#SHAREorg

SHARE
Technology · Connections · Results

WebSphere MQ 101: Introduction to the World's Leading Messaging Provider

Craig Both – bothcr@uk.ibm.com
IBM Hursley

12th March 2012
10535

Agenda

- Introduction
- Fundamentals
- The API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Business Challenges (1)
Dispersed Business Logic

- Over time, separate organisational units build their own pieces of business logic…
- …with applications developed on many different platforms.
- Connecting these together can save a lot of time and money
- ***<u>WebSphere MQ can help achieve this.</u>***

Business Challenges (1) Dispersed Business logic

N O T E S

- For example; Payroll have a program which runs 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 her annual review score and her business unit’s performance
- Sales have a program to calculate annual review scores
- Research have a program to calculate annual review scores
- Etc.
- These applications may all run using different hardware and OS and be written in different languages. Being able to connect these together reduces costs and time.

Business Challenges (2)
Process Resilience

- As systems become more integrated…
- …the reliance on and cost of failure of a process increases
- Removing dependencies and introducing redundancy reduces the risk of a failure
- ***<u>WebSphere MQ can help achieve this.</u>***

Business Challenges (2)
Process Resilience

N O T E S

- Business processes need to provide a reliable always-on experience to users. For example, revenue or trust is lost if an order can't be taken or an error occurs. This means updates need to be transported and processed reliably and processes need to continue even if one or more data sources is unavailable.

- WebSphere MQ can help decouple each component in a process from the others. This reduces the number of dependencies at each stage and allows redundancy to be introduced easily.

Business Challenges (3)
Process Scalability

- Many applications and processes start out on a single system…
- …as business grows, a single system is no longer sufficient to cope with demand
- A scaleable architecture enables the capacity to be incrementally grown to meet increasing workloads
- ***WebSphere MQ can help achieve this.***

Business Challenges (3)
Process Scalability

N

O

T

E

S

- Scalability needs to be designed in, and can add significantly to the complexity, i.e. running an application across multiple systems can be orders of magnitude harder then running on a single system.

- WebSphere MQ provides the reliable communications and administration tools needed to spread or migrate applications and processes to multiple systems.

Business Challenges (4)
Process Flexibility

- A process originally designed for one purpose…
- …needs to change to meet new requirements
- Being able to respond rapidly to internal and external challenges gives a competitive advantage
  - Building a Service Orientated Architecture
- ***<u>WebSphere MQ can help achieve this.</u>***

Business Challenges (4)
Process Flexibility

N O T E S

- Requirements may change for a number or reasons; organisational changes, market changes or changes in technology. Being able to adapt quickly gives competitive advantage.

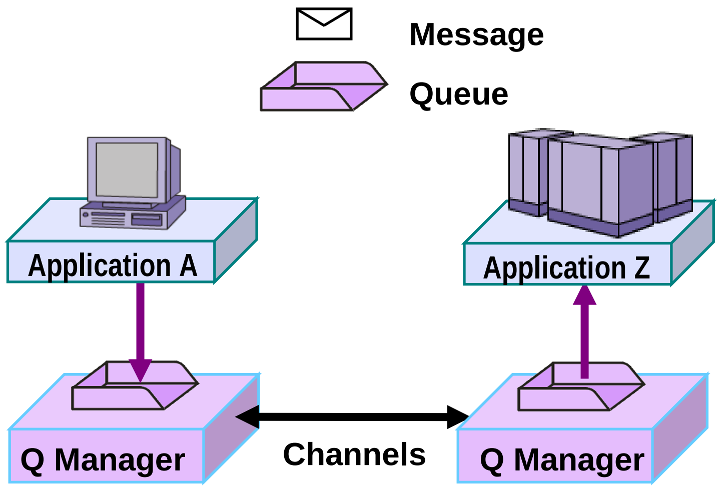
- WebSphere MQ can help to improve the responsiveness of your business by allowing individual components of your business logic to be seamlessly replaced without alterations to the rest of the process.

Agenda

- Introduction
- Fundamentals – The Key Concepts
- The API
- Example Architectures
- Other Key Features
- Related Products
- Summary

WebSphere MQ Key Concepts Messaging Goals

- 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
  - Reduce Complexity
  - Ease of use



WebSphere MQ Key Concepts
Messaging Goals

N O T E S

- What WebSphere MQ (aka MQSeries) did was to recognise that for the queuing model to be successful and applicable to a wide range of applications that it must achieve the following major goals :-
- First it must be totally reliable. A message put to an WebSphere MQ queue is as safe as a record written to a database. e-mail just isn't reliable enough
- Secondly it should be available everywhere, and support as wide a range of platforms, programming languages and common API as possible. The postal service would be severely restricted if it only covered the local city.
  - The WebSphere MQ base product is available on all major platforms such as Windows, AIX, HP-UX, Solaris, i5/OS, z/OS and many others (In all 80+ platform configurations are supported). Programming is performed using simple defacto API, such as the MQI which is available with only around a dozen verbs or via standards based interfaces such as JMS.
- Thirdly, the goals of MQ from an application standpoint is to enable as loose coupling as possible. This is achieved by providing :-
  - Location Transparency: A sending application need not know where the receiving application is nor have any knowledge of the network or communications .
  - Time Independence{ It is not necessary for both applications to be up and running at the same time (asynchronous)
  - Platform Independence: A sending application need not know what type of platform the receiving application is running on.
  - Data transparency: With the advent of Brokers and message translation it is not even necessary for the two applications to exchange messages in a shared format.
- Fourthly, it must be possible to incrementally add applications and capacity
- Finally, Reliable distributing computing is difficult, complex and error prone. Providing simple API and the right tooling makes a significant difference to the ease of application development and administration.

WebSphere MQ is not a substitute for…!

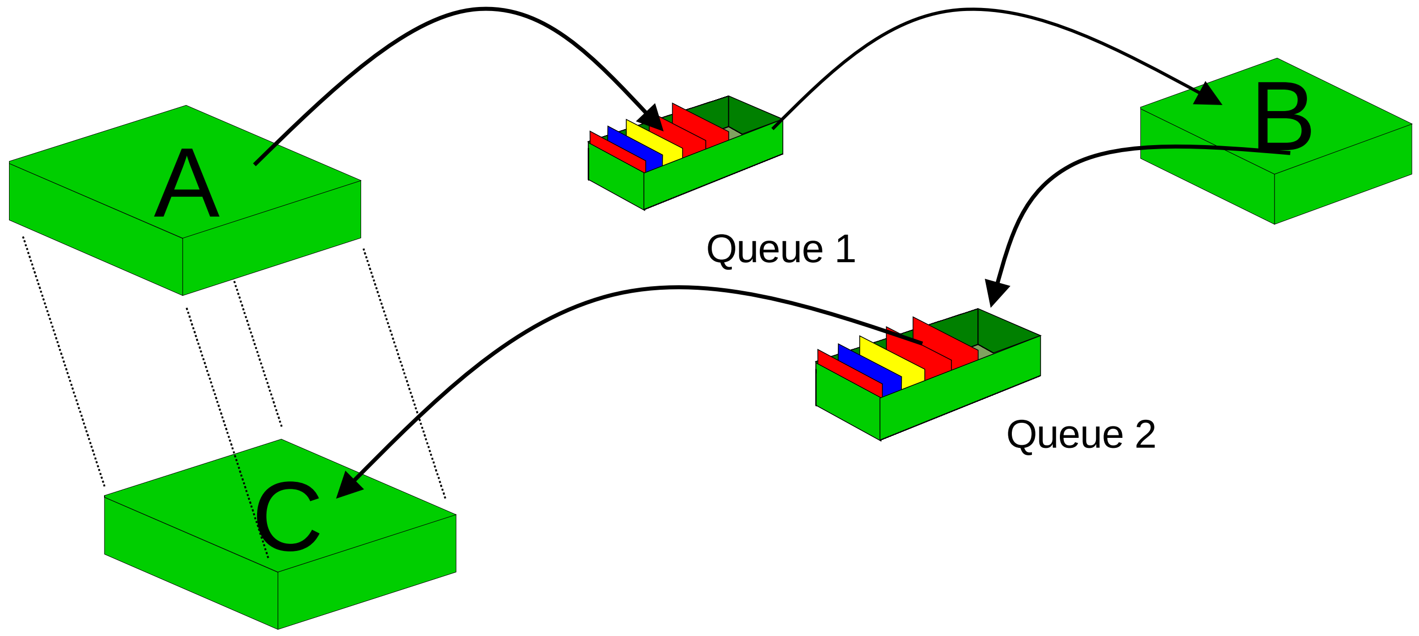
- Well written applications
- Robust network
- A Database
- Good operational procedures
- Well managed systems

WebSphere MQ is not a substitute for…!

N O T E S

- Well written applications
  - As with any middleware, MQ can be used in both good and bad ways. To conform with the desired goals application should be coded correctly
- Robust network
  - While MQ can, and will, deal with network failures it can't do so with it affecting its efficiency. Installations should always strive to have reliable communication links for critical data.
- A Database
  - MQ can provide direct access to transactional data however it is not a database and internally has a considerably different structure. Databases are designed around the 'write once/read many' principal. MQ, on the other hand, is designed around the 'write once/read once' principal.
- Good operational procedures
  - Whenever critical data is stored on system good maintenance procedures is always advised regardless of the reliability of the infrastructure.
- Well managed systems
  - Issues such as security, data backup, software maintenance need to be considered.

WebSphere MQ Key Concepts
Point-to-Point messaging



- **Messages** can be created from any source:
  - *Data, Messages, Events, Files, Web service requests / responses*
- Messages are moved asynchronously using **Queues**

WebSphere MQ Key Concepts
Point-to-point messaging

N

O

T

E

S

- The physical world is frequently organized 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.

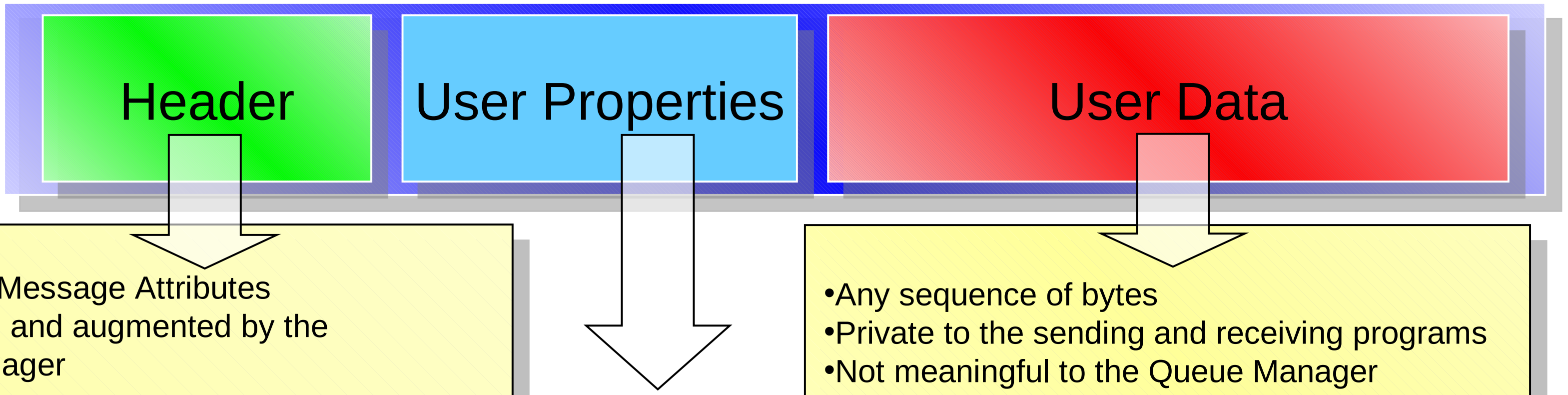
- The truth is that queuing is a natural model that allows us to function efficiently. Perhaps not surprisingly therefore it turns out that it is also a very useful model in which to organise our applications.

- 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:
  - Data in general
  - Data packaged as messages
  - It might be notification events
  - Files being moved in a managed FT application
  - SOAP messages for invoking services

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

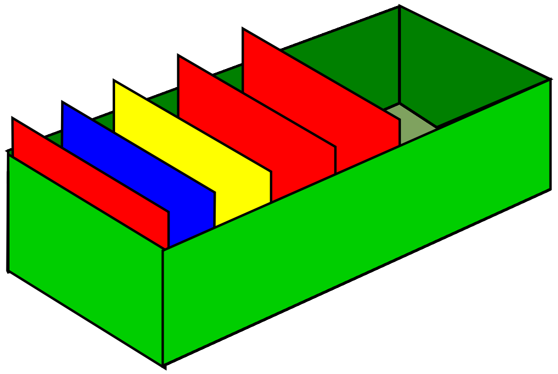
What is a Message?

