Application Servers
G22.3033-003

Session 11 – Sub-Topic Presentation
Sample Messaging Service - WebSphere MQ

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

WebSphere MQ
4000 MQ Customers
Growing at 20% Annually

Pretty set in their ways and tools

DEGREE OF MATURITY

FORMALIZATION

EXPANSION

INITIATION

TIME

IBM’s MQ Family now re-branded under WebSphere umbrella

WebSphere Process Mgr

WebSphere MQ Integrator

WebSphere MQ

Process flow
Application Services

Messaging Services
Standard Formats

Rules, Routing
API Framework
Templates
Objectives of presentation:
- What is MQ
- Sample MQ Interface to IMS

WebSphere MQ enables application programs to communicate with each other using messages and queues. This form of communication is referred to as commercial messaging.

There are two methods for applications to communicate:

- Fire and Forget
- Request/Response
**Application Communications**

**Fire and Forget**

Program A \(\xrightarrow{\text{Put Record}}\) Record Queue \(\xrightarrow{\text{Get Record}}\) Program B

**Request/Response**

Program A \(\xrightarrow{\text{Put Record}}\) Record Queue \(\xrightarrow{\text{Get Record}}\) Program B

Get Record
Processed Message

Put Record
Processed Message

**What is a message?**

Message = Header(s) + Application Data

![Message](image)

A **message consists of a header** and the attached application **data**. **Headers** typically contain elements like:

- Unique Message Id
- Routing information
- Message format

The following are examples of the **data** part of a message:

- A record from an indexed or flat file
- A row from a DB2 table
- Individual columns from DB2 tables
- Multiple rows or records
Messaging

EMAIL SYSTEM

Application A

Store

Forward

Retrieve

Application B

Data

Data

MQ Messaging

Application A

Application B

Application C

MQ Interface

Queue Manager

Queue 1

Queue 2
Enterprise-Wide Problems

- Communications required at application level
- Active connections required
- Different throughput rates
- Application Programming Interfaces not consistent between protocols and vendors
- Difficult to recover after failures

What is a Queue?

A queue is simply a place to put data.

This figure shows how messaging works in the simple case where the program putting the message and program getting the message are both on the same computer and connected to the same queue manager.
A queue is an MQ object that deals with storage or movement of messages. It has attributes that determine what processing occurs when an application accesses it through the MQI calls.

### Queue Definition Types

<table>
<thead>
<tr>
<th><strong>Application</strong></th>
<th><strong>System</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Transmission</td>
</tr>
<tr>
<td>Remote</td>
<td>Event Queues</td>
</tr>
<tr>
<td>Alias</td>
<td>– Performance</td>
</tr>
<tr>
<td>Dynamic</td>
<td>– Channel</td>
</tr>
<tr>
<td></td>
<td>– System</td>
</tr>
<tr>
<td></td>
<td>Dead letter</td>
</tr>
<tr>
<td></td>
<td>Initiation</td>
</tr>
<tr>
<td></td>
<td>Command server</td>
</tr>
<tr>
<td></td>
<td>Broker Management</td>
</tr>
<tr>
<td></td>
<td>Cluster Management</td>
</tr>
</tbody>
</table>

— Permanent

— Temporary

— Model
Local Queues

Queue Manager assures “safe” delivery

- Example of local Message Queue receiving data
- Destination is “local” or residing on same node

Transmission Queue

- A transmission queue has to be defined for each channel
- A local queue associated with a sender channel
- Usually has the same name as the destination queue manager
- Local queue manager places all messages for remote queues on a transmission queue
Transmission Queue

Queue Manager places Message on transmission queue
MCA reads message from transmission queue and delivers to remote queue manager

Remote Queue

- Is a local definition of a queue residing on another queue manager
- Definition specifies:
  - Remote Queue Name
  - Remote Queue Manager Name
  - (optionally) Transmission Queue Name
- Transmission Queue name binds Remote Queue to a Channel
Remote Queue

Queue Manager knows queue is remote

Message Queue

Comms Protocol

MQPUT

Queue Manager knows queue is remote

Application

Remote Queue Definition

Alias Queues

◆ Another name to use in reference to a queue
◆ Can refer to Local or Remote Queue
Initiation Queue

- Used in conjunction with triggers
  - A local queue where the queue manager places trigger messages
  - A trigger monitor application reads trigger events from the initiation queue
  - The trigger monitor application starts the appropriate application to process the message
  - At least one initiation queue must be defined to the queue manager if triggers are active

