Message Queue

Product Overview

Product Overview

What is Message Queue ?

Message Queue (MQ) is a professional message middleware as a core product in the enterprise-level Internet architecture. Based on the highly available distributed cluster technology, MQ provides a series of message cloud services, including message subscription and publishing, message tracing query, scheduled and delayed messages, resource statistics, monitoring, and alerts. With more than nine years of history, MQ provides asynchronous decoupling and load shifting for distributed application systems and supports features for Internet applications, including massive message accumulation, high throughput, and reliable retrying. It is one of the core products used to support the Double 11 Shopping Festival.

Multiple IDCs are deployed in a single region, ensuring extremely high availability. Even when one IDC completely fails, MQ can still provide a message distribution service for applications with no SPOF.

Currently, MQ supports access through TCP or HTTP. It also supports seven programming languages: Java, C++, .NET, Go, Python, Node.js, and PHP, to facilitate quick access to the cloud services of MQ for applications developed with different programming languages. You can either deploy your applications on Alibaba Cloud ECS instances or your own enterprise clouds or embed them into a mobile device or IoT device to connect with MQ for sending and receiving messages. Also, local developers can access MQ through the Internet to send and receive messages.

Features

MQ supports access through TCP or HTTP and multiple programming languages, and offers multidimensional management tools. It provides a series of featured functions for different application scenarios.

Feature Overview

Support for TCP, HTTP, and STOMP

- HTTP: MQ supports HTTP and RESTful, making it easy to use, fast to access, and powerful in

cross-network access. Additionally, it supports clients developed in seven programming languages.
- TCP: Different from the HTTP-based access mode of HTTP, the TCP-based SDK access mode is more specialized, reliable, and stable.
- STOMP: Similar to the text-based protocol HTTP. This protocol is used for lightweight interaction between STOMP clients using script languages (such as Ruby, Python, and Perl) and MQ brokers.

Management tools

- Web console: It supports topic management, producer management, consumer management, message query, message tracing, statistics, and monitoring.
- API: It allows you to integrate MQ management tools into your own console.
- mqadmin command set: A rich set of management commands for private clouds is provided for you to manage MQ by command.

Feature highlights

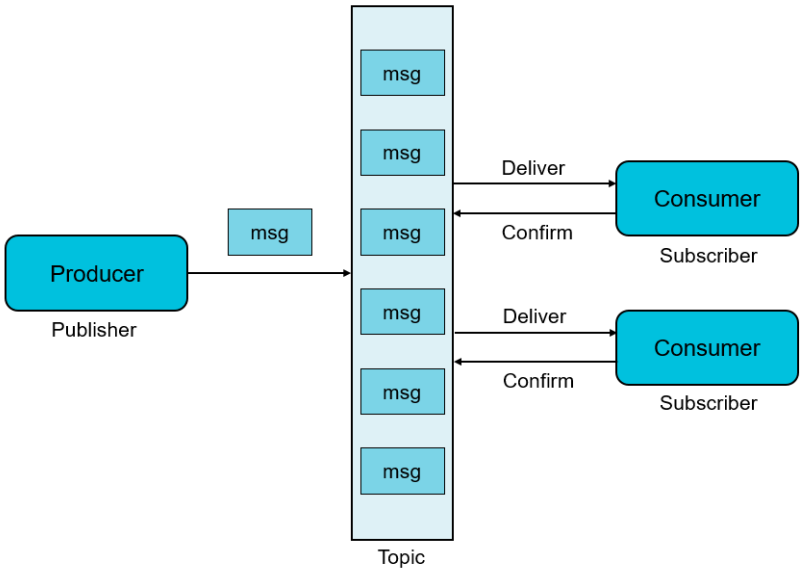
- Transactional messages: MQ provides a distributed transaction processing function similar to X/Open XA to ensure transaction consistency.
- Scheduled and delayed messages: MQ allows producers to specify the length of time to wait before a scheduled or delayed message is delivered. The maximum length is 40 days.
- Large messages: A maximum message size of 4 MB is supported.
- Message tracing: This records the complete route of a message from its publishing by the producer to the MQ broker and then to the consumer. This function facilitates troubleshooting.
- Broadcasting consumption: Consumers identified by the same group ID can consume a certain message once.
- Ordered messages: Message consumers can consume messages in the order in which messages are delivered.
- Reset consumer offset: Users can reset the consumer progress by time to trace back messages or discard accumulated messages.
- Dead-letter queues: Messages that cannot be consumed are stored to a special dead-letter queue for subsequent processing.
- Message routers: These are used to synchronize and copy messages between instances in different regions all over the world, ensuring data consistency between regions.

Apsara Stack deployment

- Customization: Technical solutions, and onsite technical support and training are provided.
- Flexible deployment: MQ can be deployed in Apsara Stack separately or in a hybrid cloud.
- O&M management: Apsara Stack supports such O&M tools as mqadmin command set and APIs to facilitate the management of platform integration and unified O&M.

Message sending and receiving model

MQ supports the publish/subscribe model. A message publisher (producer) can send a message to a topic of the broker and multiple message receivers (consumers) can subscribe to this topic to receive the message. The model is illustrated in the following figure.



For more information about MQ concepts, see Terms.

Terms

This topic introduces the MQ related terms to help you better understand and use MQ.

Topic

A topic is used to classify messages. It is the first-level classification. For more information, see Topic and tag best practices.

Message

A message is a carrier for transferring information in MQ.

Message ID

A message ID is a global unique identifier for a message, which is automatically generated by the MQ system.

Message key

A message key is a unique identifier of the message’ s service logic, which is set by the message producer.