N O T E S

- A message in WebSphere MQ is merely a sequence of bytes in a buffer of a give 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.

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?

N O T E S

A Queue is a named object (up to 48 characters) which is defined with a queue type.

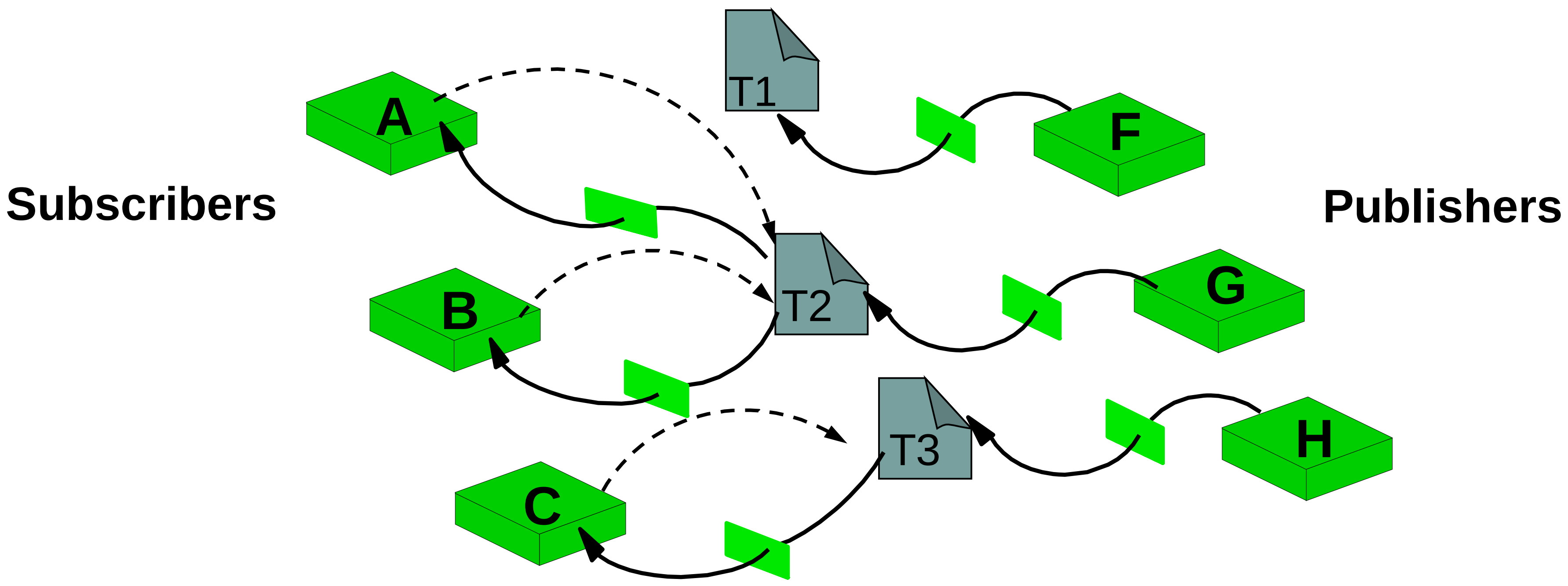
Local	Only queue type which can actually hold messages
Alias	A queue name which 'points' to another queue
Remote	A queue which 'points' to a queue on a remote machine
Model	A template definition which when opened will create a local queue with a new name

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.

WebSphere MQ Key Concepts
Publish/Subscribe



- Publish Subscribe
  - Essentially….

    ------- requesting information on a given **topic**

    _____ providing information on a given **topic**

WebSphere MQ – Key Concepts Publish/Subscribe

N O T E S

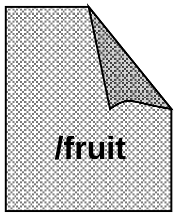
Our daily life is full of examples of requests for information on a given topic and providing information about a given topic.

The main advantage of the Publish/Subscribe model is that the publishers of the information need have no knowledge of who is interested in the information and it therefore keep their processing simple. The system maintains a list of interested parties and will send them data only when appropriate data arrives.

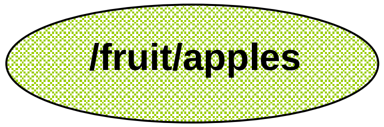
Processing for the subscribers is also simple since they need not know or care exactly where the data is coming from or who is publishing. Provided the data matches the ‘topic’ of interest the data will be received.

What is a Topic?

- A Topic can be
- a) Topic Object
  - is predefined
  - allows you to assign specific non default information to a topics
  - is an access control point



- b) Topic String
  - is a character string
  - can be made up of any characters
  - is case sensitive
    - */fruit/apples*
  - describes the information to be associated with it



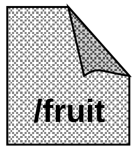
What is a Topic?

N O T E S

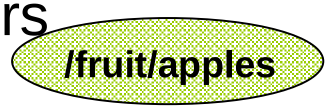
- A topic is a character string that describes the nature of the data that is published in a publish/subscribe system.
- Topics are key to the successful delivery of messages in a publish/subscribe system. Instead of including a specific destination address in each message, a publisher assigns a topic 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.
- 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.

What is a Topic Tree?

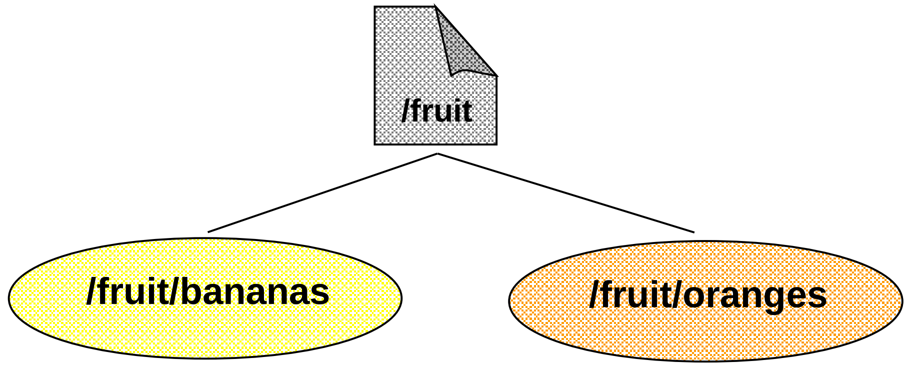
- Each topic defined is a node in a topic tree



- Topic Nodes in the topic tree can be….
  - pre-defined by administrators



  - created dynamically
- Topic Nodes contain Topic Strings



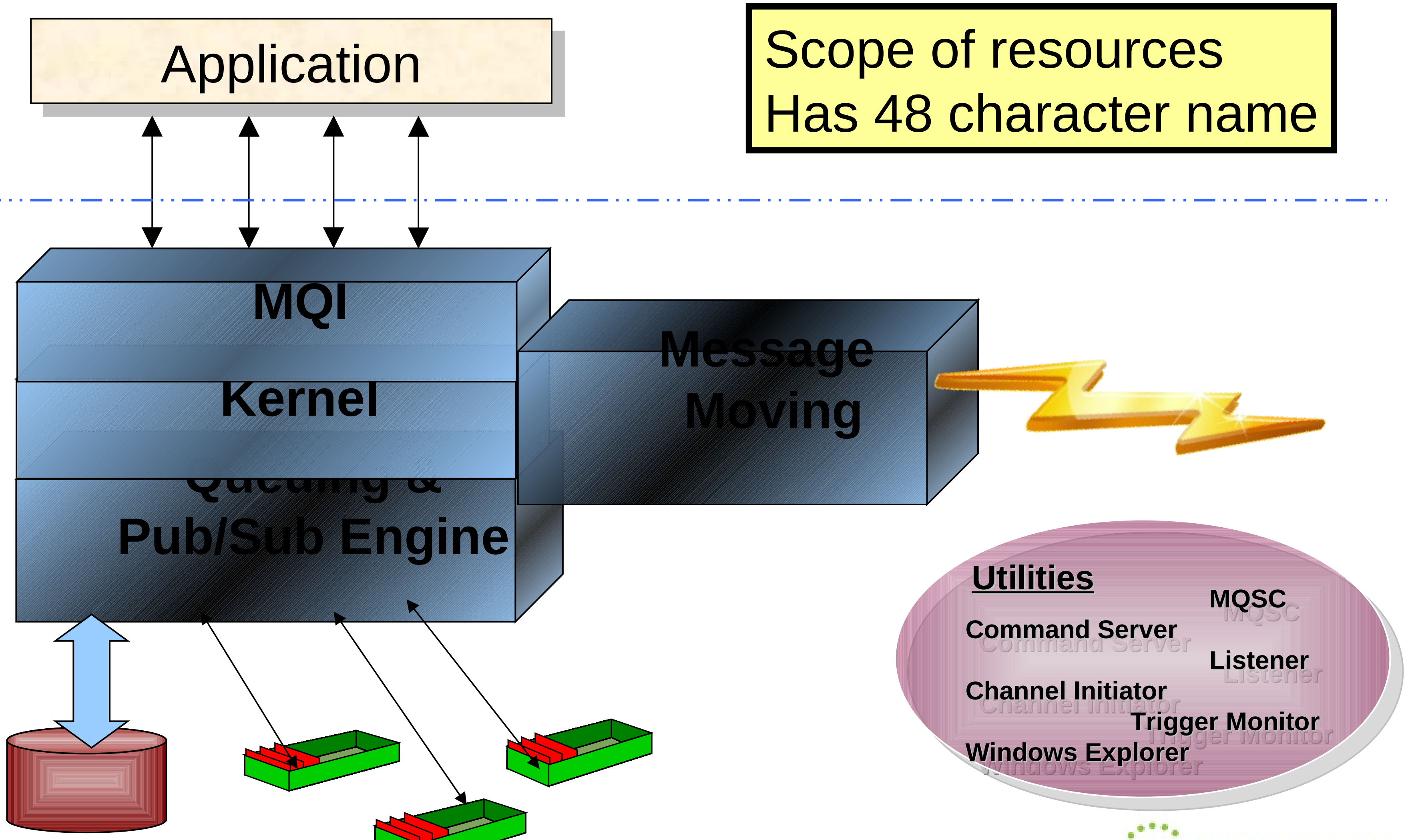
What is a Topic Tree?

N
O
T
E
S

- 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.

What is a Queue Manager ?

Scope of resources
Has 48 character name



What is a Queue Manager?

N

O

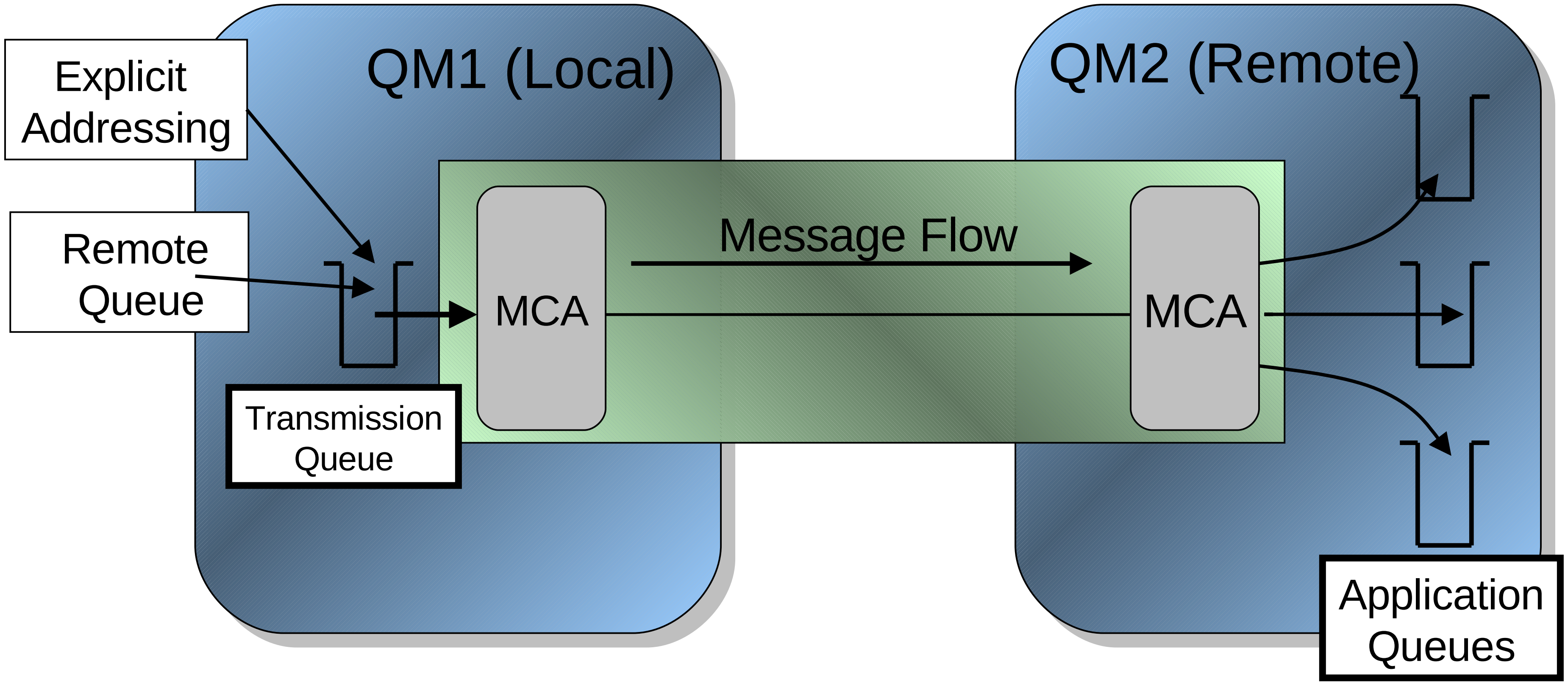
T

E

S

- 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.