Command Queue

- Command queue accepts messages to support remote administration
  - Text messages
  - Programmable Command Format (PCF)
- Command server supports program administration
  - Consistent message format across several platforms
  - Messages can be sent to remote administration queue
Event Queues

- Used to receive event messages
- The event messages indicate an instrumentation event has occurred
- Three system administration event queues
  - SYSTEM.ADMIN.QMGR.EVENTS
  - SYSTEM.ADMIN.PERFM.EVENTS
  - SYSTEM.ADMIN.CHANNEL.EVENTS

Model Queue

- Exists to specify default attributes for dynamic queue creation
- Name of model queue used to create queue, attributes of model queue are used but temporary name is generated
Dead Letter Queue

- Queue specified for undeliverable messages
- Optional facility within MQ
  - Queue Manager will stop Channel Agent if message undeliverable

What is Queue Manager?

A queue manager is the subsystem software which controls access to the individual queues assigned to it. The queue manager logs all activity with each individual queue thus creating an audit trail. Multiple queue managers can coexist with each other. The limiting factor is the availability of system resources.
**Queue Manager**

A queue manager is the MQ component that provides the messaging and queuing services to application programs through Message Queue Interface (MQI) program calls.

**Queuing Technology**

Message Oriented Middleware

External Network
MQ Benefits

- Take away the communications nightmares
- Component-built applications approach
- Allow control of load balancing
- Allow protocol independence
- Provide consistent programming interface across platforms
- More platforms than any other products

MQ System Perspective

- Time independent (asynchronous) processing
- Connectionless communications
- Assured message delivery
- Once and once only delivery
- Syncpoint control
**MQ Application Development Perspective**

- Single, multi-platform Application Programming Interface (API)
  - Procedural (COBOL, PL/1, RPG, etc.)
  - Object Oriented (C++, Java, ActiveX, etc.)

- Faster application development

- Portable code

**MQ Objects**

- Queue managers
- Queues
- Channels
- Processes
- Namelists
**Namelist**

A Namelist is an MQ object that contains a list of other MQ objects.

**Processes**

A process is an MQ object that defines an application to the MQ Queue Manager.
**Process Characteristics**

- Process definition used to identify applications to be started by a trigger monitor
- The process definition includes application ID and type, plus some application specific data

---

**MQ Channels**

A channel is a communication link providing a path on the same or different platforms. The message channel is used for the transmission of messages from one queue manager to another, and shields the application programs from the complexities of the underlying networking protocols.
Channels

- Definitions that connect 2 queue managers
- Uni-directional
- A pair of MCA programs communicating with each other
- Transmission queue ties remote queue to channel
- Can control message delivery priority through use of multiple channels
Types of Channels

- **Message Channels:**
  - Sender - Receiver
  - Server - Requestor
  - Sender - Requestor
  - Server - Receiver

- **Client Channels:**
  - Server Connection Channel

- **Cluster Channels:**
  - Cluster Sender Channel
  - Cluster Receiver Channel

Supported Platforms

WebSphere MQ supports the following platforms:

- OS/390 MVS
- OS/390 Linux
- AIX
- HP-UX
- Solaris: Intel & SPARC
- Windows NT, 2000, XP

- Windows: 3.1, 95, 98
- OS/2
- Open VMS
- Tandem NSK
- VSE
- Digital UNIX
- Compaq Tru64 UNIX
MQ Platforms

**Servers:**
- DYNIX/Pix *
- Compaq Tru64 UNIX
- Compaq (Digital) VMS
- Compaq (Tandem) NSK
- Hewlett Packard HP-UX
- Hitachi
- AIX
- VSE/ESA & MVS/ESA & OS/390
- OS/2 Warp
- AS/400 (MPI & RISC)
- Linux (Intel)
- Windows NT/2000
- NCR (AT&T GIS) UNIX
- Siemens Nixdorf SINIX and DC/OSx
- SCO OpenServer *
- Stratus Continuum/400
- NUMA-Q (Sequent)
- SCO UnixWare *
- SGI *
- Solaris
- Unisys (ClearPath OS 2200)

