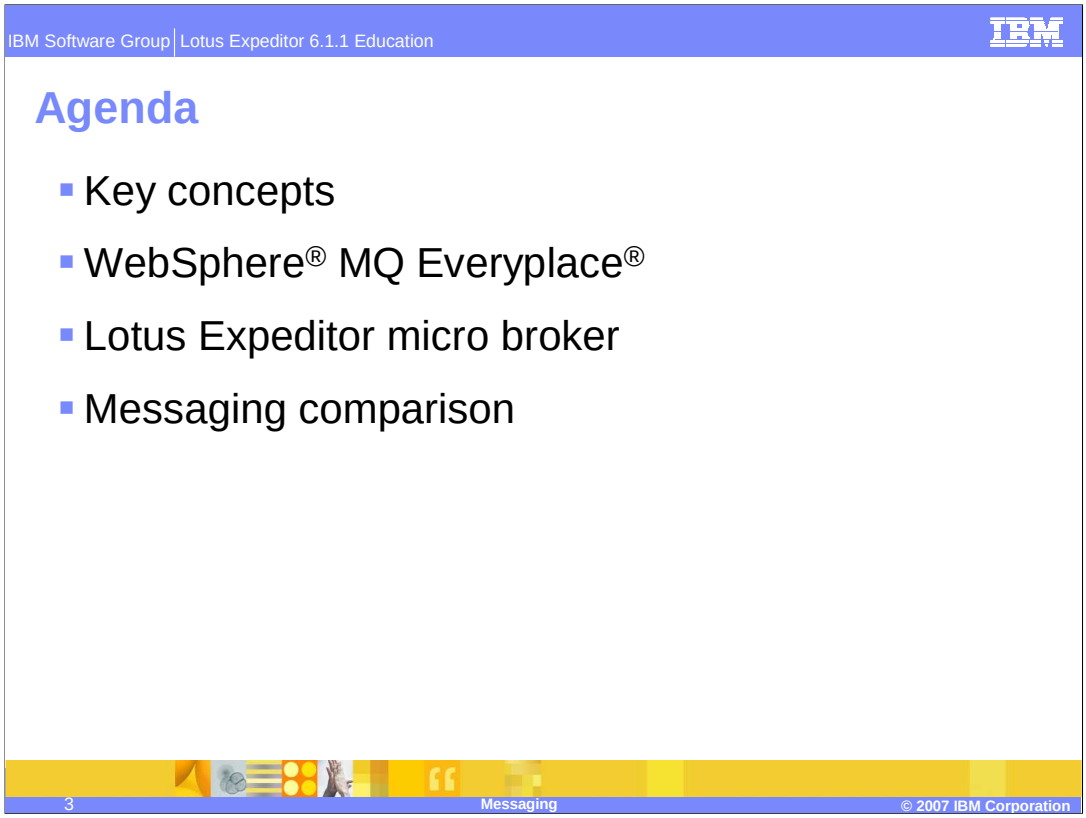
This presentation explains the messaging capabilities in the IBM Lotus Expeditor Client for Desktop.

Goal

- Understand the messaging services available in the IBM® Lotus® Expeditor Client for Desktop

The goal of this presentation is to understand the messaging services provided by IBM Lotus Expeditor Client for Desktop.

