

What's New in WebSphere MQ V7 and beyond



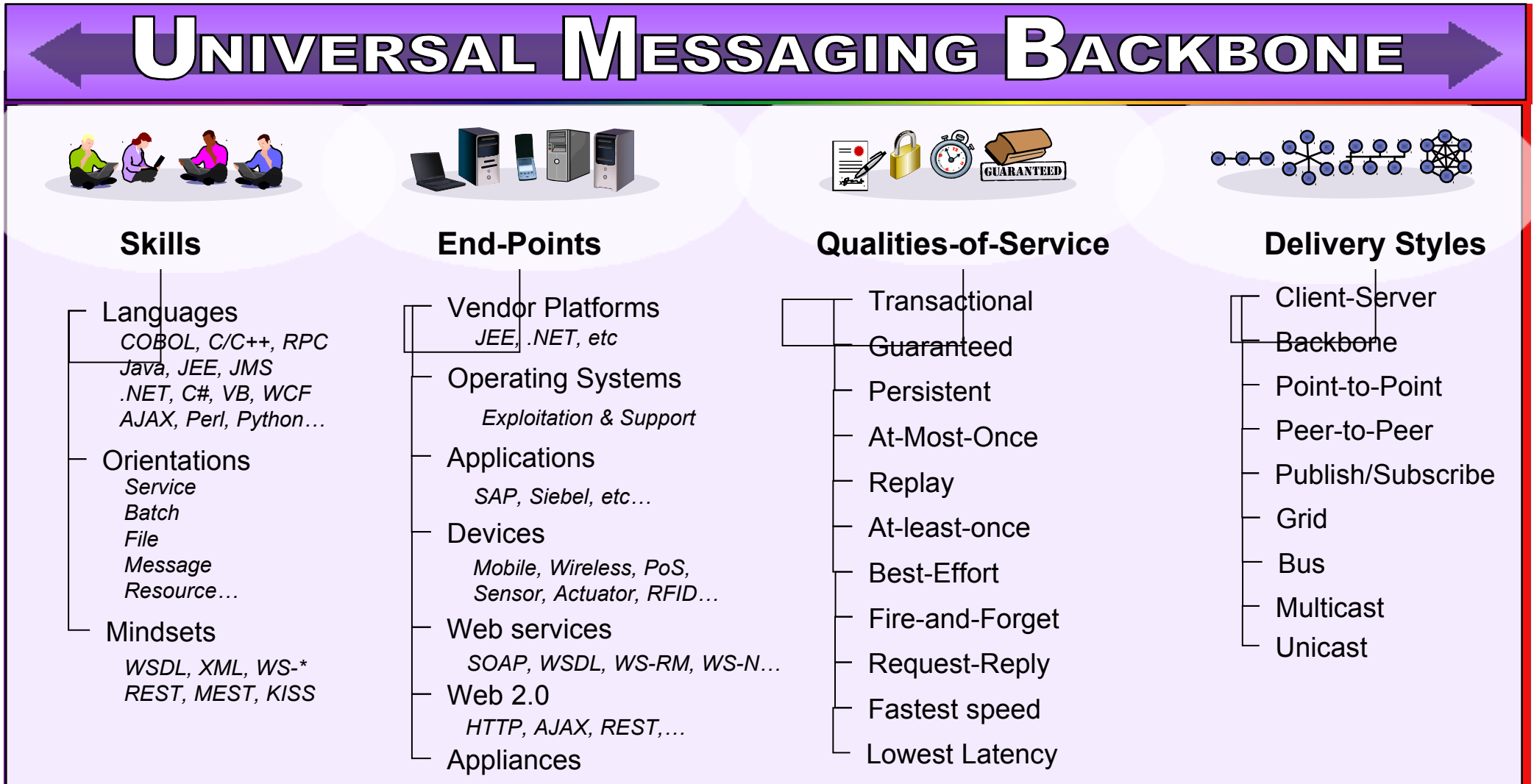
Pete Siddall

siddalp@uk.ibm.com

WebSphere MQ for z/OS Development, Hursley

IBM's Vision – Universal Messaging Backbone

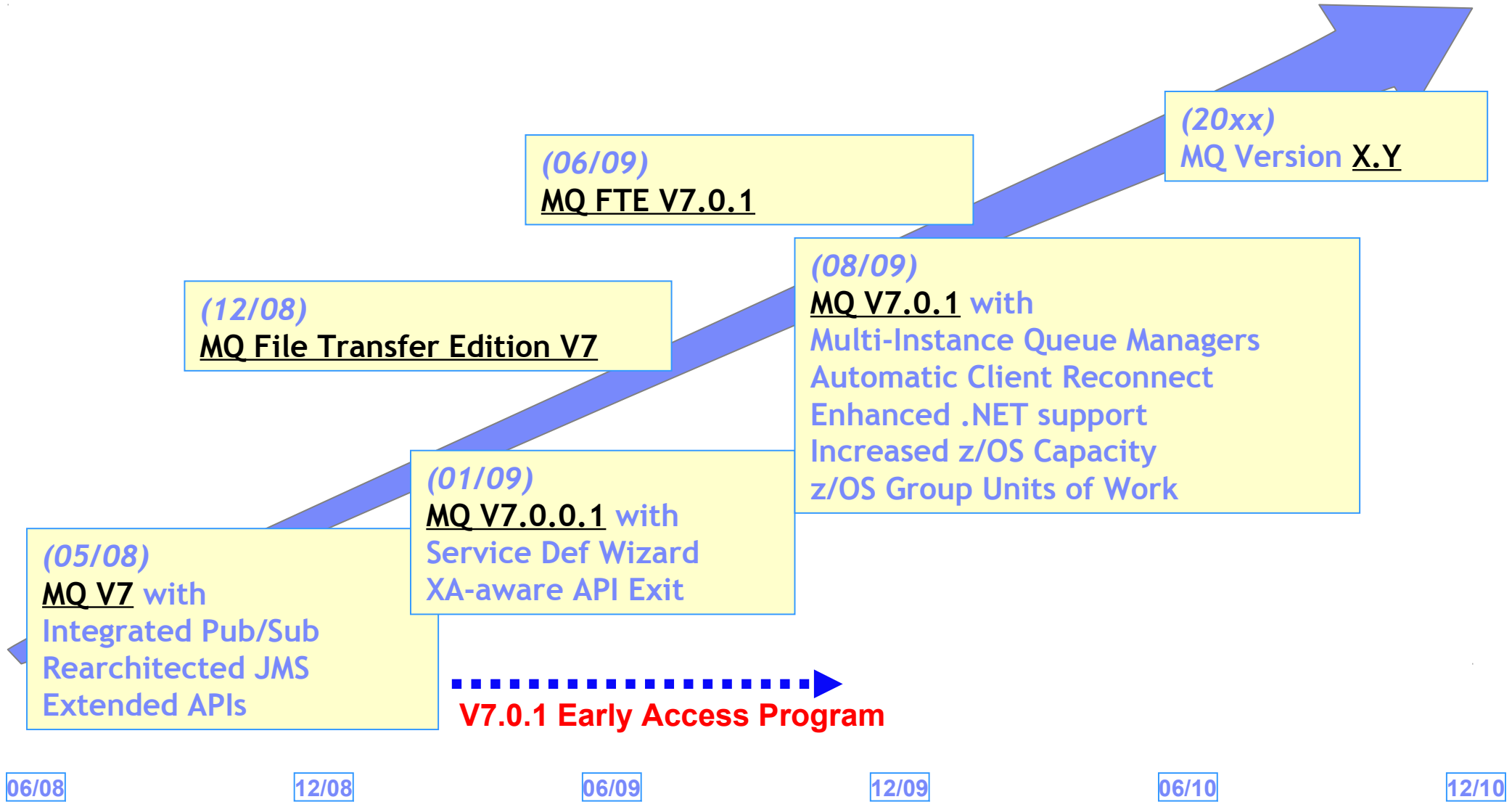
- Addressing the full spectrum of speed and availability transport requirements



IBM's Vision – Universal Messaging Backbone

- The vision for MQ is that it provides a range of capabilities, making it suitable to be a transport backbone across all environments. While MQ does not necessarily have all of these capabilities today in the core product, this picture drives the selection of features as we continue to enhance it.

WebSphere MQ Delivery Roadmap



WebSphere MQ V7

WebSphere MQ Version 7

- Central requirement was to improve JMS implementation
 - More applications being written to use this API
 - Underpins many SOA/ESB solutions needing access to messaging
- Leads to designs involving features such as
 - Ease-of-use
 - Performance
- But it also leads to enhancements for ALL applications
 - Not just JMS users
- Extension of publish/subscribe capabilities
 - Designed with Message Broker in mind
- Easier programming in any environment
 - Some features suggested by JMS requirements are useful in MQI
- Administration model and APIs natural evolution of existing interfaces

WebSphere MQ Version 7

N

O

T

E

S

- WebSphere MQ (WMQ) Version 7 takes its central theme from requirements to extend its support for JMS applications. WMQ has supported the JMS API for many years but recent growth in user applications written to use the JMS API, and growth in SOA solutions that need to exploit a JMS provider for access to messaging (for example as part of an Enterprise Service Bus), have increased the importance of this programming interface.
- This has led to incorporation of a number of JMS features into the core WebSphere MQ engine, and in turn has allowed these features to be easily accessible to applications using the native MQ API (MQI), not just those written to the JMS API.
- The publish/subscribe messaging pattern is one of the two patterns defined by the JMS API, the other being the point-to-point matter. WebSphere MQ V7 therefore extends its publish/subscribe provision. WebSphere MQ Version 7 provides publish/subscribe capabilities that exploit the natural high availability capabilities in WMQ including queue manager clustering. It incorporates a comprehensive range of qualities of service for message delivery, ranging from assured delivery to lightweight non persistent. It also permits a range of topologies, including support for very large numbers of subscribers.
- The pub/sub implementation in V7 has been designed to ensure that we can reach a simplified, and fully-integrated, relationship with Message Broker's pub/sub capabilities.
- WebSphere MQ Version 7 provides application programming interfaces that support the concepts required by JMS, including publish/subscribe, in the MQI, which JMS exploits.
- Enhancements to the programming and administrative interfaces are all modelled after existing WMQ interfaces, to deliver a familiar and easy-to-learn route towards the new features.
- This presentation is not an exhaustive list of all of the new features of WebSphere MQ Version 7, but gives an overview of major functions.

Platforms

- Essentially the same platforms as V6
 - AIX, HP-UX (x2), Solaris (x2), Linux (x4), i5/OS, z/OS, Windows
 - Updates to base OS levels
 - See <http://www-306.ibm.com/software/integration/wmq/requirements/index.html>
- Only one product for Linux/zSeries
 - 31-bit version is removed; 64-bit edition continues
- Drop Windows 2000
 - Windows XP is base level
 - Vista supported
- Windows x64
 - Adds 64-bit application support to single Windows package
 - Supporting existing 32-bit applications
 - Some exits will require recompiling to support both 32 and 64-bit modes
- Java 1.4.2 and later

Platforms

N

O

T

E

S

- The set of supported platforms is essentially the same as in the previous version of WMQ. There are changes to the base levels of operating system that are supported, driven by the current status of each OS. V7 ships 12 different versions, to handle the different operating system and hardware combinations.
- In V6 there were two versions of WMQ for Linux/zSeries, one supporting only 31-bit applications and a second that incorporated 64-bit support. For V7 the first of these has been dropped, but the 64-bit version is fully compatible with 31-bit applications.
- One of the changes to Windows support is that Windows 2000 is no longer a supported platform. Windows XP is the oldest level that can be used. With this release we include support for Vista. There is also support for the x64 versions of the operating system. When running on a 64-bit platform, the queue manager itself is still a 32-bit process, but support is now included for 64-bit applications to use this queue manager. Exits that run inside application code, namely API Exits and Data Conversion Exits, will need to have both 32-bit and 64-bit versions available. 64-bit clients would need 64-bit channel exits as well. There is only one version of WMQ for Windows; it incorporates support for all the variations of the operating system.
- Java 1.4.2 is the lowest level of runtime supported; later levels are also supported
- For z/OS the prereq level is 1.8. (z/OS 1.7 has its end of service set to September 2008.)
- For most recent information on the SOE (Supported Operating Environments), always look at the web page. That is kept updated in between releases of the product.