**Clients:**
- Apple MacOS *
- Compaq Tru64 UNIX
- EMC DG/UX *
- Compaq VMS
- Hewlett Packard HP-UX
- Hewlett Packard HP 3000
- MPE/iX *
- Hitachi
- IBM AIX
- IBM MVS/ESA
- IBM OS/2 Warp
- IBM TPF
- IBM VM/ESA
- IBM 4690 OS **
- Java
- Linux (Intel & OS/390)
- Windows 3.1/95/98
- Windows 2000/NT
- NCR (AT&T GIS) UNIX
- Pyramid DC/OSx
- Siemens Nixdorf SINIX
- Siemens Nixdorf DC/OSx
- SCO OpenServer *
- NUMA-Q (Sequent)
- SCO UnixWare *
- SGI *
- Stratus VOS (Stratus)
- Stratus Continuum/400
- Sun Solaris (Sparc/Intel)
- Unisys A (MSS)

* Available from Willow Technology  ** Available from Commerce Quest

**Leaf Node Servers (limited functionality):**
- Windows 3.1/95/98
- IBM TPF

Security

Version 5 Release 3 offers added security using Secure Sockets Layer (SSL), the Internet standard for secure communication.
**Application Programming Interfaces (API)**

**WebSphere MQ Programming APIs**

- **Application Messaging Interface (AMI)**
  - High level of abstraction, moves message handling logic into the middleware
  - Policy name: "How"
  - Service name: "Where"
  - Available for C, C++, COBOL, Java

- **Java Message Service (JMS)**
  - Emerging Java standard underpinned by WebSphere MQ
  - Abstracts MQ details
  - Interface for J2EE/WebSphere

- **Message Queue Interface (MQI)**
  - Native calls to provided functions are available in the following languages: 390 Assembler, C, C++, COBOL, COM, LotusScript, Java, PL/1, VisualBasic
WebSphere MQ Programming APIs

Message Queue Interface (Procedural)

- 13 verbs
- 7 are used most commonly
- 6 have specialized use
- Important to understand use of call parameters

(Also Object Oriented access to MQ)
### Common MQI Calls

- MQCONN: Connect to Queue Manager
- MQOPEN: Open a queue
- MQPUT: Put message to queue
- MQPUT1: Put one message on a queue
- MQGET: Get message from queue
- MQCLOSE: Close a queue
- MQDISC: Disconnect from Queue Manager

### Specialized MQI Calls

- MQBEGIN: Signals start of Unit of Work
- MQCMIT: Commits Unit of Work
- MQBACK: Rollback Unit of Work
- MQINQ: Inquire on MQ object
- MQSET: Set queue attributes
- MQCONNX: Connect with special options
Bridges/Adapters/Connectors

MQ Adapters

From IBM
- CICS Bridges, IMS Bridge, SAP Bridge, Notes Bridge, ...

From 75 other vendors

More than 150 adapters
- databases, ERP, CRM, MOM, message brokers, packaged
- applications, transaction managers
This is an example of a possible EDI processing methodology using WebSphere MQ components.

**The components used are:**

- WebSphere MQ
- WebSphere – Web transaction monitor
- WebSphere MQ Integrator
- EDI Adapter – Builds EDI transformations and load into WMQI
- OCR application to handle Fax and move content to WebSphere MQ
Electronic Data Interchange Processing

Message Brokers
The Message Flow

A Message Flow is:

- A sequence of operations on a message
- Dependant upon message content

Message Brokering Example

Application A outputs:
- Account Id, Trader Id, Price, Quantity, Date, Customer Id

Parse Rules Transform

Application B outputs:
- Customer Id, Account Id, Trader Id, Price, Quantity, Date

Application 1 inputs:
- Date, Customer Id, Account Id, Price, Quantity, Trader Id

Application 2 inputs:
- Date, Price, Quantity, Trader Id

Application 3 inputs:
- Date, Customer Id, Price, Quantity
Publish/Subscribe

WebSphere MQ
Message/Queuing

Customer Benefits of MQ

- Wide selection of available platforms
- Network is transparent to the application programmer
- Applications can be changed quickly and easily in response to changing business needs
- Applications run in an asynchronous manner - parallelism
- Assured delivery of information - anywhere in the network
- Transactional messaging support for coordinated updating of multiple data sources
- Trusted, dependable for mission critical applications
An MQ Client is a component that can be installed on a separate machine from the Base product and Server.

MQ applications can run on client.