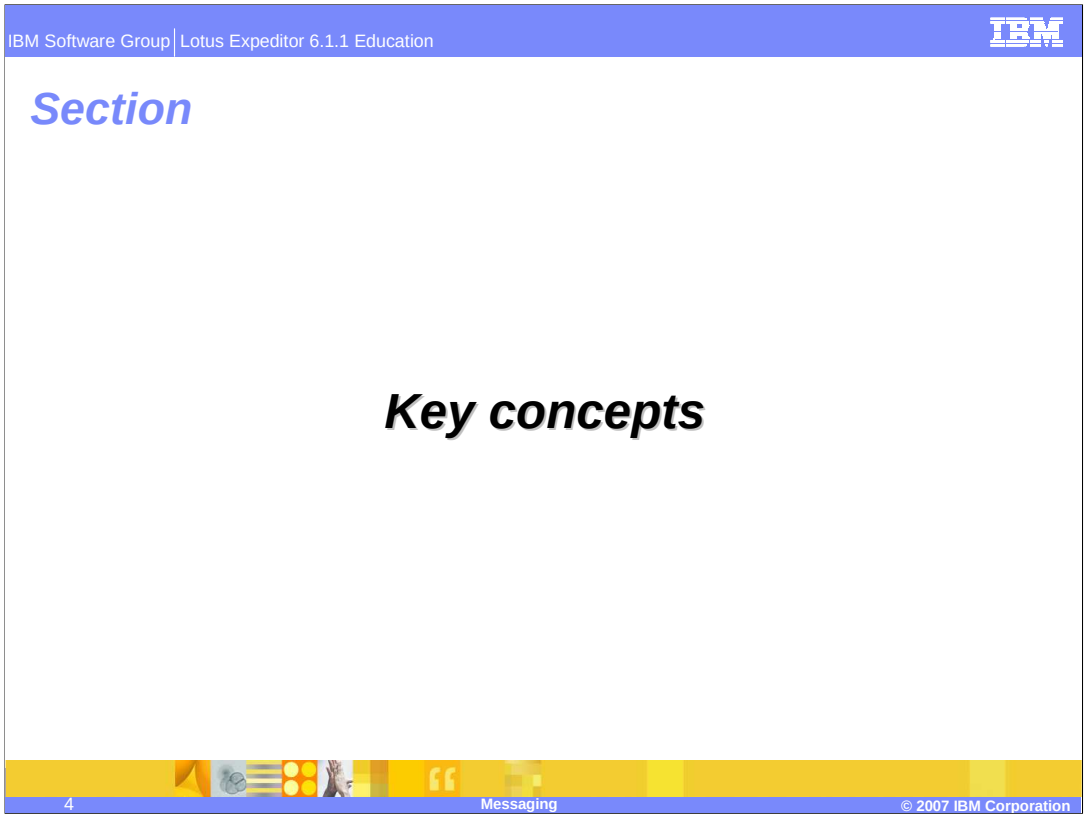


The agenda of this presentation is to explain key concepts,

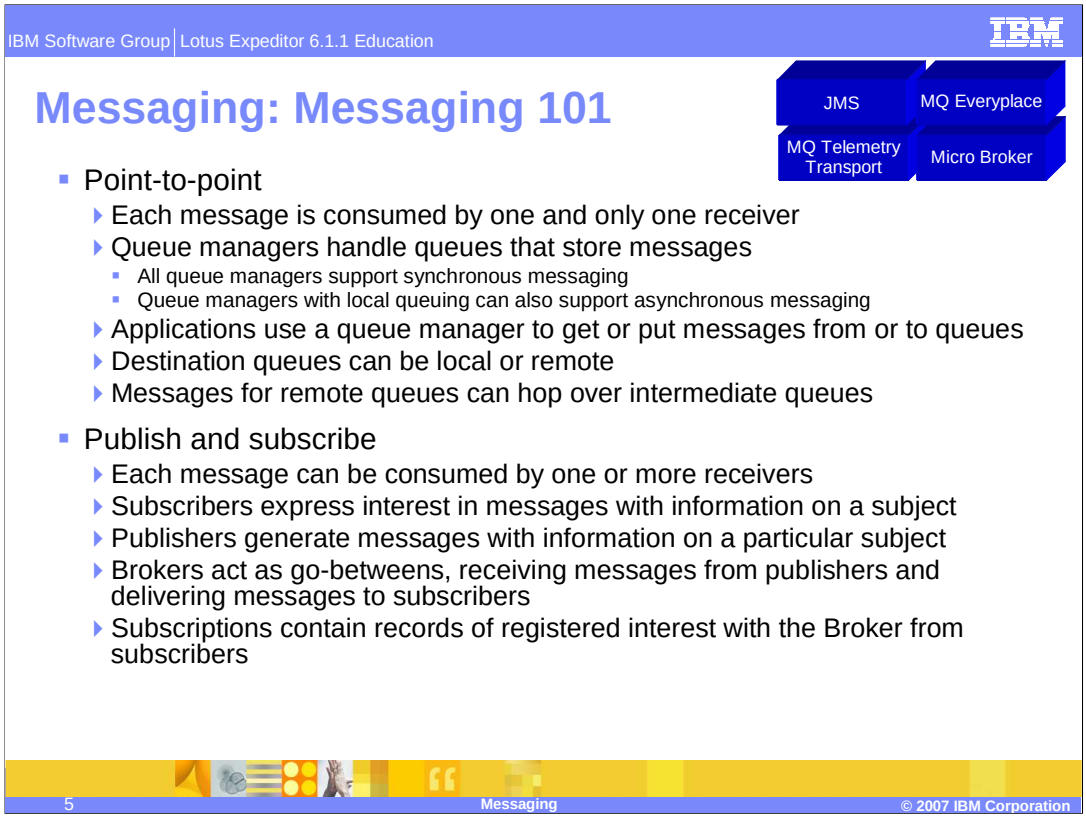
describe the WebSphere MQ Everyplace messaging service,

