#SHAREorg

SHARE
Technology · Connections · Results

Introduction to WebSphere MQ

Chris J Andrews
IBM

Monday 4th February, 2013
Session 12900

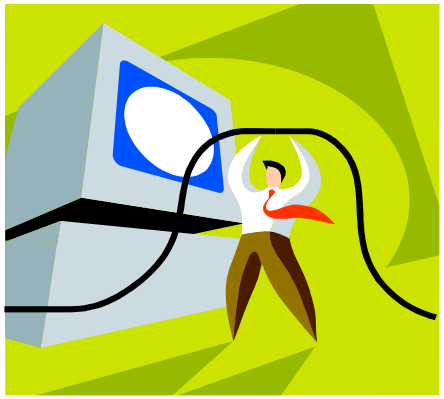
SHARE in San Francisco 2013

Agenda

- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Introduction – why use WebSphere MQ?

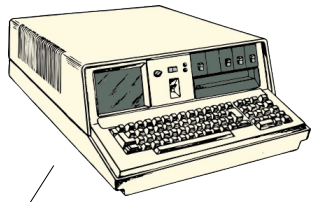
- Over time, separate organisational units build their own pieces of business logic…
- …with applications developed on many different platforms.



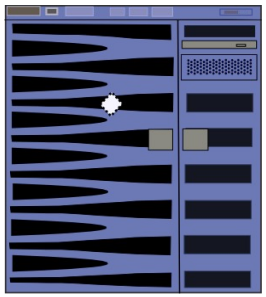
- Connecting these and managing them together can save time and money
- **WebSphere MQ can help achieve this.**

Business Challenge 1 - Universal Connectivity

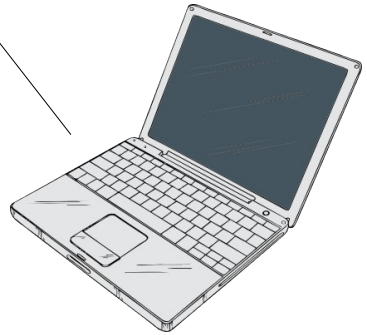
Payroll have a program to run to add a one-time payment to an employee's pay packet



HR have a program to calculate an employee's performance bonus based on their annual review score and business unit's performance



Research have a program to calculate annual review scores



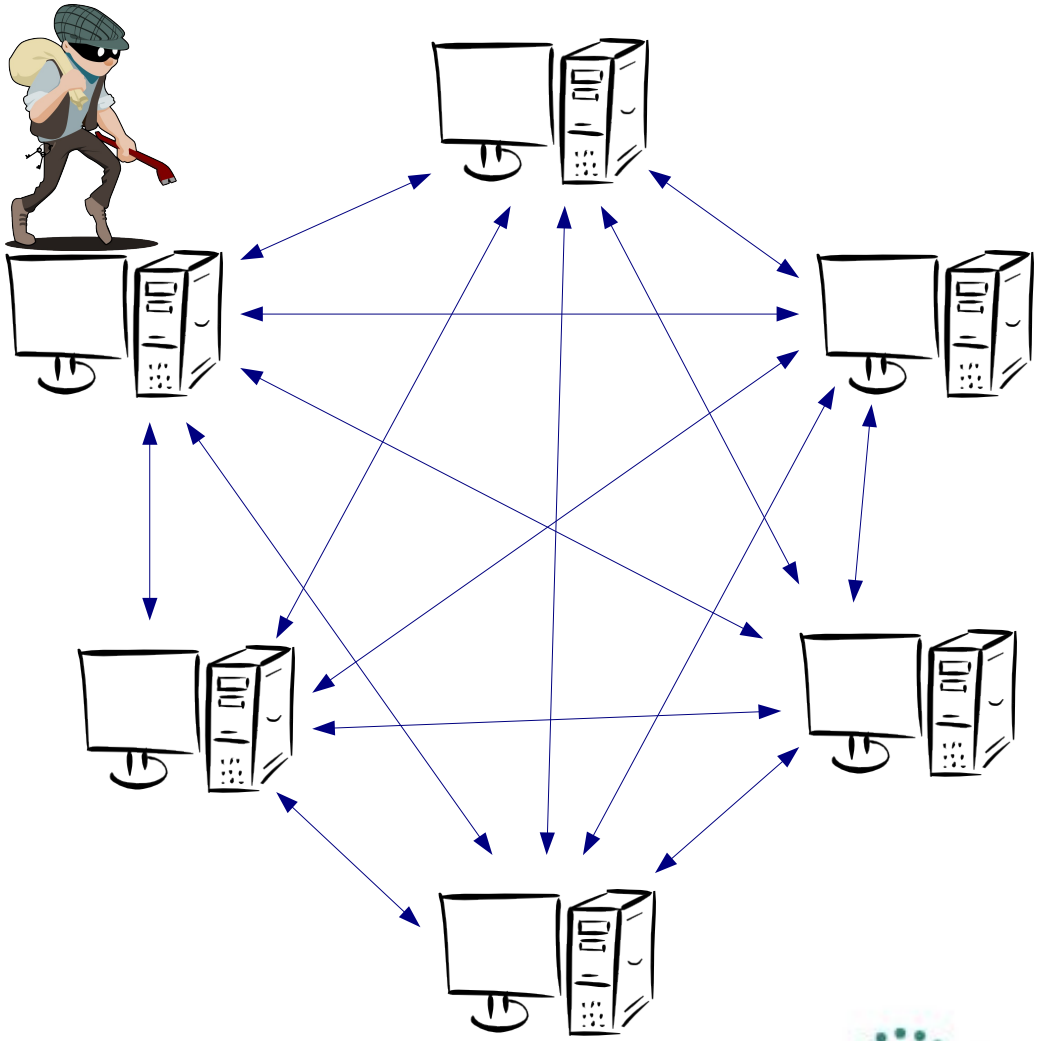
Sales have a program to calculate annual review scores

These applications run on different hardware and OS and be written in different programming languages. We want to connect the applications together in a time and cost efficient manner.

Business Challenge 2 - Process Resilience

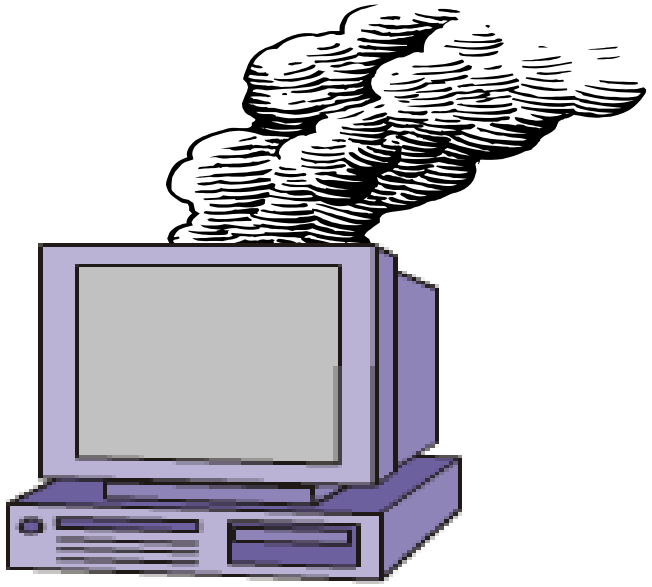
As systems become more tightly coupled, their reliance on each other increases. The cost of failure of a process increases.

- **The risk of failure can be reduced by:**
  - Removing dependencies
  - Introducing redundancy
  - Assuring data delivery
  - Providing robust security
- **WebSphere MQ can help achieve this.**



Business Challenge 3 – Process Scalability

- Many applications and processes start out on a single system.
- The business grows, and the capacity of the system can no longer cope with the workload demand.



- A scalable architecture enables the capacity to be incrementally grown to meet increasing workloads
- **WebSphere MQ can help achieve this.**

Business Challenge 4 – Flexibility

- A process was originally designed for one purpose...
- … It then needed to change to meet new requirements

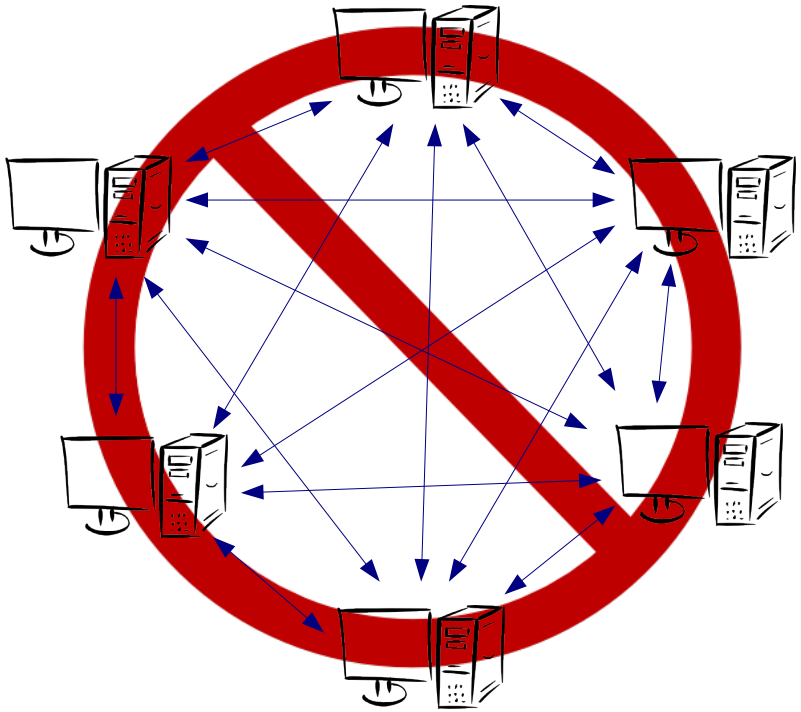
- Being able to respond rapidly to internal and external challenges by rapidly modifying existing services gives a competitive advantage.
- **WebSphere MQ can help achieve this.**

Agenda

- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Fundamentals of WebSphere MQ

- **Reliability**
  - Assured message delivery
  - Performance
- **Ubiquitous**
  - Breadth of support for platforms, programming languages and API
- **Loose application coupling**
  - Location transparency
  - Time independence
  - Data transparency (with WebSphere Message Broker)
  - Platform independence
- **Scalability**
  - Incremental growth
- **Rapid development**
  - Standards
  - Reduce Complexity
  - Ease of use



The vision for WebSphere MQ is that it provides a range of capabilities, making it suitable to be a transport backbone across all environments in an IT Infrastructure.

WebSphere MQ does not provide all these capabilities today. It evolves with new technologies as they develop and become widely adopted.

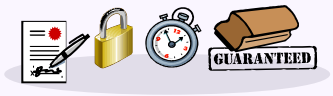
Universal Messaging Backbone

Skills

- Languages
  - *COBOL, C/C++, RPC*
  - *Java, JEE, JMS*
  - *.NET, C#, VB, WCF*
  - *AJAX, Perl, Python…*
- Orientations
  - *Service*
  - *Batch*
  - *File*
  - *Message*
  - *Resource…*
- Mindsets
  - *WSDL, XML, WS-**
  - *REST, MEST, KISS*

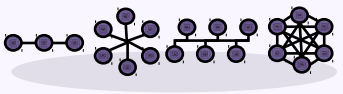
End-Points

- Vendor Platforms
  - *JEE, .NET, etc*
- Operating Systems
  - *Exploitation & Support*
- Applications
  - *SAP, Siebel, etc…*
- Devices
  - *Mobile, Wireless, PoS, Sensor, Actuator, RFID…*
- Web services
  - *SOAP, WSDL, WS-RM, WS-N…*
- Web 2.0
  - *HTTP, AJAX, REST,…*
- Appliances



Qualities-of-Service

- Transactional
- Guaranteed
- Persistent
- At-Most-Once
- Replay
- At-least-once
- Best-Effort
- Fire-and-Forget
- Request-Reply
- Fastest speed
- Lowest Latency



Delivery Styles

- Client-Server
- Backbone
- Point-to-Point
- Peer-to-Peer
- Publish/Subscribe
- Grid
- Bus
- Multicast
- Unicast

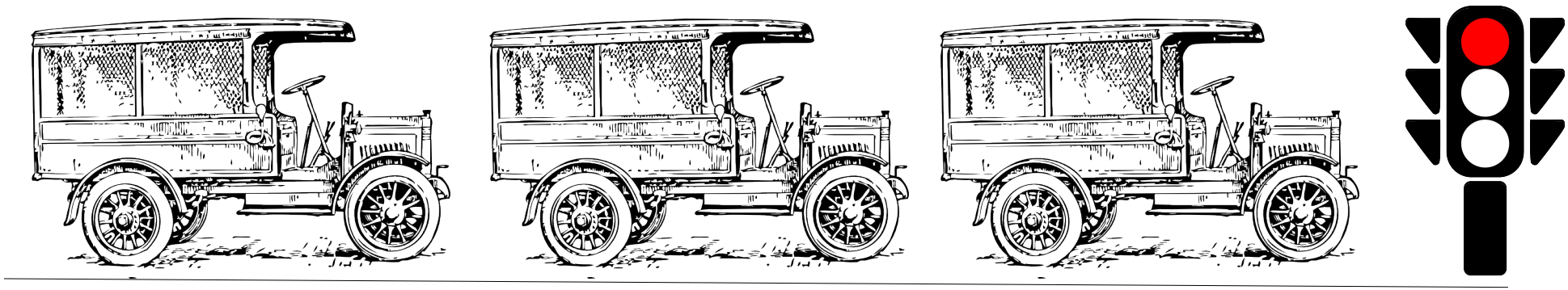
WebSphere MQ is not a substitute for:

- Well written applications
- Robust network
- Good operational procedures
- Well managed systems

What is Asynchronous Messaging?

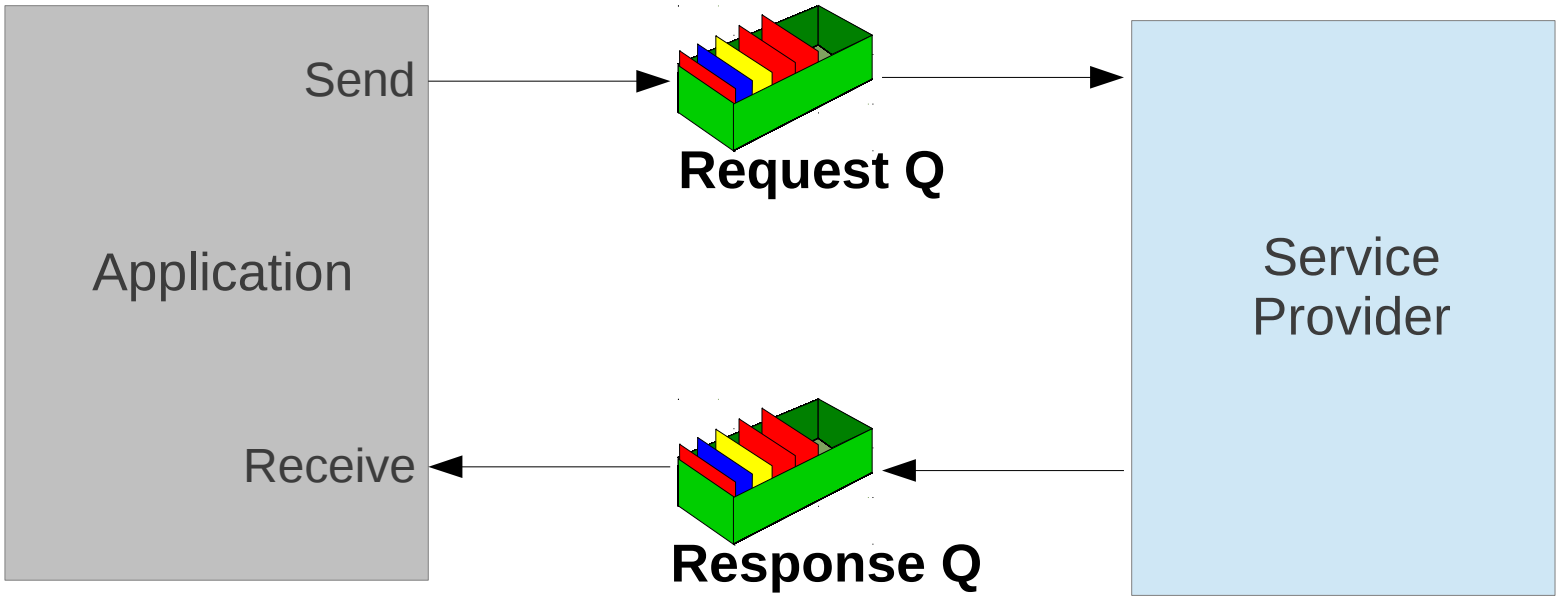
- Paradigm 1: Point to Point
- Paradigm 2: Publish Subscribe

Messaging Paradigm 1: Point to Point Messaging



- FIFO – First In, First Out
- One object in, one object out

Asynchronous Messaging – Point to Point



- Messages can be created from many sources:
  - Data, Messages, Events, Files, Web service requests / responses