Channels



- Transmission Queue accessed via Queue Resolution

Channels

N O T E S

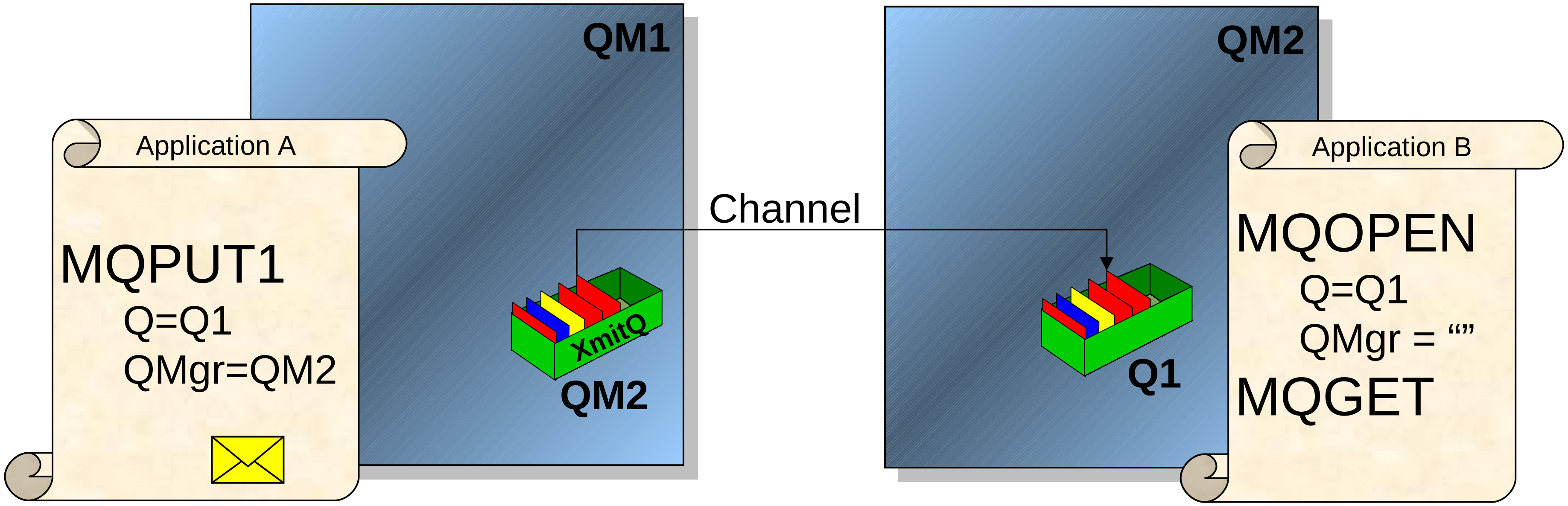
Channels are used by WebSphere MQ Queue Managers in order to exchange messages between Queue Manager implementations. This chart illustrates the various components involved.

When an application opens a queue the queue name resolution process is invoked which determines that the message should be placed on a transmission queue. The transmission queue is local queue which safely stores the message until the channel is ready and able to move the message to the remote system. A transmission queue is often abbreviated to the term ‘xmitq’. Note that **ALL** messages destined for a remote Queue Manager must pass through a transmission queue. The message might never reside of the queue though.

The process responsible for taking the messages from the transmission queue and passing them to the remote system is called a Message Channel Agent (MCA). Similarly there is an MCA at the receiving end of the channel which takes the messages from the network and puts them to the application queues. The pair of MCAs use a transport independent protocol to facilitate once-only, assured delivery of messages.

Channels are designed to be shared across applications. It is usually only necessary to have a single channel moving messages to a target Queue Manager and, usually, a single channel defined in the opposite direction.

Routing Using Direct Addressing



Routing Using Direct Addressing

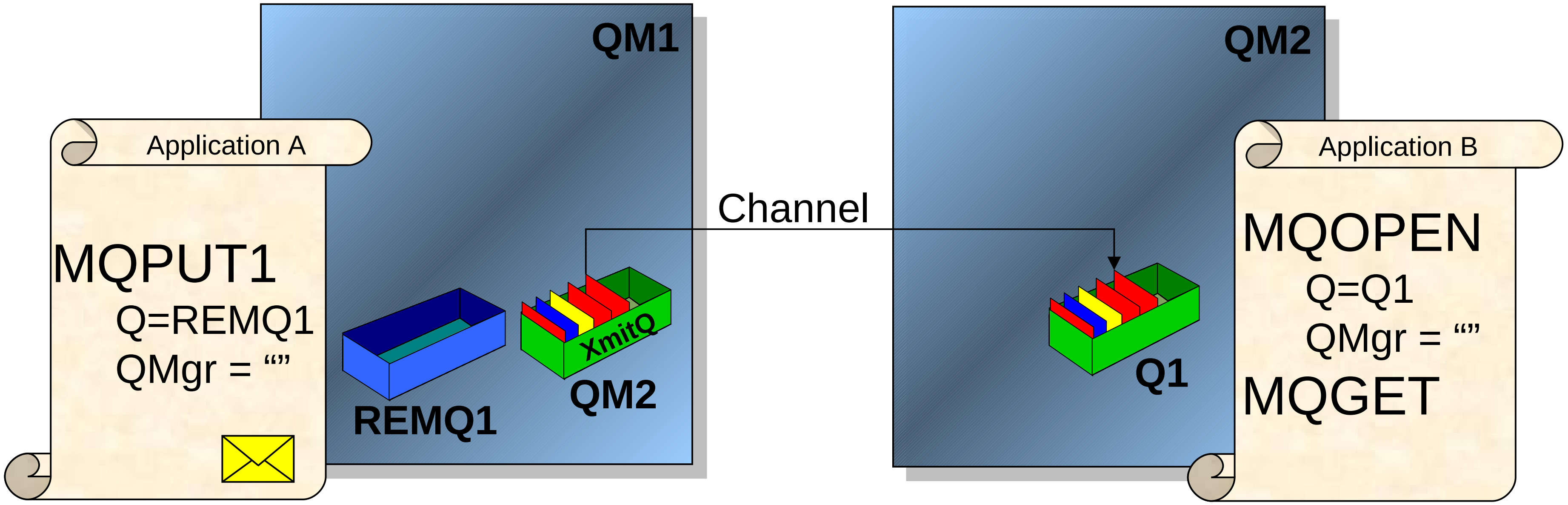
N O T E S

- Here we see an application running on QM1 issuing a PUT to QM2,Q1 (Queue Q1 on Queue Manager QM2). This should really only be done by servers who are passed the name of the Queue and Queue Manager. For clients there are other methods as we'll see later to avoid the application having to know what Queue Manager the message is being sent to. However, this is the simplest......

- Application A is connected to QM1 but issues a put to QM2. The Queue Manager recognises that this message is not destined for itself so it will try to resolve a route for QM2. Essentially what we're saying is that the Queue Manager must determine which transmission queue to put the message on. There are a number of rules the Queue Manager follows to determine the transmission queue which were noted on the previous notes page.

- The most commonly used rule, and the one in this example, is that it finds a transmission queue of the same name. The Queue Manager assumes that a transmission queue named QM2 is going to be serviced by a channel that will send the messages to QM2 (Not unreasonable!).
- So the Queue Manager places the message on the transmission queue with a little header saying that the message is destined for QM2,Q1. Sometime later the channel picks up the message and delivers it to QM2 and issues a put for QM2,Q1. The message has therefore arrived and Application B can retrieve the message.

Routing Using a Remote Queue Definition



```
DEFINE QREMOTE(REMQ1)
       RQMNAME(QM2)
       RNAME(Q1)
```

Routing Using Remote Queue Definition

N O T E S

- Here we see an application running on QM1 issuing a PUT to its own local REMQ1 definition. REMQ1 has been defined on QM1 as a remote queue definition. A remote queue definition effectively points the Queue Manager in the direction of where it should send this message. In this instance its definition contains only a queue name and a queue manager name but it could also explicitly name the transmission queue. The Queue Manager notices that the Queue Manager name in the definition is different than itself and so follows the rules as before of how to determine which transmission queue to put this message to.

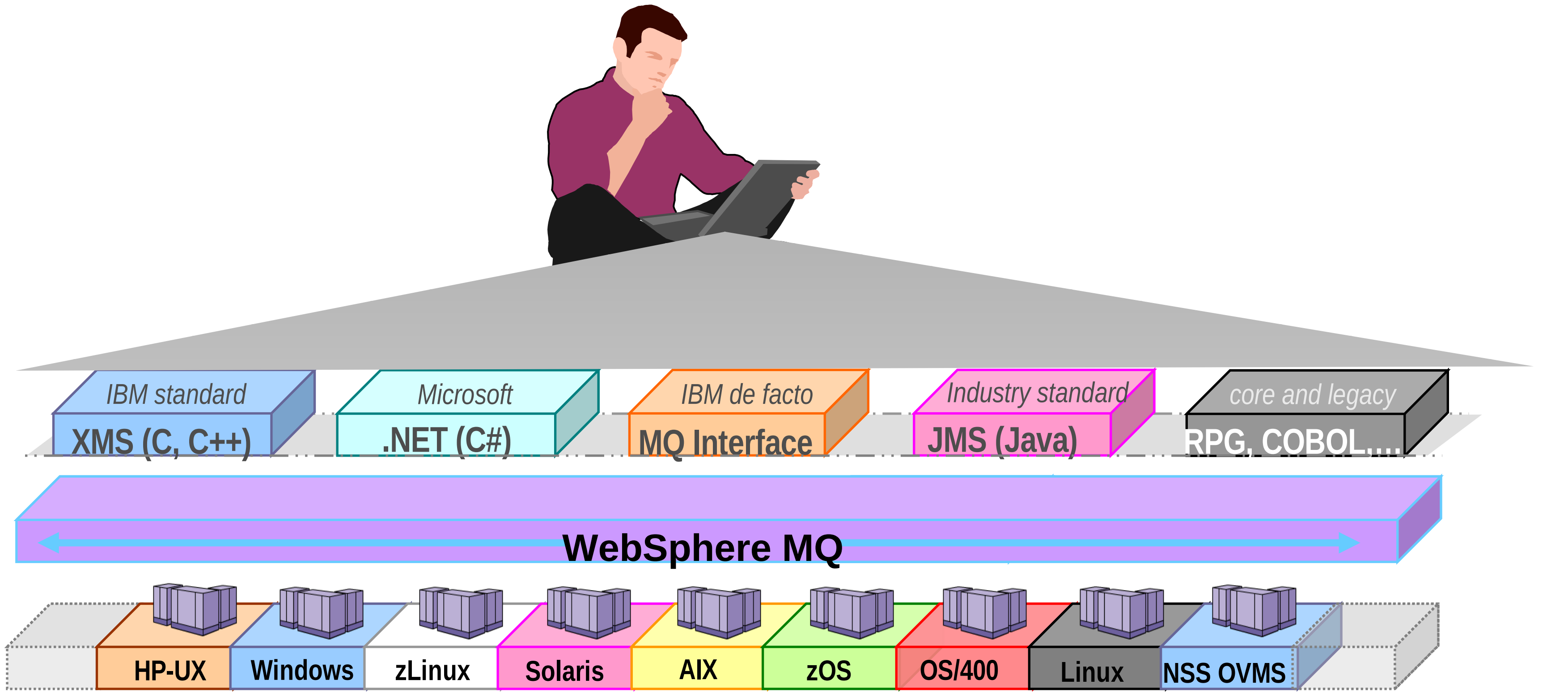
- The advantage of this method is that Application A has no need to know of the actual queue that Application B is reading or the Queue Manager it is running on. Use of remote queues is the recommended approach for client applications.