describe the MQ Telemetry Transport and Lotus Expeditor micro broker messaging services,

and to compare these messaging services so you can determine which service best meets your requirements.



Let’s start with an overview of key messaging concepts.



Messaging is separated into two main categories: Point-to-point messaging, and Publish and Subscribe messaging.

To take full advantage of the messaging capabilities of the client platform, it is important to understand the differences in these two messaging types.

In **Point-to-point** messaging, queue managers handle queues that store messages. Applications communicate with a local queue manager, and get or put messages to queues. If a message is put to a remote queue (a queue owned by another queue manager), the message is transmitted over connections to the remote queue manager. In this way, messages can hop through one or more intermediate queue managers before reaching their final destination. You can configure queue managers with or without local queuing. All queue managers support synchronous messaging operations. A queue manager with local queuing also supports asynchronous message delivery. The point-to-point messaging paradigm provides one-to-one messaging. In other words, messages are consumed by only one receiver, unlike publish-and-subscribe where messages are consumed by multiple receivers.

In **Publish and Subscribe** messaging the application programming model for a publish and subscribe messaging paradigm consists of the following:

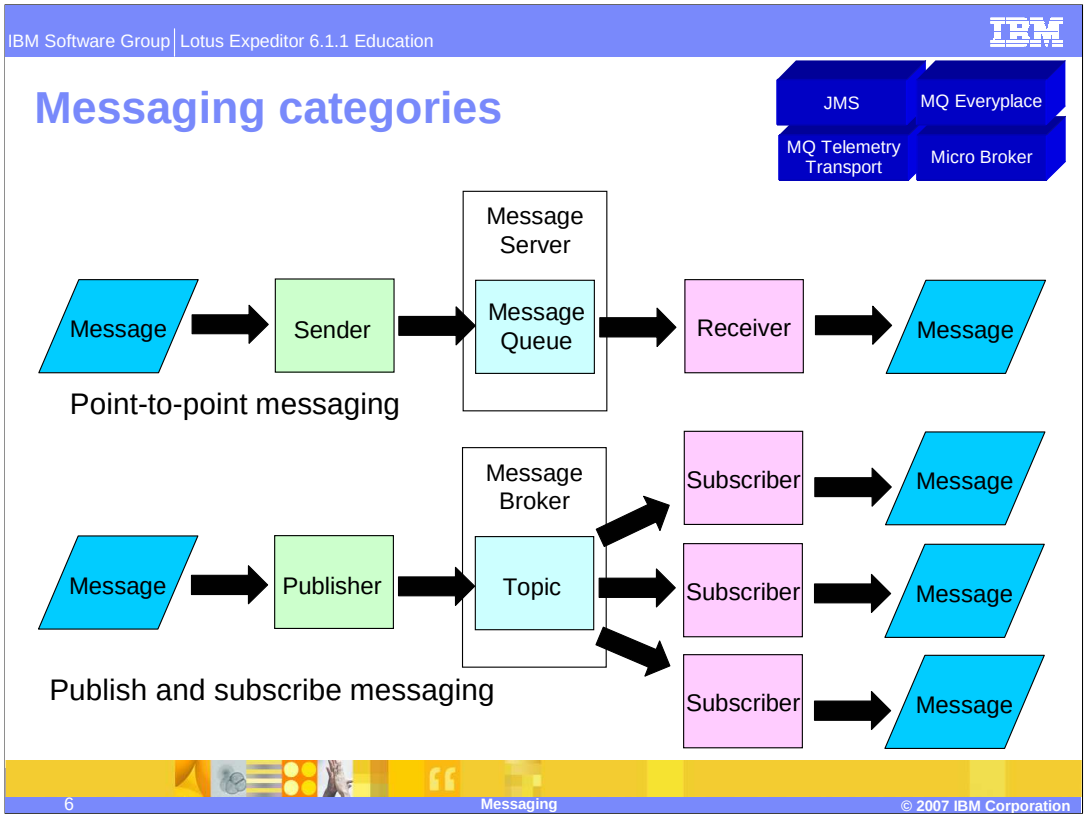
Each message may be consumed by one or more receivers