Publish/Subscribe

- A natural part of the JMS API
 - Combines both Publish/Subscribe and Point-to-Point patterns
 - Now also a natural part of the native MQI
- Point-to-point asynchronous messaging decouples applications
 - But still implies a one-one relationship between sender and receiver
- Publish/subscribe is a further stage of decoupling
 - Sender has no direct knowledge of how many (if any) apps will see a message
 - Link between applications is a **Topic**, not a **Queue**
- WMQ V6 (Distributed) included a Publish/Subscribe broker (formerly MA0C)
 - Compatibility mode available in V7
- Implementation substantially improved with V7
 - And is available for the first time on z/OS

Publish/Subscribe

N

O

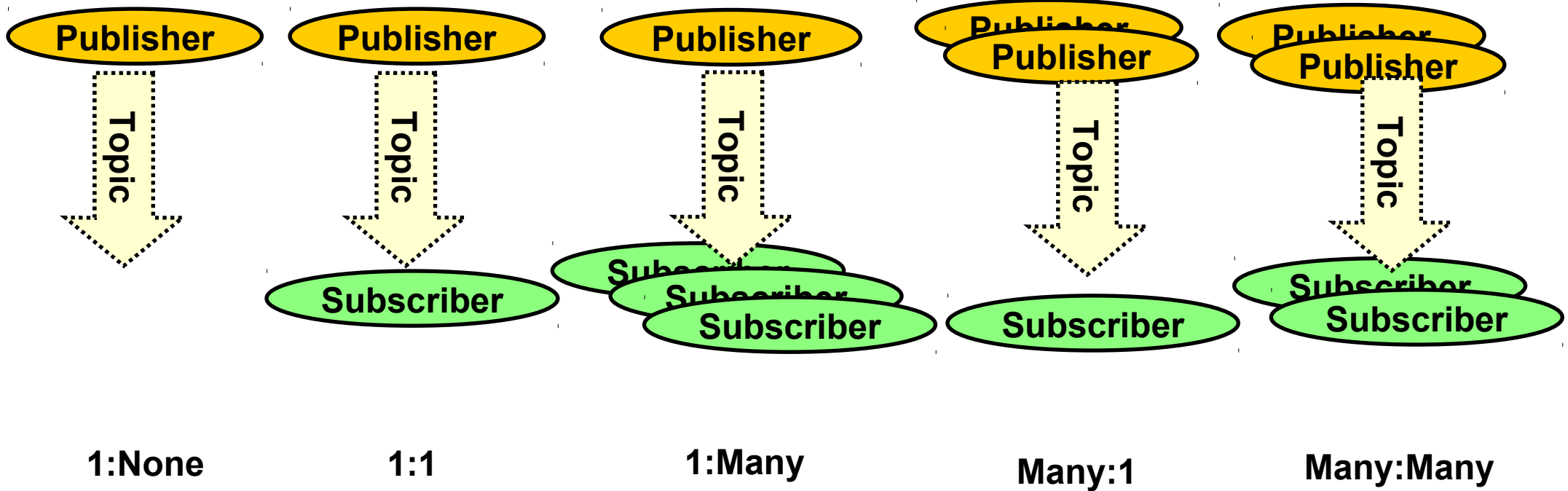
T

E

S

- Point-to-point asynchronous messaging decouples applications from each other but still implies a one-one relationship between sender and receiver. Publish/subscribe is a further stage of decoupling since the sender has no direct knowledge of how many (if any) applications will see a message. In Publish/Subscribe, the link between applications is a Topic, not a Queue. In the JMS API, both Topics and Queues are referred to as Destinations.
- Publish/Subscribe is a natural part of the JMS API and with WebSphere MQ Version 7 it is also a natural part of the native MQI. New verbs and options, that we see later, make it easy to use pub/sub from any environment.
- In WebSphere MQ Version 7 there is a publish/subscribe engine as part of the queue manager on all Distributed and z/OS systems. This is addressed directly through the MQI, replacing the queue-based interface available in WebSphere MQ V6, and previously as the MA0C SupportPac. The administration model for this uses the standard WMQ model, replacing the interfaces available in WebSphere MQ V6. Although these older interfaces are now deprecated, they are still supported to ensure coexistence with earlier versions of applications that may connect to a Version 7 queue manager.
- This publish/subscribe engine is part of the queue manager and is always available. It does not have to be manually started. A queue-based publish/subscribe daemon task is provided to ensure coexistence of the V6 style Publish/Subscribe with V7 and to migrate the use of the V6 interface to V7.
- WMQ on z/OS has not previously had a native pub/sub capability, and has had to rely on Message Broker or connecting to WMQ on a Distributed platform for access to pub/sub services. That limitation is removed in V7.

Loose Coupling with Publish/Subscribe



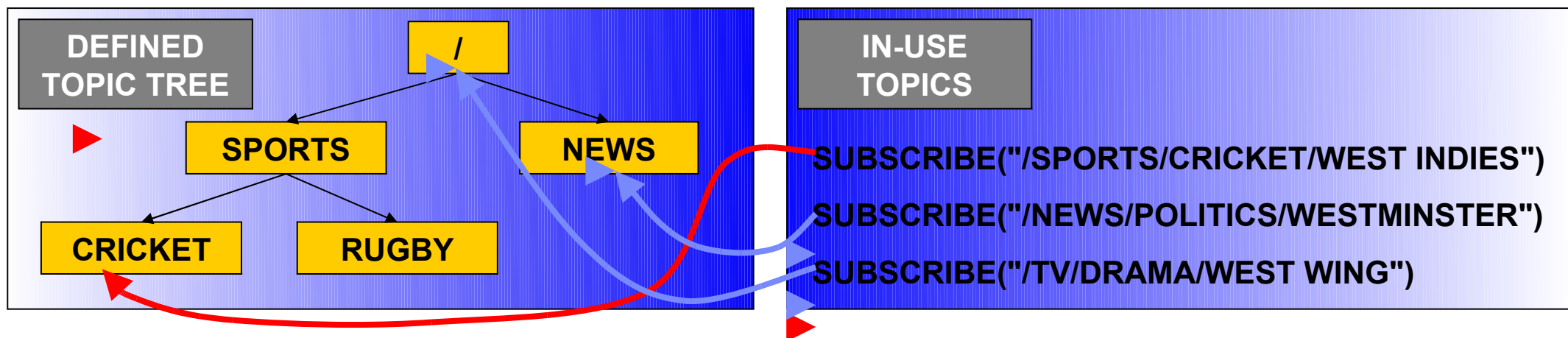
Loose Coupling with Publish/Subscribe

N**O****T****E****S**

- This picture shows some possible relationships between publishers and subscribers. There can be a one-one correspondence, but that is not part of the model. "Many" subscribers could actually be zero – the publishing application does not need to know how many subscribers exist.
- We see the topic being used as the link. In WMQ, we will often refer to Topic Strings to distinguish them from Topic Objects – discussed on the next page.

Publish/Subscribe Administration

- Based on Topic Strings
- Topic Objects
 - New object type, like queue or channel definitions
 - A 48-character name which has a longer attribute for full **topic string**
 - Defines major points in a topic tree
 - No additional definitions needed before applications can start using pub/sub
- In-use topics
 - The topic strings that applications are publishing or subscribing on
 - Inherit attributes (eg security) from the "closest" defined topic object
 - Not defined administratively, but can be viewed



Publish/Subscribe Administration

NOTES

- The construct that couples a publishing application with subscribing applications is the topic string. A topic string can be very long, and has a structure implied by the separator character '/'. Topic strings can be used directly from publishing and subscribing applications without any need to define them administratively.
- However, there may sometimes be a need to tune certain attributes of topic strings or to apply security to topic strings and as such an administrative point is required in order to do this. This administrative point is a new object type called a topic object. It is envisaged that the top layer, or the top few layers, of a topic string hierarchy is defined administratively, setting any appropriate attributes for use in that environment, and then the remaining layers of the hierarchy are defined by use, inheriting attributes from their parents in the hierarchy.
- Managing topics administratively is done by means of commands in MQSC or PCF as per the standard WMQ administration model allowing creation, alteration, deletion and display of topic objects (which contain topic string).
- In either case, whether a topic string is defined administratively or automatically created when an application uses it, status can be interrogated from the queue manager. Similarly, statistics can be gathered on the use of a topic string regardless of its creation mechanism.

Publish/Subscribe Administration (2)

- Support for durable and non-durable subscriptions
 - With durable, a client can go away and come back later without missing messages
 - Durable can cause queues to fill – generating configured depth events as warning
 - Non-durable exist only for the lifetime of the application
 - No manual "cleanup" task needed when applications end unexpectedly
- Subscriptions
 - Able to see who is subscribing to topics: DISPLAY SBSTATUS
 - Able to create subscriptions on behalf of a third party DEFINE SUB
- Security
 - Use of a topic is restricted by permissions on the associated topic object
 - On z/OS drives need for mixed-case support in RACF
 - Follows existing WMQ model for security configuration (SAF or OAM)
- Conversion of point-to-point applications without code changes
 - Administrative changes to objects
 - A queue alias can point to a topic, not just a local queue

Publish/Subscribe Administration (2)

NOTES

- When applications have subscribed to a topic, or are publishing on a topic, the queue manager can be interrogated to show the details of these applications. This interrogation is by means of MQSC or PCF commands as per the standard WMQ administration model. The details displayed are specific to publishers or subscribers and are in addition to that shown via the DISPLAY CONN command since subscribers may remain registered while they are not connected to the queue manager.
- No management or cleanup is required when subscriptions are made non-durably since the queue manager recognises when applications disconnect and cleans up any non-durable subscriptions. Queues do not continue to fill up with messages.
- The security of who can use a particular topic string is something that is restricted by permissions on the associated topic object. In the same way that attributes can be inherited from parent topics in a topic hierarchy, security settings can also be inherited from parent topics. It follows the existing WebSphere MQ model for security configuration (SAF on z/OS and the OAM on distributed). The z/OS security interface has been enhanced to permit use of mixed-case support in RACF to give more flexibility in the naming of topic objects.
- Topic security will be checked for publishing applications before they are allowed to publish, and subscribing applications before they are allowed to subscribe to a topic. Different permissions exist for creation of a new subscription against resumption of an existing durable subscription
- Applications which currently put to or get from a queue and cannot have their source code changed, can still publish messages through an alias queue which references a topic object set up on their behalf by administrative commands and receive through administratively-created subscriptions

Publish/Subscribe in the WMQ Explorer

The screenshot displays the IBM WebSphere MQ Explorer interface. The left pane shows a tree view of the queue manager hierarchy: IBM WebSphere MQ > Queue Managers > All > Development Queue Managers > v7. The main pane shows the 'Topics' view for the selected queue manager, with a table listing system topics.

Name	Topic String	Description
SYSTEM.ADMIN.QMGR.EVENT.TOPIC	SYSTEM.ADMIN.QMGR.EVENT.TOPIC	
SYSTEM.BASE.TOPIC		Base topic for resolving
		Admin stream for queue
		Default stream for queu

A context menu is open over the 'v7' queue manager icon, listing the following options:

- Compare with...
- Status...
- Delete...
- Clear Retained Publication...
- Topic Status - Subscribers...
- Topic Status - Publishers...
- Test Publication...
- Test Subscription...
- Create JMS Topic...
- Object Authorities
- Properties...

At the bottom of the interface, a 'Test Results' pane shows '0 errors, 0 warnings, 0 infos'.

Publish/Subscribe in the WMQ Explorer

N**O****T****E****S**

- The WMQ Explorer has been enhanced to show pub/sub-related operations.
- Topic objects and the subscription table can be displayed, created and modified from here.
- The same functions are available from MQSC and PCF interfaces, but this is the easy-to-use mechanism.
- When connected to older levels of WMQ, the Explorer fully supports administration of the publish/subscribe services found there. This is the equivalent of SupportPac MS0Q, the publish/subscribe plug-in for WMQ V6.

Testing Publish and Subscribe

**N
O
T
E
S**

- As part of the pub/sub operations, there are simple interfaces in the Explorer to demonstrate and check the behaviour.

Publish/Subscribe Topologies

- WMQ V6 publish/subscribe networks based on hierarchies
 - All brokers linked in parent/child tree
- WMB publish/subscribe networks based on hierarchies of collectives
 - All systems in a collective are connected to each other (mesh)
 - Also has "clones"
- New product built on **hierarchies** and **clusters**
 - With interoperability to other pub/sub systems through hierarchies
- Clusters are built on existing WMQ cluster technology
 - Cluster can be defined independently of any existing cluster used for queuing
 - Behaves in a similar way to WMB collective
- Design gives
 - Scalability
 - Availability
 - Ease of administration