- Note that the different addressing mechanisms used by Application A are completely transparent to Application B. It's code has remained unchanged in the two scenarios.

Agenda

- Introduction
- Fundamentals – The Key Concepts
- The 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

N O T E S

- 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 defacto API for MQ, providing simple common access across all platforms. Standards based interface such as JMS, and its IBM equivalent for C, C++ and .NET, XMS are also available.

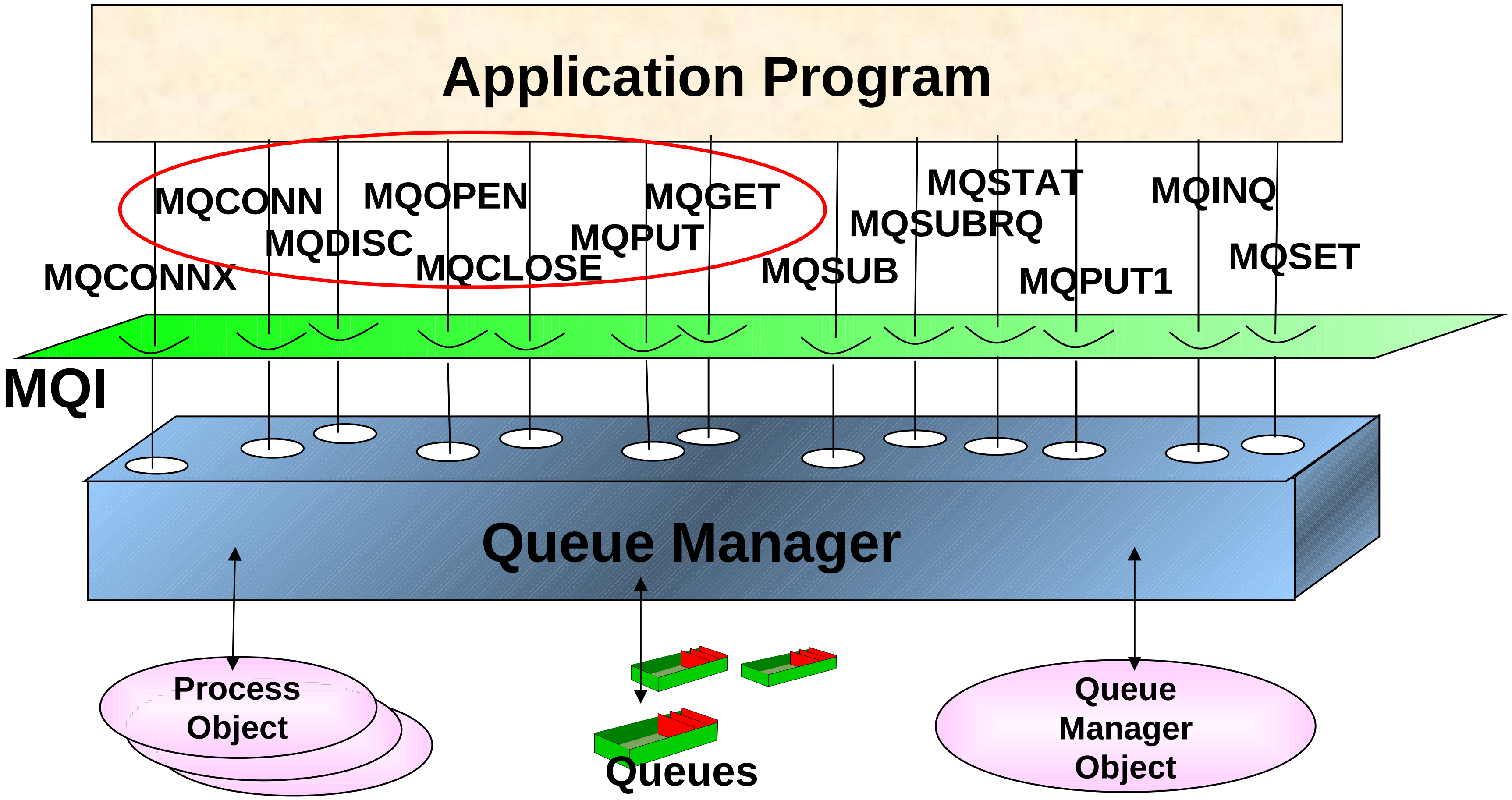
- JMS is part of the J2EE specification and is supported by all J2EE compliant applications servers including; WAS, WebLogic etc. If you are working in Java or a J2EE environment inside an app. server, then you will almost certainly use JMS to access your messaging infrastructure

- So far we've been looking at point to point queue based messaging. JMS can also offer the ability to do publish/subscribe messaging by using a pub/sub engine (broker) built into WebSphere MQ.

- 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 App Server

The MQ API (MQI)



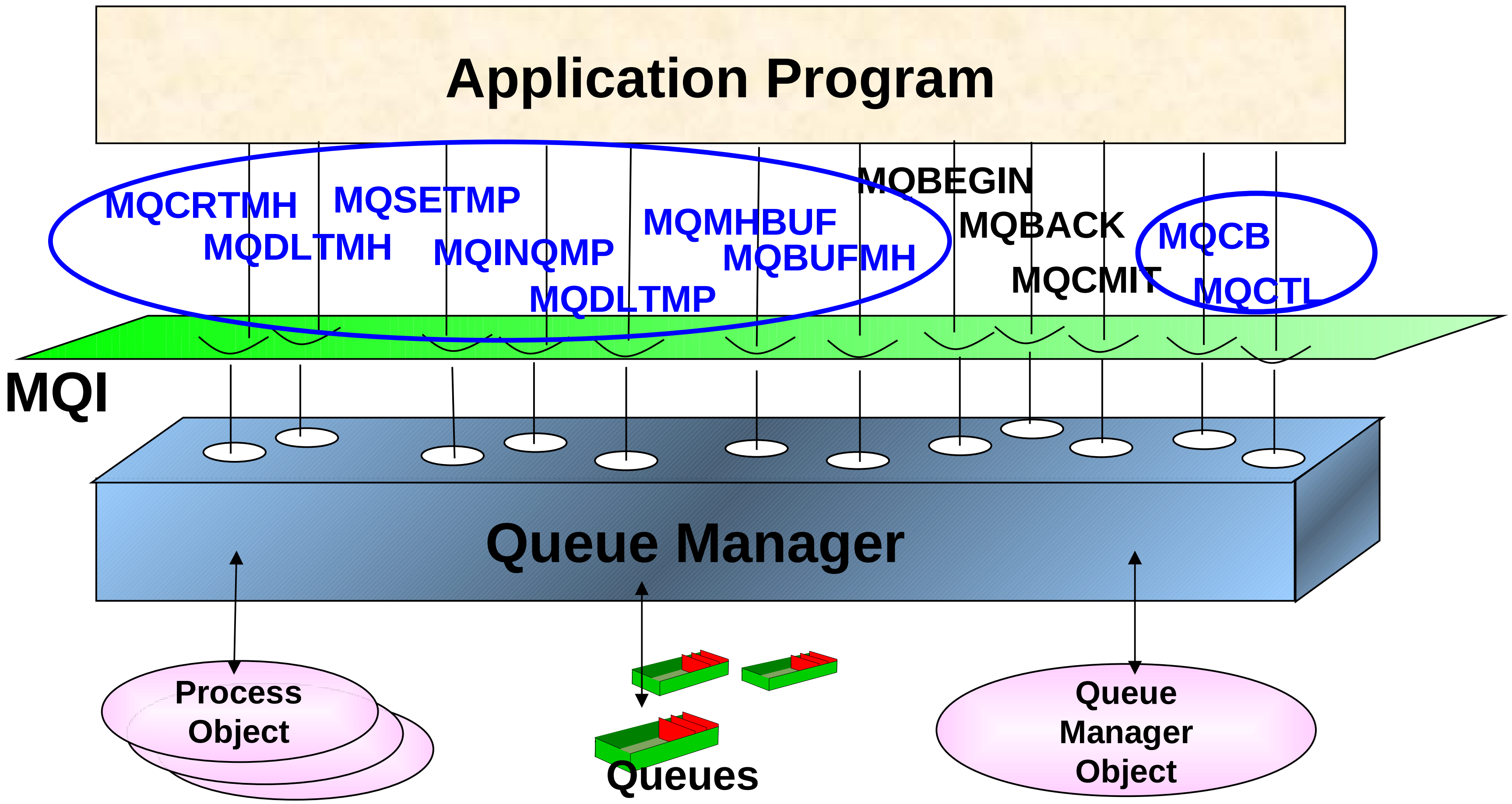
C, C++, C# (.NET), RPG, COBOL, PL/1, Java, Assembler (z/OS), Visual Basic, COM

The MQ API (MQI)

N O T E S

- There are 25 verbs in the WebSphere MQ API, known as the MQI, six of which are most heavily used and the rest have less frequent use. The most common verbs are MQOPEN, MQCLOSE, MQPUT and MQGET which are concerned with the processing of messages on queues. The other verbs have important uses but will not be used as commonly as these four.
- There are many, many options associated with these verbs - approximately 250! 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.
- The MQ API has both a procedural implementation and an object oriented implementation. This allows for straightforward usage in both of these programming environments.All of the popular languages and programming environments are supported, for example:
  - Assembler (z/OS)
  - C, C++, C#, COBOL, PL/1
  - RPG (AS/400)
  - Java, JMS
  - LotusScript, SmallTalk, Visual Basic, COM
  - Plus application environments, e.g.
  - CICS, TXSeries, WebSphere Application Server, IMS, Encina, Tuxedo, MTS, ...

The MQ API continued



The MQ API continued

N

O

T

E

S

- The first set of new MQI calls concern message properties. The functions allow the user to create, change, remove, and examine message properties.

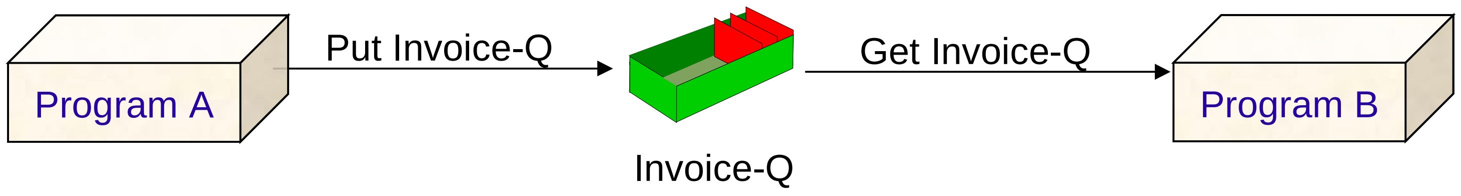
- The second set of verbs, MQCB and MQCTL, are using for asynchronously consuming messages. Essentially the programmer can register a call-back function which is called by the MQ system when either an event or messages arrives. This can greatly simplify programming.