Subscribers: Express an interest in messages containing information on a particular subject.

Publishers: Generate messages containing information about a particular subject. Messages are sent to a broker. The publish and subscribe messaging paradigm provides one-to-many messaging.

Brokers: Act as go-betweens, receiving messages from publishers and comparing them to the needs of subscribers. A message is delivered to all subscribers that have expressed an interest in the subject of the message.

Subscriptions: Contain records of registered interest with the Broker from a subscriber.



This slide illustrates how point-to-point and publish and subscribe messaging work. As you can see, the client platform provides point-to-point messaging through the Java™ Message Service (JMS) with WebSphere MQ Everyplace (MQe), and publish and subscribe messaging through the MQ Telemetry Transport (MQTT) and micro broker.

Let's explore each of these messaging services in more detail.

Section

MQ Everyplace

Next, let’s explore the capabilities provided by the MQ Everyplace messaging service.

Messaging: Point-to-point WebSphere MQ Everyplace (MQe)

JMS | MQ Everyplace

- Supports the point-to-point parts of the JMS 1.1 API
  - javax.jms
- WebSphere MQ Everyplace implements point-to-point messaging
  - Provides a small, embeddable version of WebSphere MQ for clients
  - Extends WebSphere business integration to clients
    - Configure MQe queue manager with bridge capabilities
  - Supports synchronous message delivery
  - Supports asynchronous message delivery
    - Provides once-only, assured delivery of transaction messages
    - Supports online and offline operations using local message queues
  - Secures messages using encryption, non-repudiation, authentication
  - Compresses messages to reduce transmission costs

Java Message Service (JMS) is the standard Java API for messaging. It supports the two messaging categories: point-to-point messaging and publish and subscribe messaging. JMS is defined as part of J2EE. It defines a package of Java interfaces, which allows for provider-independence, but does not necessarily allow for provider interoperability. The JMS APIs are provided with the client platform.

The client platform also includes a point to point JMS provider based on MQe messaging. The MQe classes for JMS are a set of Java classes that implement the JMS interfaces to enable JMS programs to access MQe systems.

WebSphere MQ Everyplace (MQe) is a member of the IBM WebSphere MQ family of business messaging products. It exchanges messages with various applications, providing once and once-only assured delivery leveraging the point to point message paradigm.