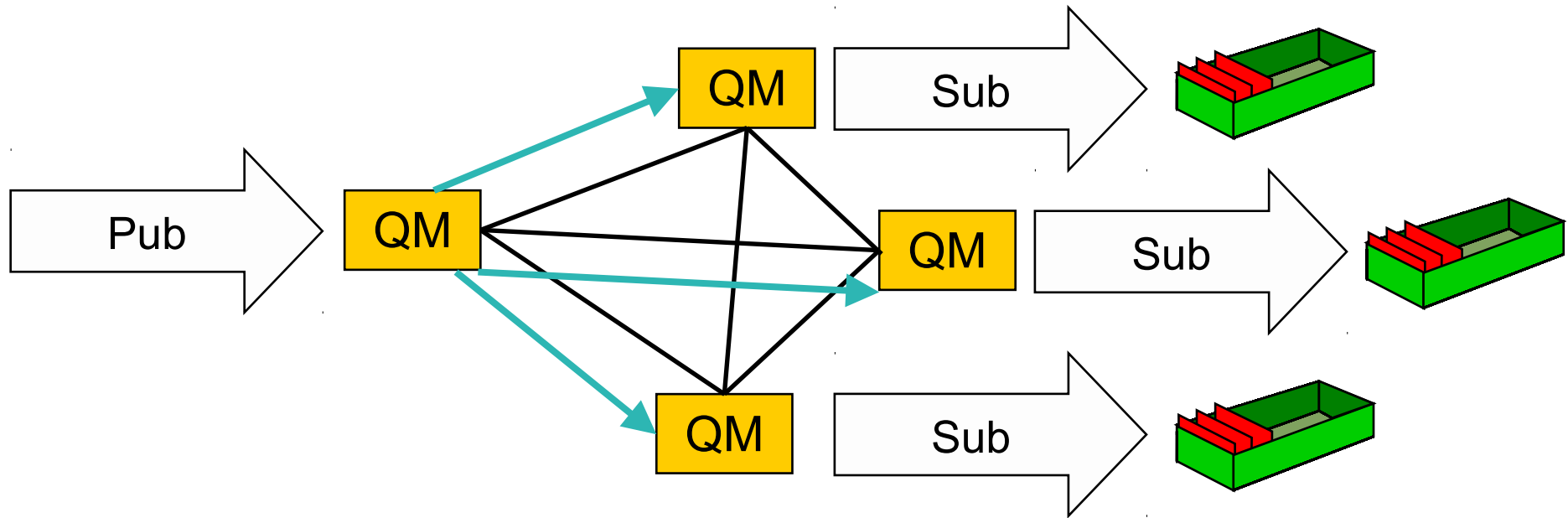
Publish/Subscribe Topologies

NOTES

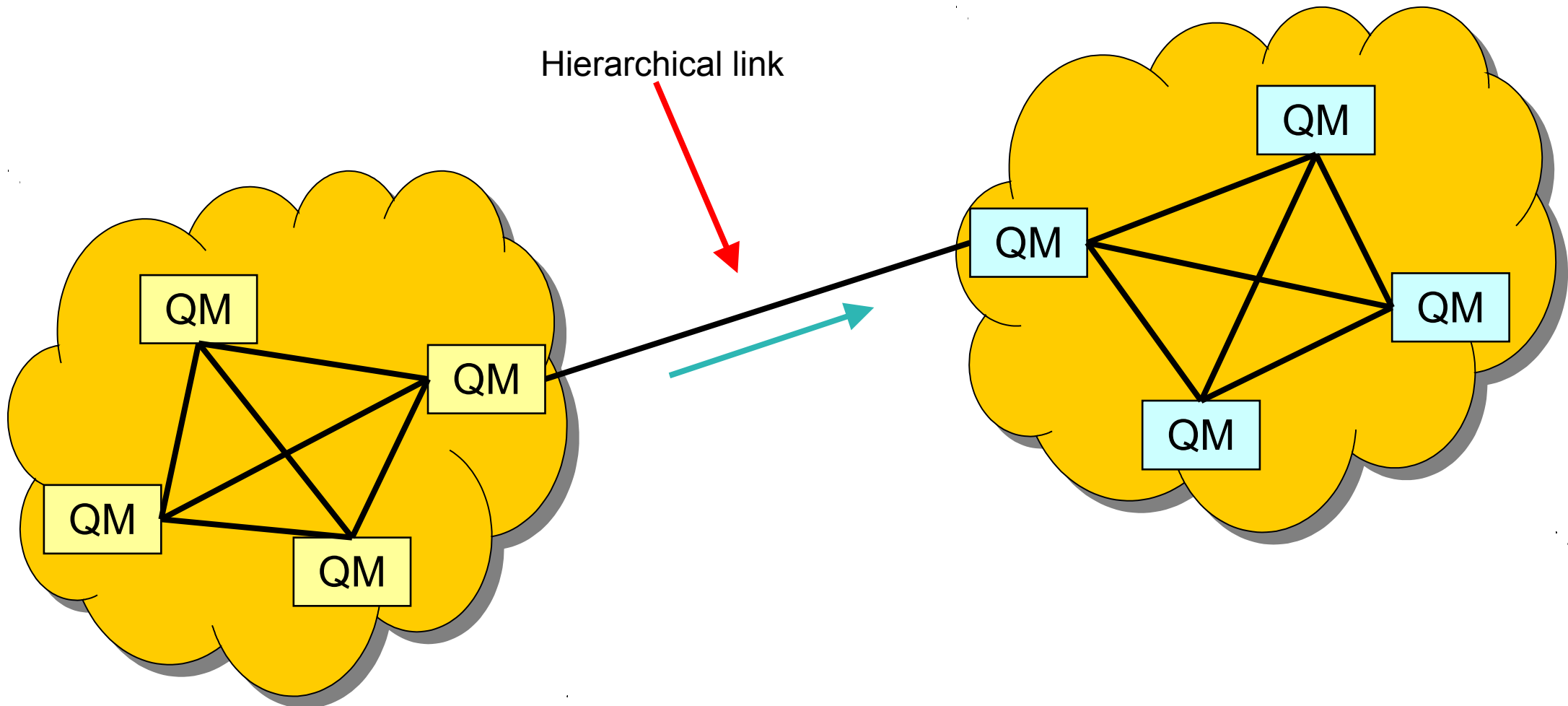
- WebSphere MQ Version 7 provides a range of topologies, including support for very large numbers of subscribers. It does this with two complementary topology designs: queue manager hierarchies, continuing support for the connection model used by WebSphere MQ V6 publish/subscribe, and publish/subscribe clusters, similar to the collectives used by WebSphere Message Broker V6.
- Queue manager hierarchies link together queue managers in a parent/child tree. Publications are delivered across these channel links to satisfy subscriptions existing on other queue managers. The availability of this topology is reliant on the links being available as there is only one route through the queue manager hierarchy from one queue manager to another.
- Publish/Subscribe queue manager clusters link together queue managers using WMQ queue manager clustering technology. A publish/subscribe cluster can be defined independently of any existing cluster used for queuing. This topology has all the availability characteristics of WMQ clusters with no reliance on any single route between two queue managers. Publications are delivered across cluster channels to satisfy subscriptions existing on other queue managers.
- WebSphere MQ Version 7 provides interoperability to other publish/subscribe systems through hierarchies, including allowing the linkage of separate publish/subscribe clusters with hierarchical links, analogous to joining up clusters with gateway queue managers for queuing. A number of attributes are available to control how much traffic is shared with other clusters; when publications are to be shared, the queue managers will attempt to minimise the traffic, allowing the destination cluster to do the fan-out as close as possible to the subscriber.
- The next two slides show some typical topologies.

Publish/Subscribe in a Cluster

- Consistent topic definitions in cluster
- Multiple routes across cluster



Publish/Subscribe in Combined Hierarchy & Clusters



Publish/Subscribe Application Programming

- Cannot significantly change the JMS API
 - But we want some of its facilities more easily available in the MQI
 - To improve MQI programming and improve (make thinner) the JMS layer
 - JMS implementation exploits new MQI functions
- New verb for subscribing
 - So you do not need to build RFH or RFH2 headers in the application
 - **MQSUB** registers a subscription
 - Includes information about where messages will be read from
 - Do not need to specify a queue – can be automatically assigned
 - Retained publications delivered immediately after subscribing
- New options on existing verbs
 - MQOPEN to get access to a topic
 - MQCLOSE deregisters a subscription
 - MQPUT, MQGET to publish and to receive publications
- Sample programs included to demonstrate use

Publish/Subscribe Application Programming

N

O

T

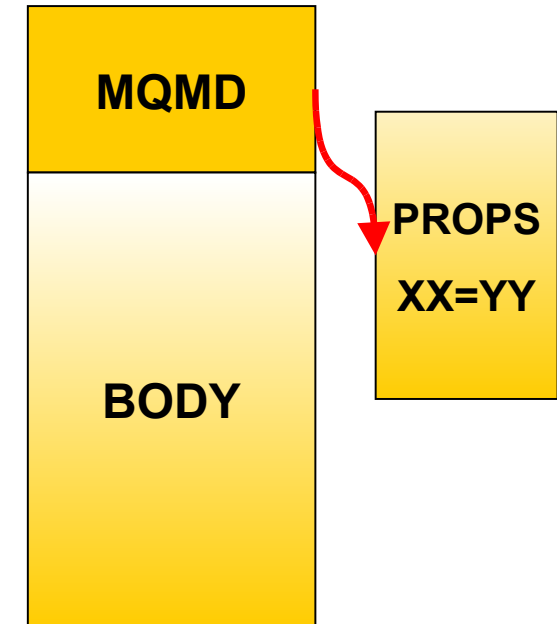
E

S

- Publish/Subscribe is part of the JMS API. We cannot change the JMS API but we want to make some of its facilities more easily available in the MQI. This has the effect of improving MQI programming and shrinks the JMS layer that is built on the MQI.
- Publishing to a topic and subscribing for publications on a topic are now part of the MQI. Subscribing to a topic object registers that applications interest in publications to that topic. This is done by means of a new verb, MQSUB. Reading from the returned object handle gives the application any publications made to that topic.
- Opening a topic for output registers that application as a publisher to that topic for the duration of the open. Putting messages to the returned object handle sends them to the subscribers of that topic.
- Options are available on the MQI to indicate whether a subscription is durable or non-durable. Those subscriptions that are made durably continue to have publications sent to their indicated target whether the application is connected or not. Those subscriptions that are made non-durably cease to exist when the application disconnects and no further publications will be sent to their indicated target.
- The indicated target for publications used by a subscriber could be a nominated queue, or a managed queue. If a queue is nominated by the subscriber then it is the responsibility of the application to tidy up that queue and remove all messages from it, even if it is using a non-durable subscription (only durable subscriptions can be made via the administrative command). If the subscriber requests a managed queue, then this queue is guaranteed to be local to this queue manager and will be tidied up after the disconnection of a non-durably subscribed application, or after the explicit removal of a durable subscription.
- If a subscription is made durably, it remains in place when the subscribing application disconnects and can be resumed by the subscribing application when it reconnects and requests the subscription again. When an application no longer needs a durable subscription it can be removed cleanly using MQCLOSE options.

Message Properties

- Arbitrary values associated with the message but not part of the body
 - Like a user-extendable MQMD
 - Already part of JMS
- New verbs including **MQSETMP** and **MQINQMP**
 - Properties can be integers, strings, boolean, etc.
- Easier to use than RFH2 folders
 - Receiving apps do not see them unless they want
 - No need to parse and skip over message headers
- Configuration options for compatibility
 - Queue and channel attributes define behaviour
 - Defaults will create RFH2 folders
- Permits explicit statement of relationships between messages
 - eg Message X is a **REPLY** to Message Y
 - Messages referred to by handles



Message Properties

N

O

T

E

S

- Message properties are arbitrary values associated with the message but not part of the body of the message. You can think of them rather like a user-extendable message descriptor (MQMD). This concept is already part of the JMS API. Properties can be any of any basic type, for example integers, strings, or Booleans, as expressed in the appropriate programming language.
- Setting and Inquiring of message properties in a message is part of the MQI with the introduction of new verbs to allow this. Message handles are introduced to allow references to be made to a message within which message properties are to be manipulated.
- For applications whose source cannot be changed, the decision to discard message properties can be made based on the consumer of messages at consumption time rather than by the producer of the messages who doesn't necessarily know whether the consumer will be able to accept message properties.
- Message properties are exposed as MQRFH2 headers when they are sent to older queue managers, as they are today. Configuration attributes also allow removal of these properties as they are transferred.
- The introduction of a message handle also permits the application to state an explicit relationship between messages. For example, Message X is a REPLY to Message Y. This then potentially allows monitoring products to tie messages together, where previously they could only guess or have external configuration.

Message Properties and Handles – Source Code

```
// This is a router app - get a message and work with it
// 1. Initial setup and read input
MQCRTMH(hConn, &CrtMsgHOpts, &hRequestMsg, &RC, &RC);
GetMsgOpts.MsgHandle = hRequestMsg;
MQGET(hConn, hObj, &MsgDesc, &GetMsgOpts, BufLen, &Buffer, &DataLen, &CC, &RC);
```

```
// 2. Forward request unchanged to a server app, named in the message
PutMsgOpts.Action = MQACTP_FORWARD;
PutMsgOpts.OriginalMsgHandle = hRequestMsg;
MQPUT(hConn, hServerObj, &MD, &PutMsgOpts, DataLen, &Buffer, &CC, &RC);
```

```
// 3. Tell requester message has been dealt with by updating existing property
Name.VSPtr = "RequestStatus";
Name.VSLength = MQVS_NULL_TERMINATED;
MQSETMP(hConn, hRequestMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING, "REQUEST
RECEIVED", 16, &CC, &RC);
PutMsgOpts.Action = MQACTP_REPLY;
PutMsgOpts.OriginalMsgHandle = hRequestMsg;
MQPUT(hConn, hReplyObj, &MD, &PutMsgOpts, DataLen, &Buffer, &CC, &RC);
```

```
// 4. Also put a completely unrelated message to a logging queue
PutMsgOpts.Action = MQACTP_NEW;
PutMsgOpts.OriginalMsgHandle = MQMH_NONE;
MQPUT(hConn, hLogObj, &MD, &PutMsgOpts, DataLen2, &LogMsgBuf, &CC, &RC);
MQCMIT(hConn, &CC, &RC);
```

Message Properties and Handles – Source Code

N

O

T

E

S

- In this chart, we see code fragments that show how to use the new message property MQI calls. While changes are needed to exploit these functions, this section demonstrates that the changes are easy to understand.
- It shows how to set properties, and also how the relationship between different messages can be expressed.
- The scenario is of an application that acts like a router, reading messages from one queue and forwarding them to another queue. At the same it sends replies to the originating application, and makes an entry to a logfile.
- Block 1 of the code reads the message, but also pulls the message's properties into a new message handle.
- Block 2 forwards the message to the destination queue. Because the same message handle it used, any properties from the inbound message are automatically copied to the outbound message, without the application explicitly needing to read and set these properties. The PMO Action field tells the queue manager that this message is being forwarded, so it knows which properties to copy.
- Block 3 sets a new property on the inbound message and sends it back to the originator.
- Block 4 creates a new unrelated message and puts it to the logging queue.
- Finally, all of these operations are committed.