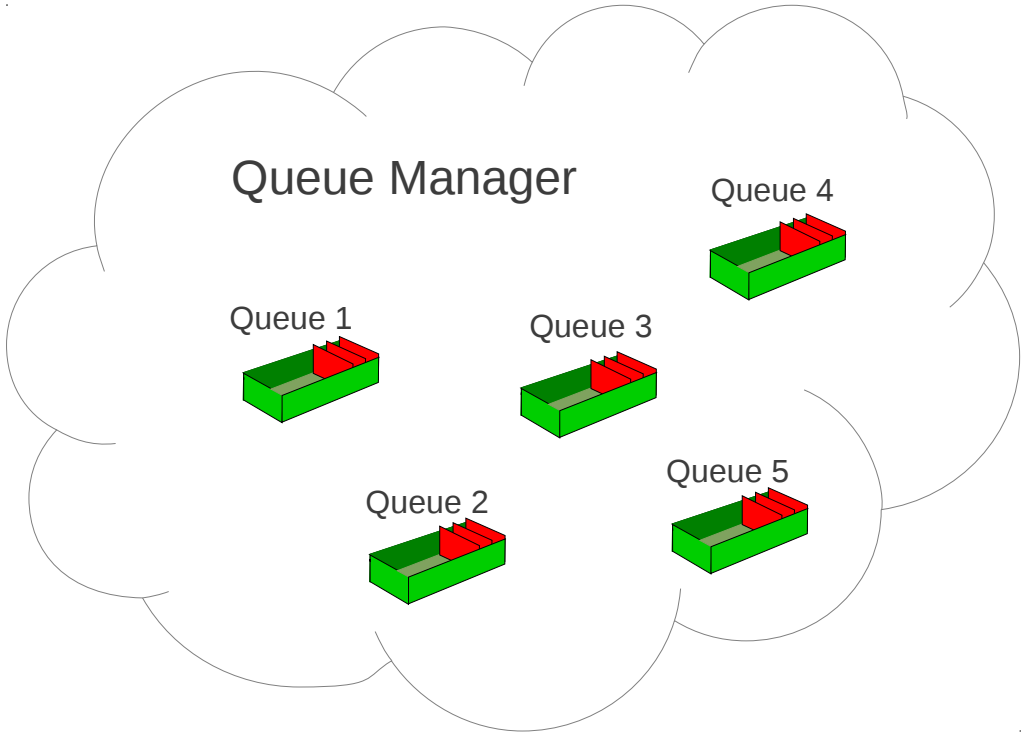
WebSphere MQ – Point to Point Messaging

- The physical world is frequently organised in queues. Consider for a moment just how many queues you have been involved in today alone. We queue at the Post Office, Supermarket checkout, at traffic lights. We write shopping lists and to do lists. We use the postal service, voice mail, and of course, the ever present e-mail.
- The truth is that queuing is a natural model that allows us to function efficiently. Perhaps not surprisingly therefore it turns out that it is also a very useful model in which to organise our applications.
- Instead of application A talking synchronously to Application B have Application A 'send a message' to a queue which Application B will read.
- Messages can be of any form, the content is not restricted, so they could contain:

  - General data
  - Data packaged as messages
  - It might be notification events
  - Files being moved in a Managed File Transfer application
  - SOAP messages for invoking services

What is a Queue?

- A queue holds messages
  - Various Queue Types
    - Local, Alias, Remote, Model
- Queue creation
  - Predefined
  - Dynamically defined
- Message Access
  - FIFO
  - Priority
  - Direct
  - Selected by Property (V7+)
  - Destructive & non-destructive access
  - Transacted
- Parallel access by applications
  - Managed by the queue manager

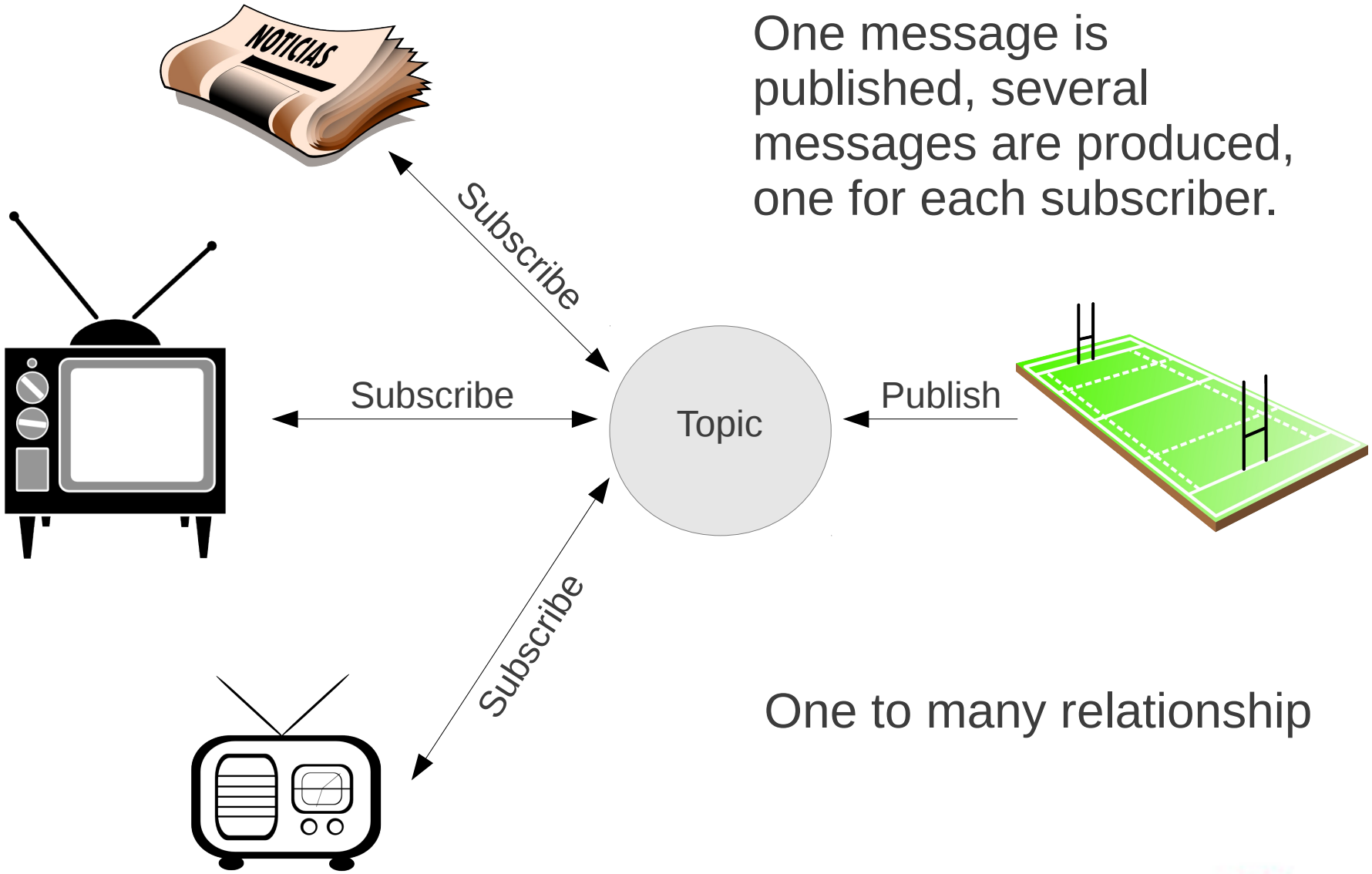


What is a Queue?

- A Queue is a named object (up to 48 characters) which is defined with a queue type.
  - Local — Only queue type which can actually hold messages
  - Alias — A queue name which 'points' to another queue
  - Remote — A queue which 'points' to a queue on a remote machine
  - Model — A template definition which when opened will create a local queue with a new name

- Applications open queues, by name, and can either put or get messages to/from the queue. Messages can be got from the queue either in FIFO order, by priority or directly using a message identifier or correlation identifier.
- As many applications as required can open a queue either for putting or for getting making it easy to have single server responding to many clients or even n servers responding to many clients.
- A key feature of WebSphere MQ is its transaction support. Messages can be both put and got from queues in a transaction. The transaction can be just local, involving just messaging operations or global involving other resource managers such as a database. A classic example, is an application which gets a message, updates a database and sends a reply message all within a single transaction. If there is a failure before the transaction commits, for example a machine crash, both the database update and the received message will be rolled back. On machine restart the request message will still be on the queue allowing the application to reprocess the request.

Messaging Paradigm 2: Publish / Subscribe



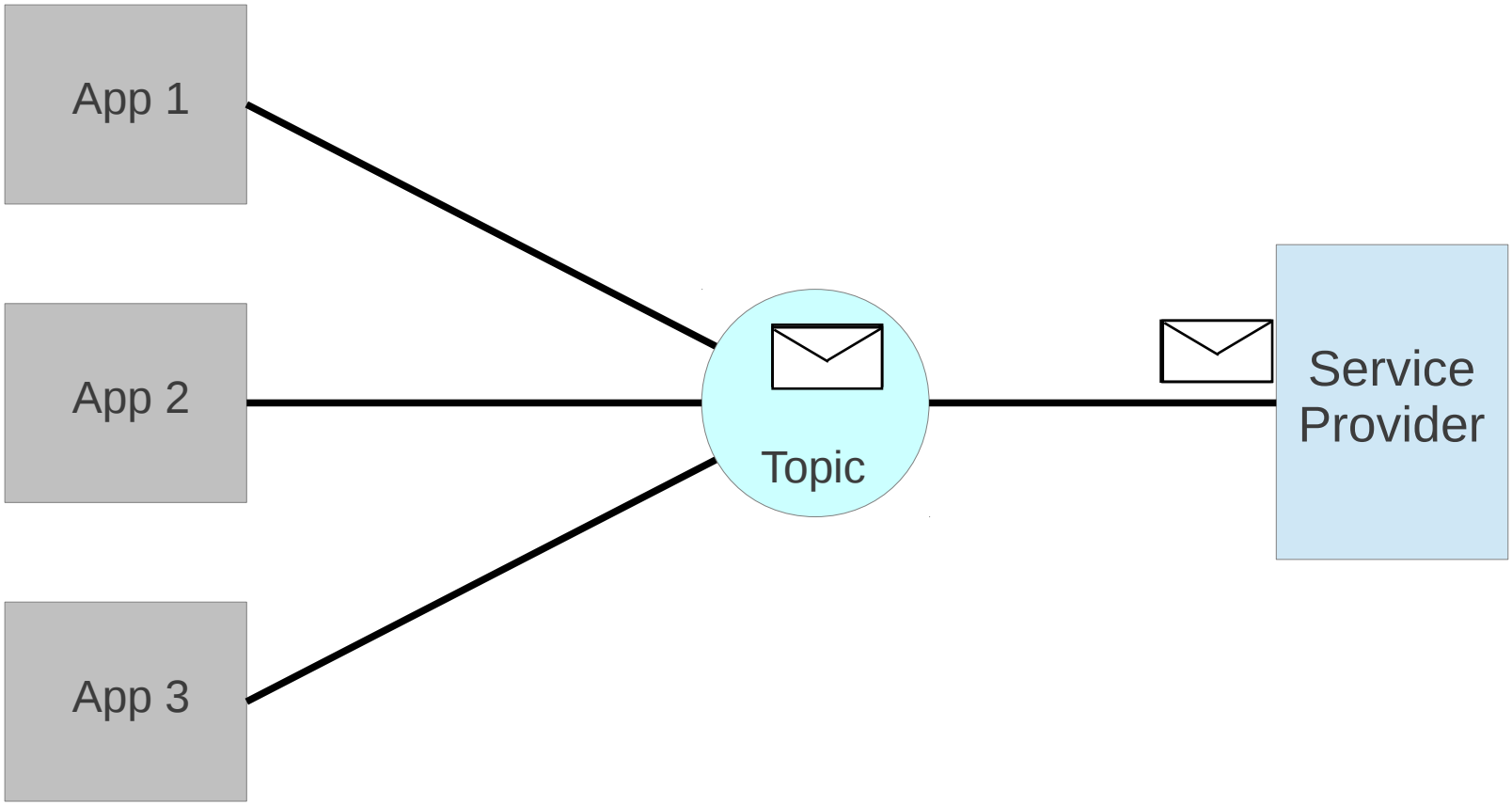
One message is published, several messages are produced, one for each subscriber.

One to many relationship

WebSphere MQ – Messaging Topology 2

- Our daily life is full of examples of requests for information on a given topic and providing information about a given topic.
-
- Let us consider an example:-
-
- You have installed a piece of software on your computer and you would like to know when there are updates available for it. The software provider has a service to inform you of updates when they occur.
- You ask the software provider to let you know when there are updates available for the software (a topic) in which you are interested (a subscription)
- The software provider informs you when updates become available (a publication)
- The software provider can use a message to provide this information
- You can provide a destination (queue) to which the information is published

Asynchronous Messaging – Publish Subscribe



What is a Topic?

- A Topic is defined by a "**Topic String**". This is a case sensitive character string, where the following characters have a special meaning:
  - '/' The topic level separator – provides structure to topic trees
  - '#' The wildcard character
  - '+' The single-level wildcard character

Example:

Price/Fruit/Apple

The Topic can be defined in a number of ways:

- Predefined by the MQSC command
- Predefined by the PCF interface (as used by the WebSphere MQ Explorer)
- Subscribing or Publishing to the Topic object

Topic Trees

By arranging Topic strings in a tree hierarchy, a 'Topic Tree' is created.

Every node in the tree is a Topic.

Topic Trees provide two benefits:

- Wildcard characters can be used to subscribe to multiple Topics.
- Security policies can be established

For example, to subscribe to both Topics:

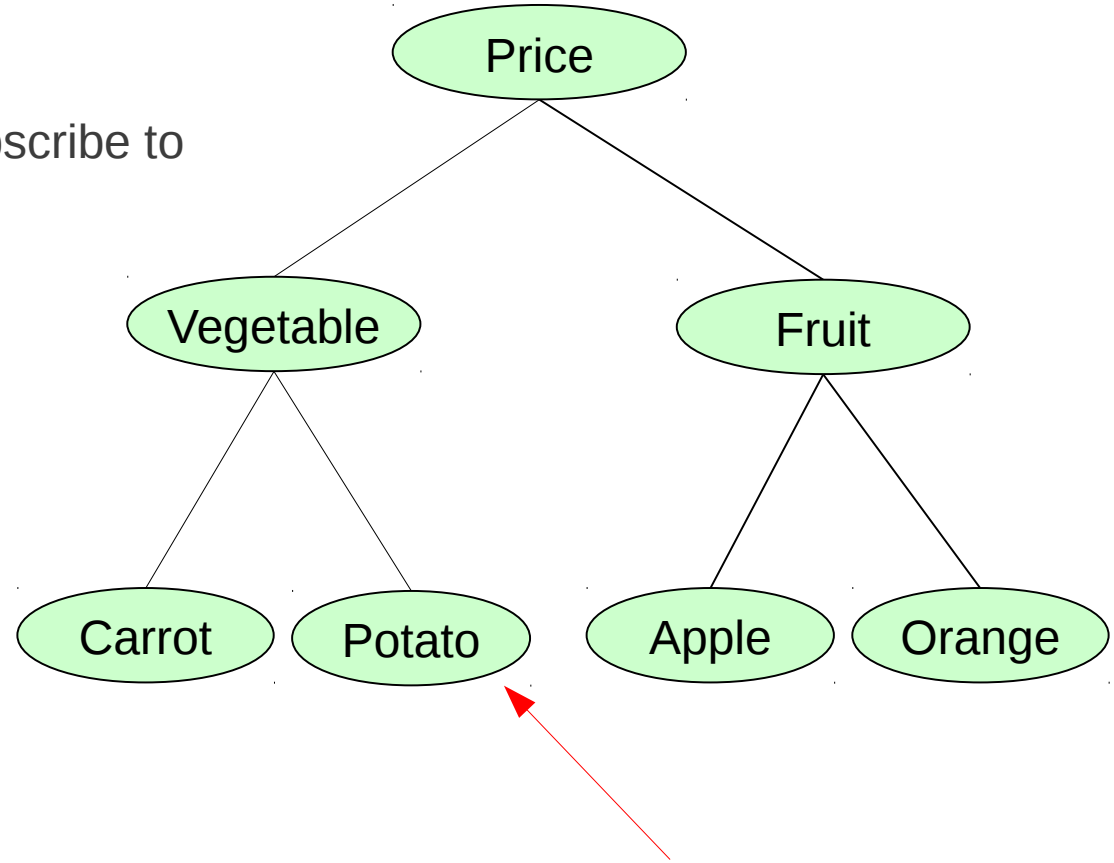
Price/Fruit/Apple
Price/Fruit/Orange

The subscription string is:

Price/Fruit/+

Note this is different to the subscription string:

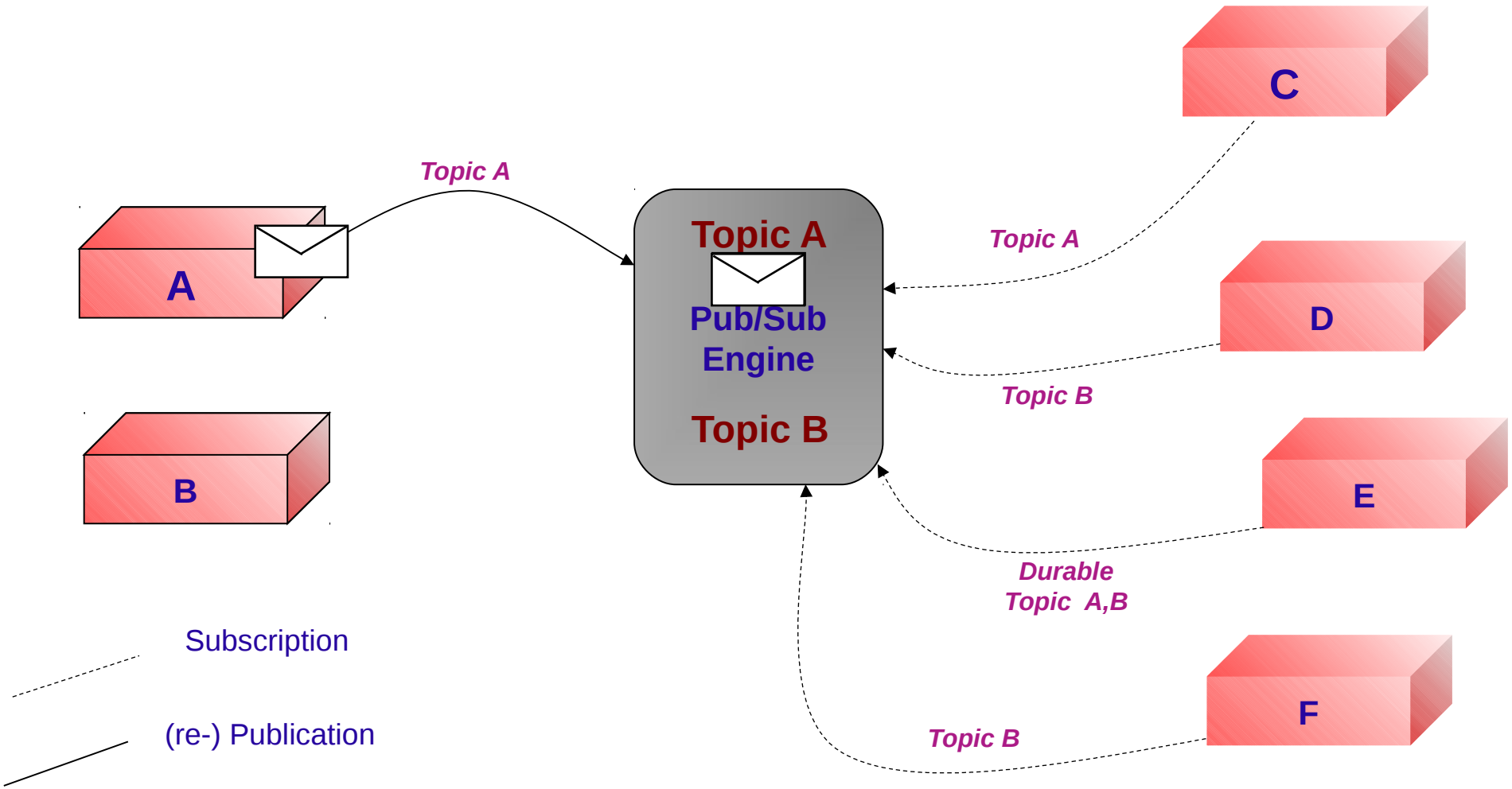
Price/Fruit/#



What is a Topic?

- A topic is a character string that describes the nature of the data that is published in a publish/subscribe system. Topics are key to the successful delivery of messages in a publish/subscribe system. Instead of including a specific destination address in each message, a publisher assigns a topic to the message. The queue manager matches the topic with a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.
- Note that a publisher can control which subscribers can receive a publication by choosing carefully the topic that is specified in the message.
- Each topic that you define is an element, or node, in the topic tree. The topic tree can either be empty to start with or contain topics that have been defined by a system administrator using MQSC or PCF commands. You can define a new topic either by using these create topic commands or by specifying the topic for the first time in a publication or subscription.
- Although you can use any character string to define a topic's topic string, choose a topic string that fits into a hierarchical tree structure. Thoughtful design of topic stings and topic trees can help you with the following operations:
  - Subscribing to multiple topics.
  - Establishing security policies.
- Although you can construct a topic tree as a flat, linear structure, it is better to build a topic tree in a hierarchical structure with one or more root topics.
- Topics can be defined by a system administrator using MQSC or PCF commands. (Topic objects)
- However, the topic of a message does not have to be defined before a publisher can use it; a topic is created when it is specified in a publication or subscription for the first time.

Durable Publish/Subscribe in Action



What is Publish/Subscribe?

- In this environment, the receiving applications notify an intermediary of their interest in particular sets of information. The receiving (or subscribing) application provides a subject and a queue where messages matching this subject may be delivered.

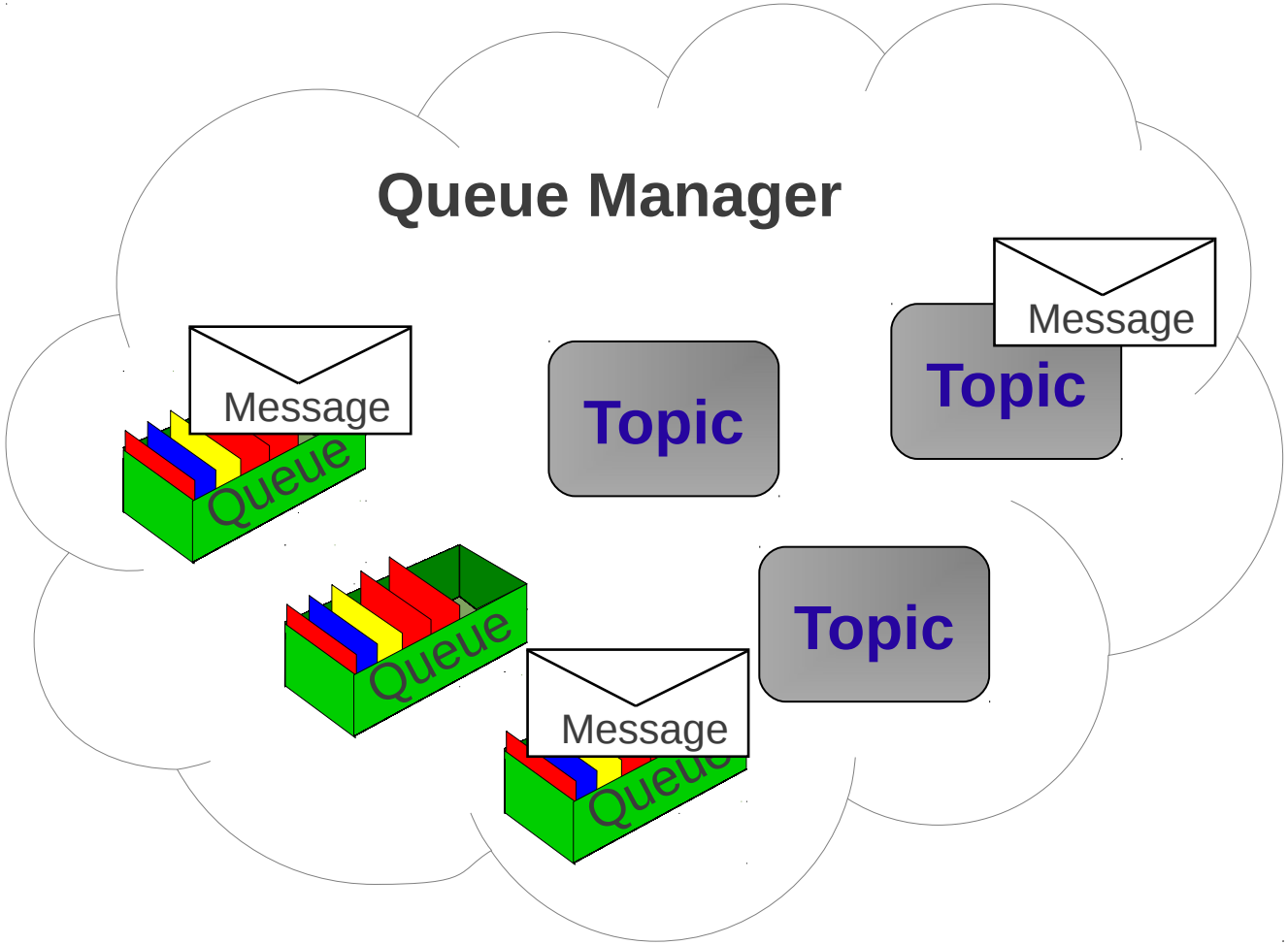
- The sending (publishing) applications generate information, together with a subject name, and sends the information to the pub/sub engine. The pub/sub engine contains a matching service which determines the subscribing applications interested in receiving this information.

- Note that the publish/subscribe model provides for the situation where a message may be published by an application using a subject which has no subscribers. In this instance the message data is discarded.

- There are many publish/subscribe products available in the marketplace today. MQ publish/subscribe differentiates itself by providing support for the publish/subscribe model and combining it with the exactly once delivery model of MQ message/queuing.

WebSphere MQ Queue Manager

The Queue Manager is the process which controls the storage and flow of messages



What is a Message?

Message = Header + User Properties + User Data

A Series of Message Attributes
Understood and augmented by the Queue Manager
- Message Id
- Correlation Id
- Routing information
- Reply routing information
- Message priority
- Message codepage/encoding
- Message format

....etc.

- User Properties require WMQ V7
  - Emulated for JMS in older versions of WMQ
- Arbitrary properties
  - For example, this is a “green” message

- Any sequence of bytes
- Private to the sending and receiving programs
- Not meaningful to the Queue Manager

- Message Types
  - Persistent ... recoverable
  - Non Persistent
- Up to 100MB message length

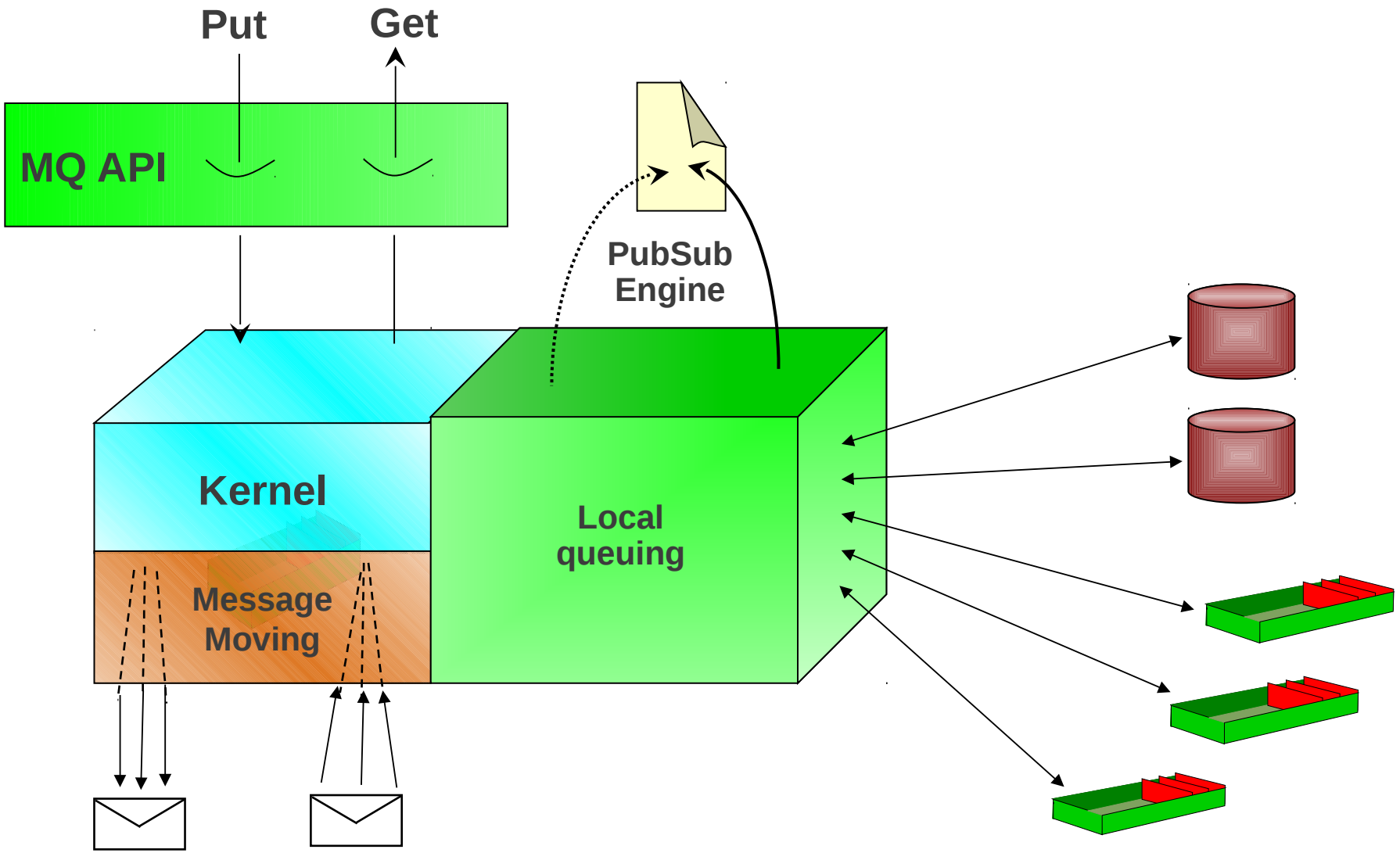
What is a Message?

- A message in WebSphere MQ is merely a sequence of bytes in a buffer of a given length. The current products support up to 100MB in a single message although the vast majority of messages are in the order of a few thousand bytes.

- Messages have various attributes associated with them such as their identifier, their priority and their format. Each application is free to define its own format for messages although there are a number of predefined formats. One common format for messages is XML for example.

- A key attribute of a message is its persistence. A message is either persistent or non-persistent. This attribute tells the Queue Manager how important the message is.
  - Persistent: persistent messages are written to disk and are logged. The Queue Manager will ensure that the messages are recovered in the case of a system crash or network failure. These messages are delivered once and only once to the receiving applications.
  - Non-persistent: The messages are identified by the application as non-critical. The Queue Manager will make every effort to deliver these messages but since they are not necessarily written to disk they will be lost in the case of a system crash or network failure. Clearly with no disk IO involved these messages are much faster than persistent ones.

The Queue Manager



What is a Queue Manager?

- A queue manager may - generally - be thought of as 3 components:

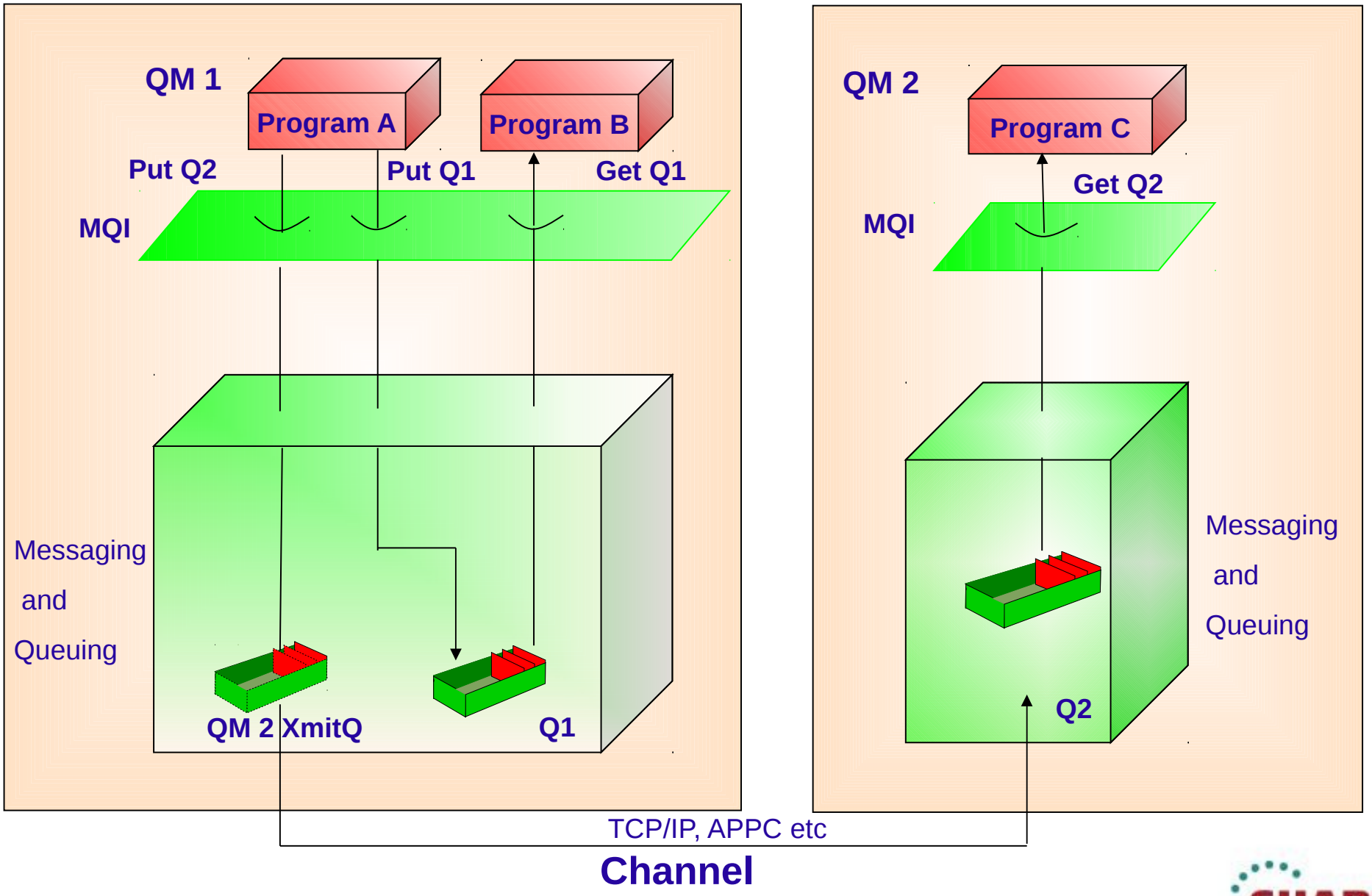
- The Kernel is the part of the queue manager that understands how to implement the MQ APIs. Given that the APIs are common across the queue manager family, it stands to reason that the Kernel is mostly common code across the set of queue managers. (The primary exception to this is the z/OS queue manager where the same functions are implemented differently to support the same APIs).