Tag

A tag is used to further classify the messages under a topic. It is the second-level classification. For more information, see Topics and tags.

Producer

A producer, also known as a message publisher, produces and sends messages.

Producer instance

A producer instance is an object instance of a producer. Different producer instances can run in different processes or on different machines. The producer instance thread is safe and can be shared between multiple threads in the same process.

Consumer

A consumer, also known as a message subscriber, receives and consumes messages.

Consumer instance

A consumer instance is an object instance of a consumer. Different consumer instances can run in different processes or on different machines. Thread pool consumption information is configured in a consumer instance.

Group

A group is a type of producers or consumers that produce or consume messages of the same type and publish or subscribe to messages based on the same logic.

Group ID

The ID of the group.

Queue

Each topic has one or more queues to store messages. The number of queues for each topic varies with the region of the instance. For more information about the exact number of queues, submit a ticket.

Note: The standard edition instances do not support queue number change, while the platinum edition instances do.

Exactly-Once delivery semantics

The Exactly-Once delivery semantics is used to specify that messages sent to the message system can be processed by consumers only once. In other words, even if the message producer sends a message to the message system again, the message can still only be consumed by a consumer once. For more information, see Exactly-Once delivery semantics.

Clustering consumption

All consumers identified by the same group ID consume messages in an even manner. For example, a topic contains nine messages and a group contains three consumer instances. In this case, each instance consumes three messages. For more information, see Clustering and broadcasting consumption.

Broadcasting consumption

Each of the consumers identified by the same group ID consumer all messages once. For example, a topic contains nine messages and a group contains three consumer instances. In this case, each instance consumes nine messages. For more information, see Clustering and broadcasting consumption.

Scheduled message

A producer sends a message to the MQ broker, expecting the message to be delivered to a consumer at a specified time in the future. The message is a scheduled message. For more information, see Scheduled and delayed messages.

Delayed message

A producer sends a message to the MQ broker, expecting the message to be delivered to a consumer after a specified period of time. The message is a delayed message. For more information, see Scheduled and delayed messages.

Transactional message

Transactional messages provide a distributed transaction processing function similar to X/Open XA to ensure transaction consistency. For more information, see Transactional messages.

Ordered message

An ordered message is a message that is published and consumed in order. Ordered messages in MQ are classified into globally ordered messages and partitionally ordered messages. For more information, see Ordered messages.

Globally ordered message

All messages under a specified topic are published and consumed in the strict first-in-first-out (FIFO) order. For more information, see Ordered messages.

Partitionally ordered message

All messages under a specified topic are partitioned by the sharding key. Messages in one shard are published and consumed strictly in the FIFO order. A sharding key is a key field used in ordered messages to distinguish different partitions. It is completely different from the key used in normal messages. For more information, see Ordered messages.

Message accumulation

A producer has sent messages to the MQ broker but a consumer cannot consume all the messages in

a short period of time due to limited consumption capability. Therefore, unconsumed messages are stored in the broker. This process is called message accumulation.

Message filtering

Consumers can filter messages by tag to receive only the message type they want. Message filtering is completed on the MQ broker. For more information, see Message filtering.

Message trace

A message trace is a complete route record of a message from its publication by a producer, to consumption by a consumer. It consists of the time, location, and other information on each node. Message traces clearly display the complete route from message delivery by message producers, to MQ broker, to message consumption by consumers. It facilitates troubleshooting. For more information, see Message tracing.

Reset consumption offset

With the timeline as coordinates, reset the consumer progress of a topic subscribed by a message subscriber over the time span specified for persistent message storage (three days by default). After the consumption offset is reset, the subscriber can receive a message sent by the message producer to the MQ broker after a set time point. For more information, see Reset consumer offset.

Dead-letter queue

Dead-letter queues are used to process messages that cannot be consumed. When a message fails to be consumed for the first time, MQ automatically retries the consumption of the message. If the message still cannot be consumed after the maximum number of retries is reached, the message cannot be properly consumed. Instead of immediately discarding the message, MQ sends it to a particular queue of the corresponding consumer.

In MQ, a message that cannot be properly consumed is called a dead-letter message, which is stored by a particular queue named dead-letter queue.

For more information, see Dead-letter queue.

Message router

Message routers are often used to synchronize messages between regions to ensure data consistency between regions. Relying on Express Connect, developed by Alibaba Cloud based on its superb infrastructure, MQ message routers allows you to efficiently synchronize messages from countries to countries and regions to regions. For more information, see Global message routers.

Limits

MQ constrains and regulates certain metrics. To avoid program exceptions when using MQ, do not exceed the maximum limits. For more information about the maximum limits of relevant items, see the following table.

Item	Limit	Description
Maximum number of instances in a single region	**Standard Edition:** 8 **Enterprise Platinum Edition:** not limited	None
Topic name length	64 characters	If the length of the name of a topic exceeds this value, consumers will fail to send or subscribe to messages under this topic.
Message size	**A normal or ordered message:** 4 MB **A transactional, scheduled, or delayed message:** 64 KB	If the size of a message exceeds this value, the message will be discarded.
Message retention period	3 days	Messages are retained for at most three days, and the system automatically deletes them after three days.
Reset consumption offset	3 days	Any message consumed within 3 days can be reset.
TPS for sending and receiving messages under a single topic	**Standard Edition:** 5000 messages/s **Enterprise Platinum Edition:** See the purchased specifications.	None
Delay of scheduled or delayed messages	40 days	The msg.setStartDeliverTime parameter can be set to any time (unit: milliseconds) within 40 days. The message cannot be scheduled to be sent after 40 days.