Other MQI Enhancements

- Asynchronous Message Reception
 - New verb **MQCB** defines a callback function
 - Automatically Invoked when a message arrives
 - No need for MQGET(WAIT) or MQGET(SIGNAL)
 - A thread can receive messages from multiple queues
 - New verb **MQCTL** to start and stop message delivery to callback
- Selectors
 - Use a SQL92 clause to select messages by properties including MQMD fields
 - Can be specified on **MQOPEN**, **MQSUB** for filtering messages
 - Selection is done inside queue manager
 - Not looking inside message body
 - Message Broker still required for content filtering
- Cooperative Browsing and Message Tokens
 - Efficient interface for applications reading from the same queue
 - Example: "master" program browses a queue telling "slaves" which message to work with, based on elements within the message
 - No races – messages locked but available to any cooperating process

Other MQI Enhancements

NOTES

- The MQI is enhanced to provide asynchronous consumption of messages removing the need for the JMS interface to build the onMessage() method on top of polling MQGET calls. This allows applications to be called back when a message becomes available, meeting their message selection criteria on one of their named queues. The callback function is also invoked when something happens that requires the application to end, for example when the connection to the queue manager is being quiesced.
- Functions to be driven are defined using a new verb, MQCB. This links a function with an object handle previously obtained by opening a queue or subscribing to a topic. The same function may be linked to many different object handles allowing the consumption of messages from a number of queues at the same time. Once all objects have been opened and all functions have been linked with object handles, another new verb is called to indicate that asynchronous consumption can start, MQCTL. The connection handle used for this call cannot be used for any other MQI verb, outside of the callback function, except for stopping asynchronous consumption.
- SQL92 selector strings are used on the JMS API to pick out specific messages to be consumed by the application. Selectors are now also available for use in the MQI both when opening a queue for input and when subscribing to a topic. Since these selectors are understood by the queue manager, it reduces the number of messages sent to the JMS interface that are discarded because they don't match the selection criteria for the application. Reducing this number of non-matching messages is especially critical when the JMS application is connected as a client, because it reduces the CPU and bandwidth used to deliver unneeded messages.
- Selection can be made based on fields in the message descriptor (MQMD) and properties. WebSphere Message Broker is still required for filtering based on the content, or body, of the message.
- Enhancements have also been made to the Browse options to allow one process to tell another which message to retrieve from a queue, removing contention that exists in some architectures today. This was used in WMQ V6 to drive CICS transactions reading from a shared queue; that interface has now been publicly exposed.

Programming in Java

- JMS read/write access to all MQMD fields as properties
 - Have to explicitly enable this in the application program
 - Allows the application to go beyond the JMS specification
- JMS access to the raw message content
 - Can treat the whole body as a byte array property
 - Can see RFH2 folders that would normally be stripped

- Message Header Classes for Java
 - Updated and supported version of MS0B SupportPac
 - Makes it easy to build and parse PCF structures
 - Extended to handle other MQI message header formats
 - eg MQCIH, MQDLH classes

Programming in Java

N**O****T****E****S**

- WebSphere MQ Version 7 does include new interfaces for the Java programmer. One group of these are intended to make it easier to create and parse MQI structures such as the PCF elements. The MS0B SupportPac has been enhanced and incorporated into the product as a fully-supported component.
- There are also options to simplify the interoperation of JMS programs with other programming languages, allowing more direct manipulation of fields in the MQMD and giving access to the raw message body. Because modifying some of these attributes is not permitted within the JMS specification, the program has to "acknowledge" that it is prepared to do this by setting another virtual property on the message or destination.

Application Migration

- New verbs remove need for some of the older interfaces
 - Which will be deprecated – though not removed immediately
 - PCF and RFH facilities for WMQ publish/subscribe applications: Identity, Streams
- A single application cannot mix new verbs with old options
 - Can't use RFH(Register Publisher) and MQPUT(topic) in same program
- Option available to translate old-style pub/sub commands
 - Both RFH and RFH2 interfaces for WMQ and WMB pub/sub applications
 - Not needed when all publish/subscribe applications have been converted
- Migration step to convert existing WMQ and WMB subscriptions
- Most common pub/sub "application" is IBM-supplied JMS layer
 - Client design to automatically take advantage of new facilities when they exist
 - So installing new JMS jar is all that's needed
- New MQI operations are not available for old VB and ActiveX programs
 - Use .Net classes instead

Application Migration

NOTES

- Having Publish/Subscribe natively in the MQI with the new verbs and options described previously removes the need for the older interface, MA0C RFH-style publish/subscribe commands. This interface is deprecated – although it is not being removed immediately.
- The PCF and MQRFH queued facilities for WebSphere MQ publish/subscribe and the MQRFH2 queued facilities for WebSphere Message Broker publish/subscribe are supported by means of a daemon task which translates these queued publish/subscribe commands into the new verbs. This path will clearly have a performance cost.
- A single application cannot mix new verbs with old options however a subscriber made through the MQRFH queued commands can interact with a publisher doing an MQPUT to a topic. This allows existing applications to be migrated one at a time.
- The most common "application" which uses these queued facilities is the IBM-supplied JMS layer. The new version of these layers will use the new verbs. So installing a new JMS jar will be all that's needed to upgrade these applications to the new interfaces. Any applications which have been written to directly put command messages to do publish/subscribe should be re-written to use the new verbs. This does not have to be done immediately since the daemon task translates the commands into the verbs.
- The VB and ActiveX interfaces are not being enhanced for WMQ V7. Existing applications will continue to work, but if they wish to exploit facilities such as message properties or the new pub/sub interfaces, then they will need to be rewritten to use a different API – probably the .Net classes. This follows Microsoft's directions with support for these older languages.

Client Performance

- Traditional WMQ non-persistent messages more reliable than some need

- "Read Ahead" for Receiving Messages/Publications:
 - Messages sent to a client in advance of MQGET, queued internally
 - Administrative choice – no application changes needed
 - Higher performance in client

- "Asynchronous Put" for Sending/Publishing Messages:
 - Application can indicate it doesn't want to wait for the real return code
 - Maybe look for return code later – **MQSTAT** verb
 - Maintains transactional semantics
 - Higher performance in client

Client Performance – Notes

N**O****T****E****S**

- While WebSphere MQ non-persistent messages do not have assured delivery, they traditionally have still been more reliable than is necessary for some application scenarios. With V7 it is possible to have additional trade-off options for performance versus reliability. WMQ provides a comprehensive range of qualities of service for message delivery, ranging from assured delivery to lightweight non persistent.

Client Performance – Read ahead – Notes

NOTES

- Message read ahead is supported between clients and servers removing the need for the WMQ client to specifically request every message that is sent to it by the server. Certain profiles of applications can benefit from providing the message criteria that they wish to consume and having these messages sent to the client without the need for the client to repeatedly tell the server the same message criteria.
- Read ahead works best when you are fairly certain that the messages really are intended for this client, you are fairly certain they will be consumed by the client, and you know ahead of time in what manner they will be consumed. The ideal scenario is a non-durable subscribe of non-persistent messages using an asynchronous consumer. By contrast, a point to point get of a persistent message in a transaction is not suitable for read ahead. As another example, a queue that has the messages processed by many cloned applications all opening the same queue at the same time is not a suitable application to use read ahead.
- Read ahead provides a significant performance improvement to an application requiring a lower Quality of Service by removing many of the network replies. The messages are stored in memory in the client process and so if the application is not ended gracefully these messages are lost. Read ahead is a negotiated option between the client and the server and therefore does not provide any support to a back level client as it will be negotiated to off. It is set by a configuration option on an individual queue level; application code does not need changes.
- Read ahead only applies to non-persistent messages, so the class of service for persistent messages is not being changed in any way. When there is a mixture of persistent and non-persistent messages on a queue, read ahead stops until the persistent message is explicitly requested by the application.

Client Performance – Asynchronous Put – Notes

N

O

T

E

S

- This is an option that allows an application to indicate that it is not immediately interested in the return code from a put of a message (although it may be interested enough to ask for the return code later). It allows unacknowledged messages to be sent removing the expense of the line turnaround on a client.
- An application can indicate that it is willing to have unacknowledged, or asynchronous, communication with the server when it is producing messages either to be placed on a named queue, or published to a topic, by supplying the put message option indicating this. Using this option does not guarantee that no line turnarounds will occur, but simply that the application does not require any synchronization with the server. For example, in the case of persistent messages outside of sync-point, this option has no effect, since a line turnaround is still required. For this case, however an error will not be given with the use of this option. This behaviour can also be configured by administrative means so that application code does not have to be changed.
- Messages that are given to the server using this option may be within a transaction. The commit of the transaction always does a line turn-around and still provides a return code, which may indicate that a previous put had failed and that the transaction has not been committed. In this case the application can request the last asynchronous return code using a verb to get the last asynchronous error to determine what the problem was. The error returned to this verb is only the error from an asynchronous call. Any subsequent synchronous calls which experience an error do not have any impact on the result provided when using this verb.
- In order for this option to be used, both the client and the server must understand it, and therefore it does not provide any advantage to a back level client. It is however applicable to both point-to-point and publish/subscribe applications and is available in the MQI. JMS applications get a benefit from this without any application code changes at all.

Client Connection Management

- Shared Client Conversations
 - Several connections from the same process can be handled on the same socket
 - Faster startup for the second and subsequent connections
- Implementation also gives us more heartbeat opportunities
 - Faster failure notification for clients

- Client Connections
 - Automatic workload distribution via CCDT
 - Control number of connected clients at a queue manager

- Free connections to z/OS for administration programs like WMQ Explorer
 - Limited number of clients permitted by V7 license without CAF

Client Connection Management (1)

N

O

T

E

S

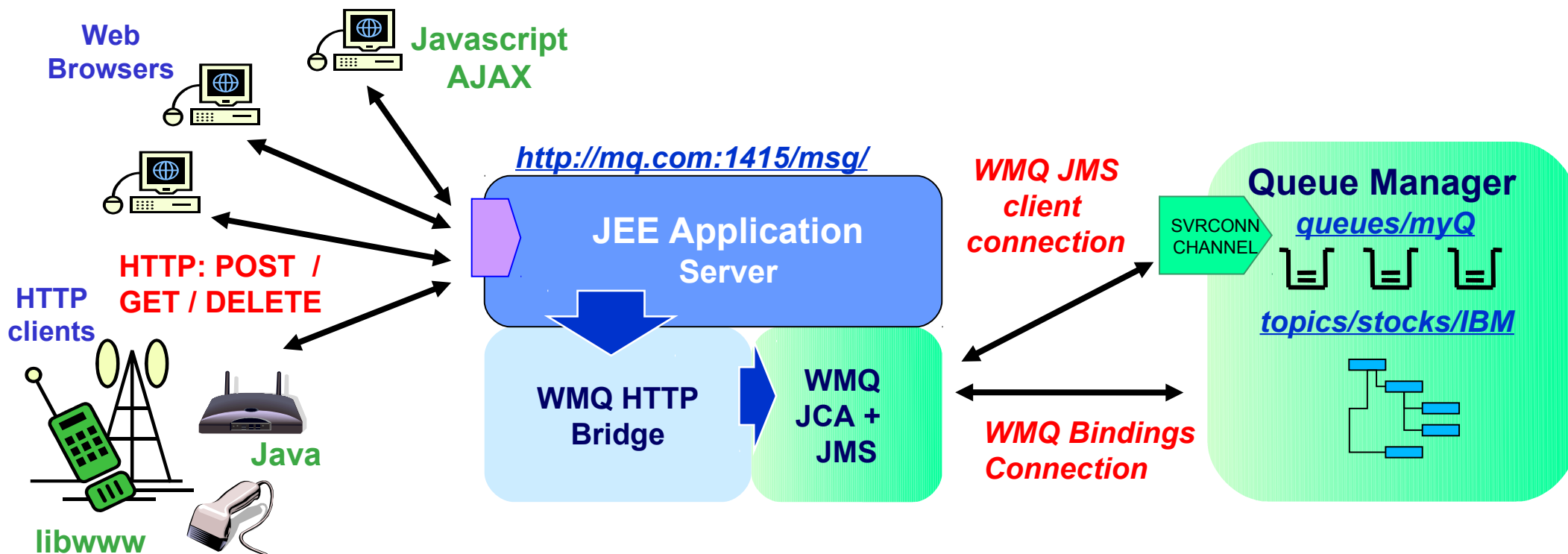
- There is a reduction in the network resources required when an application connects several times to the same queue manager by allowing all connections to be shared over the same socket. This is especially useful for JMS applications that create several JMS sessions. In order to implement these features, client channels have been changed to use a full-duplex protocol (TCP/IP only) and this implementation gives us more heartbeat opportunities allowing for faster failure notification for clients and faster tidy-up of orphaned server-connection channels.
- In order to implement conversation sharing we allow a single TCP/IP socket to carry multiple connections provided the two ends of the connection are the same process. Sharing the conversation requires a full duplex connection.
- It is not expected that large numbers of JMS sessions are created on a single JMS connection, even by larger entities such as the WebSphere Application Server. They can still benefit by having many JMS connections shared across a single socket, although of course it is not guaranteed that all their JMS connections will use the same socket, just that they could do where it is possible.
- Administrative users of the product will be aware of conversation sharing since the number of server-connection channels will be reduced compared to previous releases although they see the same number of details when displaying connections. Displaying channel status details will indicate that there are a number of connections going across a specific channel instance and there are administrative means to set limits on this behaviour, including removing it altogether.

Client Connection Management (2)

NOTES

- The Client Connection Definition Table (CCDT) has always allowed a client to connect to one of a number of queue managers, but the list was always searched sequentially for the first available system. With WMQ V7 the selection can be configured to be made randomly from the list, assisting with workload distribution. Where multiple connections might be made from the same process, an affinity attribute tells the client to attempt to connect to the same queue manager as previously – this can then exploit the multiplexing capabilities of the channel.
- The number of inbound client channels can now be restricted with WMQ SVRCONN channel attributes. You can specify independently the maximum number of instances of a channel, and the maximum number of instances from a specific partner. This is designed to prevent rogue applications from using all the system resources by continuing to connect. Existing attributes can still restrict the total number of channels across the queue manager.
- With V7, the license for WMQ on z/OS has been modified to permit a small number of client connections to the SYSTEM.ADMIN.SVRCONN without requiring purchase of the Client Attach Facility. This has been done to make it easier for customers without the CAF to use the WMQ Explorer configuration program. System messages have been extended to make it obvious when you are using one of these connections. If you want to use other named SVRCONN channels, or to have more connections, then the CAF is still required.