- The Local Queuing component is the part of the queue manager responsible for interacting with the local operating system. It manages memory, the file system and any operating system primitives such as timers, signals, etc. This component insulates the Kernel from any considerations of how the underlying operating system provides services and so enables the Kernel to be operating system independent.

- The Message Moving component is responsible for interacting with other queue managers and with MQI clients. For environments where all of the message queuing activity is local to a system then this component is unused - though this is a very rare case.

- The message moving functions are provided by specialised MQ applications, called Message Channel Agents.

Local and Cross-System Communication with WMQ



Cross-System Communication with WebSphere MQ

- In the diagram we see Program A sending messages to two other programs.

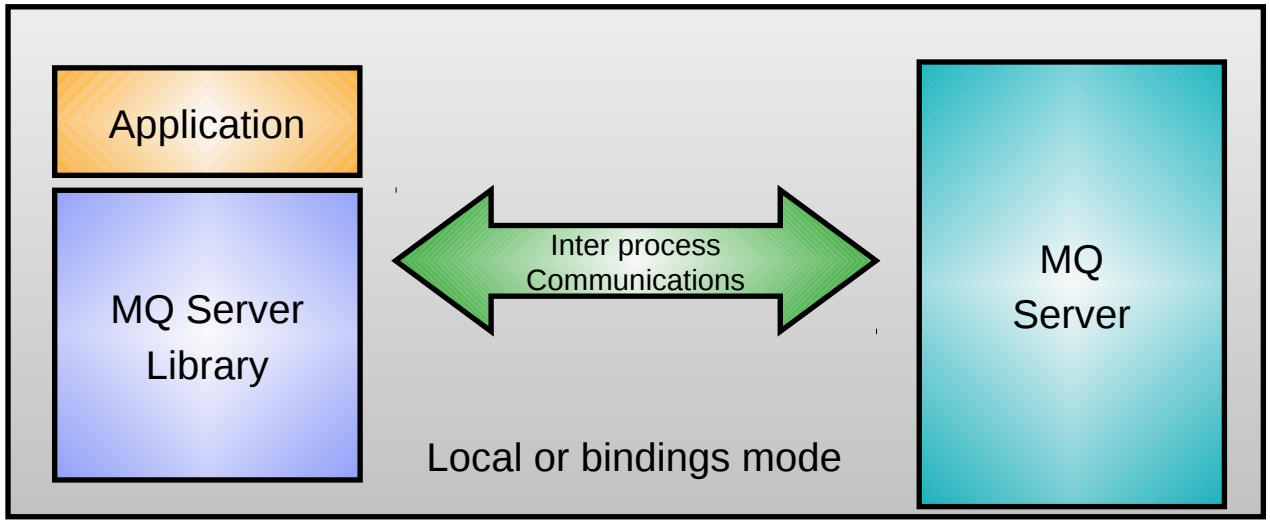
- To Program B: in this case the actual physical queue that both applications access are the same. This therefore does not require any network communication.

- To Program C: in this case Program A wants to put a message to queue Q2 on Queue Manager QM2. It can't do this directly without requiring that the network and QM2 Queue Managers are available so instead the message is put to a 'holding' queue called a transmission queue. Asynchronously, another part of WebSphere MQ called a channel will read this transmission queue and deliver any messages to the queues on QM2.

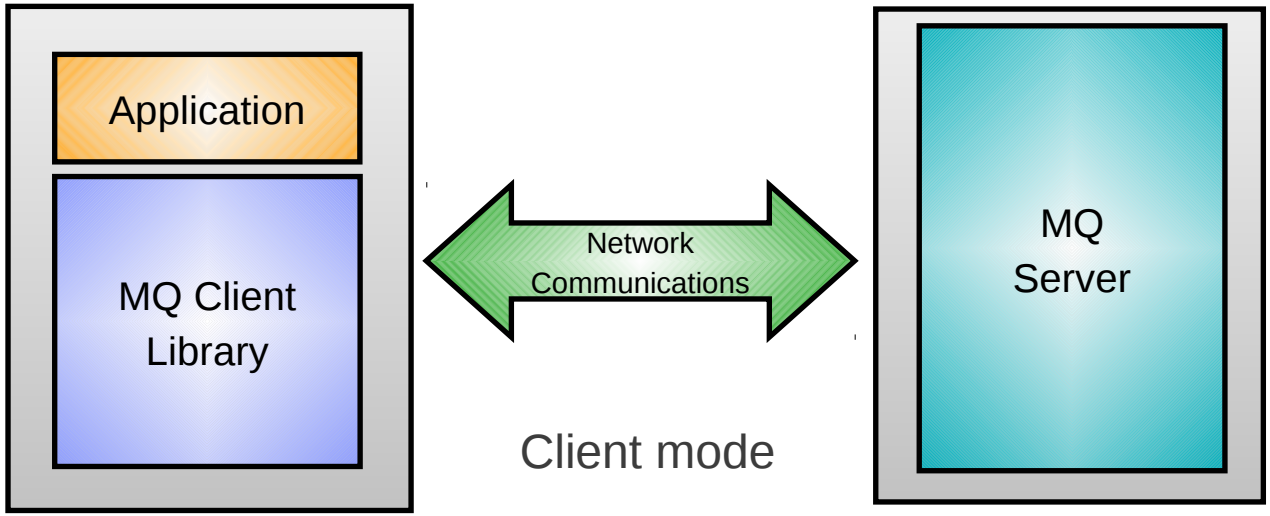
- Any number of applications running on QM1 can send messages to QM2 via the same transmission queue and channel.

Communicating with the Queue Manager

Server Model



Client Model



Application code is independent of the client to queue manager connection mode

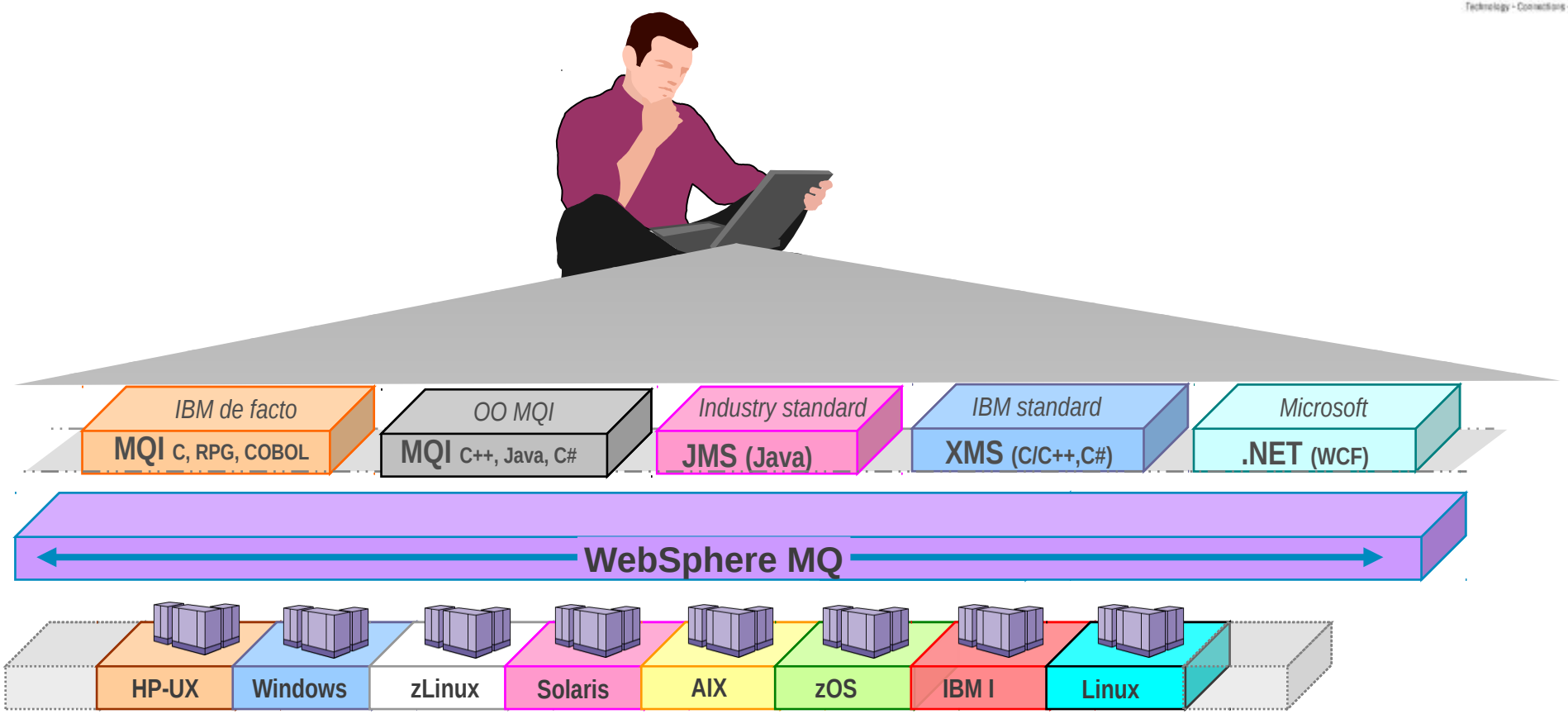
What is an MQ Client?

- WebSphere MQ clients provide a low cost, low resource mechanism to gain access to MQ facilities. The client provides a remote API facility, allowing an WebSphere MQ application to run on a machine that does not run a queue manager.
- Each MQ API command is passed to a Server queue manager where a proxy executes the required API command. The connection between client and server is entirely synchronous providing an 'rpc-like' mechanism - though NO regular (well-known) rpc mechanism is used !
- The client machine does not own any MQ resources - all resources are held by the Server machine. Thus, if local queuing capability is required then a server (rather than a client) must be used.
- The WebSphere MQ Client support is part of the WebSphere MQ product that can be installed and used separately from the MQ server. It provides a set of libraries which can be linked with your applications to provide access to WebSphere MQ queues without requiring the application to run on the same machine as the queues.
- Generally speaking an application is linked either with the client libraries or with the server libraries (often called ‘local’ or ‘bindings’ mode). In bindings mode the application communicates with the Queue Manager via an inter-process communications link of some kind. In client mode the application communicates via a network connection. However, as can be seen from the diagram, the two models are logically equivalent. For this reason the functionality provided at the client is almost identical to that provided by local applications.

Agenda

- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Programming API



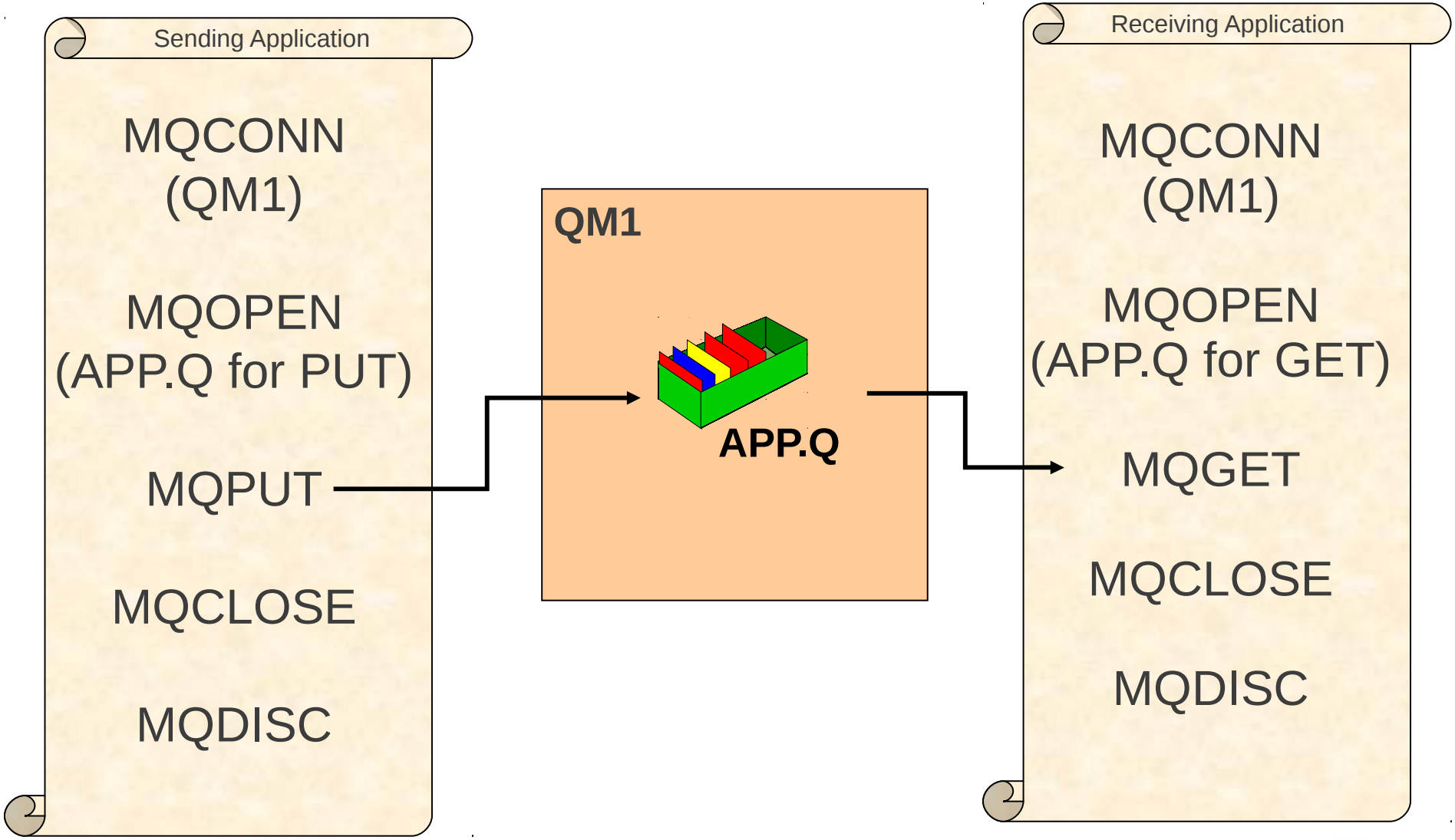
- Broad support for:
  - programming languages, messaging interfaces, application environments and OS platforms.

Programming API

- One of WebSphere MQ key strengths is its breadth. It can run on virtually any commercially available platform and is accessible through a wide number of programming languages and API. The MQ Interface (MQI) is the de facto API for MQ, providing simple common access across all platforms. It has both a procedural implementation and an object oriented implementation.

- Standards based interfaces such as JMS, and its IBM equivalent for C, C++ and .NET, XMS are also available.

