IBM WebSphere MQ V7.0

Introduction and Technical Overview

Unit Agenda

- Why is Messaging Important to the Enterprise?
- What is WebSphere® MQ? Why use it?
- What are:
  - Messages
  - Queues
  - Queue Managers
  - Channels
- Lab 1 – Exploring WebSphere MQ
What is WebSphere MQ - IBM

- A proven way of bridging between the components of your Service Oriented Architecture (SOA)
- Like a strong, broad bridge it robustly links your applications and your Web services
- It connects virtually any commercial IT system
- Helping you to share and exchange critical business information with ease, confidence and security

Why is this a Challenge?

“Computers are really dumb. You have to tell them everything.”

- How do you...?
  - Move data across different systems, platforms, and devices when the HW, SW configurations and programming models are different?
  - Overcome network failures?
  - Deliver information when the target application is not online or is busy?
  - Ensure transmission integrity and recovery?
  - Handle lost or duplicate data?
  - Ensure a secure connection?
  - Ensure multi-step transactions either happen completely or not at all?
  - Apply qualities of service based on different requirements? e.g., assured delivery, fast delivery?
  - Manage a session (request/response)?
  - Efficiently distribute events?
  - Scale to handle volumes?
  - Deal with data in unlike formats?
  - Determine which data to send where?
  - Audit who sent what, where and when?

You either:

1. Program it all into your applications
2. Build your own middleware
3. Buy middleware to do it for you
TechWorks

History

- IBM MQSeries® Introduced in 1992, initial release in 1994
  - C, COBOL, PL/I language initially supported by MQI
    - Limited initial set of API verbs (MQCONN, MQOPEN, MQPUT, MQGET, etc)
- MQ Publish/Subscribe support introduced in 1998
- JMS Specification published around the same time
  - Java™ Message Service (JMS) provided a higher level OO abstraction for messaging and simplified Pub/Sub
  - MQSeries implementation of JMS introduced shortly after specification published (1999)
- Additional languages, platforms and protocols supported over time
  - VB, .NET, C++, C
  - Tandem (now HP NonStop Server), DEC (now HP OpenVMS), etc
  - Message Service API (XMS) introduced to provide non-Java developers the benefits of JMS
  - Java Connector Architecture (JCA) and HTTP support extends reach
- Product renamed WebSphere MQ in 2002
- Latest release introduces new MQ API (MQI) functions closely mapped to JMS specification.

What does WebSphere MQ do?

- Provides **messaging services** to applications and Web services that need to exchange data and events with:
  - Proven reliability
  - Transactional integrity
  - Consistency
  - Time independence
  - Ease and Speed
  - Flexibility
  - High-performance
  - Security
  - Scalability
  - Auditability

*WebSphere MQ is like email for SOA applications ...but email you can bet your business on*
How do you use WebSphere MQ?

Developers attach applications and Web services to WebSphere MQ using a choice of cross-platform languages and interfaces — such as JMS.

Application and technology adapters accelerate this activity.

Integration specialists use cross-platform graphical tooling to configure their messaging networks — these tools are based on open source Eclipse.

How does WebSphere MQ work?

- Messaging services are based on Queues that store and forward data based on simple programming commands.
- Uses the proven database technique of two-phase commit transactions to ensure messages are not lost or duplicated.
- Uses publish/subscribe to route messages dynamically based on keywords or “topics”.
- Uses multi-processor threading and clustering to accelerate throughput of messages.
The Value of Loose Coupling

Synchronous vs. Asynchronous Communications

Synchronous communications = Telephone conversation

You can't simultaneously have separate conversations without:
- Experiencing overlapping conversations and losing track of what is going on or
- Waiting for the other person to finish before responding.

Neither can your applications!

Asynchronous communications = Voicemail / Email

Please leave a message and I will listen to it when I am available.
Elements of Messaging and Queuing

- Programs communicate by putting messages in message queues

“A building block for distributed processing”

Elements of Messaging and Queuing

- Communication can be one way or two way
Elements of Messaging and Queuing

Either program can be busy or unavailable

There can be a one to many relationship between applications

Or a many to one relationship between applications
What is a Message?

- A message is considered to be the unit of data to be moved from one application to another
- A message is built by an application
- A message is consumed by a different application
- Message can contain **any** kind of data:
  - Binary data
    - A video clip, a song, a photograph, a sensor reading, etc…
  - Text data
    - Raw text
    - XML
  - Structured data (C Structures, COBOL Copybook, Serialized Java objects)
  - The source data is the choice of the application

The structure of an MQ Message

**Message Headers**
- A set of Message Attributes understood and augmented by the Queue Manager
  - Unique Message Id
  - Correlation Id
  - Routing Information
  - Reply Routing Information
  - Message Priority
  - Message Persistence
  - Persistent
  - Non-persistent
  - Message Codepage
  - Message Format
  - Etc...