WebSphere MQ Bridge for HTTP - Architectural Overview



- 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 non-servlet implementation available as MA94

HTTP Connectivity to WMQ

NOTES

- The first goal of the HTTP feature 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
- Shipped with WMQ V7 is the J2EE/JCA-based implementation, previously SupportPac MA0Y. An alternative native implementation, SupportPac MA94 is also available as a separate download.
- 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 quirks. For example the bridge 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

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)

N

O

T

E

S

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

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

Message Data

Required Headers

WMQ Explorer Enhancements

- **Sets**
 - Queue Managers can be partitioned into sets within the Navigator
 - For example "Test", "Production"
- **Security Configuration**
 - Easy to define channel exits, userid/password configurations
 - Configured for each queue manager or for all queue managers in a set
 - Password manager included
 - Still recommend security exit or service for authentication at the server
- **Tighter JMS integration**
 - Creating an queue/topic can define a JMS destination at the same time
- **Message browser configuration**
 - Number, size of messages
- **Plug-in Migration**
 - Explorer now based on Eclipse 3.3 – compatibility not guaranteed
 - Major change is availability of supported PCF classes
- **Using MQ Explorer with WebSphere MQ for z/OS**
 - you can use the MQ Explorer, which runs on the Windows and Linux on x86 platforms, to manage the z/OS parts of your WebSphere MQ network without having to buy the Client Attach Feature

WMQ Explorer Enhancements

NOTES

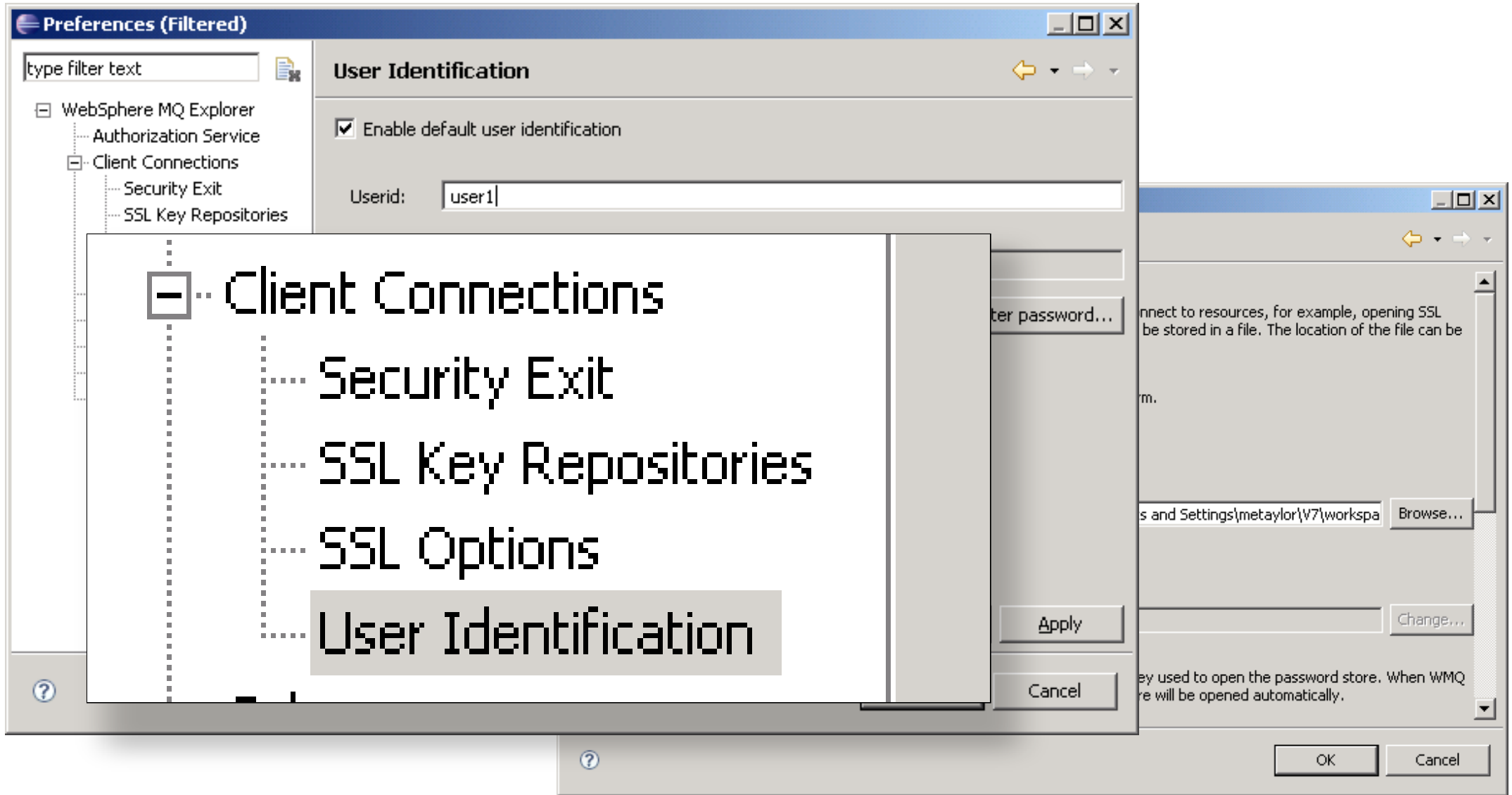
- Along with support for new features such as the publish/subscribe objects and their corresponding status displays, the Explorer has been enhanced with more general capabilities.
- One of the most frequently requested features has been the ability to partition the list of queue managers into sets, for example to show test and production systems separately in the navigation tree. This version of the Explorer allows sets to be defined, with queue managers assigned to sets manually or automatically based on some attribute such as the CommandLevel value. Queue managers can appear in more than one set. There are some operations that can be applied at the set level, such as connecting to all of them with a single click.
- The security controls allow configuration of not just a userid and password, but SSL and channel exit parameters. These can be defined globally or for each queue manager. A password manager component stores information securely, so that you do not need to re-enter passwords on each restart of the Explorer.
- When a queue or topic is created, the wizard can optionally also drive the corresponding JMS operation. This assist with integration of JMS configuration objects with the WMQ configuration.
- The message browser now has preferences that say how many messages to read from the queue instead of the previously hard-coded 500; similarly the length of messages can be configured.
- The Explorer has been updated to use the current version of the Eclipse runtime, V3.3. User-written plug-ins ought to work unchanged, but Eclipse has deprecated some interfaces that existed in the previous version, so compatibility cannot be guaranteed. One major change is that PCF classes are now officially available and supported; plug-ins should be changed to use those interfaces.

Queue Manager Sets

The screenshot displays the WebSphere MQ Explorer interface. On the left, a tree view shows the hierarchy: IBM WebSphere MQ > Queue Managers > Queue Managers > Development Queue Managers. The 'Development Queue Managers' folder is highlighted with a dashed border. Below it, other folders like 'All', 'v7', 'Production Queue Managers', 'audi', and 'QMA' are visible. On the right, a table titled 'Queue Managers Set Development Queue Managers [manual set]' is shown with the following columns: Queue manager name, Command level, Queue manager status, Platform, and Queue-sharing group.

Queue manager name	Command level	Queue manager status	Platform	Queue-sharing group
v7	700	Running	Unix	

WMQ Explorer Preferences



WMQ as an SOA Asset - Service Descriptions

- A standard way to describe all WMQ apps as SOA assets (services)
 - To be inventoried, and catalogued in Service Registry
 - To be re-used as services in composite SOA applications
 - To be managed and traced with SOA tools
- IBM has created the WMQ Service Definition and SOAP binding
 - IRI for WMQ addresses (“wmq:”)
 - Message destinations - Queues or Topics
 - Other resources - Qmgrs, channels, channel status etc.
 - WSDL bindings to define application properties
 - Also defines the Message Exchange Pattern; Request queue; Response queue; Correlation style; Message format; Message persistence, priority etc.
- Published as SupportPac MA93

WMQ as an SOA Asset - Service Descriptions

N

O

T

E

S

- 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 (ie not hosted in environments such as CICS or WAS) native WMQ applications
- This will allow applications to:
 - Be inventoried, and catalogued. For example, the WSDL description of an app can be stored in WSRR
 - Be managed and traced with SOA tools. which could monitor the queues associated with a service
 - Be re-used in composite applications. For example, once the service definition has been implemented by web services tools, it will be possible to drop a WMQ application into a composite Web services application, and the tools will generate the code required to invoke the application
- IBM is creating the WMQ service definition specification consisting of two documents.
 - An IRI specification defining the address of WMQ message destinations (Queues or Topics) and the address of other resources such as 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
- Initially this has been published as a SupportPac, without immediate plans to add tooling support in the base WMQ product, but it provides a standard to be implemented by IBM and vendor tooling products.

Some Performance Information

- Persistent pub/sub throughput increased up to 60%
- Non-persistent client throughput increased up to 300%
- JMS Selector rates improved up to 250%
- Message Listener throughput improved up to 45%
 - Latency also improved

- Measurements taken from pre-release code
- Performance reports will be published as usual on SupportPac site

Some Performance Information

**N
O
T
E
S**

- These numbers come from the announcement letter and are based on pre-release code
- See final performance reports on the SupportPac site

IBM WebSphere MQ V7.0.0 - Summary

- Enhanced JMS
 - More applications being written to use this API
 - Underpins many SOA/ESB solutions needing access to messaging
 - Improved performance & ease-of-use
- Enhanced Publish-and-subscribe
 - Ease-of-use
 - New support for z/OS
- Extended verbs and behaviors for MQI programming interface
- Enhanced MQ clients for increased throughput resilience and availability

- Web 2.0 support to help create richer user experience

- Evolutionary – if you know V6, you will know V7

IBM WebSphere MQ V7.0

N**O****T****E****S**

- WMQ V7 was released in 2008. This presentation does not discuss any of the details of the new features in that release; other presentations exist that provide that level of information. Instead we just have a single slide to remind you of the major areas that were enhanced.

WebSphere MQ V7.0.0.1

Initial V7 Updates (7.0.0.1)

- Complete some elements of V7 that didn't make GA
 - Migration/coexistence PTFs for z/OS

- Service Definition Wizard
 - Describing MQ applications in WSDL for better governance

- Standalone MQ Explorer download (SupportPac MS0T)
- XA-aware API Exit

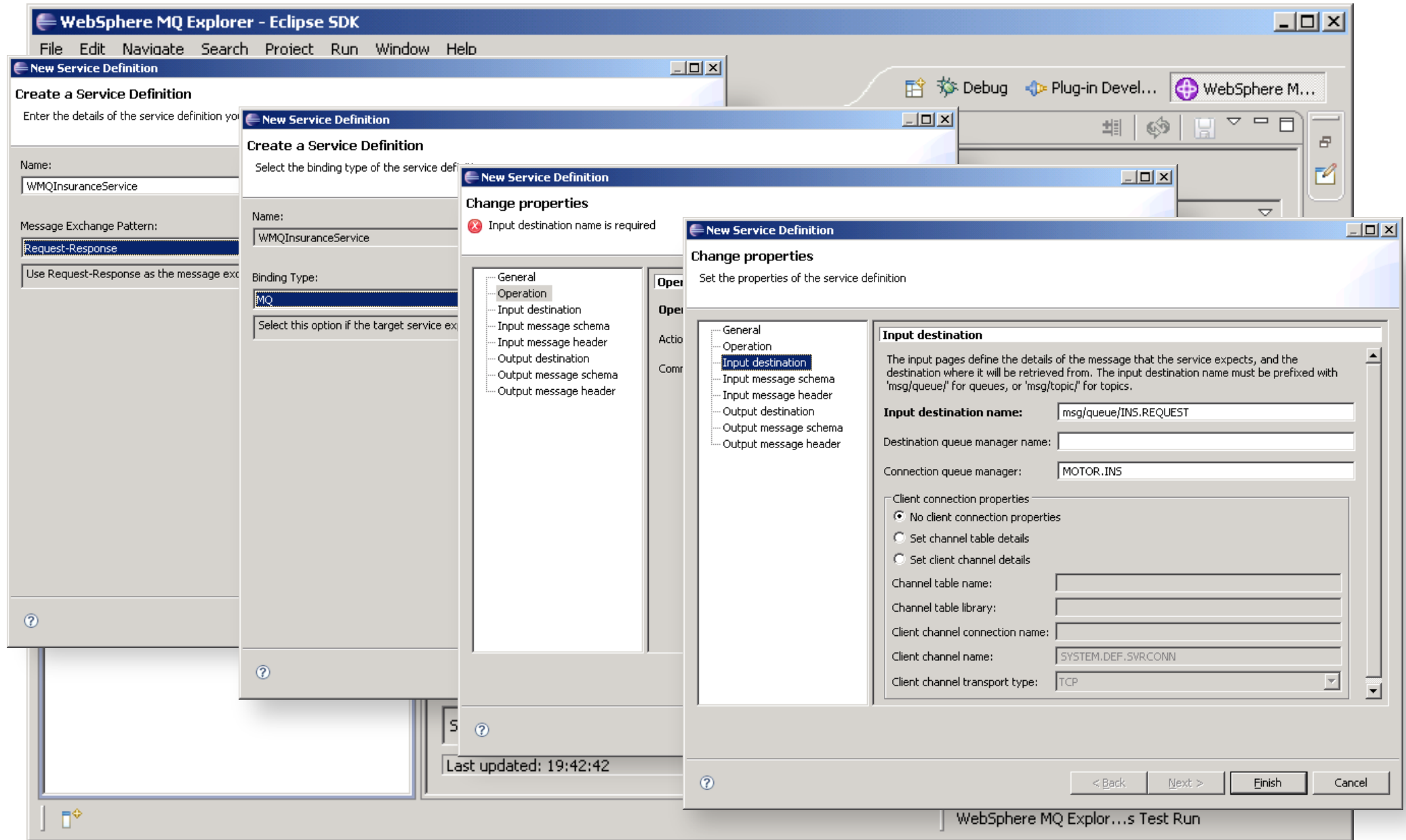
- Continued performance enhancements
- SupportPac updates
 - MS81 (MQIPT)
 - Performance Reports
 - Various Cat2 packages: MS03, MA01, MS0P, MO71, MO72, MC91 ...