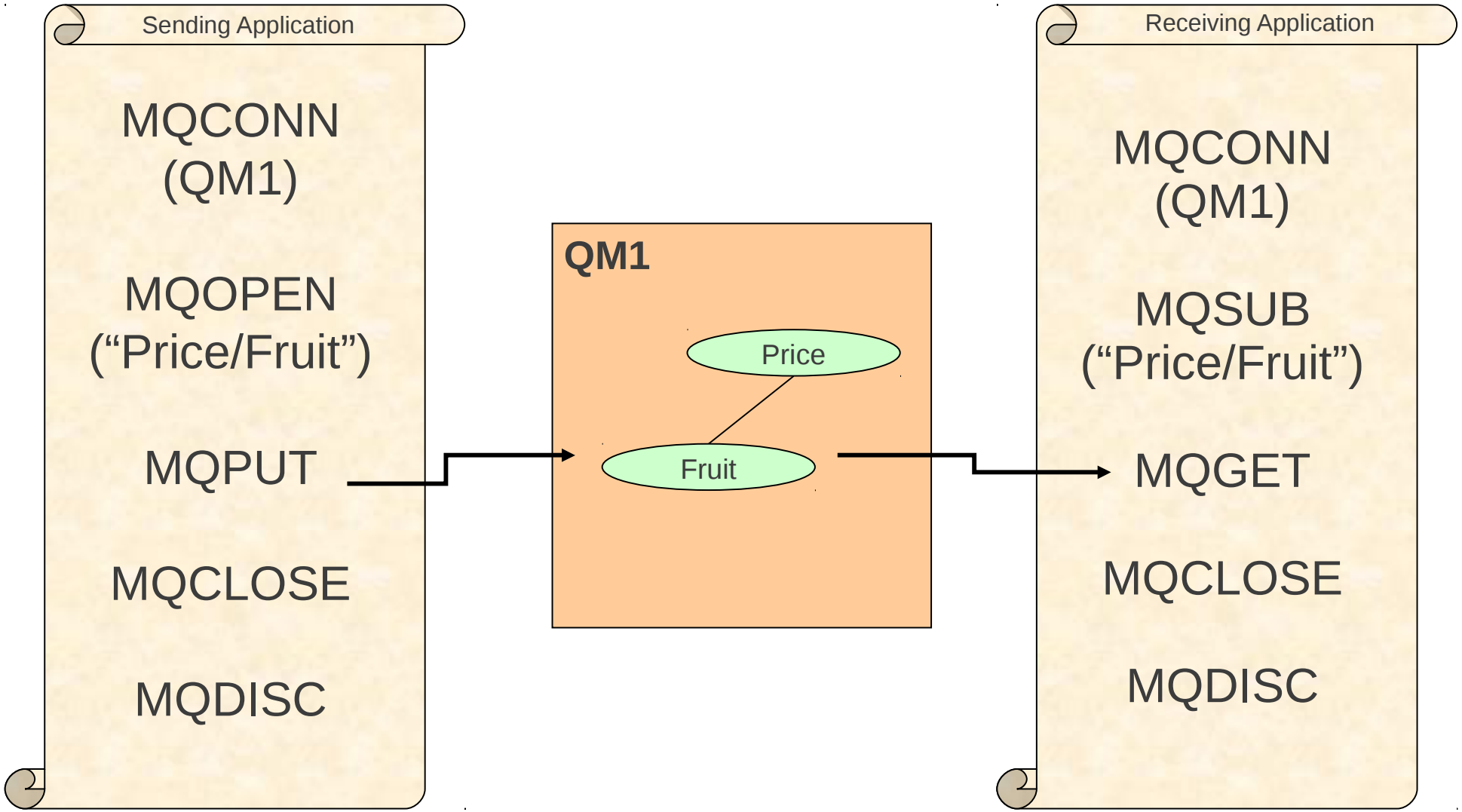
The WebSphere MQ API (MQI)



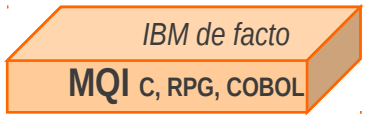
The WMQ API (MQI)

- The most common verbs are MQOPEN, MQCLOSE, MQPUT and MQGET which are concerned with the processing of messages on queues. The first example shows an application putting a message to a queue and another getting the message off the queue. We refer to this as the Point-to-Point application model. The second example shows an application publishing a message to a topic and another subscribing to messages about that topic. We refer to this as the Publish/Subscribe application model.
- There are many, many options associated with these verbs. However, in general, most of these options may be left to take their default values - and MQ provides a set of default structures to allow for easy assignment of these default values.
- There are 24 verbs in total in the WebSphere MQ API, known as the MQI. We have briefly illustrated the most common ones. The rest have less frequent use and we have summarised them in a table.
- To use the MQ verbs in your application you link with the MQ library provided with MQ, which will send your call to the MQ queue manager to process.

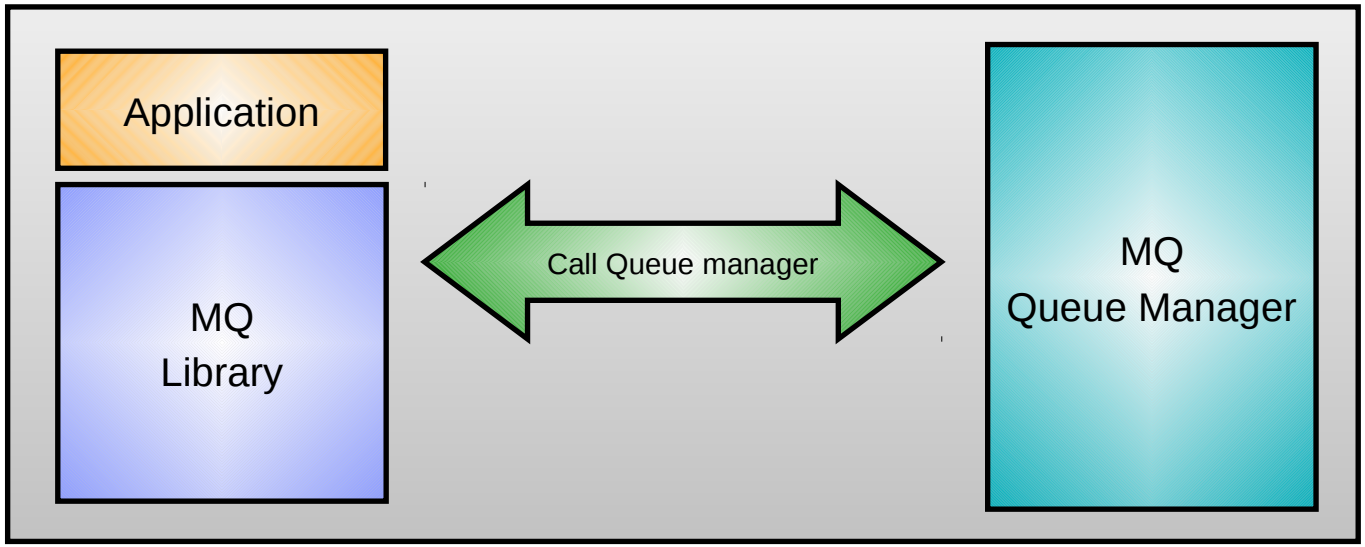
The WMQ API (MQI) – Publish/Subscribe



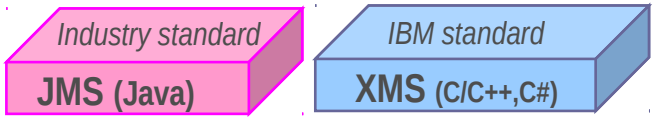
The WebSphere MQ API (MQI) – Summary of all verbs



Connection	Resource Use	Messages	Object attributes	Transactions	Message Properties
MQCONN	MQOPEN	MQPUT	MQINQ	MQBEGIN	MQCRTMH
MQCONNX	MQSUB	MQPUT1	MQSET	MQCMIT	MQCLTMH
MQCTL	MQSUBRQ	MQGET		MQBACK	MQSETMP
MQDISC	MQCLOSE	MQCB			MQINQMP
					MQDLTMP
					MQMHBUF/MQBUFMH



Java Message Service (JMS) and XMS

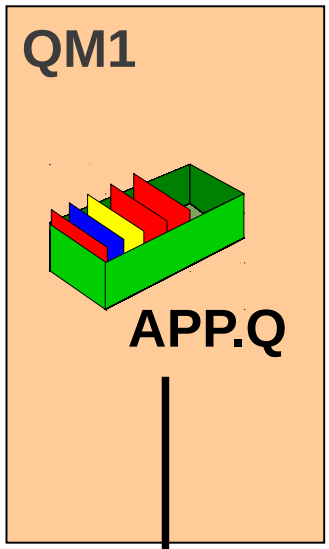


- JMS is the standard Java API for messaging
  - Point-to-point and Publish/subscribe messaging (application can be agnostic)
  - Enables greater portability between messaging providers
    - *Vendor-independent messaging API in Java*
    - *Managed by The Java Community Process*
    - *Expert Group includes IBM*
  - WMQ supports all Java Enterprise Edition (JEE) 1.4+ application servers
  - Features such as message-driven beans greatly simplify creation of messaging applications

- IBM Message Service Clients (XMS) renders a JMS-like API in non-Java languages
  - (Almost) full compatibility with JMS 1.1 API
  - Full interoperability with IBM JMS implementations on WMQ and WPM
  - Shared administered objects in JNDI with JMS
  - Current implementations include: C, C++ and .NET

Example JMS receiving application

Some client APIs need no MQI programming knowledge!




```
// Lookup the WMQ specific objects in JNDI
Context jndiContext = new InitialContext();
ConnectionFactory cf = jndiContext.lookup("jms/QM1");
Destination dest = jndiContext.lookup("jms/APP.Q");


// Establish a connection with the queue manager
Connection conn = cf.createConnection();
conn.start();
Session session =
  conn.createSession(false, Session.AUTO_ACKNOWLEDGE);


// Get a message
MessageConsumer consumer = session.createConsumer(dest);
consumer.receive();
```

Java Message Service (JMS) and XMS

- JMS is part of the J2EE specification and is supported by all J2EE compliant applications servers including; WAS, WebLogic etc. If you are working in Java or a J2EE environment inside an app. server, then you will almost certainly use JMS to access your messaging infrastructure. JMS 1.1 is the current version of the standard and is fully supported by MQ.

- It simplifies programming – providing simple to use Pub/Sub messaging in addition to point-to-point, although there are many similarities with the MQI (Connection = MQCONN(), Session = UOW)

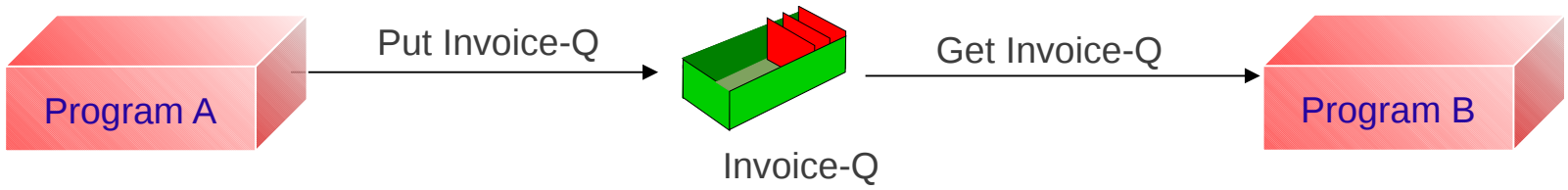
- XMS syntactically the same as JMS V1.1, but for C, C++ and C#. It offers good interoperability between JMS & non-Java applications, and they share administration models – it is ideal for sending message to JMS application running in an Application Server.

Agenda

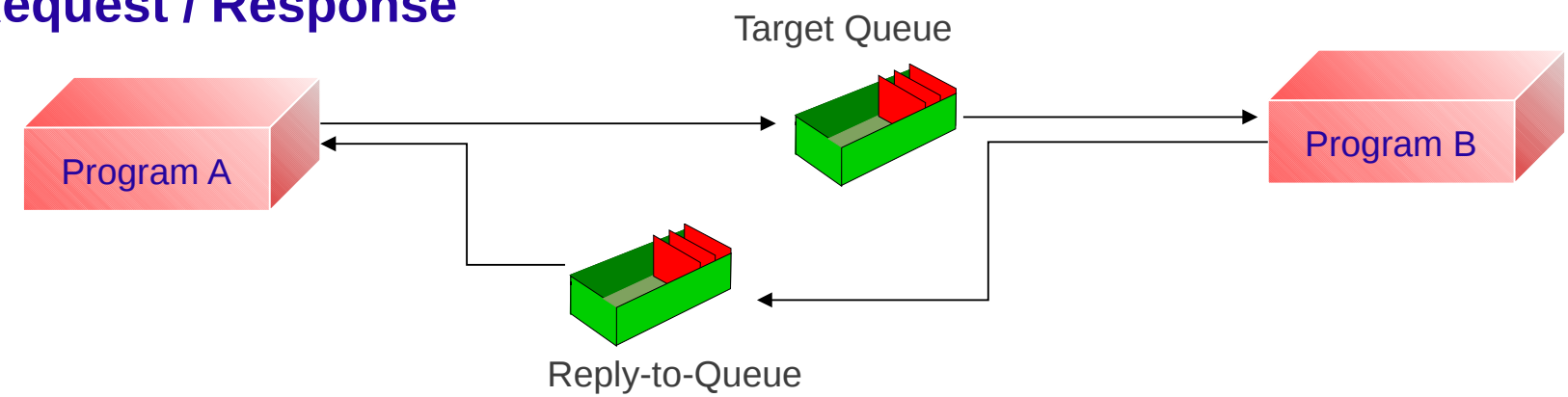
- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Example application architectures (1)

‘Send and Forget’



Request / Response

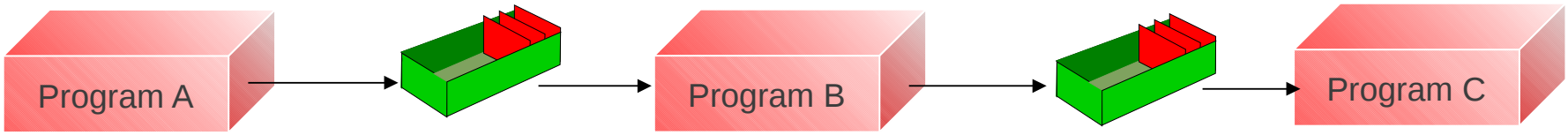


Example application architectures (1)

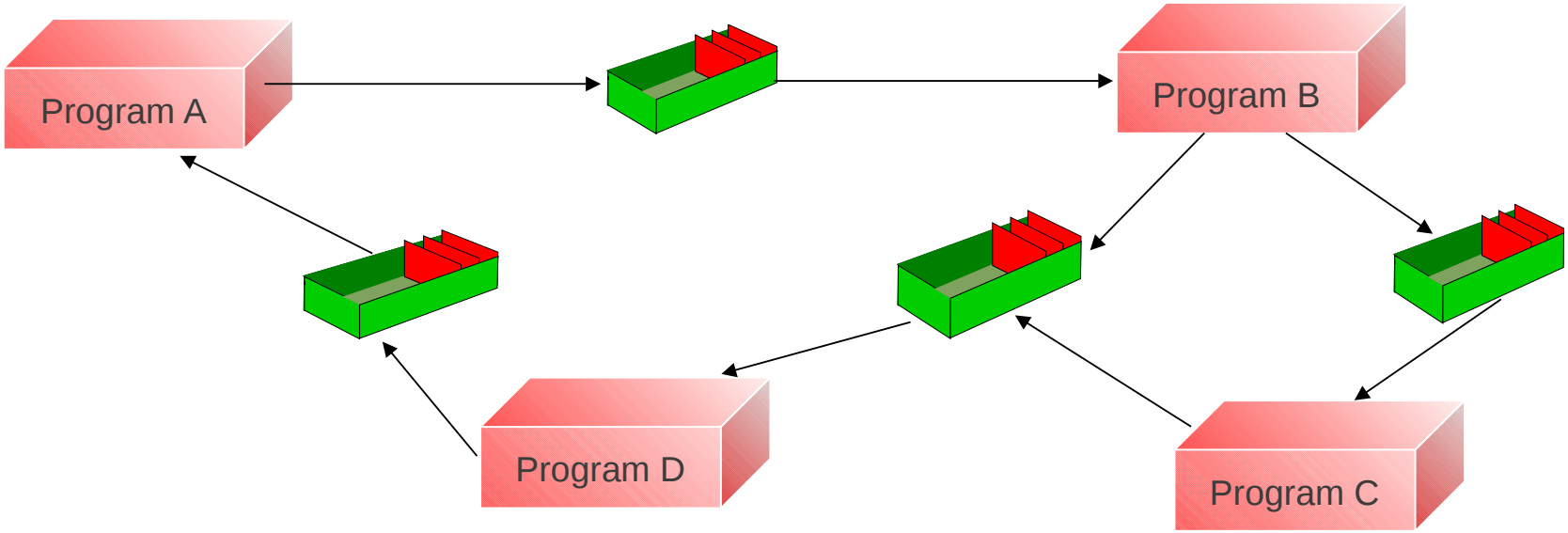
- These examples show some of the ways in which MQ queues can be used and, thereby, shows some of the styles of applications that may benefit from the use of a message/queuing model.
- 'Send and Forget'
- This style is one where there is no (direct) response required to a message. The message/queuing layer will guarantee the arrival of the data without the application having to solicit a response from the receiver.
- Request/Response
- This style is typical of many existing synchronous applications where some response is required to the data sent. This style of operation works just as well in an asynchronous environment as in a synchronous one. One difference is that - in this case - the sender does not have to wait for a response immediately. It could pick up the response at some later time in its processing. Although this is also possible with the synchronous style, it is less common.

Example application architectures (2)

Chain



Workflow



Example application architectures

- **Chain**

  Data does not have to be returned to the originating application. It may be appropriate to pass a response to some other application for processing, as illustrated in a chain of applications.

