The **Jakarta Messaging** API (formerly **Java Message Service** or **JMS** API) is a [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) for [message-oriented middleware](https://en.wikipedia.org/wiki/Message-oriented_middleware). It provides generic messaging models, able to handle the [producer–consumer problem](https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem), that can be used to facilitate the sending and receiving of messages between [software systems](https://en.wikipedia.org/wiki/Software_system).[[1]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-1) Jakarta Messaging is a part of [Jakarta EE](https://en.wikipedia.org/wiki/Jakarta_EE) and was originally defined by a specification developed at Sun Microsystems before being guided by the [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process)

General idea of messaging[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=1)]

*See also:*[*Message-oriented middleware*](https://en.wikipedia.org/wiki/Message-oriented_middleware)*and*[*Message passing*](https://en.wikipedia.org/wiki/Message_passing)

Messaging is a form of [*loosely coupled*](https://en.wikipedia.org/wiki/Loosely_coupled#In_computing) distributed communication, where in this context the term 'communication' can be understood as an exchange of messages between software components. Message-oriented technologies attempt to relax *tightly coupled* communication (such as [TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) network [sockets](https://en.wikipedia.org/wiki/Stream_socket), [CORBA](https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture) or [RMI](https://en.wikipedia.org/wiki/Java_remote_method_invocation)) by the introduction of an intermediary component. This approach allows software components to communicate with each other indirectly. Benefits of this include message senders not needing to have precise knowledge of their receivers.

The advantages of messaging include the ability to integrate heterogeneous platforms, reduce system bottlenecks, increase scalability, and respond more quickly to change.[[3]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-3)

**Version history**[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=2)]

* **JMS 1.0**[[4]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-jms-101-spec-4)
* **JMS 1.0.1** (October 5, 1998)[[4]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-jms-101-spec-4)
* **JMS 1.0.1a** (October 30, 1998)[[5]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-6)
* **JMS 1.0.2** (December 17, 1999)[[7]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-7)
* **JMS 1.0.2a** (December 23, 1999)[[8]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-8)
* **JMS 1.0.2b** (August 27, 2001)[[9]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-9)
* **JMS 1.1** (April 12, 2002)[[10]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-10)
* **JMS 2.0** (May 21, 2013)[[11]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-11)[[12]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-12)
* **JMS 2.0a** (March 16, 2015)[[13]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-13)[[14]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-14)

JMS 2.0 is currently maintained under the [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process) as [JSR](https://en.wikipedia.org/wiki/Java_Specification_Request) 343.[[15]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-15)

JMS 3.0 is under early development as part of Jakarta EE.[[16]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-16)

Elements[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=3)]

The following are JMS elements:[[17]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-oraclejms-17)

**JMS provider**

An implementation of the JMS interface for message-oriented middleware (MOM). Providers are implemented as either a Java JMS implementation or an adapter to a non-Java MOM.

**JMS client**

An application or process that produces and/or receives messages.

**JMS producer/publisher**

A JMS client that creates and sends messages.

**JMS consumer/subscriber**

A JMS client that receives messages.

**JMS message**

An object that contains the data being transferred between JMS clients.

**JMS queue**

A staging area that contains messages that have been sent and are waiting to be read (by only one consumer). As the name queue suggests, the messages are delivered in the order sent. A JMS queue guarantees that each message is processed only once.

**JMS topic**

A distribution mechanism for publishing messages that are delivered to multiple subscribers.

Models[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=4)]

The JMS API supports two distinct models:

* Point-to-point
* Publish-and-subscribe

**Point-to-point model**[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=5)]

Under the *point-to-point* messaging system, messages are routed to individual consumers who maintain queues of incoming messages. This messaging type is built on the concept of [message queues](https://en.wikipedia.org/wiki/Message_queue), senders, and receivers. Each message is addressed to a specific queue, and the receiving clients extract messages from the queues established to hold their messages. While any number of producers can send messages to the queue, each message is guaranteed to be delivered, and consumed by one consumer. Queues retain all messages sent to them until the messages are consumed or until the messages expire. If no consumers are registered to consume the messages, the queue holds them until a consumer registers to consume them.