Agenda

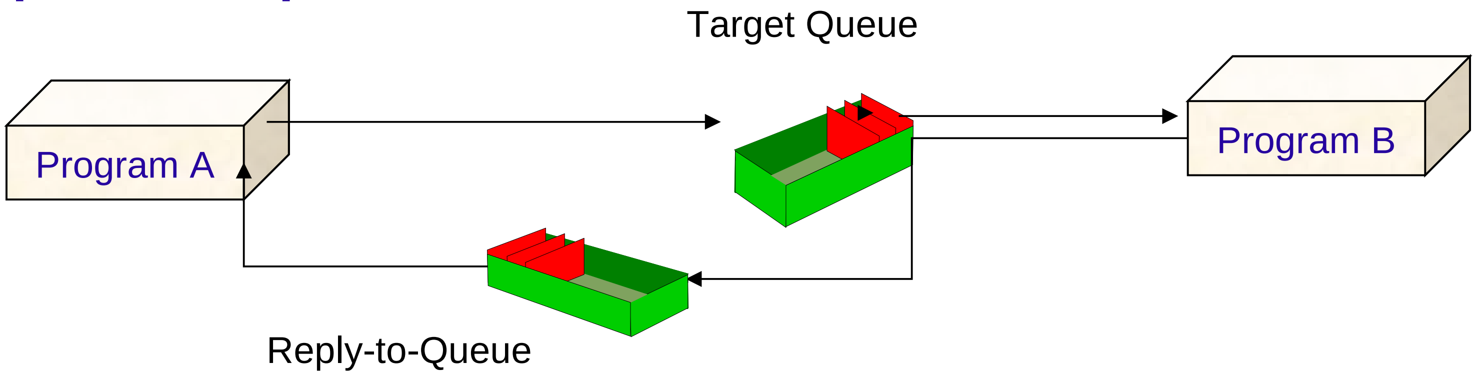
- Introduction
- Fundamentals – The Key Concepts
- The API
- Example Architectures
- Other Key Features
- Related Products
- Summary

Example application architectures (1)

‘Send and Forget’



Request / Response



Example application architectures (1)

N O T E S

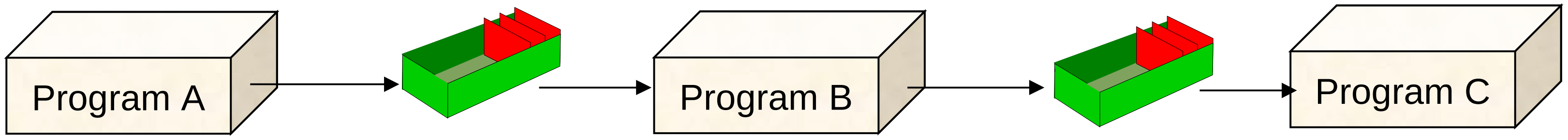
- These examples show some of the ways in which MQ queues can be used and, thereby, shows some of the styles of applications that may benefit from the use of a message/queuing model.

- '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.

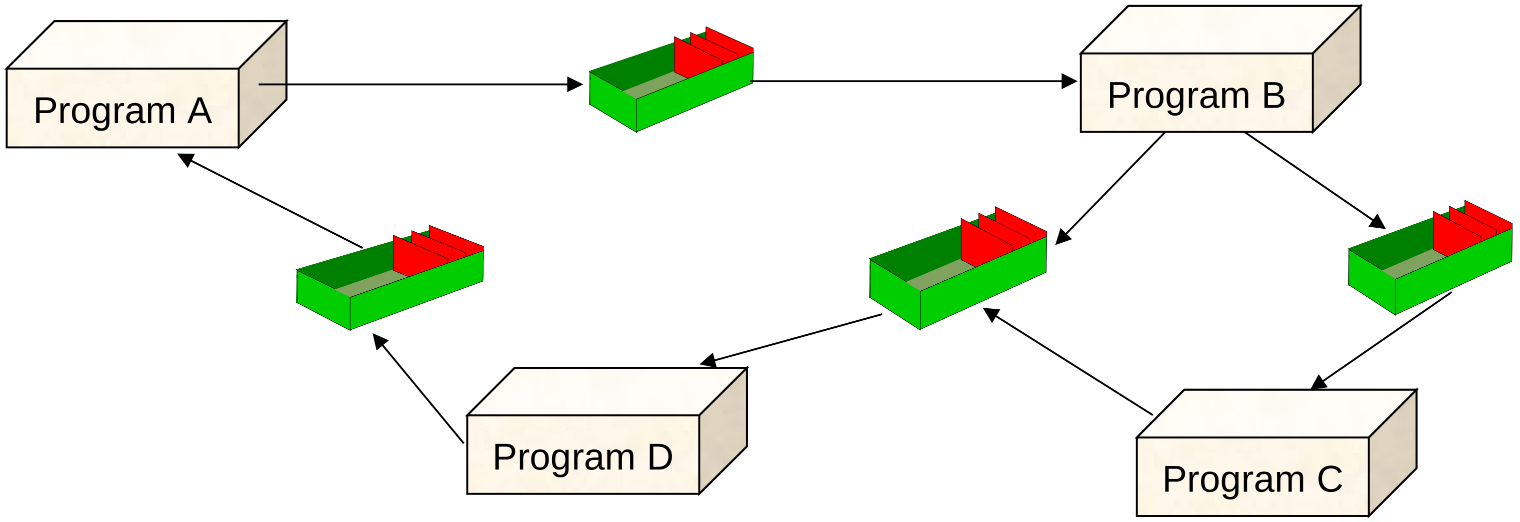
- Request/Response
  This style is typical of many existing synchronous applications where some response is required to the data sent. This style of operation works just as well in an asynchronous environment as in a synchronous one. One difference is that - in this case - the sender does not have to 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 (2)

N O T E S

•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.

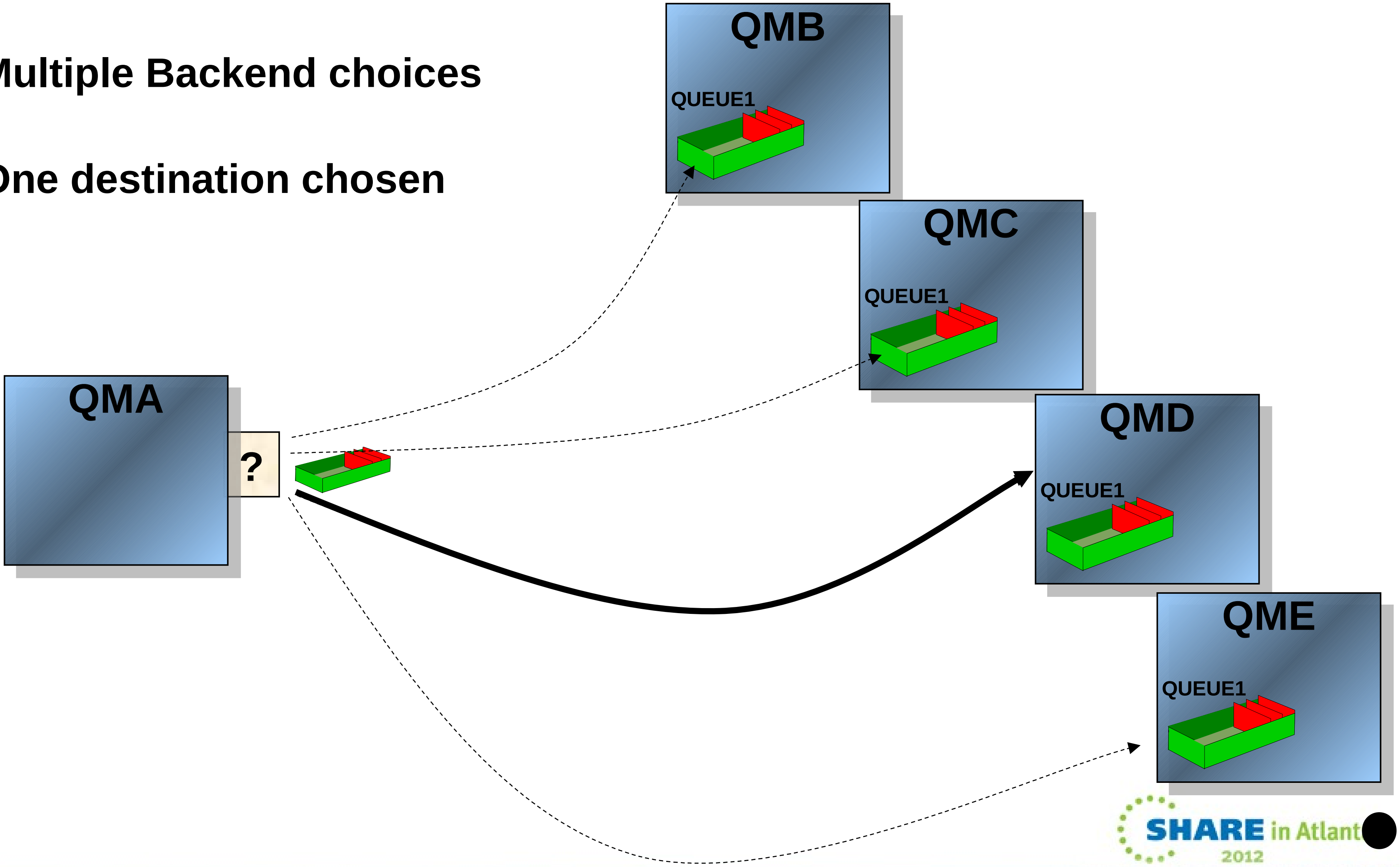
•Workflow
There may be multiple applications involved in the processing before a response comes back to the originating application.

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.

Example application architectures (3) Workload Balanced

Multiple Backend choices

One destination chosen



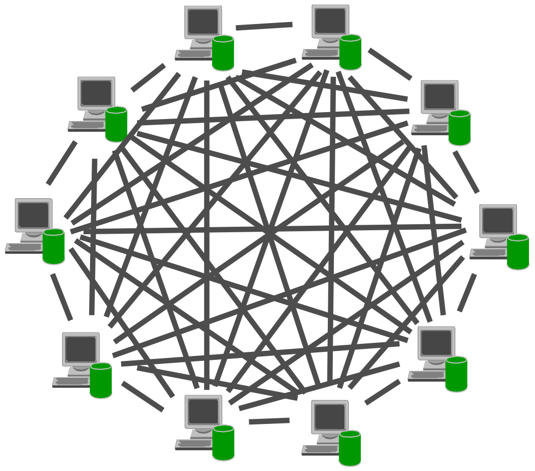
Example application architectures (3) Workload Balanced

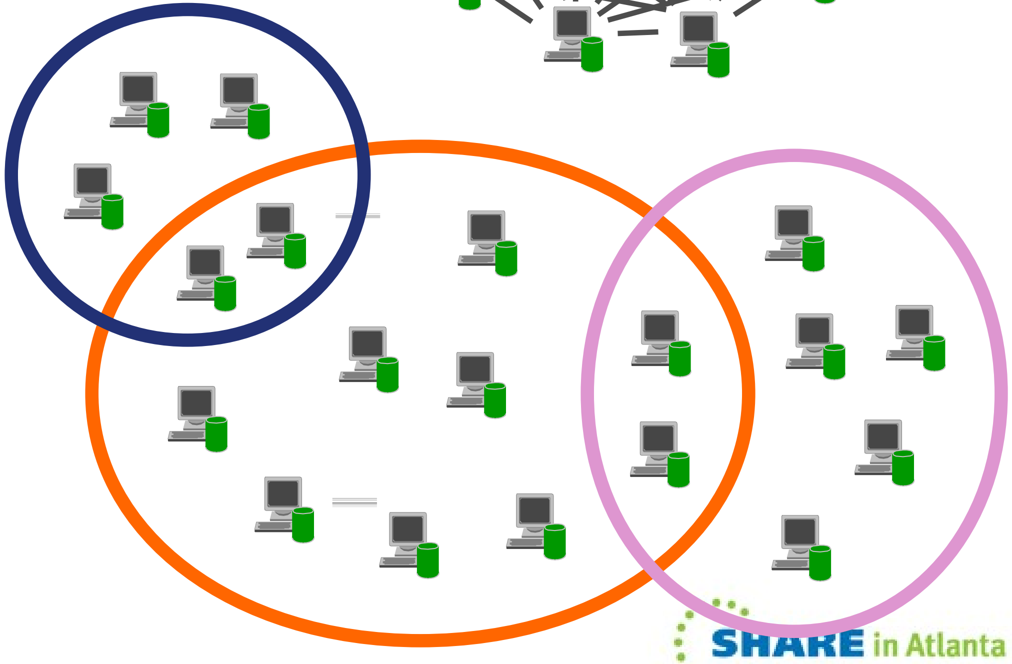
N O T E S

- The final example given here (though not the last possibility by any means) is workload balancing. In order to enable highly scalable applications it is useful to be able to spread the work across multiple backend serving applications.
- To feature in MQ which provides this capability is called 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 MQ resources, such as channels, automatically and provides automatic notification of failed or new queue managers in the environment.

MQ Clusters

- Simplified administration
  - Large WMQ networks require many object definitions
    - Channels
    - Transmit queues
    - Remote queues
- Workload balancing
  - Spread the load
  - Route around failures
- Flexible connectivity
  - Overlapping clusters





MQ Clusters

N O T E S

- It would be nice if we could place all the queues in one place. We could then add processing capacity around this single Queue manager as required and start multiple servers on each of the processors. We would incrementally add processing capacity to satisfy increased demand. We could manage the system as a single entity. A client application would consider itself to be talking to a single Queue manager entity.