- **Workflow**

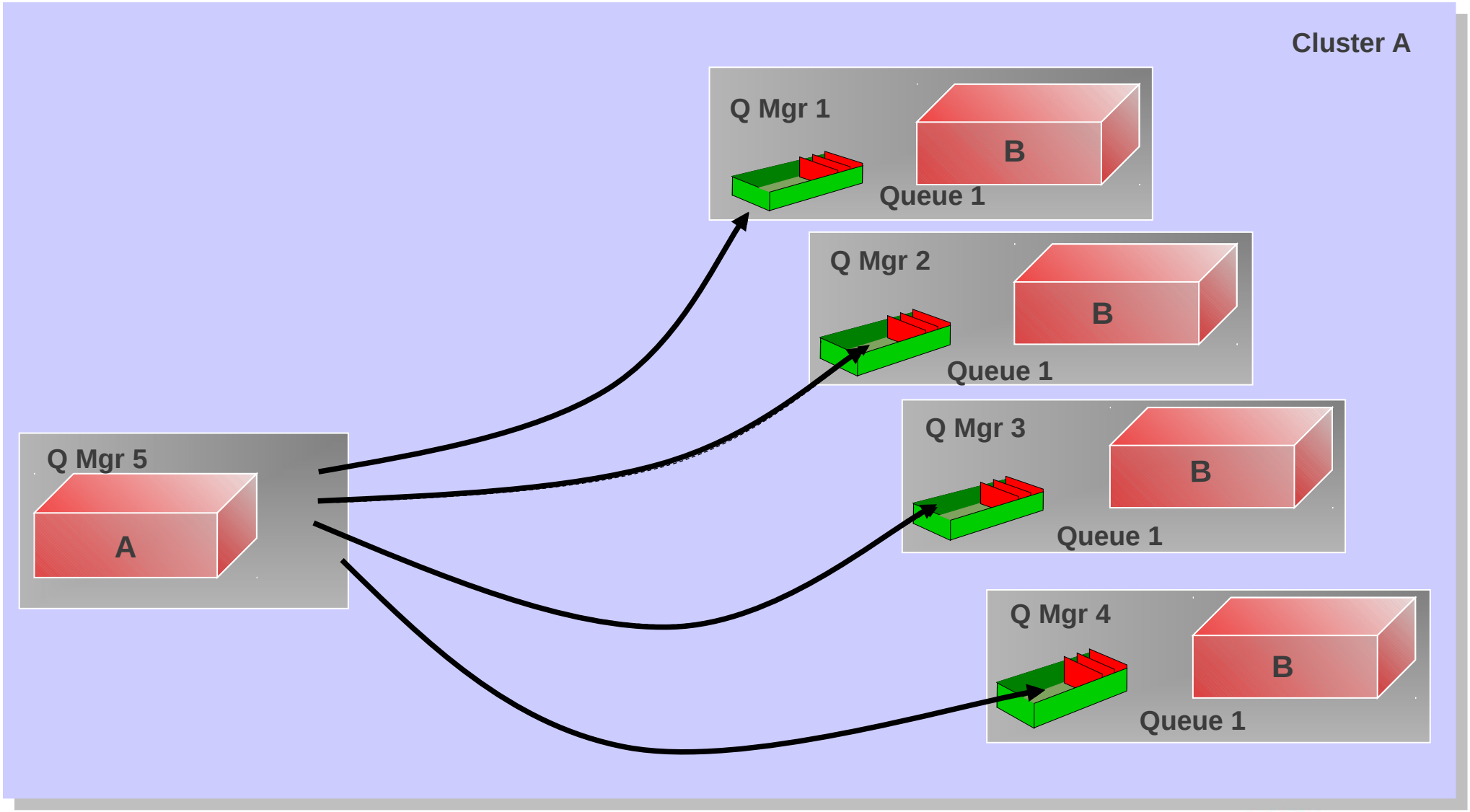
  There may be multiple applications involved in the processing before a response comes back to the originating application.

  These various modes of interaction may be arbitrarily combined to provide as complex/sophisticated topology as is necessary to support a particular application. The loosely coupled nature of the message queuing model makes it ideal for this style of interaction. Furthermore, it makes it straightforward to develop applications in an iterative style.

Agenda

- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Example application architectures – Clustering



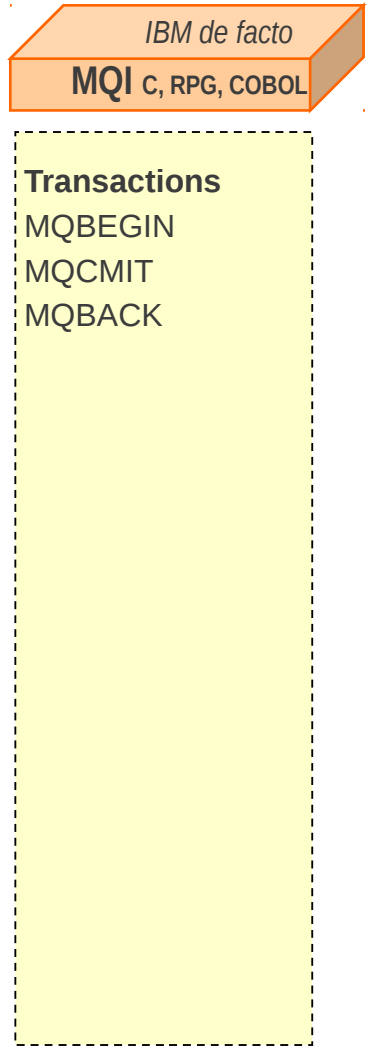
Example application architectures – Clustering

- In order to enable highly scalable applications, MQ queue managers provide support for MQ Clusters. In this environment, there are several copies (or clones) of a particular target queue and each message is sent to exactly one of the possible choices.

- WebSphere MQ Cluster support also defines and manages all WMQ resources, such as channels, automatically and provides automatic notification of failed or new queue managers in the environment.

WebSphere MQ Transactions

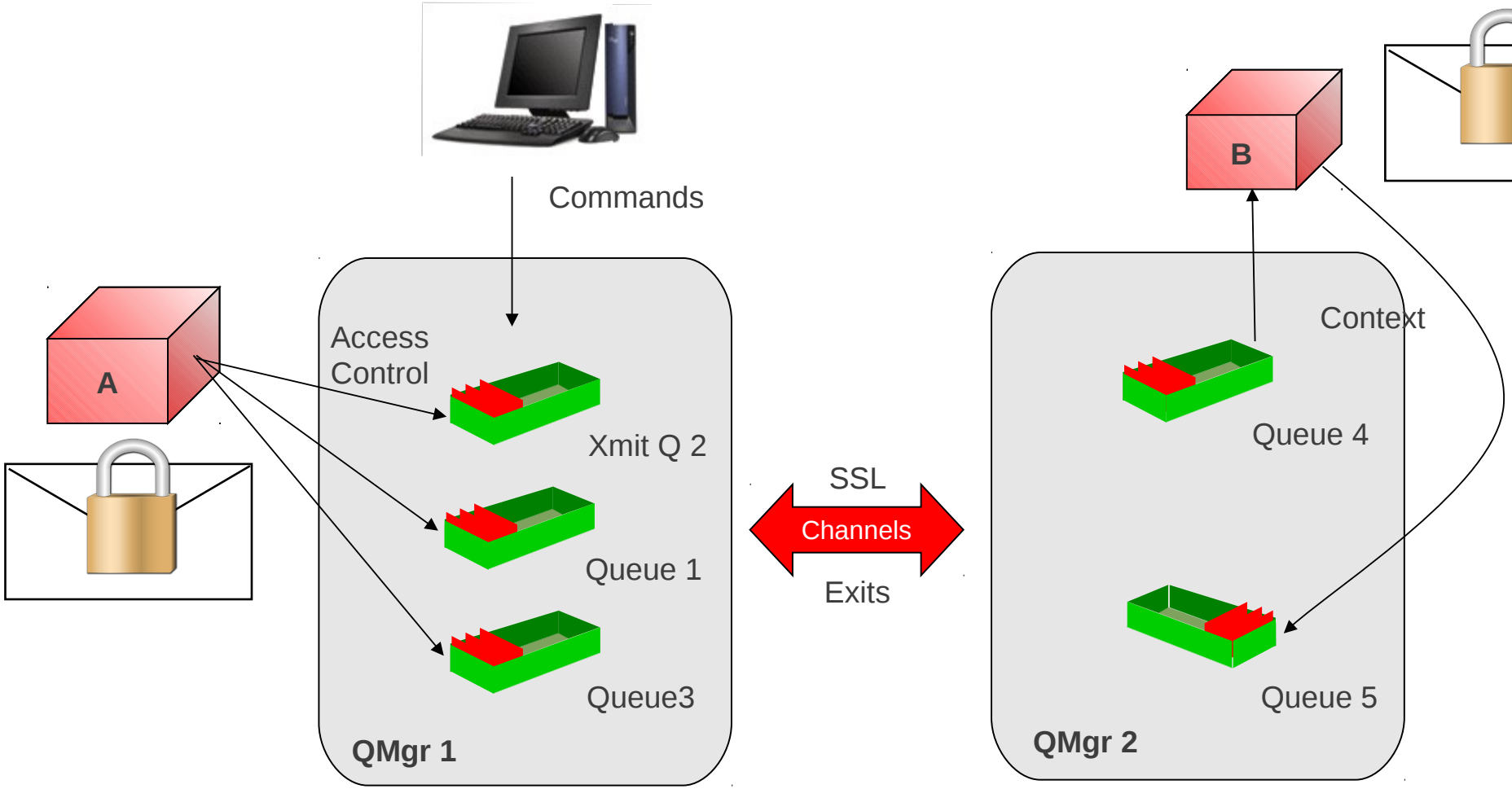
- Message level inclusion/exclusion in unit of work
- Single UoW active per connection at any one time
- WebSphere MQ local units of work
  - MQCMIT and MQBACK control the unit of work
- Messages and other resources in a global unit of work
  - Managed by a Transaction Manager
    - *WebSphere Application Server, CICS, IMS, z/OS RRS*
    - *Microsoft Transaction Server*
    - *Any XA or JEE App Server Transaction Manager*
  - Managed by WebSphere MQ
    - *WebSphere MQ is an XA Transaction Manager*
    - *MQBEGIN, MQCMIT and MQBACK control the unit of work*



WebSphere MQ Transactions

- WebSphere MQ supports units of work (UoW) where a set of resource updates may be considered as an atomic unit - either all of the changes are made or none of the changes are made. This support is particularly important when using WebSphere MQ in a commercial environment (it's primary focus) as transactions play a major part in this arena.
- WebSphere MQ allows messages to be included/excluded from a UoW at the message level. This differs from some other environments where a UoW starts and all subsequent actions are included in the UoW. Thus, a set of messages may be considered to be a UoW. Often, it is necessary to include both MQ messages and some other recoverable resources (typically database updates) in a UoW. Typically, this has required the use of some Transaction Monitor and WebSphere MQ works well with CICS and IMS on z/OS and with any XA compliant Transaction Manager. In situations where a Transaction Manager product is not available/suitable, WebSphere MQ itself may be used as the Transaction Manager. This does not mean that WebSphere MQ is transforming itself into a Transaction Monitor, it is just providing the Transaction Manager aspect of a Transaction Monitor product.
- The API used in handling transactions differs according to the environment. WebSphere MQ provides some verbs to handle UoWs. If a Transaction Monitor is used, however, its UoW verbs are used in place of the MQI.

WebSphere MQ Security



WebSphere MQ Security

- There are several aspects to WebSphere MQ security.
- Control of WebSphere MQ commands :

  Access to MQ commands, like creating and starting queue managers, can be controlled through operating system facilities and also by MQ facilities; it is necessary to be in a particular authorisation group to be allowed to use these commands.

- Access to Queue Manager objects:

  There is an access control component that is provided by the MQ Queue Manager, called the Object Authority Manager (OAM), which controls access to Queue Manager objects, particularly queues. The OAM can control access to resources at a very granular level, allowing access for different actions, such as GET, PUT, INQ, SET, etc. This access is (generally) based upon group memberships.

- This security service is a pluggable component of MQ. Thus, if the OAM does not meet the requirements of the environment it is possible to provide a different (or additional) component. Note that the OAM is used for all queue managers except for the z/OS queue manager which uses any SAF compliant security manager.

- Encryption of message data through Advanced Message Security (AMS)

WebSphere MQ Security (contd)

- Channel Security (Authentication)
- WebSphere MQ 6.0 provides built-in SSL link level security
- MQ also provides a number of exit points during the transfer of messages between systems. The key exits concerned with security are :-

  Security Exit : This exit allows for (mutual) authentication of partner systems when they connect to one another.

  Message Exit: This exit allows allows for customisation at the message level, allowing individual messages to be protected, in terms of message integrity, message privacy and non-repudiation

WebSphere MQ Security (contd)

- Application Security
- This level of security is not implemented directly by the Queue Manager but such facilities may be implemented at the application level, outside of the direct control of WebSphere MQ.

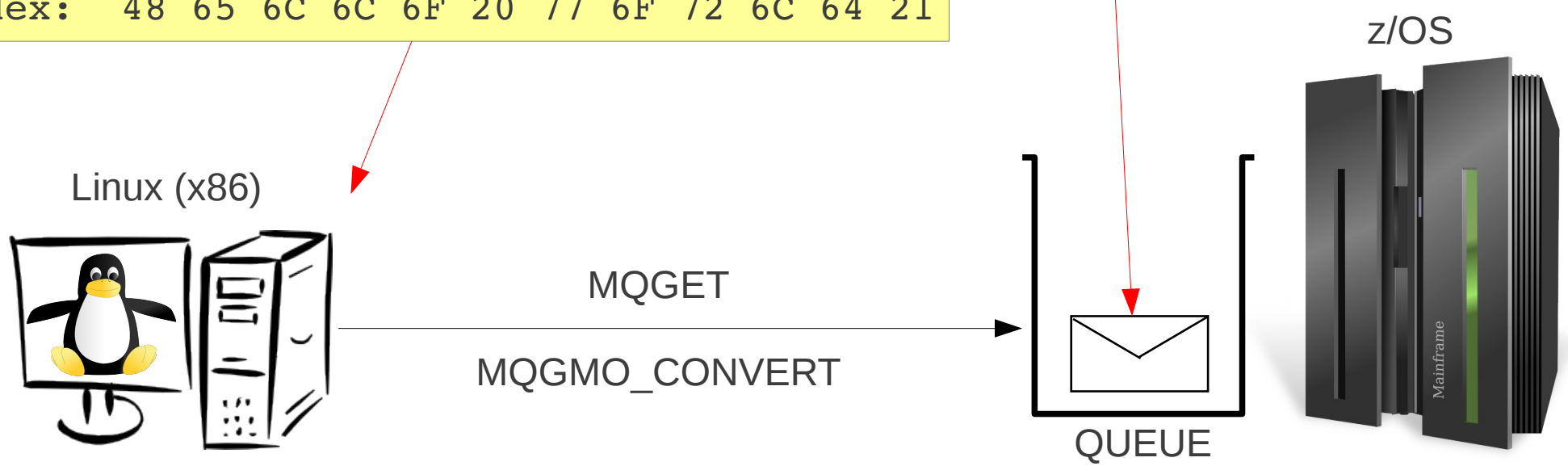
- Advanced Message Security
- Provides end to end security, enabling messages to be encrypted from the time they are PUT by the sending application to when they are GET by the receiving application, so messages are help encrypted when at rest on queues as well as when in transit.