**Publish-and-subscribe model**[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=6)]

The [*publish-and-subscribe*](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) model supports publishing messages to a particular message "topic". *Subscribers* may register interest in receiving messages *published* on a particular message topic. In this model, neither the publisher nor the subscriber knows about each other. A good analogy for this is an anonymous bulletin board.

* Zero or more consumers will receive the message.
* There is a timing dependency between publishers and subscribers. The publisher has to create a message topic for clients to subscribe. The subscriber has to remain continuously active to receive messages, unless it has established a durable subscription. In that case, messages published while the subscriber is not connected will be redistributed whenever it reconnects.

JMS provides a way of separating the application from the [transport layer](https://en.wikipedia.org/wiki/Transport_Layer) of providing data. The same Java [classes](https://en.wikipedia.org/wiki/Class_(computer_science)) can be used to communicate with different JMS providers by using the [Java Naming and Directory Interface](https://en.wikipedia.org/wiki/Java_Naming_and_Directory_Interface) (JNDI) information for the desired provider. The classes first use a *connection factory* to connect to the queue or topic, and then use populate and send or publish the messages. On the receiving side, the clients then receive or subscribe to the messages.

URI scheme[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=7)]

[RFC 6167](https://tools.ietf.org/html/rfc6167) defines a jms: [URI scheme](https://en.wikipedia.org/wiki/URI_scheme) for the Java Message Service.

Provider implementations[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=8)]

To use JMS, one must have a JMS provider that can manage the sessions, queues and topics. Starting from Java EE version 1.4, a JMS provider has to be contained in *all* Java EE application servers. This can be implemented using the message inflow management of the [Java EE Connector Architecture](https://en.wikipedia.org/wiki/Java_EE_Connector_Architecture), which was first made available in that version.

The following is a list of common JMS providers:

* [Amazon SQS](https://en.wikipedia.org/wiki/Amazon_Simple_Queue_Service)'s [Java Messaging Library](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/jmsclient.html)
* [Apache ActiveMQ](https://en.wikipedia.org/wiki/Apache_ActiveMQ)
* [Apache Qpid](https://en.wikipedia.org/wiki/Apache_Qpid), using [AMQP](https://en.wikipedia.org/wiki/AMQP)[[18]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-18)
* [IBM MQ](https://en.wikipedia.org/wiki/IBM_WebSphere_MQ) (formerly MQSeries, then WebSphere MQ)
* [IBM WebSphere Application Server](https://en.wikipedia.org/wiki/IBM_WebSphere_Application_Server)'s Service Integration Bus (SIBus)[[19]](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_note-19)
* [JBoss Messaging](https://en.wikipedia.org/wiki/JBoss_Messaging) and [HornetQ](https://en.wikipedia.org/wiki/HornetQ) from [JBoss](https://en.wikipedia.org/wiki/JBoss_(company))
* [JORAM](https://en.wikipedia.org/wiki/JORAM) from the [OW2 Consortium](https://en.wikipedia.org/wiki/OW2_Consortium)
* [Open Message Queue](https://en.wikipedia.org/wiki/Open_Message_Queue) from [Oracle](https://en.wikipedia.org/wiki/Oracle_Corporation)
* [OpenJMS](https://en.wikipedia.org/wiki/OpenJMS) from the OpenJMS Group
* [Oracle WebLogic Server](https://en.wikipedia.org/wiki/Oracle_WebLogic_Server) and [Oracle AQ](https://en.wikipedia.org/wiki/Oracle_AQ)
* [RabbitMQ](https://en.wikipedia.org/wiki/RabbitMQ) from [Pivotal Software](https://en.wikipedia.org/wiki/Pivotal_Software)

See also[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=9)]