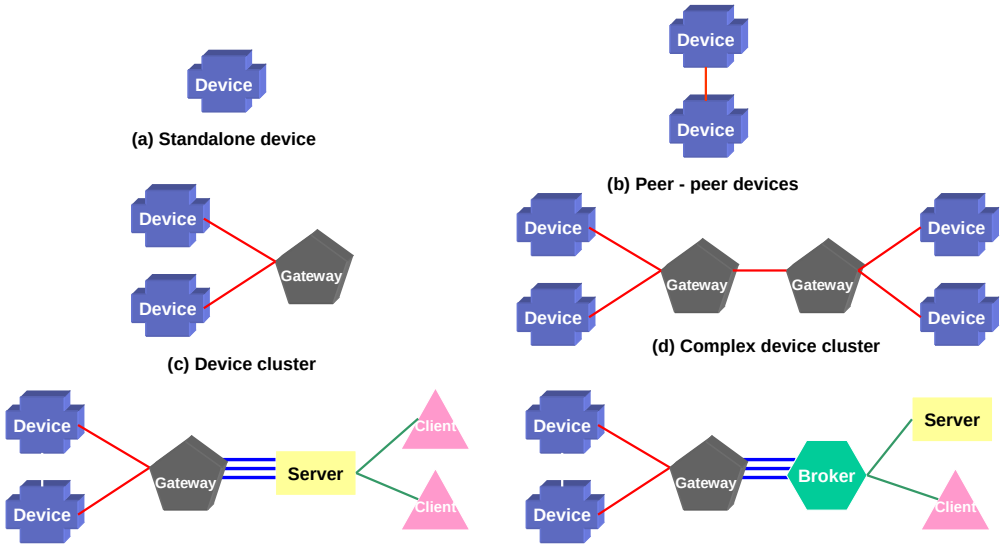
MQe provides an integrated set of security features enabling the protection of message data both when held locally and when being transferred. With *synchronous* message delivery, the application puts the message to MQe for delivery to the remote queue. MQe simultaneously contacts the target queue and delivers the message. After delivery, MQe returns immediately to the application. If the message cannot be delivered, the sending application receives immediate notification. MQe does not assume responsibility for message delivery in the synchronous case (non-assured message delivery).

With *asynchronous* message delivery, the application puts the message to MQe for delivery to a remote queue. MQe immediately returns to the application. If the message can be delivered immediately, or moved to a suitable staging post, it is sent. If not, it is stored locally. Asynchronous delivery provides once and once-only assured delivery. After the message is provided to MQe, control is returned to the application. MQe next takes responsibility for assured delivery of the message. Delivery occurs in the background allowing the application to carry on its processing.

MQe also has the ability to exchange messages with WebSphere MQ host queue managers and brokers. To do this, configure a MQe queue manager with bridge capabilities. Without the bridge, a queue manager can communicate directly only with other MQe queue managers. However, it can communicate indirectly through other queue managers in the network that have bridge capabilities.

MQe secures messages using encryption, non-repudiation, authentication, and compresses messages to reduce transmission costs.