- Even though this is highly desirable, in practice it is almost impossible to achieve. Single machines cannot just have extra processors added indefinitely. Invalidation of processor caches becomes a limiting factor. Most systems do not have an architecture that allows data to be efficiently shared between an arbitrary number of processors. Very soon, locking becomes an issue that inhibits scalability of the number of processors on a single machine. These systems are known as "tightly coupled" because operations on one processor may have a large effect on other processors in the machine cluster.

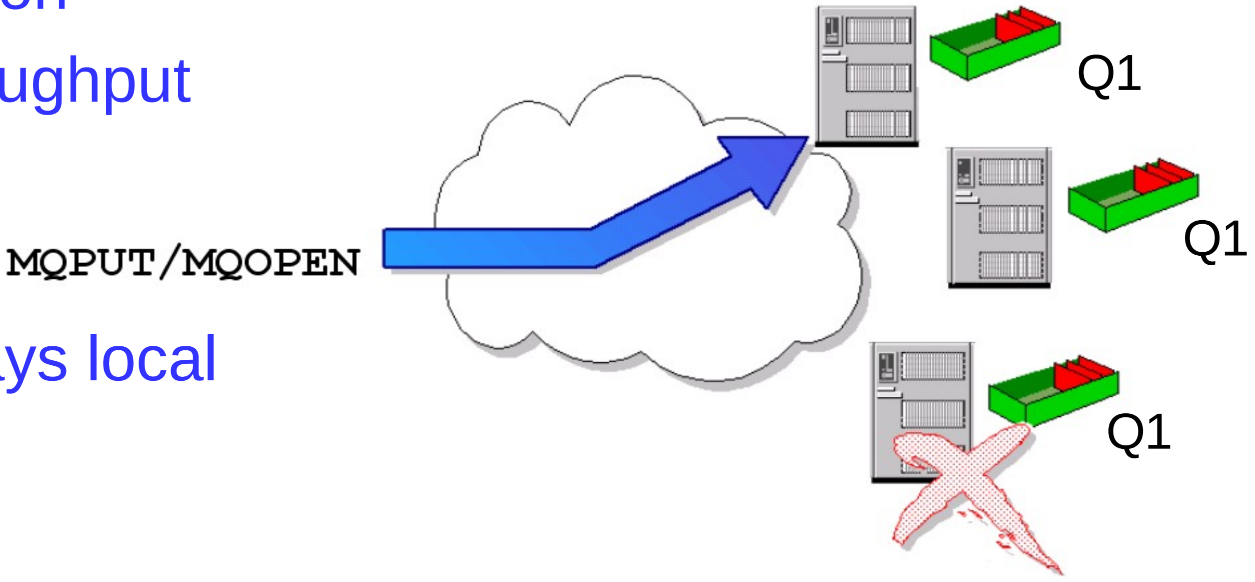
- By contrast, "loosely coupled" clusters (e.g. the Internet) have processors that are more or less independent of each other. Data transferred to one processor is owned by it and is not affected by other processors. Such systems do not suffer from processor locking issues. In a cluster solution, there are multiple consumers of queues (client queue managers) and multiple providers of queues (server queue managers). In this model, for example, the queue ‘Q1’ is available on multiple servers. Some clients use ‘Q1’ on both servers, other clients use the queue ‘Q1’ on just one server.

- A cluster is a loosely coupled system. Messages flow from clients to servers and are processed and responses messages sent back to the client. Servers are selected by the client and are independent of each other. It is a good representation of how, in an organization, some servers provide many services, and how clients use services provided by multiple servers.

- The objective of WebSphere MQ clustering is to make this system as easy to administer and scale as the Single Queue Manager solution.

Goals of Clustering

- Multiple Queues with single image
- MQI applications to exploit clusters transparently
- Definition through usage (MQOPEN)
- Failure isolation
- Scalable throughput



- MQGET always local

Goals of Clustering

N O T E S

- Consider a client using the queue ‘Q1’ that is available in the cluster on three server queue managers. A message is MQPUT by the client and is delivered to *one* of the servers. It is processed there and a response message sent to a ReplyToQueue on the client queue manager.
- In this system, if a server becomes unavailable, then it is not sent any further messages. If messages are not being processed quickly enough, then another server can be added to improve the processing rate.
- It is important that both these behaviors are achieved by existing MQI applications, i.e. without change. It is also important that the administration of clients and servers is easy. It must be straight forward to add new servers and new clients to the server.
- We see how a cluster can provide a highly available and scalable message processing system. The administration point in processing is MQOPEN as this is when a queue or queue manager is identified as being required by an application.
- Note that only **one** message is sent to a server; it is not replicated three times, rather a specific server is chosen and the message sent there. There are methods of sending a message to multiple destinations, such as Pub/Sub or Distribution Lists, but that is beyond the scope of this presentation.
- Also note that MQGET processing is still local, we are not extending MQGET into the network.

Agenda

- Introduction
- Fundamentals – The Key Concepts
- The API
- Example Architectures
- Other Key Features
- Related Products
- Summary

WebSphere MQ Transactions

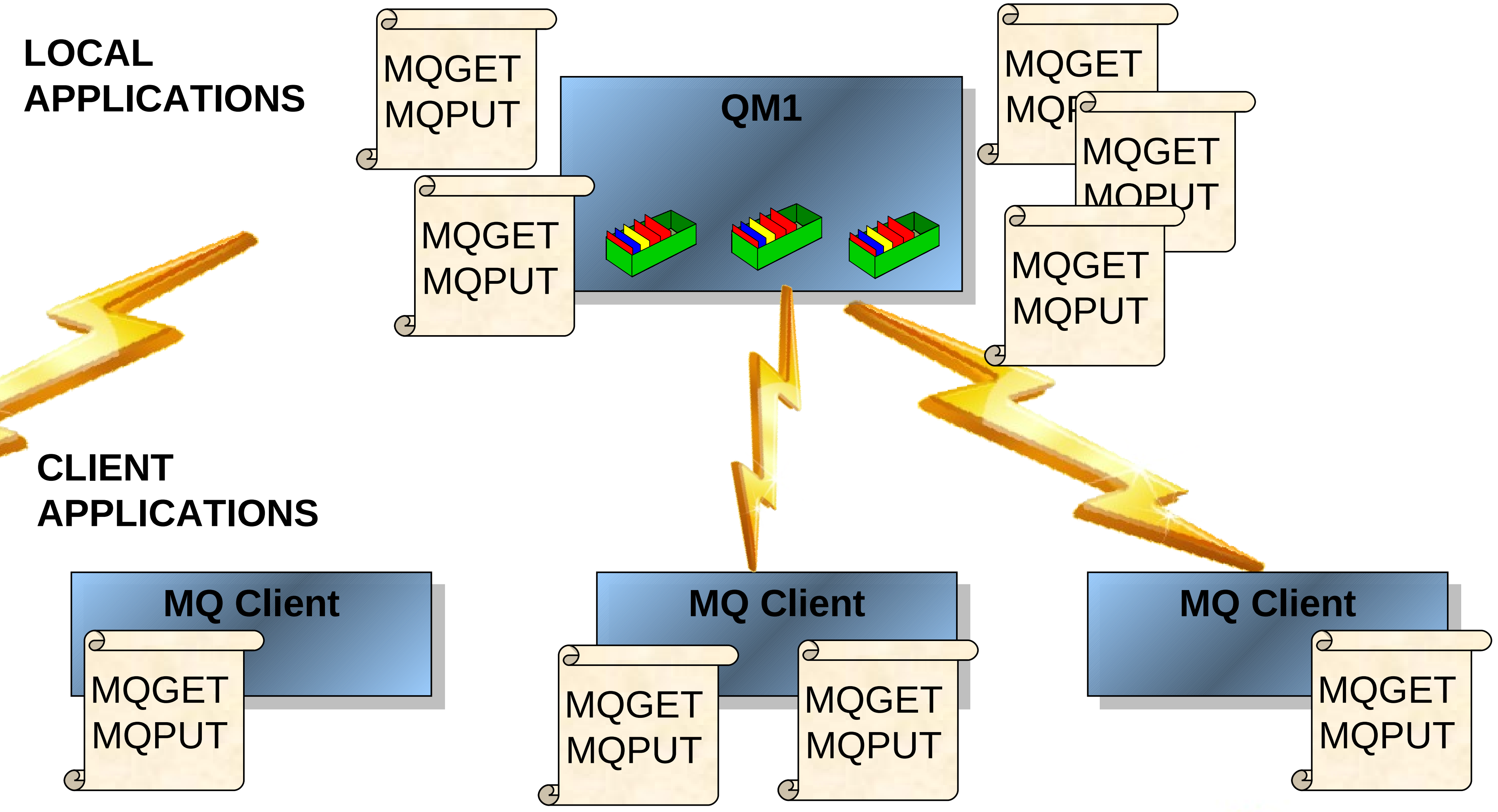
- Message level inclusion/exclusion in unit of work
- Single UoW active per connection at any one time
- WebSphere MQ local units of work
  - MQCMIT and MQBACK control the unit of work
- Messages and other resources in a global unit of work
  - Managed by a Transaction Manager
    - WebSphere Application 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

N O T E S

- WebSphere MQ supports logical 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.

What is an MQ Client?



What is an MQ Client?

N O T E S

- 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.

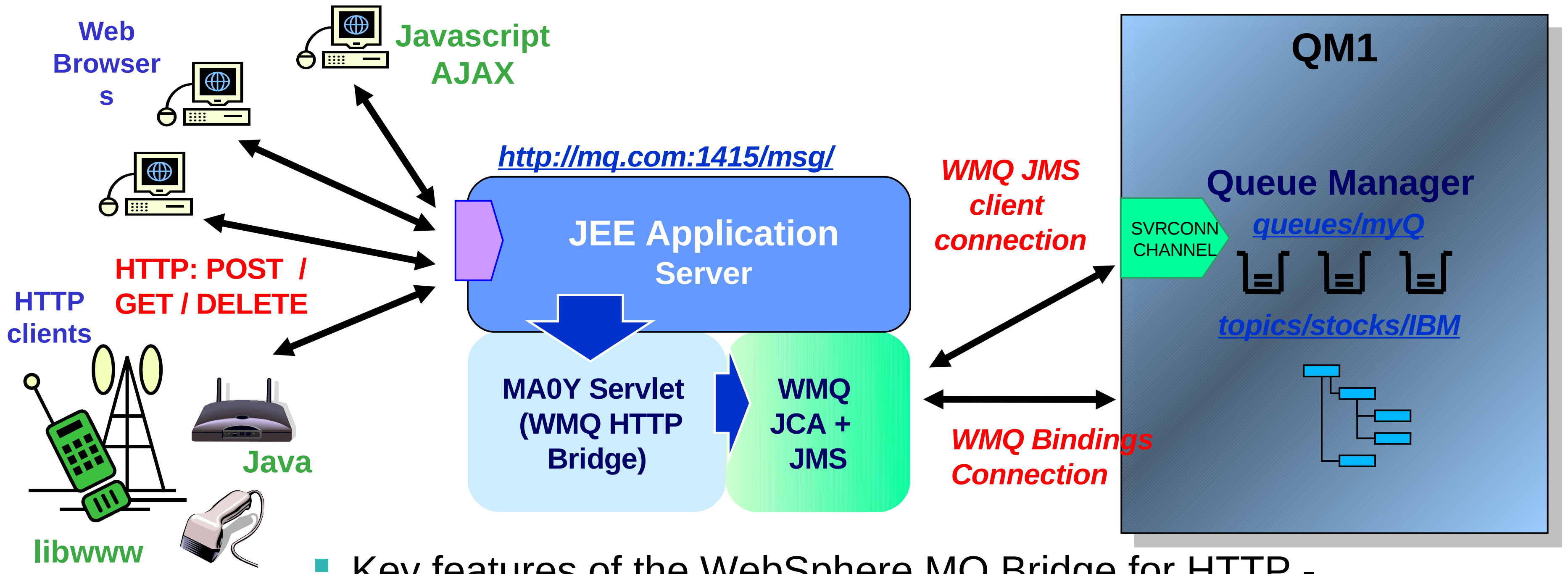
- Each MQ API command is passed to a Server queue manager where a proxy executes the required API command. The connection between client and server is entirely synchronous providing an 'rpc-like' mechanism - though NO regular (well-known) rpc mechanism is used !