* [Message Driven Beans](https://en.wikipedia.org/wiki/Message_Driven_Bean)
* [Message queue](https://en.wikipedia.org/wiki/Message_queue) — the concept underlying JMS
* [Service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture)
  + [Event-driven SOA](https://en.wikipedia.org/wiki/Event-driven_SOA)
* Messaging technologies that do *not* implement the JMS API include:
  + [Advanced Message Queuing Protocol](https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) (AMQP) — standardized message queue protocol with multiple independent implementations
  + [Data Distribution Service](https://en.wikipedia.org/wiki/Data_Distribution_Service) (DDS) — An [Object Management Group (OMG)](https://en.wikipedia.org/wiki/Object_Management_Group) standardized real-time messaging system with over ten implementations that have demonstrated interoperability between publishers and subscribers
  + [Microsoft Message Queuing](https://en.wikipedia.org/wiki/Microsoft_Message_Queuing) — similar technology, implemented for [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework)

References[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=10)]

* 1. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-1) Curry, Edward. 2004. ["Message-Oriented Middleware"](http://www.edwardcurry.org/publications/curry_MfC_MOM_04.pdf). In Middleware for Communications, ed. Qusay H Mahmoud, 1-28. Chichester, England: John Wiley and Sons. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1002/0470862084.ch1](https://doi.org/10.1002%2F0470862084.ch1). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-0-470-86206-3](https://en.wikipedia.org/wiki/Special:BookSources/978-0-470-86206-3)
  2. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-2) [*"JSR 914: Java Message Service (JMS) API"*](https://jcp.org/en/jsr/detail?id=914)*. The Java Community Process Program. Retrieved July 31, 2018.*
  3. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-3) Richards et al., pages 3–5.
  4. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-jms-101-spec_4-0) [***b***](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-jms-101-spec_4-1) [*"Java Message Service"*](http://java.sun.com/products/jms/jms-101-spec.pdf)*(PDF).*[*Sun Microsystems*](https://en.wikipedia.org/wiki/Sun_Microsystems)*. October 5, 1998.*[*Archived*](https://web.archive.org/web/19990224001845/http:/java.sun.com/products/jms/jms-101-spec.pdf)*(PDF) from the original on 1999-02-24. Retrieved July 31, 2018.*
  5. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-5) [*"Java Message Service Documentation"*](http://java.sun.com/products/jms/docs.html)*. Sun Microsystems. October 30, 1998.*[*Archived*](https://web.archive.org/web/19990224001845/http:/java.sun.com/products/jms/docs.html)*from the original on 1999-02-24. Retrieved July 31, 2018.*
  6. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-6) [*"Java Message Service Source - Version 1.0.1a"*](https://web.archive.org/web/20000816170233/http:/www.javasoft.com/products/jms/jms-101a-src.zip)*. Sun Microsystems. October 29, 1998. Archived from*[*the original*](http://www.javasoft.com/products/jms/jms-101a-src.zip)*(ZIP)on August 16, 2000. Retrieved July 31, 2018.*
  7. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-7) [*"Java Message Service"*](http://java.sun.com/products/jms/jms1_0_2-spec.pdf)*(PDF). Sun Microsystems (published December 17, 1999). November 9, 1999.*[*Archived*](https://web.archive.org/web/20000823085445/http:/java.sun.com/products/jms/jms1_0_2-spec.pdf)*(PDF) from the original on 2000-08-23. Retrieved July 31, 2018.*
  8. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-8) [*"Java Message Service Documentation"*](http://java.sun.com/products/jms/docs.html)*. Sun Microsystems. December 23, 1999.*[*Archived*](https://web.archive.org/web/20000229161639/http:/java.sun.com/products/jms/docs.html)*from the original on 2000-02-29. Retrieved July 31, 2018.*
  9. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-9) [*"Java Message Service"*](http://download.oracle.com/otn-pub/jcp/7543-jms-1.0.2b-spec-oth-JSpec/jms-1_0_2b-spec.pdf)*(PDF). Sun Microsystems. August 27, 2001. Retrieved July 31, 2018.*
  10. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-10) [*"Java Message Service"*](https://github.com/javaee/jms-spec/raw/9383d33ff79a156d09fb5c9c0ccdfd6b844b1d54/jms1.1/specification/jms-1_1-fr-spec.pdf)*(PDF). Sun Microsystems. April 12, 2002. Retrieved July 31, 2018.*
  11. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-11) [*"Java Message Service"*](https://github.com/javaee/jms-spec/raw/9383d33ff79a156d09fb5c9c0ccdfd6b844b1d54/jms2.0/specification/word/JMS20.pdf)*(PDF).*[*Oracle*](https://en.wikipedia.org/wiki/Oracle_Corporation)*. March 20, 2013. Retrieved July 31, 2018.*
  12. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-12) [*"JMS 2.0 Final Release"*](https://javaee.github.io/jms-spec/pages/JMS20FinalRelease)*. Java Message Service Specification. June 9, 2017. Retrieved July 31, 2018.*
  13. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-13) [*"Java Message Service"*](https://github.com/javaee/jms-spec/raw/9383d33ff79a156d09fb5c9c0ccdfd6b844b1d54/jms2.0a/specification/word/JMS20.pdf)*(PDF). Oracle. March 10, 2015. Retrieved July 31, 2018.*
  14. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-14) [*"JMS 2.0 errata release (Rev a)"*](https://javaee.github.io/jms-spec/pages/JMS20RevA)*. Java Message Service Specification. July 5, 2017. Retrieved July 31, 2018.*
  15. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-15) [*"JSR 343: Java Message Service 2.0"*](https://jcp.org/en/jsr/detail?id=343)*. The Java Community Process Program. Retrieved July 31, 2018.*
  16. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-16) *Monson-Haefel, Richard (December 6, 2018).*[*"JMS 3.0: Get Involved!"*](https://www.tomitribe.com/blog/jms-3-0-get-involved/)*. Tomitribe. Retrieved July 17, 2020.*
  17. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-oraclejms_17-0) [Java Message Service (JMS)](http://www.oracle.com/technetwork/java/jms/index.html)
  18. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-18) [*"Apache Qpid™: Open Source AMQP Messaging"*](http://qpid.apache.org/)*.*
  19. [**^**](https://en.wikipedia.org/wiki/Jakarta_Messaging#cite_ref-19) *Wallis, Graham.*[*"Choosing a messaging system: WebSphere MQ vs. the WebSphere Application Server Service Integration Bus"*](http://www.ibm.com/developerworks/websphere/library/techarticles/1109_wallis/1109_wallis.html)*. IBM developerWorks.*