Initial V7 Updates (7.0.0.1)

NOTES

- V7.0.0.1 was the first fixpack released on Distributed platforms. At around the same time, some capabilities were released for z/OS.
- The main new feature that was part of the fixpack was the Service Definition Wizard, part of the Eclipse environment. And we will see more about that in following slides.
- If you write API Exits for Distributed platforms, V7.0.0.1 extended the interfaces so you can now intercept the XA operations invoked by external transaction managers. The particular problem that needed these operations to be exposed was that a TM might call the xa prepare/commit operations on connection handles other than the main one used for MQPUT/GET. So an API Exit could not know which messages were being committed. Now they can.
- SupportPac MS0T is a repackaging of the MQ Explorer that can be directly downloaded from the IBM web site. The goal of this repackaging was to simplify distribution of the Explorer, which was previously only available on the MQ Server CD images – even if you didn't want to install the server runtime code.
- MS81 (Internet Pass-Thru) has been updated to deal with the new MQv7 protocol flows, particularly between clients and servers.
- Other SupportPacs have been updated for currency with new features provided in V7.

Service Definition Wizard: Creating the WSDL



Service Definition Wizard: Creating the WSDL

N**O****T****E****S**

- During 2007 documents were released defining how an MQ application could be described using the standard mechanism for services, WSDL. They are included in SupportPac MA93.
- WMQ V7.0.0.1 makes it easy to create the WSDL without needing to write XML directly in a text editor. A new plug-in for the MQ Explorer provides a wizard, where you fill in information about your application and it will write the WSDL directly.
- That WSDL can then be imported into other tools such as WSRR for further processing such as impact analysis and runtime decision-making.

Generated WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://tempuri.org/InsuranceRequest"
  xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  name="InsuranceRequest"
  targetNamespace="http://tempuri.org/InsuranceRequest"
  xmlns:importinms=""
  xmlns:importoutms="">
  <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import schemaLocation="InsuranceQuote.xsd"/>
      <xsd:import schemaLocation="Insurance.xsd"/>
    </xsd:schema>
  </types>
  <message name="RequestQuote_Input">
    <part name="RequestQuote_Input_Part"
      type="importinms:xsd:string"/>
  </message>
  <message name="RequestQuote_Output">
    <part name="RequestQuote_Output_Part"
      type="importoutms:xsd:string"/>
  </message>
  <portType name="InsuranceRequest_PortType">
    <operation name="RequestQuote">
      <input message="tns:RequestQuote_Input"/>
      <output message="tns:RequestQuote_Output"/>
    </operation>
  </portType>
```

```
<binding name="InsuranceRequest_Wmq_Binding"
  type="tns:InsuranceRequest_PortType">
  <wmqservice:binding/>
  <operation name="RequestQuote">
    <wmqservice:targetAction>RequestQuote</wmqservice:targetAction>
    <input>
      <wmqservice:body/>
    </input>
    <output>
      <wmqservice:body/>
    </output>
  </operation>
</binding>
<service name="InsuranceRequest">
  <port binding="tns:InsuranceRequest_Wmq_Binding"
    name="InsuranceRequest_Wmq_Port">
    <wmqservice:address
      location="wmq:/msg/queue/INS.REQUEST"/>
    <wmqservice:replyTo>wmq:/msg/queue/INS.REPLY?
      connectQueueManager=MOTOR.INS</wmqservice:replyTo>
    <wmqservice:connectQueueManager>MOTOR.INS</wmqservice:connectQueueManager>
  </port>
</service>
</definitions>
```

Generated WSDL: Detail