Client uses a server queue manager via a network protocol (SNA, TCP/IP, NetBIOS, DECNet, SPX).
**MQ Clients**

**Why Clients**

- **Supported on:**
  - PCs (MS-DOS, Windows 3.11/95/98/2000)
  - MacOS (Willow Technology)
  - Unix Workstations

- Reduces client hardware requirements

- Client license typically cheaper than server license
MQ Client Weakness

- Requires Real-time connection to Server
- What about Travelling Users?
- User maintains connection to MQ until MQDISC call is issued

Securing a WMQ infrastructure
Session Layout and Aim

- Define the scenario
- MQ specific security
  - Issues
  - Vulnerabilities
  - Threats
- Using cryptography to secure an MQ infrastructure

What Is Information Security?

- Information security is a process, not a product!
- Information security can be defined as the process of balancing risk to the business with the cost of implementation, management, and reward.
The Linear Security Process.

Why Do We Need Security in WebSphere MQ?

- Systems based on WebSphere MQ form the core of many businesses
- The data flowing is often of a high risk value
- The enterprise WebSphere MQ environment crosses many boundaries
- Many vulnerabilities in WebSphere MQ exist
MQ Vulnerabilities

- Access control policies
  - QM, Channels, Queues
- Network level security
- Application level security
- Clusters!
  - A QM joining a cluster inherits all the privileges from the repository
- Clients!
  - C/S connection is OPEN!
  - Use of MCAUSER may be difficult
  - Strong Authentication may be required

Cryptographic services

Modern Cryptographic services can help us most of the above issues.

CAVEAT:
The System Context bears functional consequences
Cryptographic services

- Privacy (Encryption)
- Integrity (no tampering)
- Authentication
- Non Repudiation
- Mutual Authentication (PEA)

Again: The System Context bears functional consequences
## E2E vs P2P Security

### Functional issues

<table>
<thead>
<tr>
<th>Service/Context</th>
<th>P2P</th>
<th>E2E</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEA</td>
<td>Appropriate and can be implemented. Since PEA is a protocol, P2P is the correct context for this functionality. The entity that is authenticated is the channel (node).</td>
<td>Cannot be implemented. Since the users and applications leave the communications to MQ, PEA does not make sense at this level.</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Appropriate and can be implemented. Messages in the queue are not encrypted.</td>
<td>Appropriate and can be implemented. Messages in the queue are encrypted.</td>
</tr>
<tr>
<td>Integrity</td>
<td>Appropriate and can be implemented. You are assured that messages are not counterfeited on the channel.</td>
<td>Appropriate and can be implemented. You are assured that messages are not counterfeited on queues or channels.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Appropriate and can be implemented. The authentic origin of the message is the original point of transmission (the sender channel or node).</td>
<td>Appropriate and can be implemented. The authentic origin of the message is the originating user or application process.</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>Not appropriate, but can be implemented. Non-repudiation is meaningful at the application level.</td>
<td>Appropriate and can be implemented. The originating user or application process cannot deny having sent the message.</td>
</tr>
</tbody>
</table>

### Management issues

<table>
<thead>
<tr>
<th>Service/Context</th>
<th>P2P</th>
<th>E2E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants in the communication</td>
<td>Channels or nodes</td>
<td>Users or application processes</td>
</tr>
<tr>
<td>Key managers</td>
<td>Site administrators, therefore suitably skilled persons.</td>
<td>Users: if you wish to guarantee non-repudiation the managers must be the users themselves, usually not highly skilled.</td>
</tr>
<tr>
<td>Number of participants</td>
<td>Equal or proportional to the number of nodes.</td>
<td>Significantly greater than the number of nodes.</td>
</tr>
<tr>
<td>Implementation costs</td>
<td>Low to very low. Deployment of a channel solution can be quantified as a few hours for each node involved.</td>
<td>Low to medium if a transparent solution is applicable. If code changes are required than the cost would range from high to very high. In some cases it would be impossible (the source may not be available)</td>
</tr>
<tr>
<td>Management costs</td>
<td>Same as for E2E (except for key management, which is greater in the E2E context on account of the greater number and lower skill level of participants).</td>
<td>Same as for P2P (except for key management, which is greater in the E2E context on account of the greater number and lower skill level of participants).</td>
</tr>
</tbody>
</table>
Implementing MQ Security

- Any modern and secure crypto implementation must be standard PKI based.
- Any modern crypto implementation must use the relevant standards (PKCS, S/MIME, X.509) to ensure robustness and interoperability be standard PKI based.
- The most relevant may be:
  - X.509 for certificates
  - PKCS#7 for digital signatures (and ciphers)
  - PKCS#11 for tokens (key management)

