate broker objects. The CLI tool can also be used for invocation of batch scripts and easy access when GUI is unavailable. |
|  | GUI Management Tool | GUI management tool is made available to view statuses and manipulate broker objects. |
|  | Object Filtering | The broker objects can be filtered for reporting and manipulation via the GUI or CLI management tools. |
|  | Channel Multi-Paths | Ability to define multiple routes between brokers using the same underlying transportation but isolating specific flows for prioritization.  |
|  | COTS Monitoring | Interoperate with a least one of the best breed monitoring solutions on the market. |
|  | Weighted Client Selection | A connected client application can automatically connect to an alternative broker if the primary broker is not available. |

## General Criteria

| **ID** | **Criteria** | **Description** |
| --- | --- | --- |
|  | Route Retry | Upon any kind of network failure with the broker is offline, existing routes in progress will continue to retry until network failure is resolved or until retry timeout is reached. |
|  | Dynamic Storage | Storage containing queue information can be easily moved from one location to another location without any loss of messages. |
|  | Dynamic Thread Spawning | An additional thread will be initiated upon reaching a specified depth threshold increment. |
|  | Auto Re-connect | The ability for the client application to reconnect automatically after the connection is loss. |
|  | Retrieved Message with wait period | A client application connects to a queue and waits a specified amount of time until the message is placed onto that queue so it can be consumed. If no messages are placed onto that queue within the specified amount of time, the client application disconnects from that queue.  |
|  | Log Recoverability | The functionality to recover queue messages when there has been hardware failure. |
|  | Dual Logging | The functionality for logging in order to minimize the likelihood of data loss problems in the log files. Currently this feature is supported in MQ in z/OS environment. |
|  | Attachment | Ability to support attachment. |
|  | Interoperability | Provide standardized way to interoperate with widely used messaging protocols like IBM MQ. |