```
<service name="InsuranceRequest">
  <port binding="tns:InsuranceRequest_Wmq_Binding"
        name="InsuranceRequest_Wmq_Port">
    <wmqservice:address
      location="wmq:/msg/queue/INS.REQUEST"/>
    <wmqservice:replyTo>
      wmq:/msg/queue/INS.REPLY?connectQueueManager=MOTOR.INS
    </wmqservice:replyTo>
    <wmqservice:connectQueueManager>MOTOR.INS
    </wmqservice:connectQueueManager>
  </port>
</service>
```

WSRR Impact Analysis

WebSphere Service Registry and Repository

Perspective: Administrator Go Support Help

Graphical View Page Help

Graph for: mq-binding.wsdl

mq-binding.wsdl WSDL Document

Impact Analysis for: QUEUEMGR.1

CONNECTQMGR MQ Queue Manager

QUEUEMGR MQ Queue Manager

QUERY.CUSTOMER.INPUTQ MQ Queue

QUEUEMGR.1 MQ Connection

queryCustomerService Service

queryCustomerPort Port

mq_insurance_port Extension Logical Object

mq_insurance_ops Binding

mq_insurance_po Type

Input

Options

Object Display Mode

- All Objects
- Derived Objects
- External Relationships

Graph Display Depth

- Automatic

List History

WebSphere MQ V7.0.1

WebSphere MQ V7.0.1 Rationale

- Objectives

- Early delivery of non-disruptive function
- Providing platform for other products to build on
- Minimise migration costs

- Major themes

- Enhanced availability options on Distributed platforms
- Constraint relief on z/OS
- IBM portfolio exploitation, and extension of reach, for V7 features
- Ongoing performance, consumability and serviceability enhancements
- Keeping pace with industry evolution in areas such as platforms and SSL

WebSphere MQ V7.0.1 Rationale

N**O****T****E****S**

- WMQ V7.0.1 is designed to be an vehicle for delivery of function earlier than might have been expected with a traditional release cycle (for example with a V7.1 or a V8).
- Some of the function included in this release is a prerequisite on which other products such as Message Broker will build.

WMQ V7.0.1 Content Summary

New Feature	Benefits	Details
Multi-Instance Queue Managers	Increases availability Does not require specialist skills Can help ease system maintenance	Enables automatic failover to a standby Queue Manager instance in the event of an incident or planned outage
Automatic Client Reconnect	Increases availability Simplifies programming	Provides Client-connected applications with automatic detection of failures and reconnects to alternative Queue Managers
Enhanced Governance	Increases visibility of changes Enables SOA Governance	Emits events whenever configuration changes are made or commands are run Service Definition wizard generates WSDL describing MQ apps
Enhanced SSL Security	Simplifies security certificate management	Supports certificate checks with Online Certificate Status Protocol (OCSP) as well as to Certificate Revocation Lists (CRL)
Enhanced .NET support	Increases ease-of-use for .NET developers	Provides IBM Message Service Client for .NET developers Supports use of WebSphere MQ as custom channel within Windows Communication Foundation
Increased 64-bit z/OS exploitation	Increased use of z/OS system resources Provides constraint relief for virtual storage	Extends use of 64-bit storage by Queue Manager enabling more capacity such as number of open queues
z/OS Log Compression	Increased use of z/OS system resources Increased log performance & bandwidth	Compresses message logs produced by persistent messages
z/OS Group Units of Work	Increased resilience	Enables Units of Work to be owned collectively by Queue Sharing Groups so that any Queue Manager in the group can process two-phase transactions from clients
Publish/Subscribe Interfaces	Additional control of pub/sub behaviour Simplified integration for Message Broker	Exit point to dynamically modify routing and content Tools to migrate pub/sub state from MB to MQ



Installation and Delivery

- WMQ V7.0.1 is a modification release on the V7 base
 - Which means limited scope for new objects/attributes
 - Minimises migration aspects

- On Distributed platforms, it is available in two ways
 - A fixpack for upgrade from existing V7 installations (which can be backed out)
 - A replacement V7 installation image
 - Customers ordering V7 will now get V7.0.1
 - Single service stream for V7.0.x.y

- On z/OS, it is available as a modification level release
 - Migration supported from V6 and from V7.0.0
 - Customers ordering V7 will now get V7.0.1
 - New ZPARM OPMODE option to control whether new function is available

End of Service and Component Deprecation

- End of Service has been announced for V6 – Sept 2011

- Components being removed in next full release
 - File Transfer (Windows/Linux utility) – not to be confused with FTE product
 - API exerciser, Windows performance monitor integration
- C++ API (the Imq classes) to be stabilised
 - Cannot extend current code while maintaining application compatibility
 - Many C++ applications use the C MQI today.

- Next full release will not support HP-UX on PA-RISC hardware

- SupportPac **MA0F** (AMI) will inherit EoS for V6
- SupportPac **MS0E** (Admin Wrapper) will inherit EoS for V6
- SupportPac **MC91** (HA) being withdrawn immediately
 - Still downloadable, but not on main SupportPac page

End of Service and Component Deprecation

N**O****T****E****S**

- End of Support for V6 (and V7.0.0 on z/OS) is end of Sept 2011.
- <http://www.ibm.com/software/support/lifecycle/> contains details.
- The C++ classes will continue to be supported, and kept in step with any necessary compiler version changes, but the function available through those classes will not be enhanced.
- Later slides show why MC91 is being withdrawn.

Distributed Platforms: Multi-instance Queue Managers

- Basic failover support without HA coordinator
 - Faster takeover: fewer moving parts
 - Cheaper: no specialised software or administration skills needed
 - Windows, Unix, Linux platforms

- Queue manager data is held in networked storage
 - NAS, NFS, GPFS etc so more than one machine sees the queue manager data
 - Improves storage management options: formal support for these even without failover config
- Multiple (2) instances of a queue manager on different machines
 - One is “active” instance; other is “standby” instance
 - Active instance “owns” the queue manager’s files and will accept app connections
 - Standby instance does not “own” the queue manager’s files and apps cannot connect
 - If active instance fails, standby performs queue manager restart and becomes active

- Instances share data, so it’s the SAME queue manager

Distributed Platforms: Multi-instance Queue Managers

N

O

T

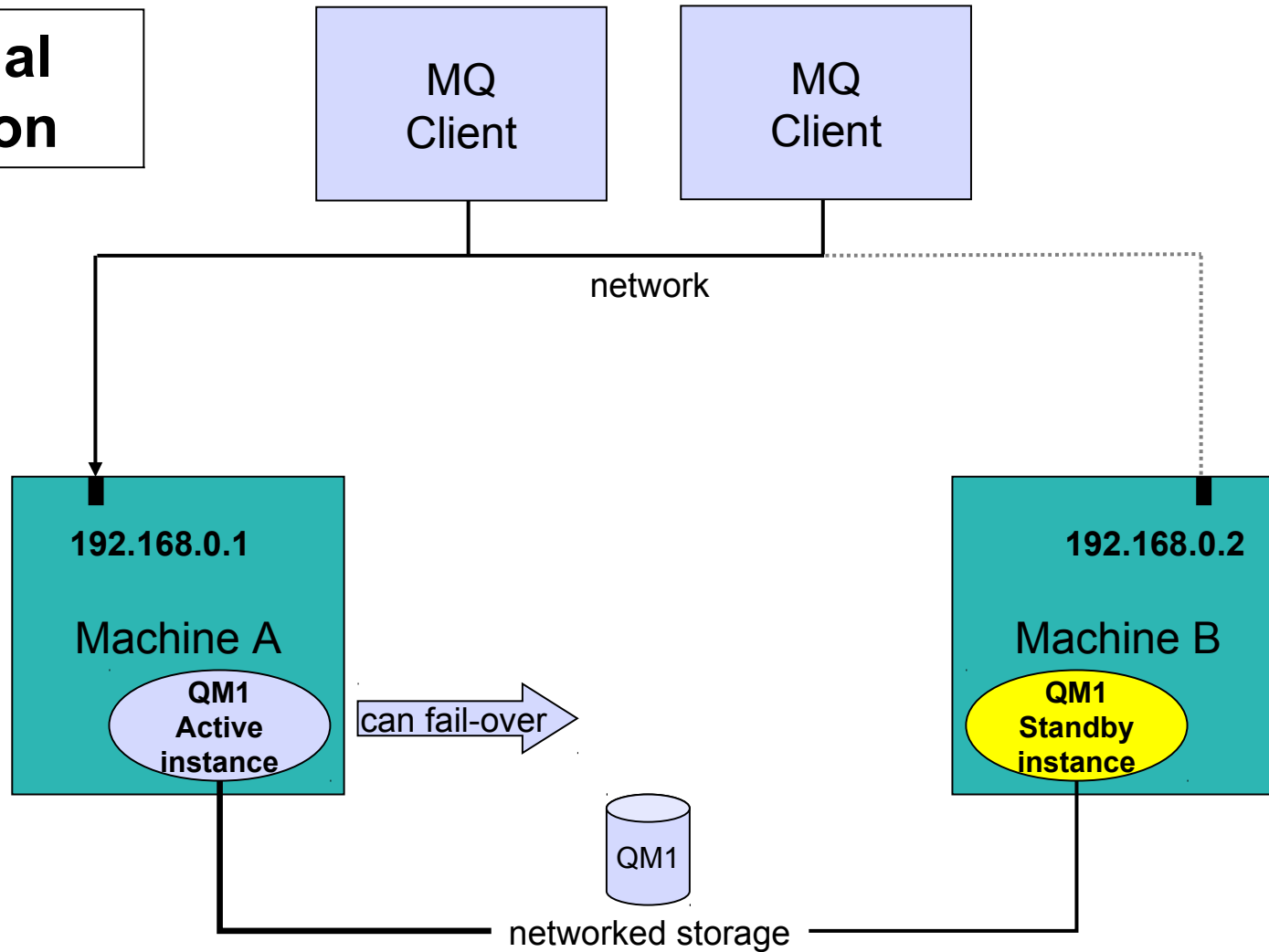
E

S

- “Basic failover”: no interaction with other resources like disks, IP addresses, databases, user applications etc. There is also no sophisticated control over where the queue managers run and move to (eg like a 3-node HACMP setup).
- Currently no System i: waiting for PTFs for their NFSv4 implementation.
- Architecturally, this is essentially the same as an existing HACMP/VCS setup, with the data shared between systems. It does not give anything “stronger” in terms of availability – but we do expect the typical takeover time to be significantly less. And it is much simpler to administer.
- Just as with an HACMP/VCS configuration, the takeover is at heart a restart of the queue manager, so non-persistent messages are discarded, queue manager channels go into retry etc.
- HACMP/VCS configurations are still a valid choice – but now you have an extra choice. You can continue to use HACMP/VCS solutions using MQ V7.0.1. While we do not (currently) ship, as standard, the start/stop/monitor commands that are currently in MC91, these are simple enough that you should be able to write them anyway. The queue manager is now more sensitive to when its internal processes die, so the MC91 “kill” script that iterated through all process names should not be necessary.

Multi-instance Queue Managers

1. Normal Execution

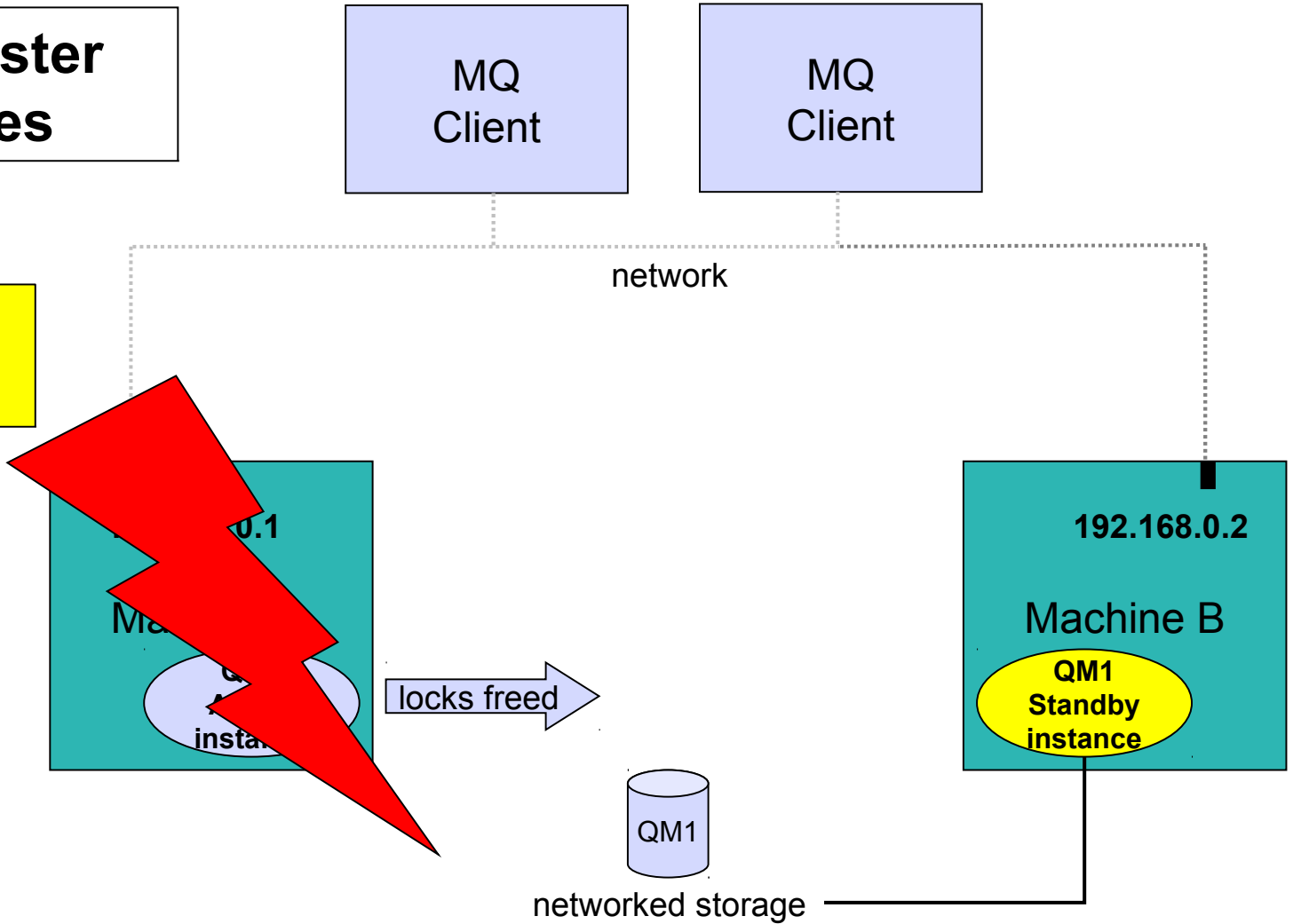


Owne the queue manager data

Multi-instance Queue Managers

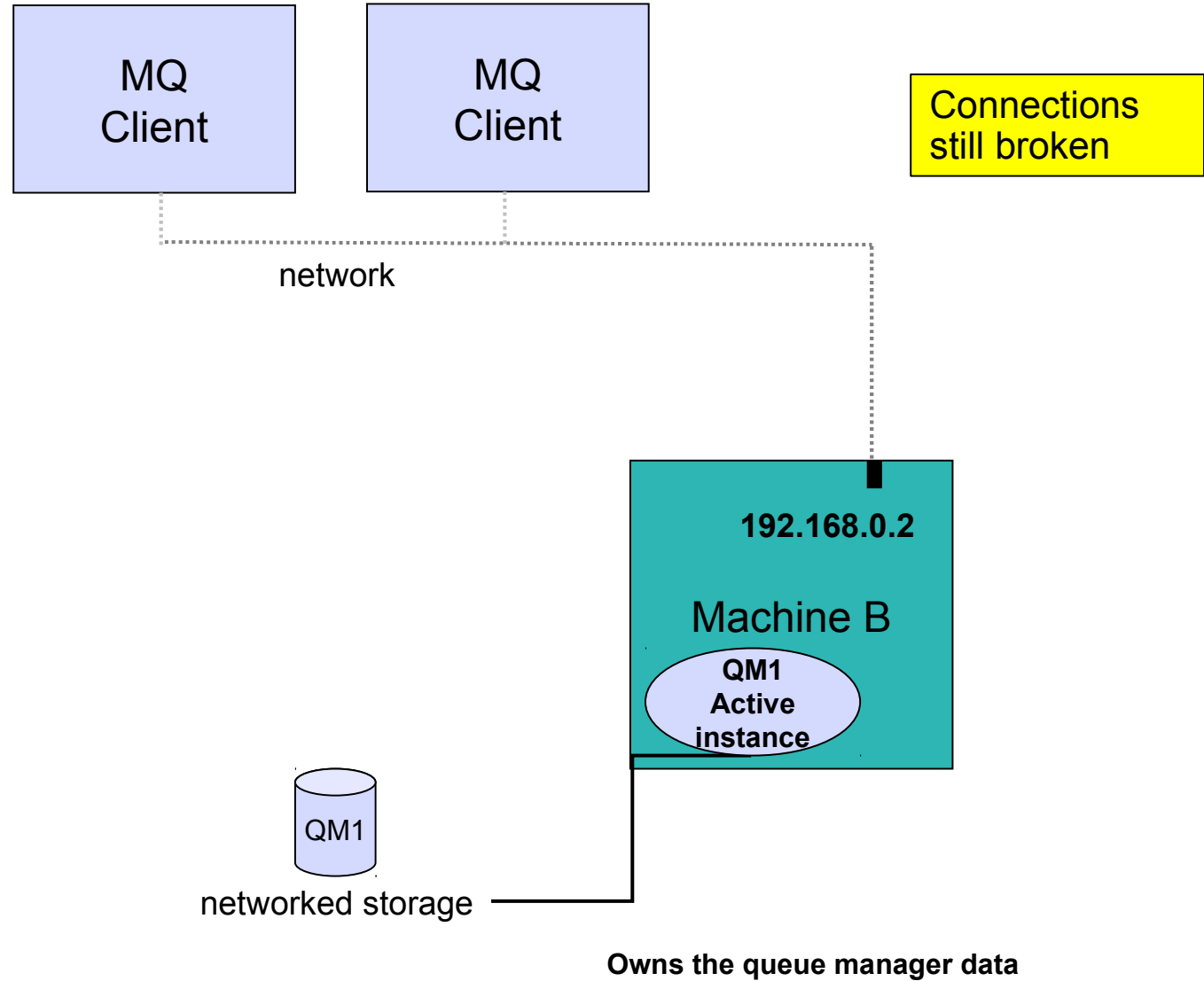
2. Disaster Strikes

Connections broken from clients



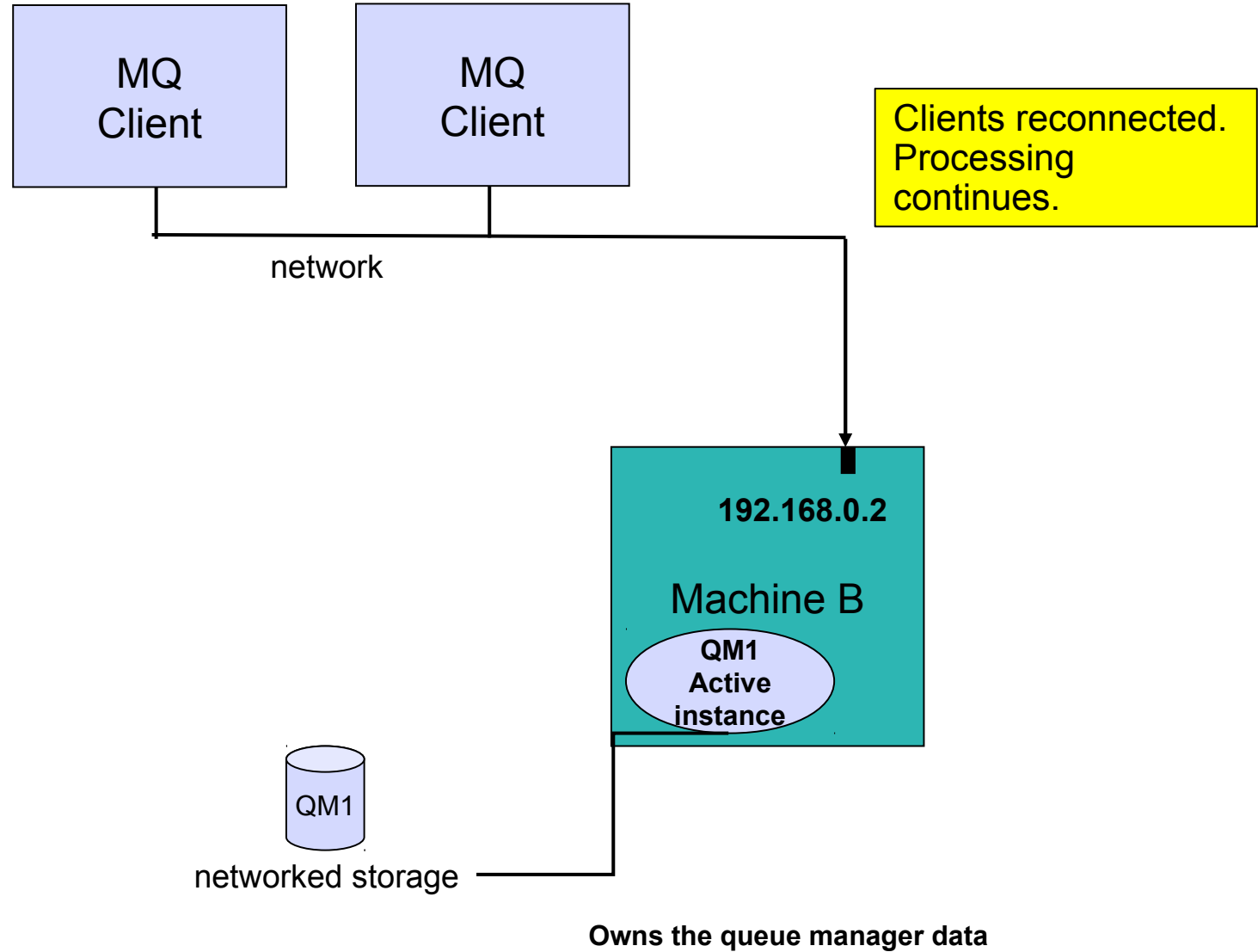
Multi-instance Queue Managers

3. Standby Comes to Life



Multi-instance Queue Managers

4. Recovery Complete



Multi-instance queue managers: Details

- MQ is NOT becoming an HA coordinator
 - Generally, if other resources also required, use an HA coordinator such as HACMP
 - Service objects can restart applications with qmgr but limited control
 - Message Broker will integrate with and exploit this MQ function
- The IP address is not taken over
 - Channel config needs all possible addresses unless you use external IPAT or intelligent router
 - CONNAME('host1(port1),host2(port2)') syntax extension on all platforms including z/OS
- Support for networked storage over modern network file system protocols
 - For example, NFS v4 (not v3)
 - Tool shipped to validate configuration
- New options for crtmqm/strmqm/endmqm to control operations
 - Cannot guarantee which instance becomes the primary
- Removes need for MC91, which will be withdrawn
 - crtmqm now does equivalent of MC91's hacrtmqm

Multi-instance queue managers: Details

N

O

T

E

S

- QMgr Service Objects have no enforced or implied sequence which make them difficult to use for dependency management in these scenarios. They also force the started applications to run with mqm authority (unless sudo-ish things are done).
- Failover time depends on 2 things:
 - a) How quickly the standby notices the active has gone. This will be near-instantaneous when just the qmgr fails or is explicitly ended. Will probably be longer, based on filesystem timeouts, when the whole box fails.
 - b) The amount of work that has to be done during restart, typically dependent on any long-running transactions that need to be replayed or backed out.
- From the announcement letter:
 - Unix: For multi-instance queue managers, you will need a networked storage device (such as a NAS). The storage must be accessed by a network file system protocol which is Posix-compliant and supports lease-based locking. NFS v4 and IBM GPFS both satisfy this requirement. Earlier versions of NFS do not satisfy this requirement and must not be used with multi-instance queue managers.
 - Windows: A networked storage device accessed by the Common Internet File System (CIFS) protocol used by Microsoft Windows networks is required.
- MC91 is being moved to the “withdrawn” page for SupportPacs so it is still available, but it is no longer as prominent for HA capabilities. The regular commands like crtmqm now do the necessary work previously done by hacrtmqm (although in a different way); there is a new addmqinf command that is the equivalent of halinkmqm. The scripts continue to work with V7.0.1 both for existing/upgraded qmgrs, and for newly created ones. But they are not recommended.

Multi-instance queue managers: How it looks

- Enhanced dspmq
- New option for dspmq to output English-only text
 - Useful for programmable parsing