Implementing P2P Security

- Functionally equivalent to SSL
- Use the security exit for Principals Authentication (PEA)
- Use msg or S/R receive exit for Encryption, No tampering, Authentication
- Shut down the channel if PEA or Authentication fails; you may be under attack!
- LOG and Audit
Implementing E2E Security

- Use the API crossing exits whenever possible (WMQ 5.3)
- Use a robust, standard based crypto toolkit
- Refer to PKCS based, PKI compliant toolkits
- By using standards you leave yourself independent from vendors (!)
- Perform PKCS#7 - S/MIME operation when putting
- Perform symmetrical operation when getting
- Using API crossing exits allows for application transparent implementations, although it is somewhat a “lower level” security
- Use a commercially available product for transparent operation. Implement your own solution with Standard Toolkits

Example: How MQ works with IMS applications
MQ and IMS interfaces

- MQ offers two (2) interfaces to IMS Applications:
  - The IMS Adapter
  - The MQ-IMSBridge

MQ and IMS - The IMS Adapter

- The IMS adapter is the interface between IMS application programs and an MQ subsystem.
- It makes it possible for IMS application programs to use the MQ API (MQI).
- The adapter supports a two-phase commit protocol for changes made to resources owned by MQ with IMS acting as the syncpoint coordinator.
**MQ and IMS - The IMS Adapter**

- The IMS adapter provides access to MQ resources for programs running in:
  - Task (TCB) mode
  - Problem state
  - Non-cross-memory mode
  - Non-access register mode

- The adapter provides a connection thread from an application task control block (TCB) to MQ.

**MQ and IMS - The IMS Bridge**

- Component of MQ for OS/390 that allows direct access from MQ applications to applications on your IMS system
  - Enables implicit MQI support

- An IMS *Open Transaction Manager Access* (OTMA) client.
The MQ-IMSBridge: Benefits

- You can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by MQ messages.
- You do not have to rewrite, recompile, or re-link your IMS programs.

How do MQ applications use the IMSBridge to access IMS applications?

- In bridge applications there are no MQ calls within the IMS application.
- The IMS application gets its input using a GET UNIQUE (GU) to the IOPCB.
- Sends its output using an ISRT to the IOPCB. MQ applications using the IMS header (the MQIIH structure) in the message data to ensure that the applications can execute as they did when driven by nonprogrammable terminals.
How do MQ applications use the IMSBridge to access IMS applications?

- MQ applications put the message to a “special” MQ queue
- The messages contain IMS transaction data
- The storage class of the MQ queue determines whether the queue is an **OTMA queue** *(contains the XCF Group Name and IMS Target System name)*
- Once the message is written to the bridge queue, an IMS transaction can be started

The MQ-IMSBridge Flow
What is OTMA?

- IMS OTMA (Open Transaction Manager Access)
- The IMS OTMA facility is a transaction-based connectionless client/server protocol
- Runs on IMS Version 5.1 or later. It functions as an interface for host-based communications servers accessing IMS TM applications through the OS/390 Cross Systems Coupling Facility (XCF).
- How do you connect a client to a server so that the client can support a large network (or a large number of sessions), while maintaining high performance?

What is OTMA?

- MQ as a client can be used to support a large IMS network supporting several sessions simultaneously
- OTMA is implemented in an OS/390 sysplex environment.
- The domain of OTMA is restricted to the domain of MVS Cross-coupling facility (XCF).
- XCF is the transport layer of OTMA
Benefits of OTMA?

- Full-duplex processing provides an environment in which transactions and output messages are sent and processed in parallel.
- You can implement IMS device support outside IMS.
- You can also implement device support for your IMS subsystem that is different from what IMS provides, or enable device support that IMS does not provide.

Benefits of OTMA?

- Solves IMS problem:
  - How do you connect a client to a server so that the client can support a large network (or a large number of sessions), while maintaining high performance?
- Supports several clients by enabling them as gateway to IMS applications (MQ for OS/390, OEM, TCP/IP, DCE/RPC, and other IBM applications)
**MQ as client of OTMA**

- MQ as a client can connect to IMS in a high performance manner to support a large IMS network.
- OTMA is implemented in an OS/390 sysplex environment.
- The domain of OTMA is restricted to the domain of XCF.

**IMS Message Flow in an OTMA Environment**

![Diagram depicting IMS Message Flow in an OTMA Environment]

The Transaction Pipe is created dynamically when OTMA receives output.