- 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.

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
- Alternative implementation as SupportPac MA94

HTTP Connectivity to WMQ

N O T E S

- 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 modelled 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

Example HTTP Flow - POST (= MQPUT)

Put to destination

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
```

Type and length of message (60 char string)

reply Queue

Headers to include on reply

Message Data

Response code

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
```

Required Headers

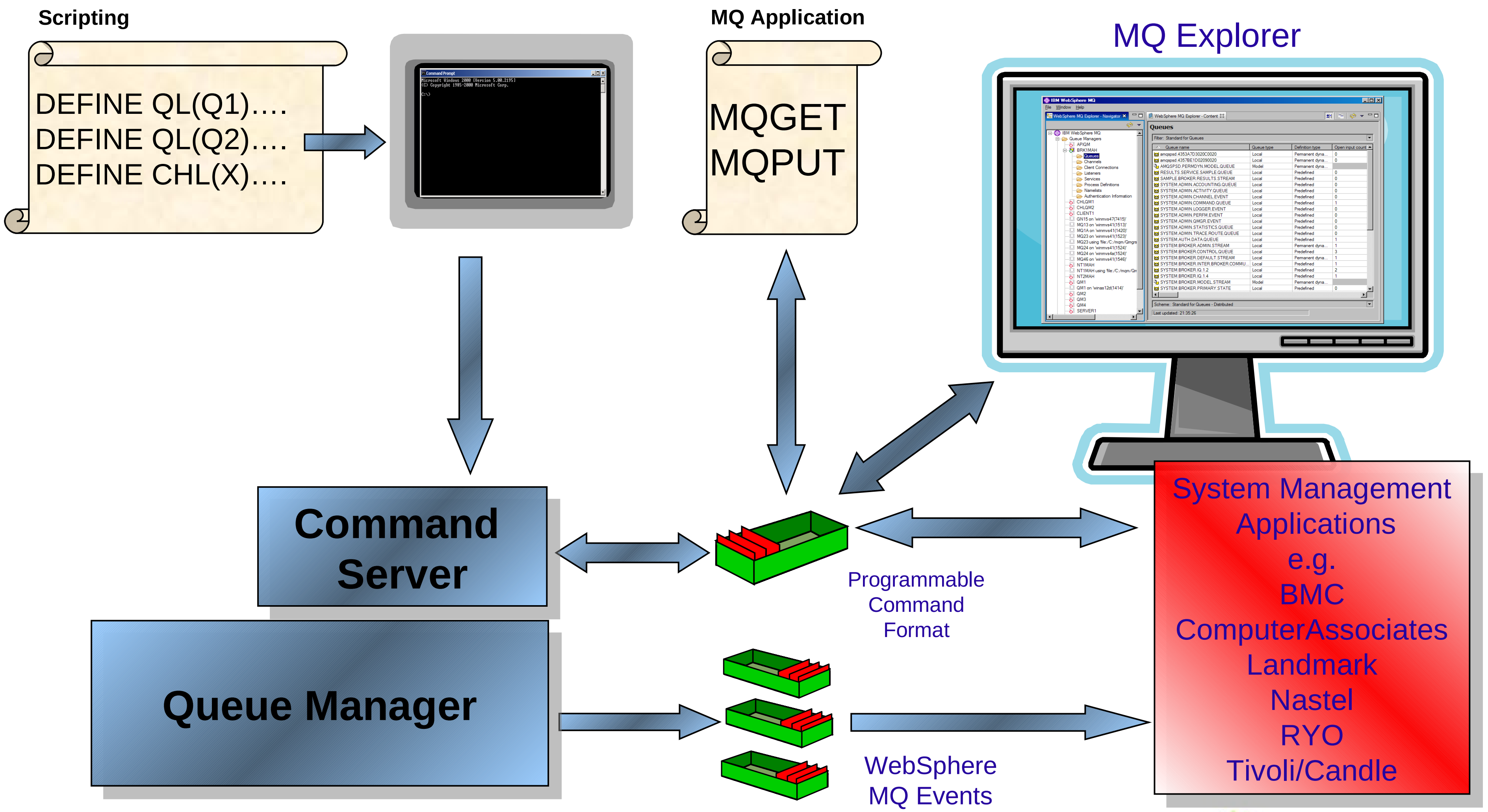
HTTP-MQI Verb / Resource Mapping

N O T E S

- 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**

WebSphere MQ System Management



WebSphere MQ Systems Management

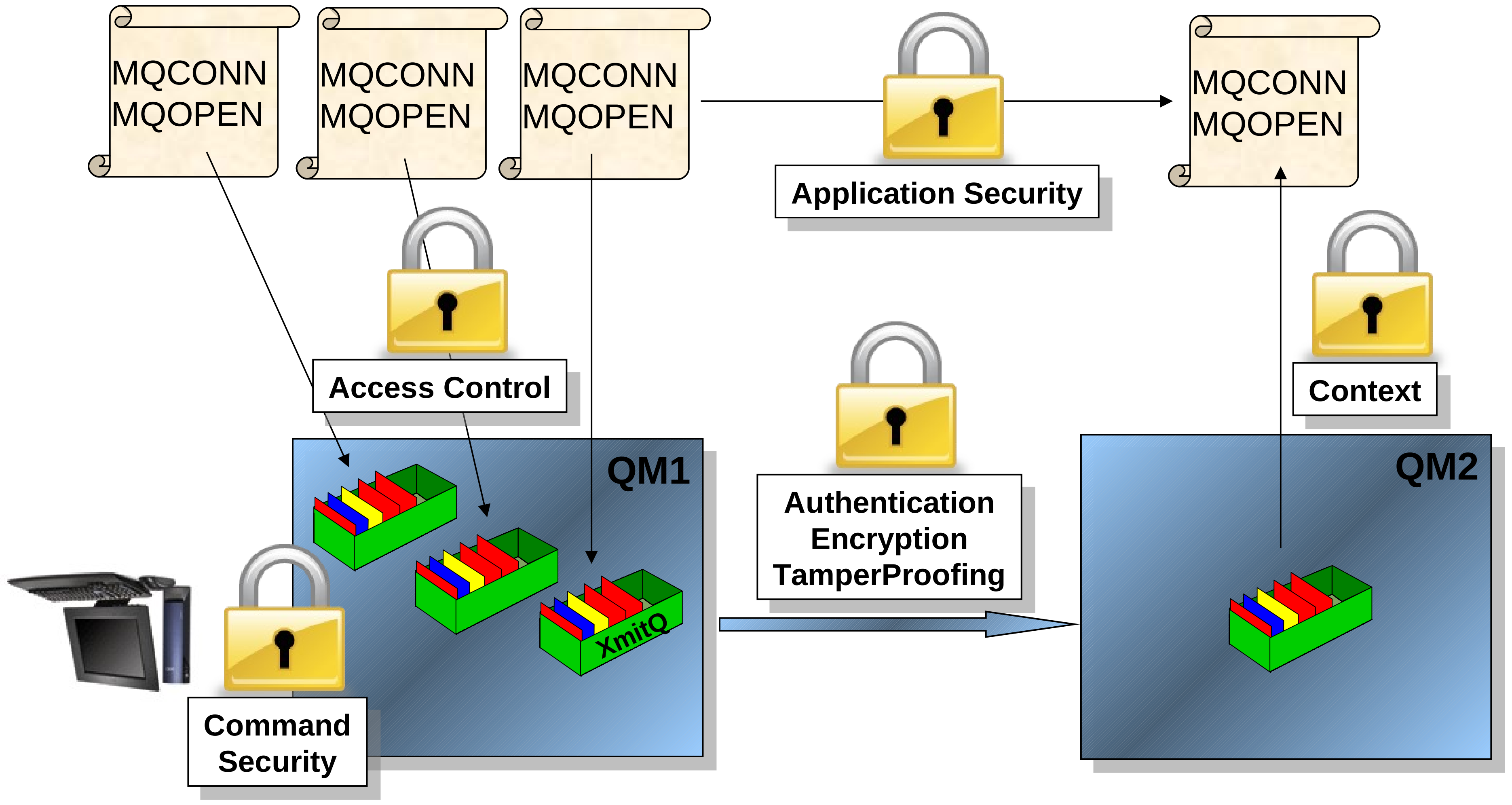
N O T E S

- 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.

MQ Security


MQCONN
MQOPEN
MQCONN
MQOPEN
MQCONN
MQOPEN
Application Security
MQCONN
MQOPEN
Access Control
Context
QM1
QM2
Authentication
Encryption
TamperProofing
XmitQ
Command
Security

MQ Security

N

O

T

E

S

- There are several aspects to WebSphere MQ security:

- 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.

- 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.

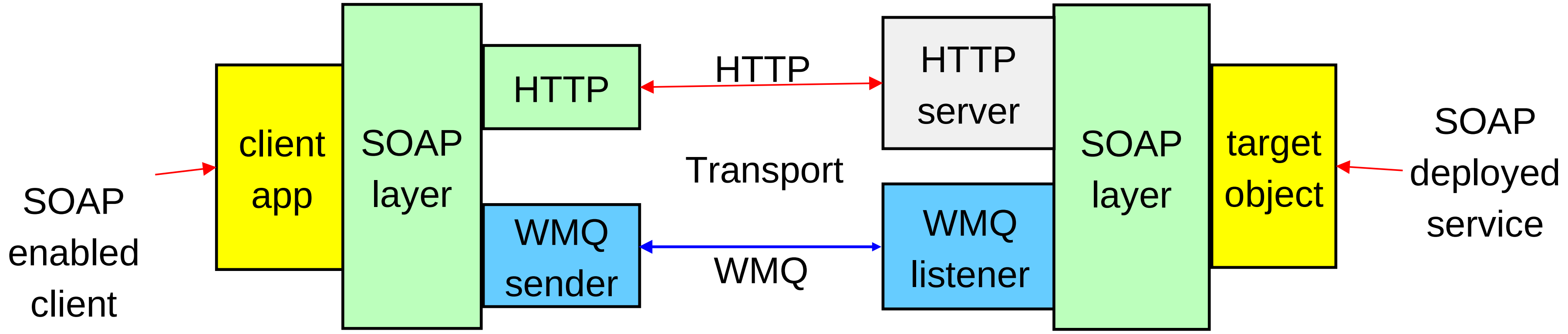
WebSphere MQ Security (contd)

N
O
T
E
S

- 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
- 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.

Extended Security Edition (TAMBI)
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.

SOAP and Web Services over WebSphere MQ



- Transport SOAP messages over a reliable transport instead of http
- Integrates directly into:
  - Axis Web Services environment
  - .NET Web Services environment
  - WebSphere Application Server Services environment
  - CICS Services environment
- Heterogeneous
  - if services interoperate using HTTP, they will interoperate using WMQ
- SOAP / JMS Message Format
  - Soon to be standardized at API level across Vendors
  - Sonic, TIBCO, BEA, Axis

SOAP and Web Services over WebSphere MQ

N
O
T
E
S

- If you had a Web service and client, typically they would communicate using HTTP, but WMQ can be used seamlessly instead.

- The benefits this gives are:
  - Improved Quality of service
  - Better management of a WMQ network than an HTTP network
  - No application changes are required. It is facilitated at deployment time by changing the URI details. If it works over HTTP, it will work over WMQ (administration)

- The SOAP messages are carried formatted as JMS messages, called SOAP over JMS and so can interoperate with other compatible IBM products such as WebSphere Application Server and CICS.

- Other vendors also offer the same ability to carry SOAP messages formatted as JMS, but at present they do not interoperate, however, the leading vendors are now driving a standardisation effort which will help.

- This won’t mean that the messages from different vendors will look the same on the wire, but it will mean that the way that SOAP messages are stored within a JMS message will be the same i.e. by using the same property names, body types etc.

