Message Queues

By: **IBM Cloud Education**

8 August 2019

Integration

What is a message queue?

Message Queues

A message queue is a component of messaging middleware that makes it easier to develop resilient connections between applications.

What is a message queue?

A message queue is a component of messaging middleware solutions that enables independent applications and services to exchange information. Message queues store “messages”—packets of data that applications create for other applications to consume—in the order they are transmitted until the consuming application can process them. This enables messages to wait safely until the receiving application is ready, so if there is a problem with the network or receiving application, the messages in the message queue are not lost.

This model, known as asynchronous messaging (15:11), prevents data loss and enables systems to continue to function if processes or connections fail. This allows developers to keep processes and applications separate, keeping their communications self-contained and event-driven to make the architecture more reliable.

Message queues are available in messaging solutions across numerous deployment options, including optimized physical appliances, cloud services, mainframes, and as software.

Benefits

Message queuing solutions are widely used across industries and can offer an array of benefits to developers and system administrators alike, including the following:

- **Reliable message delivery:** Using a message queue can ensure that business-critical messages between applications will not be lost and that they will be only be delivered to the recipient once. With this feature in place, additional de-

duplication or loss-prevention logic is unnecessary.

- **Inter-application connectivity:** Some message queue solutions can handle message encryption, transactionality, and other communication aspects between applications and services. This simplifies application development and enables disparate architectures to work together.

- **Versatility:** Message queue solutions can support multiple languages, such as Java, Node.js, COBOL, C/C++, Go, .NET, Python, Ruby, and C#. They can also support numerous application programing interfaces (APIs) and protocols, including MQTT, AMQP, and REST, as well as others.

- **Resilience:** Asynchronous messaging ensures application-specific faults won't impact the system. If one component in the system stalls, all others can continue interacting with the queue and processing messages. This decreases the chance that the entire system's stability will be impacted by one part's failure.

- **Improved security:** A message queue may be able to identify and authenticate all messages, and in some message queue solutions, they can be set to encrypt messages at rest, in transit, or end-to-end. This can contribute to the overall security of the applications and/or infrastructure.

- **Integrated file transfers:** Some message queue solutions include additional features, such as the ability to transfer files. This can be used as an alternative to FTP within enterprises where such solutions are in use.

Use cases

Today's enterprise computing environments are complex and highly decentralized. Messaging makes it easier to integrate applications and services on diverse platforms by providing a single, robust, and secure shared messaging backbone. This protects against data loss and ensures systems will continue to function even with

unstable connectivity.

Message queues are uniquely suited for integrating on-premises backend systems with cloud services. In cloud architectures, applications are often broken down into small, independent components. This makes it easier to design and code them, and also easier to manage their performance. Message queues enable these decoupled cloud-based applications to communicate with each other or with on-premises systems.

Message queuing increases architecture resilience because the messages can have persistence. This means they're stored to disk until the service receiving the message confirms processing. Messaging queues can be used in scenarios requiring high levels of security, fault tolerance, and accuracy, such as financial transaction processing, air-travel booking, or updating healthcare patient records.

Message queues can also be used to enable applications and systems residing in different clouds (whether public or private) to communicate, even if they are located in different countries or even on remote continents. Using a message queue increases fault tolerance and can be used to prevent data being duplicated or lost across geographically and technically disparate systems. Because each service within the system is decoupled, or logically separated from the others, each can continue to function if other services or applications fail or stall.

Message queues work across disparate applications such as mobile, IoT, and traditional transaction system records. They also support various platforms—such as virtual machines and containers—and can enable integration between legacy applications and today's latest solutions.

Message queue vs. ...

Message queue vs. pub/sub

Message queues use a point-to-point messaging pattern, in which one application (called the sender) submits a message to the queue and another application (called

the receiver) gets the message from the queue and consumes it. There should be a tightly coupled one-to-one relationship between the sender and consumer, and each message should be consumed only once.

If your applications require messages to be distributed to multiple parties, either multiple message queues can be combined or a publish/subscribe (pub/sub) messaging model can be used.