Data Conversion

When receiving messages, WebSphere MQ can convert the message payload data. This is most commonly used to convert character data so that it is in a format which is consumable by the receiving application.

```
CCSID 500 (EBCDIC Latin-1-charset)
Data: H  e  l  l  o     w  o  r  l  d  !
Hex:  C8 85 93 93 96 40 A6 96 99 93 84 4F
```

```
CCSID 1208 (UTF-8)
Data: H  e  l  l  o     w  o  r  l  d  !
Hex:  48 65 6C 6C 6F 20 77 6F 72 6C 64 21
```



WebSphere MQ Systems Management

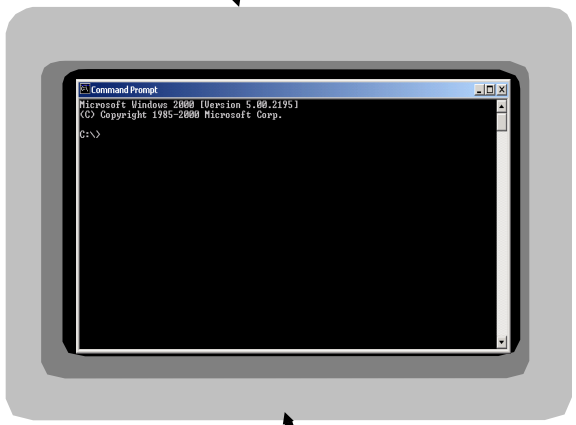
MQ Explorer

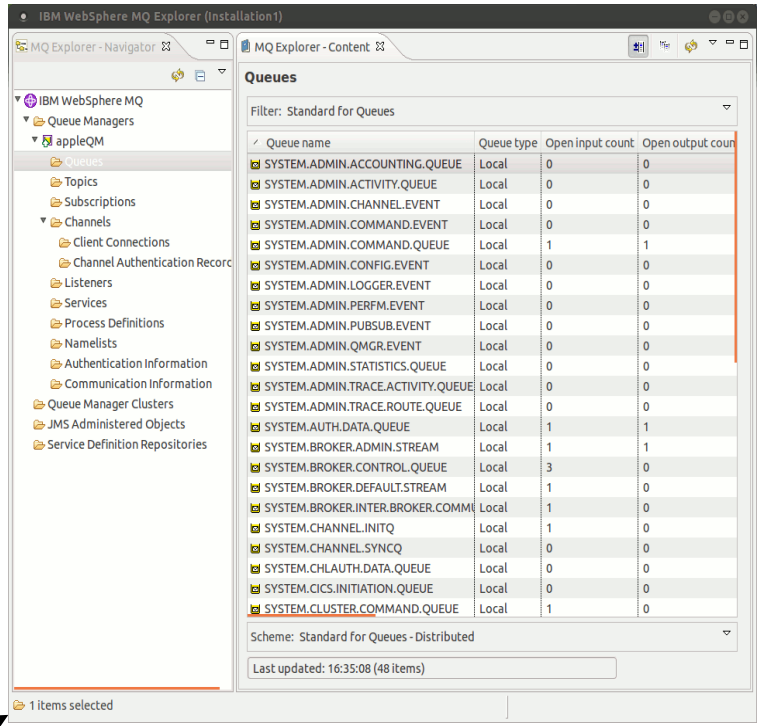
Scripting

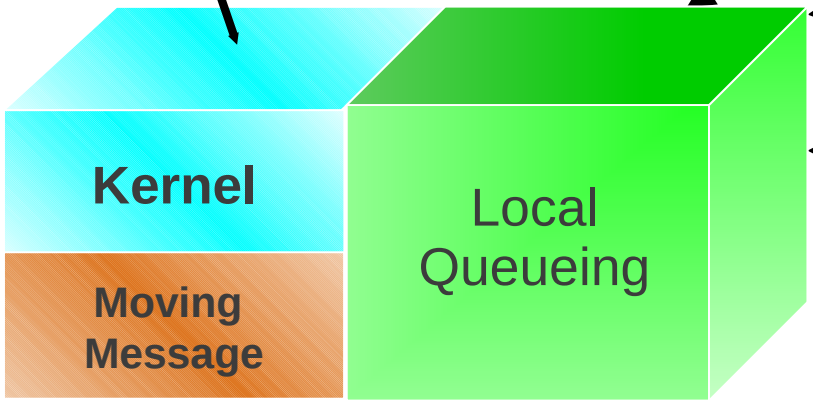
MQSC

MQ Application

Programmable Command Format (PCF)







System Management Applications:

BMC, CA, Landmark, RYO, Tivoli

WebSphere MQ Systems Management

- One of the key operational components of any system is management. WebSphere MQ enables systems management in a number of ways:

- There are facilities provided by the MQ base to enable MQ resources to be managed. There are 'internal' utility programmes (for example, MQSC, the TSO interface for z/OS and the command line interface for AS/400). There are also documented interfaces, most notably Programmable Command Format messages which are PUT to a well known queue and are processed accordingly by the queue manager.

- WebSphere MQ provides events. These events are themselves MQ messages which are PUT (by the queue manager) to well known queues and provide information on state changes for various queue manager resources. The format of the event messages is documented. Text based message logs (and Windows events) are also provided.

- So, WebSphere MQ queue managers provide a set of documented interfaces to allow control and configuration of resources and to inform external processes of state changes within the queue manager. These interfaces may be used by any application program. Typically, this occurs in 3 ways:

- There are MQ utilities which make use of these interfaces. Most notably, the MQ Explorer (provided for Windows and Linux for Intel environments) enables management and configuration of both local and remote queue managers using PCF messages.

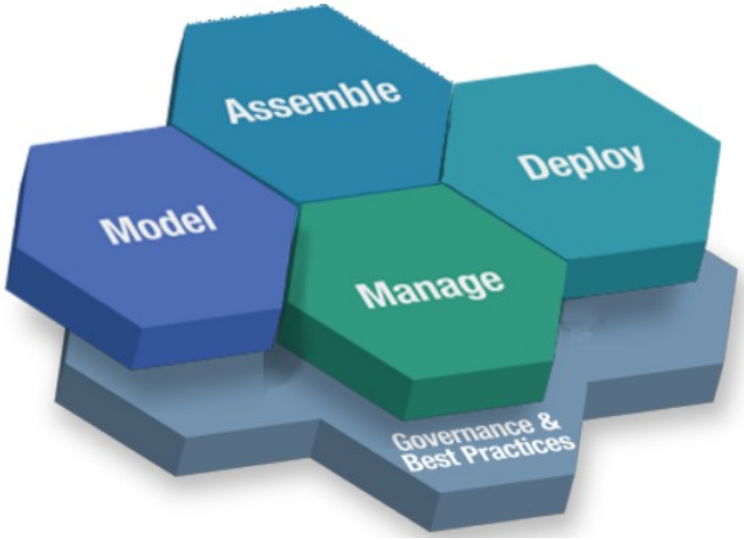
- The majority of the established systems management vendors use the facilities described above to provide MQ 'personalities' for their products.

- Customers may write their own utilities to provide systems management capabilities within their organisations. This style often makes use of the messaging APIs to utilise PCF and event messages. Also scripting languages (most notably PERL) are used to provide systems management scripts for WebSphere MQ and other environments.

WebSphere MQ Service Definition

- A Specification which allows WMQ applications to be described as SOA assets using WSDL and URI, in the same way as web-services. Enabling:
  - Inventory and cataloguing in a Service Registry
  - Re-use as a services in a composite SOA applications
  - Management and tracing with SOA tools
  - Impact Analysis

- Tooling to generate Service Definitions supplied in WebSphere MQ Explorer.



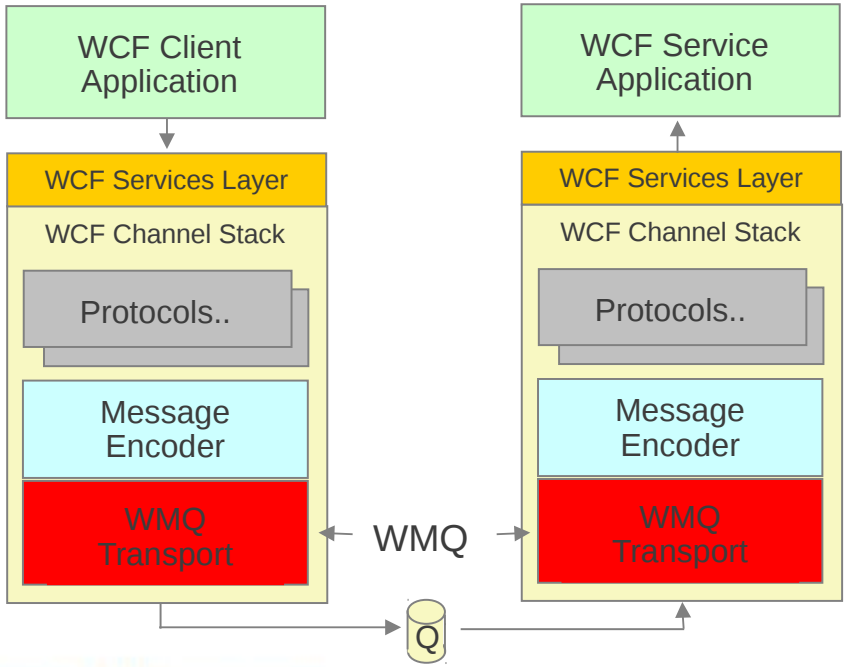
WebSphere MQ Service Definition

- WMQ users have requested guidance from IBM on how they should describe their WMQ applications as services for use in service oriented architectures.
- There has been particular interest in applying this to unmanaged native WMQ applications (i.e. those coded to the MQI - not JMS - and those running outside of an application server / CICS etc.)
- This will allow applications to:
  - Be inventoried, and catalogued in Service Registry. For example, the WSDL description of an application can be stored in WSRR
  - Be managed and traced with SOA tools. which - for example – will be able to monitor the queues associated with a service
  - Be re-used in composite applications. For example, once the MQ service definition has been implemented by web services tools, it will be possible to drop an MQ application into a composite Web services application, and the tools will generate the code required to invoke the MQ application
- IBM has created the WMQ service definition specification to address this. This consists of two documents.
  - An IRI specification, which defines :
    - *The address of a WMQ message destinations i.e. Queues or Topics for use by messaging applications*
    - *The address of other WMQ resources i.e. Qmgrs, Queues, channels, channel status etc. for use by admin tools*
  - A Bindings Specification, which defines :
    - *Properties which may be used to describe and connect to a WMQ app.*
    - *The mapping of properties to message headers for the construction and interpretation of SOAP and non-SOAP messages*
    - *Supported message exchange patterns*
    - *A WSDL binding for SOAP/WMQ and non-SOAP/WMQ*
    - *Examples of IRIs, Messages, and WSDL documents*
- The specification is published in SupportPac MA94 with tooling to help generate service definitions being available in the WMQ Explorer since v7.0.0.1

WMQ Custom Channel for WCF

- Windows Communication Foundation (WCF) underpins Web services and Messaging in .NET 3
  - Built-in Transports e.g. MSMQ, HTTP(S), Named Pipes, TCP/IP, etc.
  - Transports can be extended with ‘custom channels’
  - Allows alternative transports (like MQ) to be slotted into WCF seamlessly

- Primary focus is for service orientated architectures



WMQ Custom Channel for WCF

- Initially released on Alphaworks in 2007, the WMQ Custom Channel for WCF is now available in WMQ v7.0.1.

- Its primary role is for use as a transport for web services, interoperating with clients and services hosted in WCF, WAS, CICS, Axis and .NET (.asmx) enabling web services to be invoked with messaging qualities of service.

- Integrates seamlessly with the built-in WCF channels provided by Microsoft and so shares the same tooling such as the svcutil.exe client proxy generator.

- Supports both one-way (fire and forget) and request-reply message exchange patterns.

Agenda

- Introduction – why use WebSphere MQ?
- Fundamentals of WebSphere MQ
- Using the WebSphere MQ API
- Example Architectures
- Other Key Features
- Related Products
- Summary

WebSphere MQ and the Wider World

- For a messaging engine to be really useful it should allow access to the messages from many different environments. We have already discussed MQs programming language and API support but what about the environments.

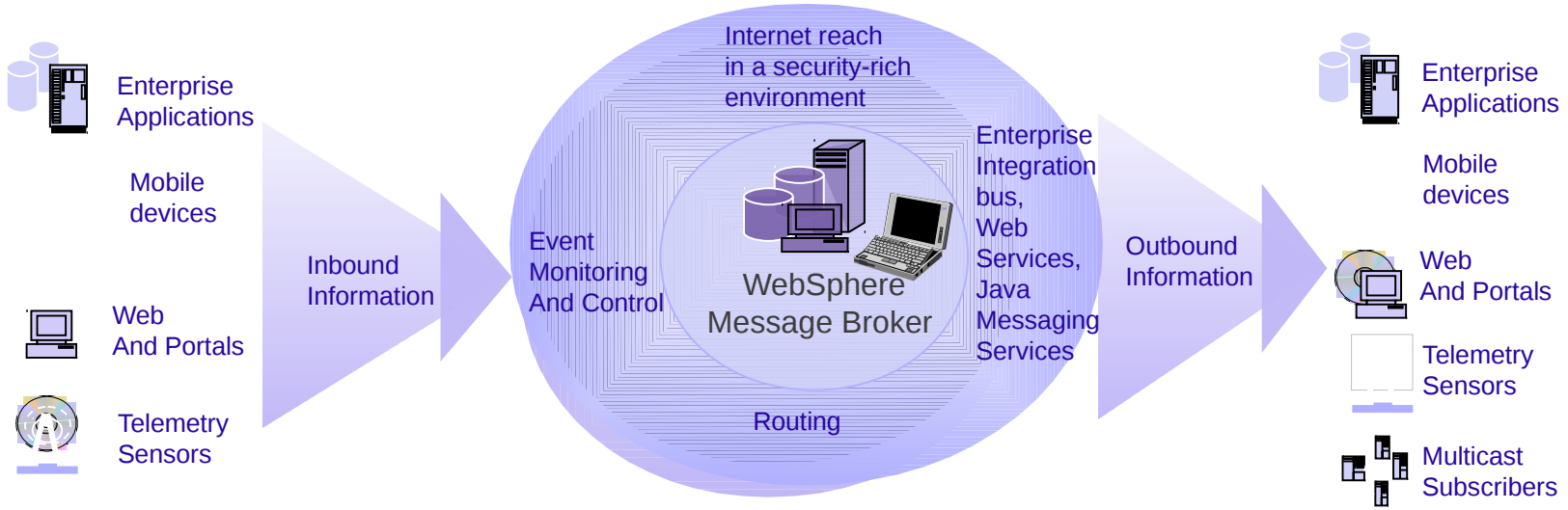
- The complexity of overall business applications is increasing every year as more and more applications are linked together in some way. WebSphere MQ dramatically reduces an individual applications complexity by providing a consistent, reliable and transactional method of communicating between applications from hundreds of different environments.

- We are now going to look briefly at some of the other WebSphere Business Integration products that make up the portfolio, and how WebSphere MQ fits in

The WebSphere MQ Product Family

- WebSphere MQ (formally MQSeries)
  The messaging system.
- WebSphere MQ: Managed File Transfer (MFT / WMQ-FTE)
  For a managed file transfer solution which leverages WebSphere MQ.
- WebSphere MQ: Advanced Message Security (AMS)
  End to end application security.
- WebSphere MQ: Telemetry (MQTT)
  For extending the reach of enterprise messaging to devices.
- WebSphere MQ: Low Latency Messaging (LLM)
  For low latency, extreme high throughput multi-cast messaging.

WebSphere Message Broker



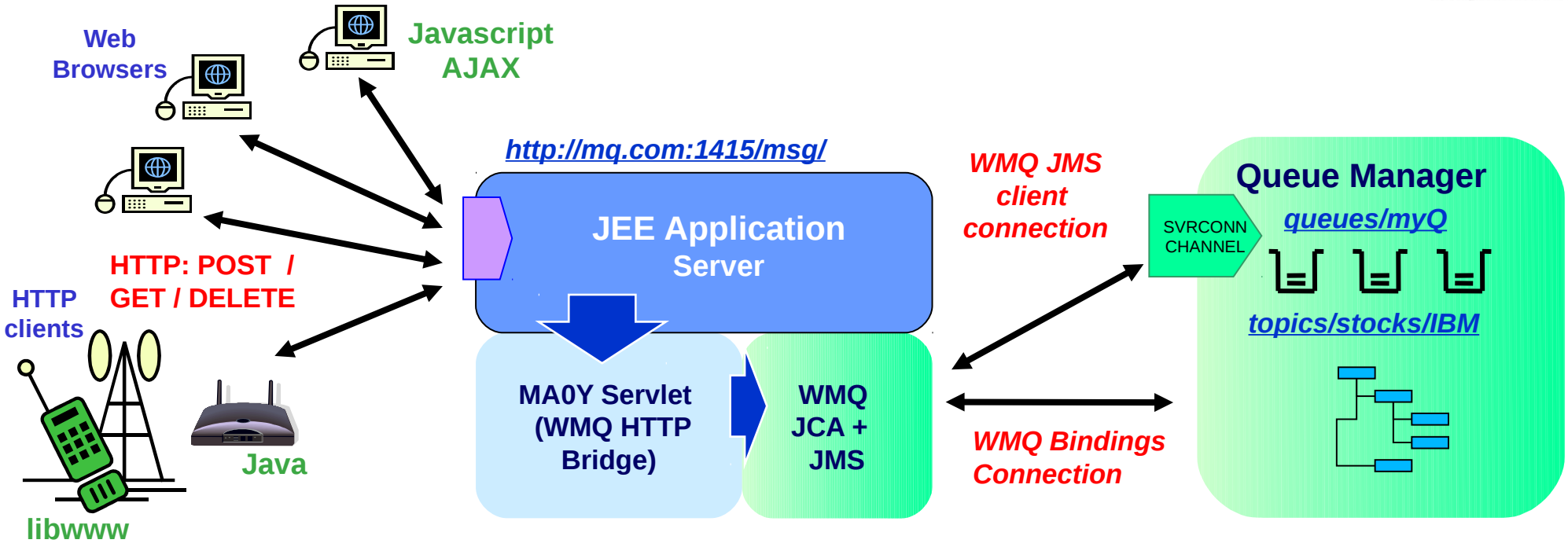
- WebSphere Message Broker
  - Message transformation (mediations)
    - *Combine data sources: databases, files, etc.*
    - *Update other data stores: databases, files, etc.*
  - Content based filtering and routing
  - Adapters - SAP, PeopleSoft, ORACLE, Files, e-mail…
  - WebSphere Transformation Extender

WebSphere Message Broker

- WebSphere MQ provides the assured delivery backbone to an Enterprise Service Bus. The queue managers are message content agnostic. Consequently, any data may be exchanged between applications. However, many applications are dependent upon their data being routed to particular destinations and are dependent upon particular data formats. So, the fact that applications may exchange data (via WebSphere MQ and/or WebSphere Adapters) does not solve all possible problems. For the general case of any to any application integration, an intermediary is required to handle message routing issues and to handle (both simple and very complex) message transformation issues.