MQ Everyplace topologies

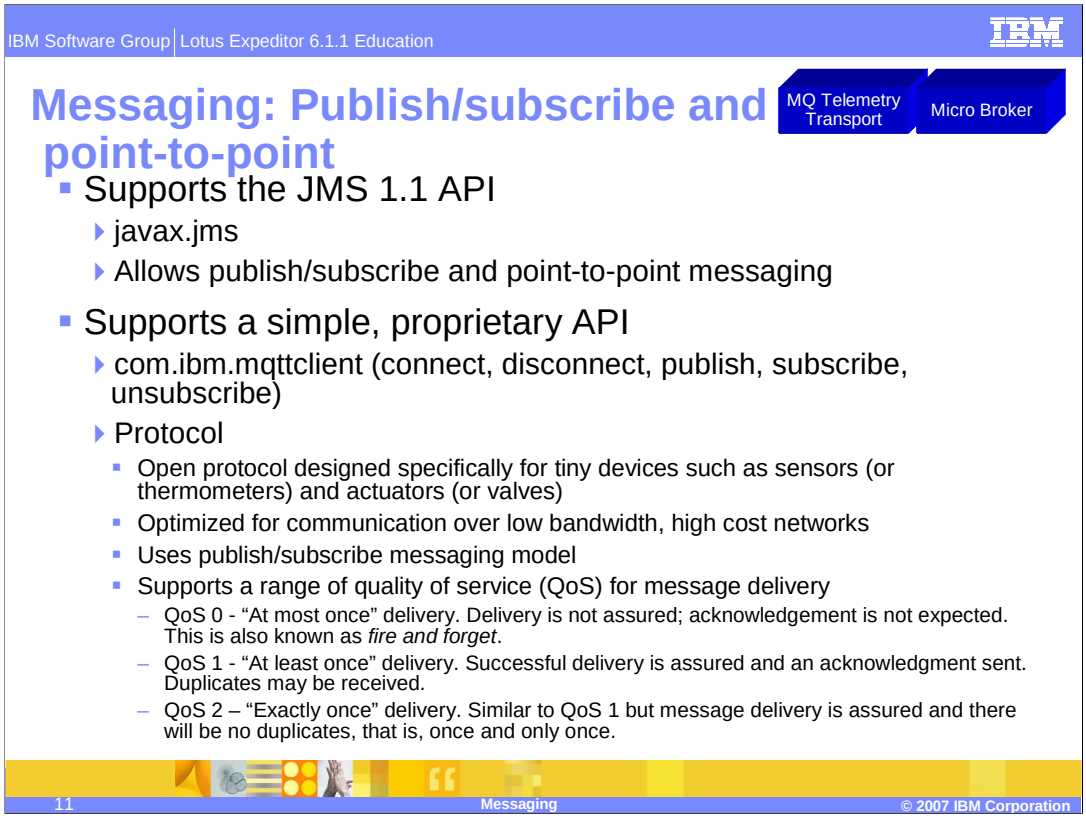


This slide shows topologies supported by WebSphere MQ Everyplace.

Section

Lotus Expeditor micro broker

Next, let's cover the capabilities provided by the MQ Telemetry Transport and the micro broker messaging service.



MQ Telemetry Transport (MQTT) is an open protocol designed for resource-constrained devices and networks, providing publish and subscribe messaging over TCP/IP.

Clients operate in conjunction with a suitable message broker, such as the micro broker, WebSphere Business Integration Message Broker, or WebSphere Business Integration Event Broker, which are responsible for the syndication of messages.

As a wire protocol, no device API is mandated; rather, the implementations expose a simple semantic including: connect/disconnect, publish, and subscribe/unsubscribe.

Provision is made for assurance of message delivery using one of three levels of service; fire and forget, at most once, and exactly once.

By minimizing the requirement on network bandwidth, it is practical to use MQTT in wide area networks, which typically have lower link speeds than wired networks. This facilitates not only using MQTT for the collection of data, but also for the presentation of data on handheld devices.

A Java client implementation of the MQTT wire protocol is provided to simplify MQTT client programming. For more information, see the URL: http://www.mqtt.org.