Agenda

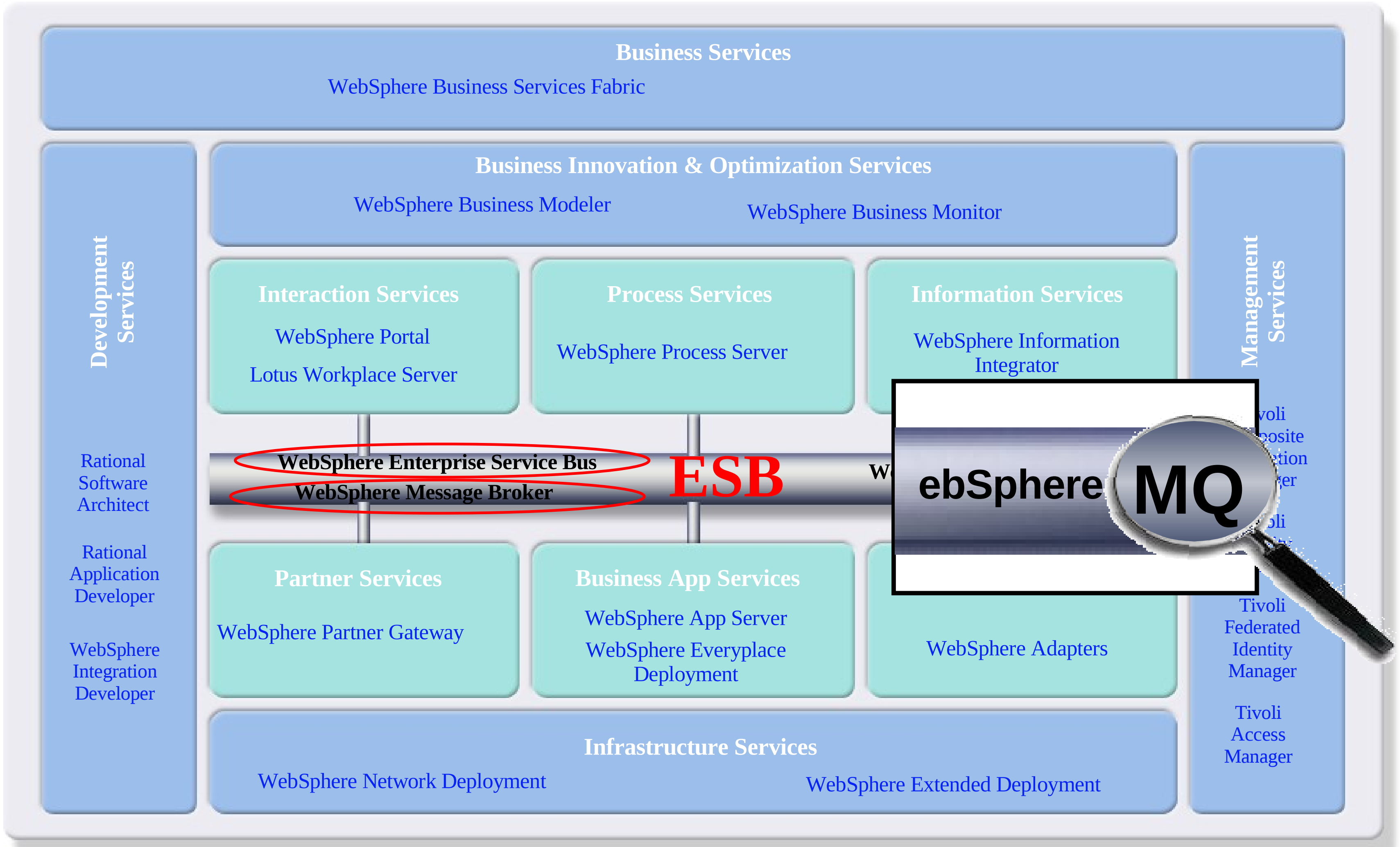
- Introduction
- Fundamentals – The Key Concepts
- The API
- Example Architectures
- Other Key Features
- Related Products
- Summary

WebSphere MQ and the Wider World

N O T E S

- 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

SOA - Reference Architecture

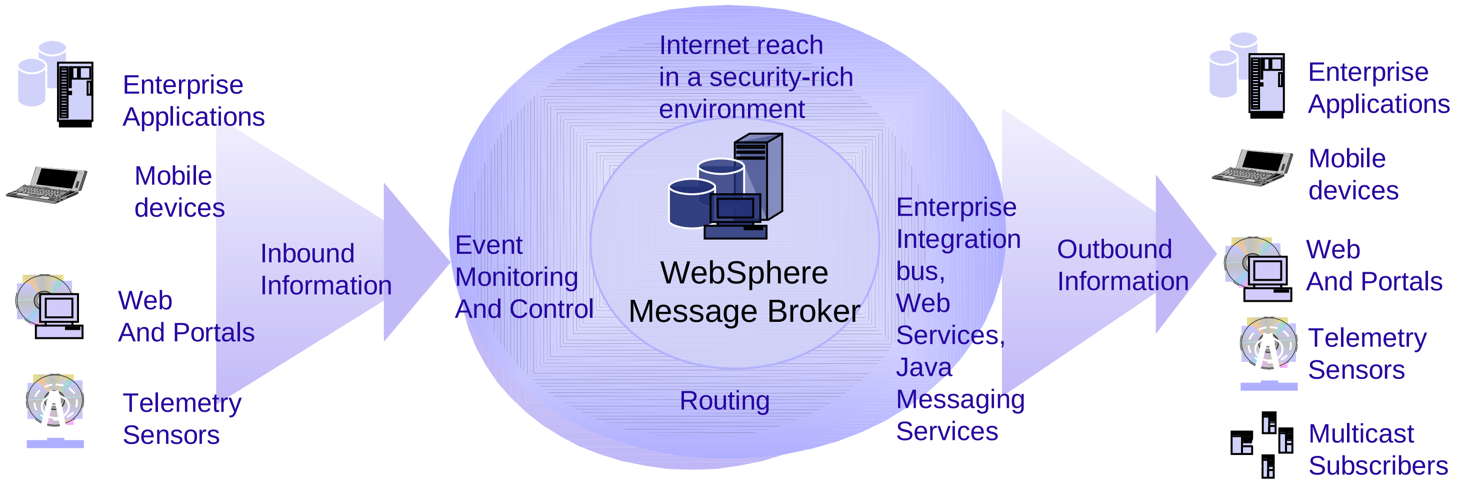

Business Services
WebSphere Business Services Fabric
Business Innovation & Optimization Services
WebSphere Business Modeler
WebSphere Business Monitor
Development Services
Rational Software Architect
Rational Application Developer
WebSphere Integration Developer
Interaction Services
WebSphere Portal
Lotus Workplace Server
Process Services
WebSphere Process Server
Information Services
WebSphere Information Integrator
WebSphere Enterprise Service Bus
WebSphere Message Broker
ESB
ebSphere MQ
Management Services
Tivoli Federated Identity Manager
Tivoli Access Manager
Partner Services
WebSphere Partner Gateway
Business App Services
WebSphere App Server
WebSphere Everyplace Deployment
WebSphere Adapters
Infrastructure Services
WebSphere Network Deployment
WebSphere Extended Deployment

SOA – Reference Architecture

N
O
T
E
S

- This SOA reference architecture diagram is something that you will see time & again. It provides a good way of laying out a roadmap for pursuing SOA.
- At its core is the Enterprise Service Bus. This delivers all of the inter-connectivity capabilities. Many products can be used in this space, however their functionality can be broken down into two main services.
  - The transport backbone, for moving raw data.
  - The mediation and routing services
- WebSphere MQ is an ideal fit for implementing the transport backbone, providing reliable, scaleable, any-to-any linkage.
- It is also tightly integrated into IBMs ESB offerings including the WebSphere Enterprise Service Bus and the WebSphere Message Broker in order to create an ESB with full mediation and routing services.

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 (Mercator)

WebSphere Message Broker

N O T E S

- 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.

WebSphere ESB and WebSphere Process Server

- WebSphere ESB
  - Built on WebSphere Application Server
  - Standards based Mediation, Routing and Adapters for J2EE environments
- WebSphere Process Server
  - Seamless upgrade from WebSphere ESB
  - Process choreography

WebSphere ESB and WebSphere Process Server

N O T E S

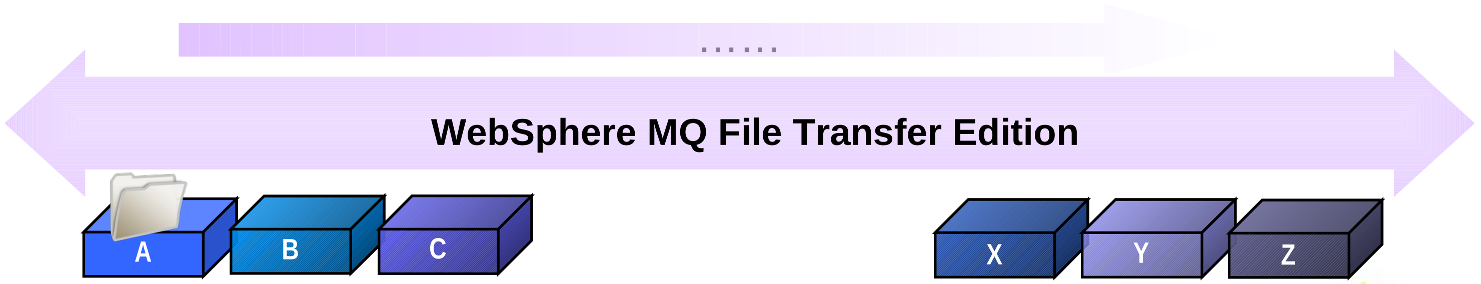
- WebSphere ESB is part of the integrated SOA stack for J2EE environments, using WAS for clustering, load-balancing and integrated with WebSphere Process Server for process orchestration. They are based on standards-based integration, so support J2EE, JMS, XML SCA/SDO.
- By default, these products use the messaging system provided within WebSphere Application Server - WebSphere Platform Messaging. This is a complementary technology to WebSphere MQ for the Java world.
- WebSphere MQ is a plug-in replacement, the choice of which will depend on your requirements:
- Organisations in which the vast majority of components are outside J2EE may benefit from configuring WebSphere MQ as their default JMS provider.
- Organisations in which the vast majority of components are within J2EE may wish to consider using one of the client offerings to connect their native code to the SI bus.
- Organisations where there is not a clear majority either way may want to consider using both messaging systems and connecting them with MQ link or MQ servers.
- Individual circumstances vary – these are just rough guidelines

WebSphere MQ File Transfer Edition

- Adds managed file transfer services to WebSphere MQ
- Enables reliable, secure and traceable file transfers
- Replaces costly, ad hoc solutions that lack management controls

File transfer capabilities

- **Any file size (KB, MB, GB…)**
- **Powerful graphical tooling**
- **No need for programming**
- **Reliability leveraging MQ**
- **Full logging for audit**
- **High-performance**
- **Code page conversion**
- **SSL security**
- **Distributed job automation**
- **Multi-purpose solution – transports both messaging and files**
- **Many supported MQ environments**



WebSphere MQ File Transfer Edition

N O T E S

- The file transfer product is one of the newest products to the family but has already proved to be very popular. A large number of installations still use File Transfer as the basis of their data distribution. However, standard FTP has significant drawbacks.
  - **Limited reliability**
    - transfers can fail without notification, files can be lost
  - **Limited Security**
    - often userid/passwords sent in plain text
    - authentication and encryption often not available
  - **Limited Flexibility**
    - Changes to transfers often require many ftp script updates across multiple machines
    - Often only one FTP transfer can run at a time
    - Typically transfers can not be prioritized
  - **Limited Visibility**
    - Transfers can not be centrally monitored and managed
    - Logging capabilities often very limited

- MQ FTE solves these problems in a easy to use manner which leverages your existing MQ network.

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 r g
- Provides secure, reliable and high-speed infrastructure

This was session WMQ 101 - The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00			Free MQ! - MQ Clients and what you can do with them.	MQ Performance and Tuning on distributed	
09:30		The MQ API for dummies - the basics	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation	The even darker arts of SMF	CICS Programs Using WMQ V7 Verbs
11:00		Putting the web into WebSphere MQ: A look at Web 2.0 technologies The Doctor is in. Hands-on Lab and Lots of Help with the MQ Family	Message Broker administration	The Do's and Don'ts of z/OS Queue Manager Performance	
12:15		WebSphere MQ: Highly scalable publish subscribe environments		MQ & DB2 – MQ Verbs in DB2 & Q-Replication	
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in WebSphere Message Broker V8.0	The Do's and Don'ts of Message Broker Performance	Diagnosing problems for MQ	
03:00	WebSphere Message Broker 101: The Swiss army knife for application integration	What's new in WebSphere MQ V7.1	WebSphere MQ Security - with V7.1 updates	Diagnosing problems for Message Broker	
04:30	Introduction to the WebSphere MQ Product Family - including what's new in the family products	Under the hood of Message Broker on z/OS - WLM, SMF and more	MQ Java zero to hero	Shared Q including Shared Message Data Sets	
06:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	

SHARE
Technology · Connections · Results