- Message Broker provides the function that enables complex message routing and transformation functions to be encapsulated outside of applications, in a (logically) central component.

HTTP Connectivity to WMQ



- Key features of the WebSphere MQ Bridge for HTTP
  - Maps URIs to queues and topics
  - Enables MQPUT and MQGET from
    - *Web Browser*
    - *Lightweight client*
- Can be used as a SOAP Web Services entry point

HTTP-MQI Verb / Resource Mapping

- Define URI to identify queue (or topic)
- Modelled on REST principles
  - Simple translation of HTTP to MQI
- Message Format:
  - Header fields (MQMD) conveyed in HTTP headers
  - Body is passed in HTTP entity body
  - Message type is conveyed in HTTP Content-Type
    - *"text/plain" or "text/html" equate to WMQ string messages (MQFMT_STRING)*
    - *All other media types map to WMQ binary messages (MQFMT_NONE)*

		HTTP verb mapping			
Resource	**Sample URIs**	**GET**	**POST**	**PUT**	**DELETE**
Messages	**http://host/msg/queue/*qname/*** **http://host/msg/topic/*topic_path/***	**MQGET w. browse**	**MQPUT**	**-**	**MQGET**

Example HTTP Flow - POST (= MQPUT)

Request:

```
POST /msg/queue/requestQ/ HTTP/1.1
Host: www.mqhttpsample.com
Content-Type: text/plain
Content-Length: 60
x-msg-replyTo: /msg/queue/replyQ/
x-msg-requiresHeaders: msgID, priority, timestamp
Message body which will appear on the queue as an MQSTR
```

Put to destination

Type and length of message (60 char string)

reply Queue

Headers to include on reply

Message Data

Response:

```
HTTP/1.1 200 OK
x-msg-msgID: 1234567890
x-msg-timestamp: Thu, 22 Mar 2007 08:49:37 GMT
x-msg-priority: 4
```

Response code

Required Headers

HTTP Connectivity to WMQ

- The first goal of the HTTP feature (originally SupportPac MA0Y) is to extend the reach of WMQ applications to more environments such as web browsers. This will give Rich Internet Applications simplified access to the Enterprise. Eliminating the WMQ client reduces the cost of application deployment, though this is not a complete replacement for the WMQ client
  - It is missing many MQI features and does not offer transactionality, assured delivery etc.
  - But in many cases where applications have resend logic and check for duplicates it will be good enough
- The API is modeled after REST ("Representational State Transfer") principles. REST offers a different integration style to WS-* standards based web services. Qualities of service are sacrificed for simplicity and scalability to keep barriers-to-entry low. REST APIs are typically simple and can be used spontaneously and incrementally – for example in Web 2.0 mash-ups. The HTTP/WMQ API is largely based on REST, though it has some quirks. For example this component transfers message representations, but messages are not ideal REST resources
  - They do not necessarily have a unique identifier, and so cannot be addressed individually
  - Not generally amenable to caching etc. because they must be delivered only once
  - They are very transient
- It is a stateless / connectionless API with one HTTP verb corresponding to one WMQ operation
  - HTTP headers = Message headers
  - request headers (get and put options) – wait, requires-headers
  - entity headers (MQMD options) – priority, expiry, timestamp, persistence, msgId, correlId, replyTo
  - HTTP request payload = Message body as either text or binary
- No client libraries are provided – apps code directly to HTTP verbs using whatever APIs are in the environment.
- REST was described by Roy Fielding in http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

WebSphere MQ Advanced Message Security

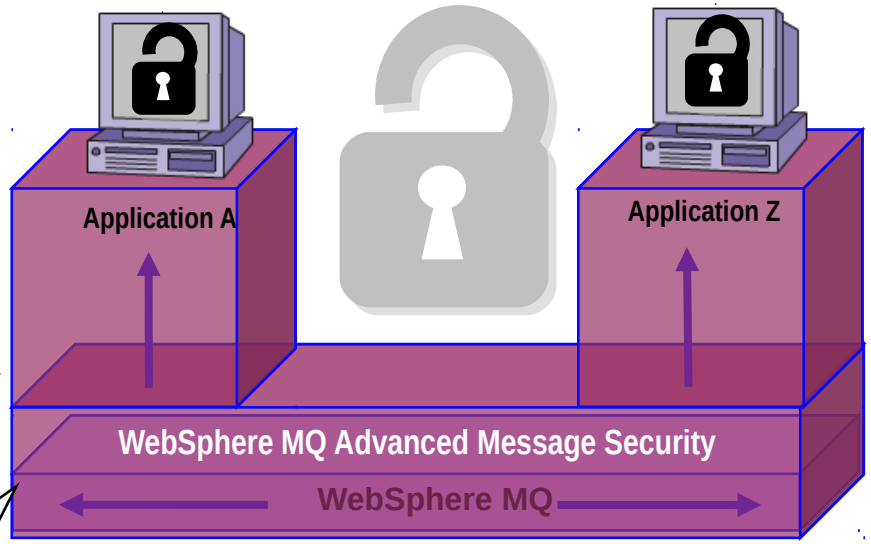
- Secures application data even before it is passed to MQ
- Upgrade from base WMQ – No changes to existing applications or network required

WebSphere MQ standard security:

- Industry standard SSL channels (128-bit)
- Certified for Common Criteria
- Authentication is based on Operating System identifier of local process
- Security admin has to be done at each server
- Message data can be encrypted in transport but not when it resides in the queues

WebSphere MQ Advanced Message Security adds:

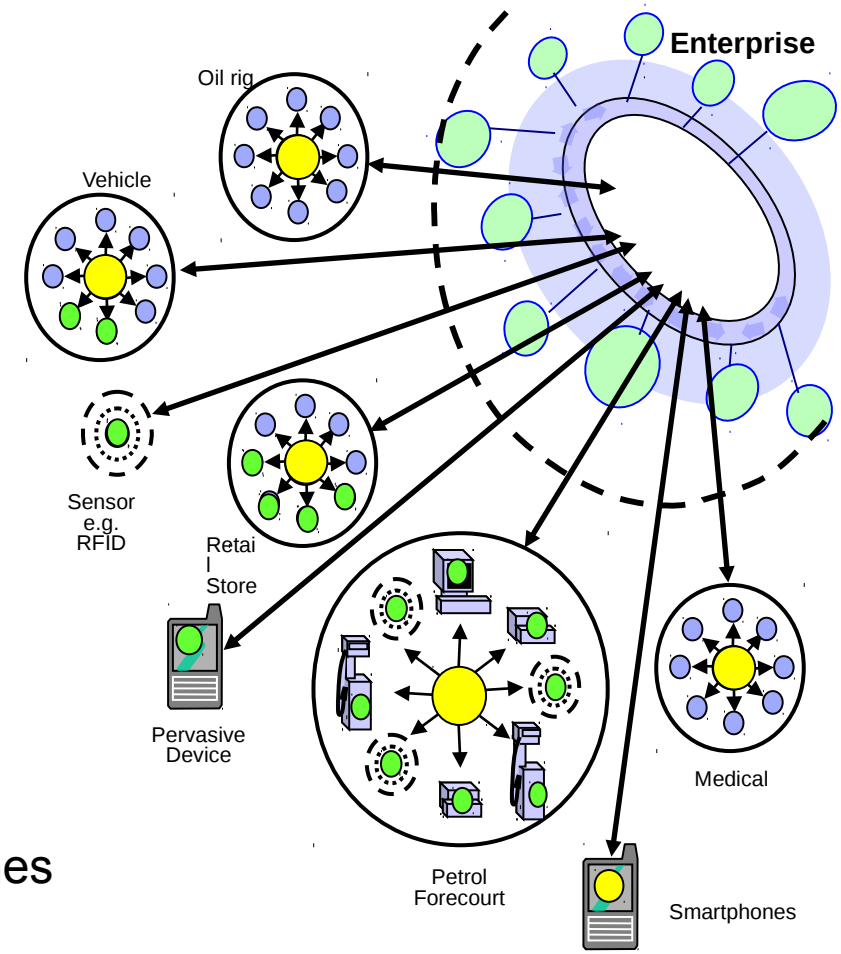
+ Authentication policies are based on certificates associated with each application
+ Message data is protected end-to-end – including when it resides in queues
+ Centralized admin of security policies across all servers
+ Much finer granularity in security policies
+ Audit logs of data and queue access
+ No changes needed to applications or queues



Securing the data and the applications

WebSphere MQ Telemetry

- Product extension included in WMQ 7.1 supporting mass connectivity for smart devices to the enterprise
- Utilises MQTT protocol
  - a lightweight, public, low bandwidth messaging protocol for scenarios where enterprise messaging clients are too big or bandwidth intensive.
  - Established for >10 years
- Java and C API provided, but you can "roll your own"
- Ideally suited to:
  - Fragile / Expensive networks such as "sometimes connected" devices / satellite phones
  - Niche platforms such as tiny sensors, personal devices, edge/small servers
  - Mass Scalability (> 50,000 clients per queue manager)

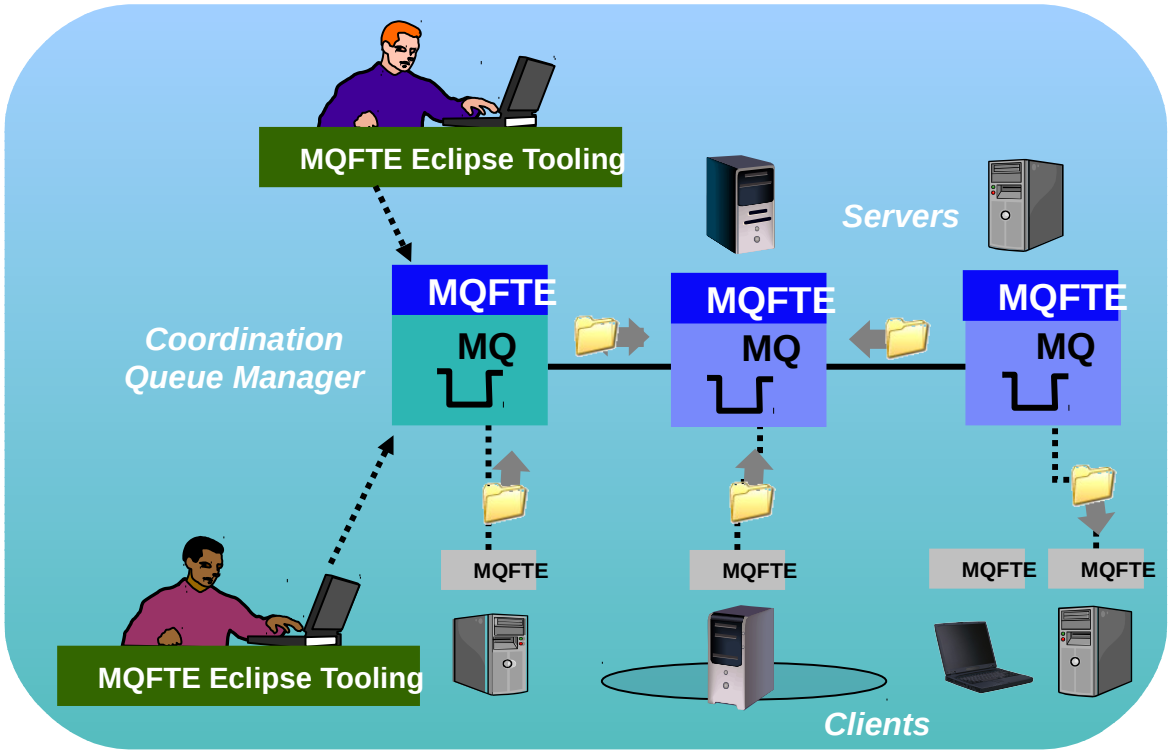


WMQ Managed File Transfer

- MQ MFT/FTE solves problems of auditing, monitoring, scheduling, security …
  - Automated bulk data transfer between distributed heterogeneous systems.
  - Capabilities for integrating, managing, and controlling data movement.

- Built on WebSphere MQ
  - Assured delivery of data over MQ backbone

- Simplicity and ease-of-use
  - GUI Driven
  - WMQ Explorer Integration
  - Scheduled, or Triggered transfers
  - Scriptable
- Complements MB File Nodes



Product page:
  - http://www.ibm.com/software/integration/wmq/filetransfer/v7/

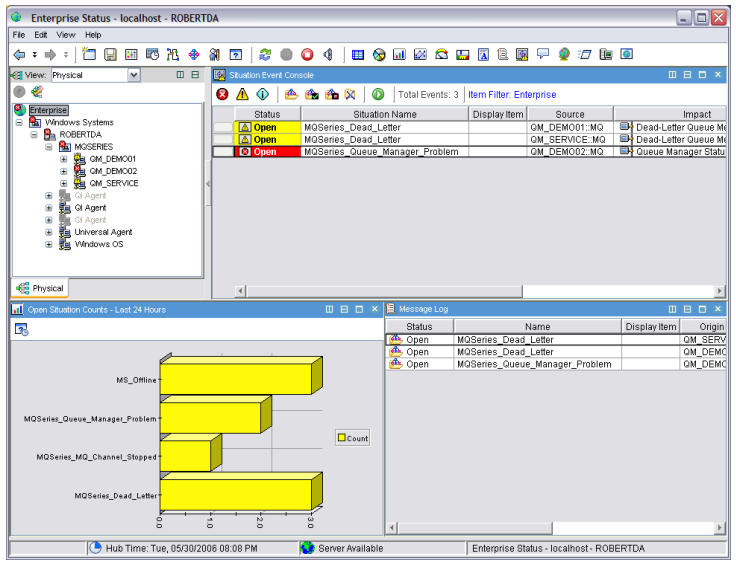
Tivoli Omegamon and ITCAM

- Range of IBM products for monitoring and managing
  - Common core technologies with product-specific integration
  - eg Omegamon for Messaging deals with WMQ and Message Broker

- Enterprise-scale Management with Omegamon
  - Much larger environments than the MQ Explorer will handle
  - Allows joining of multiple products into single views
    - *eg there might be a situation only if both WMQ and DB2 show specific issues*

- Part of the "extended" WMQ development team
  - Make sure Tivoli can support new features
  - WMQ V7 support available

- Monitor SLAs
  - Drill down to appropriate product/OS levels

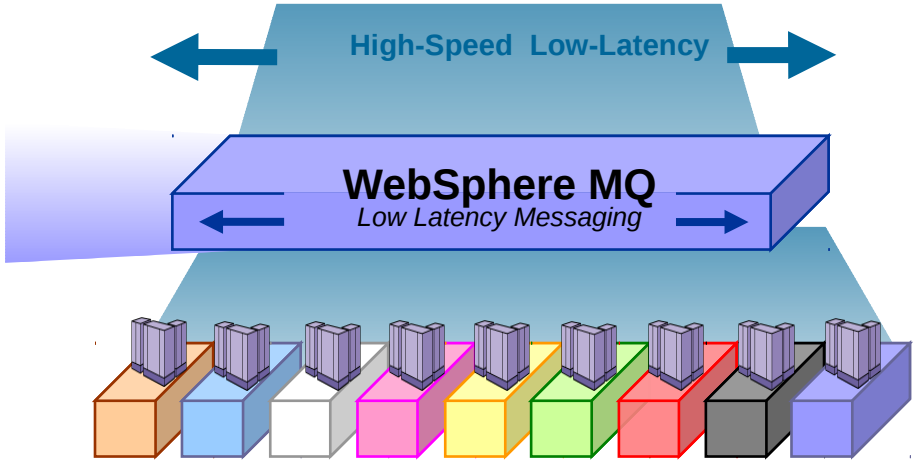


WebSphere MQ Low Latency Messaging

- Extends the WebSphere MQ messaging family
  - New product that provides a messaging transport optimized for low latency, high-throughput delivery

- Provides low Latency, high-throughput messaging
  - Capable of 91 million messages per second
  - Less than 30µs latency at high throughput rates
  - Traffic control with static & dynamic rate control

- Delivers semi-reliable delivery
  - Choice of Multicast and Unicast transport with range of topology, speed and reliability characteristics
  - Ordered (FIFO) delivery
  - Stream failover for high availability



- Filters messages flexibly
  - Coarse-grained, topic-based and fine-grained filtering

- Included in WebSphere Front Office for Financial Markets

Getting WebSphere MQ : Free Trial

http://www.ibm.com/developerworks/websphere/downloads

Summary

- WebSphere MQ - World leader in messaging technology
- Runs everywhere your applications do
- Simplifies application communication
  - From simple connectivity…..
  - ….. to complex workload balancing, transformation and routing
- Provides secure, reliable and high-speed infrastructure

Copyright and Trademarks