**Message Properties**
- Arbitrary values associated with the message but not part of the body
- Properties can be integers, strings, boolean, etc.
- Receiving applications do not see them unless they want
- Permits explicit statement of relationships between messages
  - e.g. Message X is a **REPLY** to Message Y

**Message Data**
- Any sequence of bytes
- Defined by the sending program
- Understood by the receiving program
- NOT meaningful to the Queue Manager
- Can contain any data
- Structured
  - XML, Tagged, Tagged Delimited, C or Cobol defined, etc.
- Unstructured
  - Binary
    - A video, a picture, etc.
  - Any content
What is a Queue?

- **Messages** are delivered asynchronously to a **Queue**
- **A Place to hold messages**
- **Queue creation**
  - Pre-defined
  - Dynamic definition
- **Message Access**
  - FIFO (first in first out)
  - Priority (FIFO within Priority)
  - Direct
  - Destructive & non-destructive access
- **Parallel access by applications**
  - Managed by the queue manager

What is a Queue Manager?
How are Messages Persisted?

Messages may be written to queue files
- Persistent messages are logged
- Persistent Messages are always recoverable
- Logging has implication on performance
- Non-persistent Messages have 2 classes of service:
  - Messages are retained for the life of the Qmgr
  - Messages can survive a normal shutdown and restart of the Queue Manager

What are Channels?

- Queue Manager to Queue Manager
  - Uni-directional
  - Usually defined in pairs for example:
    - One Sender
    - One Receiver
  - Asynchronous
- Client to Queue Manager
  - Bi-directional
  - Defined as a single channel
  - Synchronous
- A building block for a reliable Enterprise Service Bus (ESB)

**Note: Client to Client communication must go via a Queue Manager**
Reliable, asynchronous communication with WebSphere MQ

Accept Message
- Receive message from application
- Manage “unit of work”

Apply Security (optional)
- Access Control
  (permission to get/put by queue or topic)

Deliver Message(s)
- Deliver message to application
- Ensure Exactly Once Delivery
  (even after a failure)
- Manage “unit of work”

Queues can be Local or Remote
Parallel Processes

Asynchronous Syncpoint

Synchronous Model

Asynchronous Model
Triggering (Overview)

Queue Manager

Appl.Q

Trigger Mechanism

External Facility

START

A

B

Parallel Processes
Discovering the value of WebSphere MQ V7 for Your Enterprise Messaging Needs
Developing WebSphere MQ Applications

- WebSphere MQ supports a wide range of platforms
  - Windows®, UNIX®, Linux®, z/OS®, i5OS®, HP NonStop, etc...
- With a whole range of programming languages
  - Java, C/C++, C#, .NET, COBOL, RPG, TAL, etc...
- And a wide range of Interfaces
  - MQI, JMS, .NET, etc...
- A large number of sample programs are provided to show how to develop MQ applications for various languages and environments, for example:
  - Put sample: amqsput
  - Get sample: amqsget
  - Browse sample: amqsgbr
  - Publish sample: amqspub
  - Subscriber sample: amqssuba
  - ...and many others

The solution to Universal Connectivity → IBM WebSphere MQ

WebSphere MQ can dramatically reduce application infrastructure costs by providing a single manageable distributed infrastructure for all application messaging traffic.

Features:
- WebSphere MQ supports the broadest range of APIs, programming languages and OS platforms
- Provides the only JMS engine that can be implemented on “any” standards-compliant JEE server
- Provides rich web services interfaces for customers needing reliable SOAP message delivery
- Offers a broad range of qualities of service and messaging methods including publish/subscribe
- Supports major transaction monitors and database managers
- Offers the most scalable, most manageable messaging system available
- Assures transactional message delivery end-to-end
**WebSphere MQ Enterprise Class Messaging**

- Proven Scalability
  - Grow your network incrementally one server at a time
- Performance
  - Many clients are moving millions of messages per day
- Administrate massive networks
  - Cross-platform, remote configuration tooling
  - Tivoli® CAM for enterprise-wide systems administration
- Support for virtually any commercial IT platform
  - MQ for z/OS
    - Built from the ground up to exploit zSeries platform
    - Consistent with MQ on distributed platforms
  - Clustering on distributed, shared queues on z/OS
    - For High-Availability and workload balancing
    - Easier to set up than you may think!
- Multi-threading
  - Exploits multi-processors for high-speed throughput
- Security
  - Industry-standard SSL support
  - Certified for Common Criteria
  - Policy-based security with MQ Extended Security Edition
- IBM’s worldwide 24x7 support

- 90% of the Fortune 100
- 300 of the Fortune 500
- 66% of NA and European banks
- Banking clients move transactions worth $35 Trillion over MQ
- Government clients move 675+ million messages per day over MQ