In pub/sub messaging, the application producing the message is called a publisher and applications consuming it are the subscribers. Each message is published to a topic, and every application that subscribes to that topic gets a copy of all messages published to it.

Most messaging middleware solutions support both the message queue (point-to-point) and pub/sub messaging models.

Message queue vs. message bus

A message bus—a type of enterprise service bus, or ESB—allows services ubiquitous access to data while ensuring they remain decoupled and independently functional within a distributed system architecture. When you employ a message bus, all the services or applications must share common data types, a common command set, and common communication protocols (although, they may be written in different languages). Consumers can determine how they use messages.

If decoupled applications are to communicate via a message bus, the messages must be transformed so that they are all of the same type. In contrast, message queues transport messages, no matter whether they are of the same or different types.

Message queue vs. web services

Applications can communicate directly through web services or APIs based on standard protocols—such as Simple Object Access Protocol (SOAP) or HTTP—instead of via messaging middleware. Web services are widely used in distributed systems, relatively simple, and easy to implement, making them a viable alternative to message queues in certain use cases and scenarios.

Unlike message queues, however, web services cannot guarantee message delivery. If the server or connection fails, you must build the capability to handle the error

within the client. Web services also lack pub/sub-distribution models. Messaging middleware offers greater fault tolerance and better ability to handle heavy traffic or activity bursts.

To learn more about when to use APIs, when to use messaging, or when to use both, see "An introduction to APIs and messaging."

Message queue vs. databases

Databases can be used as an alternative to message queues in certain situations, but most of the time they serve different purposes and are not readily interchangeable. Databases are most commonly used for storage, and they allow you to access the same information over and over again. Message queues cannot be used for storage purposes. Once a message has been consumed, it is deleted from the queue.

Designing message queue-like functionality into a database is possible, but it requires a great deal of coding effort and knowledge. Databases can only be used to replicate simple queue structures and aren't scalable for larger applications.

Check out "A Brief Overview of the Database Landscape" for more info on databases and their capabilities.

Message-queuing-as-a-service

Message queuing is traditionally administered by dedicated teams within IT. But "as-a-service" delivery—using a cloud-hosted message queue—can enable individuals or line-of-business (LOB) users to request changes to the messaging infrastructure on their own, through a portal, which can increase agility.

Message-queuing-as-a-service naturally works well within serverless or microservices architectures common in cloud-native development. Because it's

offered in a cloud-hosted service model, the cloud provider handles all the provisioning, installation, and maintenance of your messaging infrastructure, and it is hosted in the cloud.

Tutorials

These tutorials will help if you're new to developing applications that communicate via IBM MQ:

- MQ Essentials: Getting started with IBM MQ
- Ready, set, connect! Connect a simple MQ application to a queue manager

These additional resources will give you a more comprehensive overview:

- IBM MQ Developer Essentials Course
- MQ cheat sheet for developers
- MQ best practices to make your life easier

Message queues and IBM

IBM message queue solutions make life easier for developers by supporting hybrid cloud environments, agile development processes and microservices architectures with an all-in-one messaging backbone. This is especially important as enterprises embark upon application modernization on the journey to cloud.

Take the next step:

- Learn about about IBM Cloud Pak for Integration, which was built on the core capability of IBM MQ, the gold standard among enterprise messaging solutions.

Get started with an IBM Cloud account today.

Why IBM Cloud

Why IBM Cloud

Hybrid Cloud approach

Trust and security

Open Cloud

Data centers

Case studies

Products and Solutions

Cloud Paks

Cloud pricing

View all products

View all solutions

Learn about

What is Hybrid Cloud?

What is Cloud Computing?

What is Confidential Computing?

What is a Data Lake?

What is a Data Warehouse?

What is Artificial Intelligence (AI)?

What is Machine Learning?

What is DevOps?

Resources

Get started

Docs

Architectures

IBM Garage

Training and Certifications

Partners

Cloud blog

Hybrid Cloud careers