Further reading[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=11)]

* *Richards, Mark; Richard Monson-Haefel; David A. Chappell (2009). Java Message Service, Second Edition. O'Reilly.*[*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier))[*978-0-596-52204-9*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-596-52204-9)*.*

External links[[edit](https://en.wikipedia.org/w/index.php?title=Jakarta_Messaging&action=edit&section=12)]

|  |  |
| --- | --- |
|  | The Wikibook [*Java EE Programming*](https://en.wikibooks.org/wiki/Java_EE_Programming) has a page on the topic of: [***Java Message Service***](https://en.wikibooks.org/wiki/Java_EE_Programming/Java_Message_Service) |

* [Official website](https://projects.eclipse.org/projects/ee4j.jms) [Edit this at Wikidata](https://www.wikidata.org/wiki/Q755515#P856)
* [JSR 343: Java Message Service 2.0](https://jcp.org/en/jsr/detail?id=343)
* [javax.jms](https://javaee.github.io/javaee-spec/javadocs/javax/jms/package-summary.html) API [Javadoc](https://en.wikipedia.org/wiki/Javadoc) documentation
* [Oracle's Java EE 7 JMS tutorial](https://docs.oracle.com/javaee/7/tutorial/partmessaging.htm)
* [A historical comparison matrix of JMS providers](https://www.theserverside.com/matrix)