Messaging: Micro broker

MQ Telemetry Transport | Micro Broker

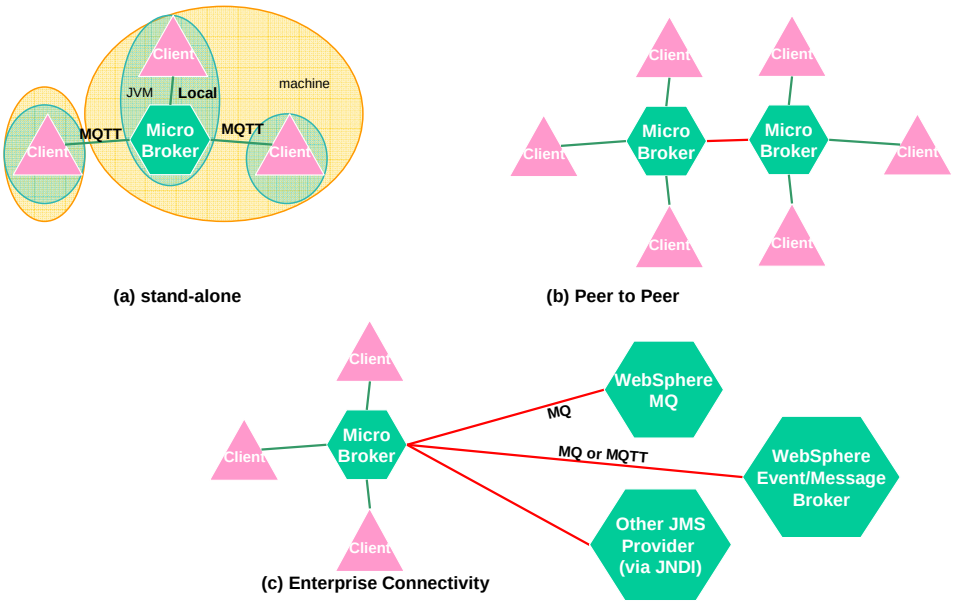
- Small footprint, 100% Java message broker
- Supports messaging, notification, and event services
- Enables lightweight messaging clients to communicate with each other, on one host, across a network, or with enterprise brokers (with bridge capabilities)
- Uses the MQ Telemetry Transport protocol over TCP/IP
- Provides a “bridge” to transform and route messages to other messaging systems, such as the WebSphere Message Brokers or WebSphere MQ

The micro broker is a very small footprint, 100% Java message broker, capable of running in resource-constrained environments. It is suitable for embedding in applications and solutions that have a need for messaging, notification and event services. Micro broker supports the publish and subscribe messaging paradigm. It provides a messaging infrastructure, which enables lightweight messaging clients to communicate with each other, on one host or across a network, and with enterprise brokers through its bridging capabilities.

Micro broker uses the MQ Telemetry Transport (MQTT) protocol over TCP/IP and, optionally, uses a DB2e database to provide persistent storage of publications and state information. The Micro broker also provides a “Bridge” to transform and route messages to WebSphere Business Integration Message Brokers or WebSphere MQ, thus enabling connection to an Enterprise Service Bus.

Micro broker topology

MQ Telemetry Transport | Micro Broker



This slide shows topologies supported by the micro broker.

Green ovals indicate JVM boundaries

Orange ovals indicate machine boundaries

Section

Messaging comparison

Finally, let’s compare these messaging services to help you determine which service best meets your requirements.

Messaging comparison

	WebSphere MQ Everyplace	Micro Broker
JMS provider included	Yes, point-to-point provider	Yes
Messaging paradigm	Point-to-point (queue-based)	Point-to-point and publish/subscribe
Implementation type	Java-based (platform-independent) implementation	Java-based (platform-independent) implementation
Bridging – provides bridge to ESB	Bridge to MQe and WebSphere Message and Event Brokers	Supports bridge to any generic JMS provider, or specifically WebSphere MQ and WebSphere Message and Event Brokers
Wire protocol	MQe-specific	MQTT standards based
Small footprint client	No	Yes, Java and 'C'
QoS for message delivery	At least once and exactly once	Fire and forget, at least once, and exactly once
Local & remote queues	Yes	Yes
Built in security	Yes	No

Use this table to help you decide which messaging service meets your requirements. However, note that this table is not an exhaustive comparison of the two products. See the product documentation for more complete information about these products.

Notes

- MQe permits only a single QueueManager to be defined per JVM. Applications that define their own queue manager infrastructure cannot co-exist with other applications that do the same.
- Avoid non Latin 1 characters in QueueManager or queue information
- Micro broker permits only a single broker instance per JVM
- MQTT topics are limited to the ASCII character subset of UTF-8

This slide covers some additional notes regarding messaging.

MQe permits only a single QueueManager to be defined per JVM. Applications that define their own queue manager infrastructure cannot co-exist with other applications that do the same.

Avoid non Latin 1 characters in Queue Manager and Queue information

Micro Broker permits only a single broker instance per JVM

Topics are limited to ‘ASCII’ character subset of UTF8

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about xpdv6.1.1_access_services_messaging.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Everyplace IBM Lotus WebSphere

Java, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.