```
$ hostname
rockall
$ dspmq -x
QMNAME (V7)          STATUS (Running)
      INSTANCE (rockall)  MODE (Active)
QMNAME (V7B)        STATUS (Running)
      INSTANCE (rockall)  MODE (Active)
QMNAME (V7C)        STATUS (Running as standby)
      INSTANCE (llareggub)  MODE (Active)
      INSTANCE (rockall)  MODE (Standby)
```

Multi-instance queue managers: How it looks

- As a graphical example, SupportPac MS0P V7.0.1

The screenshot shows the WebSphere MQ Explorer interface. The main content area displays a table titled "Queue Managers on rockall". The table has the following columns: Name, Listener Port, QMID, Admin Obj..., State, and Instances. The data rows are as follows:

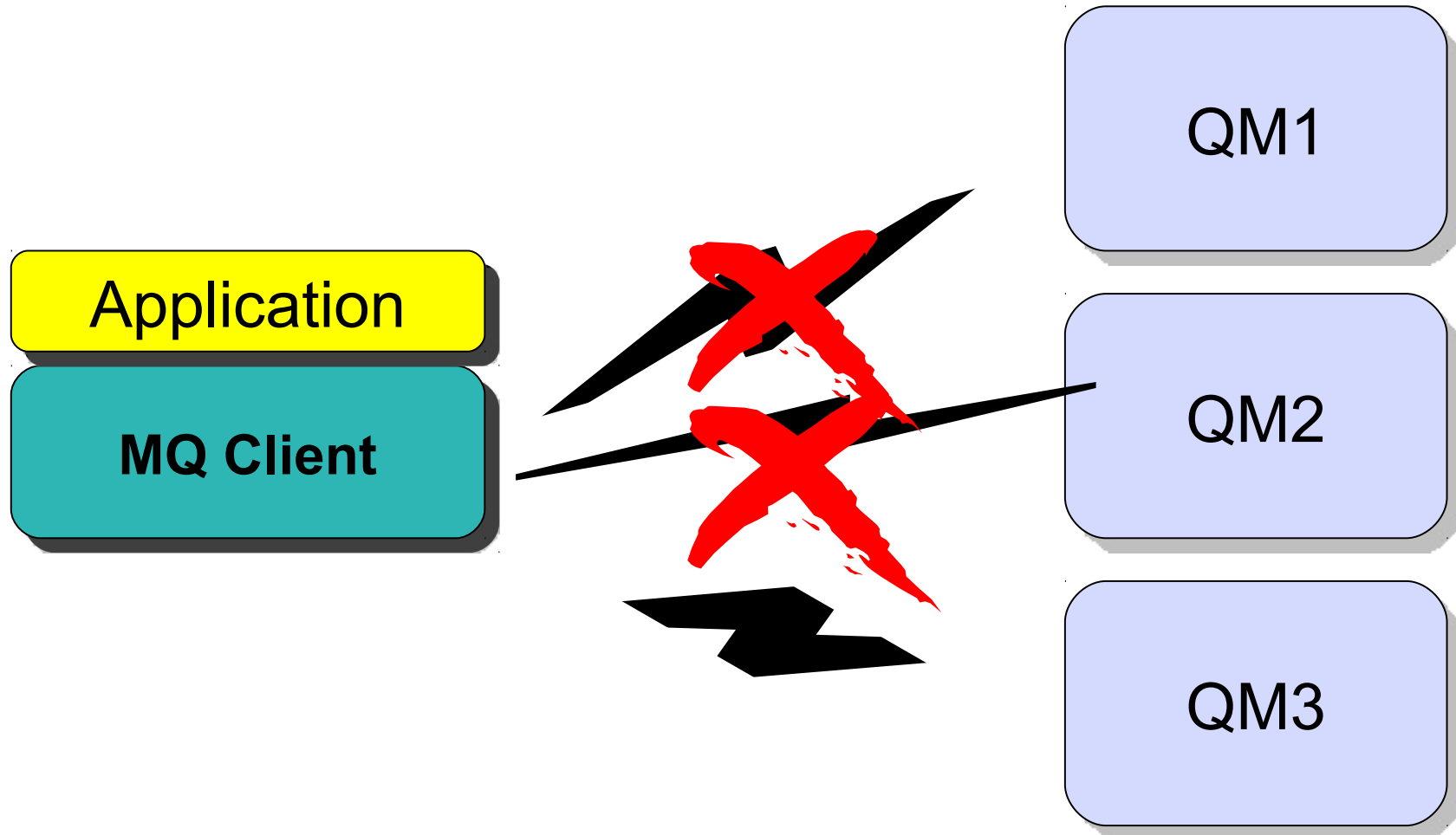
Name	Listener Port	QMID	Admin Obj...	State	Instances
ATS_AIX	3414	ATS_AIX_2009-04-21...	No	Running	rockall(Active)
linear	1414	linear_2009-04-21_1...	Yes	Running	rockall(Active)
V53	Unknown	Unknown	Unknown	Stopped	Unknown
V7	2414	V7_2009-04-21_12.3...	Yes	Running	rockall(Active)
V7B	2415	V7B_2009-04-21_12....	Yes	Running	rockall(Active)
V7C	Unknown	Unknown	Unknown	Running as standby	llareggub(Active), rockall(Standby)

The status bar at the bottom of the window shows a table with the following columns: Source, Destination, Current File, File Number, Progress, Rate, and Started (Europe/London). The status bar is currently empty.



Automatic Client Reconnection

- Client library provides necessary reconnection logic on detection of a failure
- Hides failure from application code



Automatic Client Reconnection: Details

- Enabled in application code or ini file
 - Event Handler callback shows reconnection is happening if app cares

- Tries to keep dynamic queues with same name
 - So replies may not be missed
- Not all MQI is seamless, but majority repaired transparently
 - eg a browse cursor would revert to the top of the queue, non-persistent messages will have been lost during restart, non-durable subscriptions may miss some messages, in-flight transactions backed out, hObj values maintained
- Some MQI options will fail if you have reconnection enabled
 - Using MQGMO_LOGICAL_ORDER, MQGET gives MQRC_RECONNECT_INCOMPATIBLE

- Initially just in MQI and JMS – not the other OO classes
 - Requires both client and server to be V7.0.1 level with SHARECNV>0
 - Server can be z/OS



Client Reconnection

NOTES

- A TDQ is not deleted immediately when a connection fails, so a reconnecting client should find it if it lost connection due to a network failure. If the queue manager has failed over to the standby instance, the original TDQ will have been deleted but a new one will be created automatically by the client using the same name. So depending on the nature and timing of the failure, reply messages may still be accessible.
- MQCNO flags: RECONNECT, RECONNECT_Q_MGR, RECONNECT_DISABLED
- Options not supported: MQPMO_LOGICAL_ORDER, MQGMO_LOGICAL_ORDER
- MQPUT of PERSISTENT message outside of syncpoint
 - May return MQRC_CALL_INTERRUPTED
- MQSTAT may return only a 'subset' of information
 - Reconnection may have occurred during a message sequence
 - Can be used to query reconnection status
- Failure 'during' transaction
 - Transaction is 'doomed' – final MQCMIT will always cause a rollback
 - MQRC_BACKED_OUT returned to application
 - Non-transactional operations are allowed to complete
- XA and reconnection are mutually exclusive
 - The XA interface is too restrictive to guarantee data integrity between multiple resource managers if a connection to MQ is re-established silently

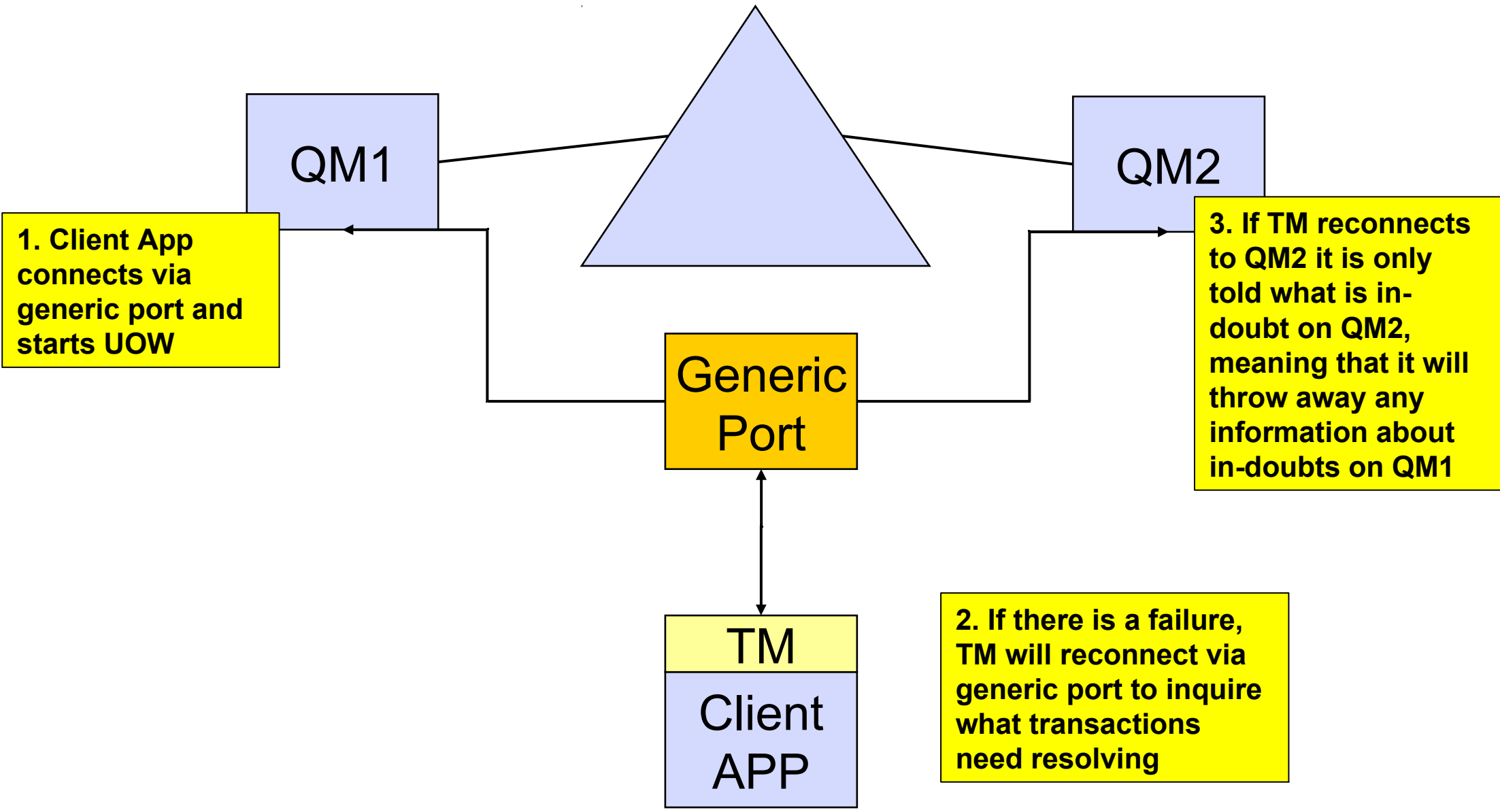
Group-level Units of Recovery for z/OS

- A client's two-phase/global transaction can now be owned by a QSG
 - Instead of by individual queue managers

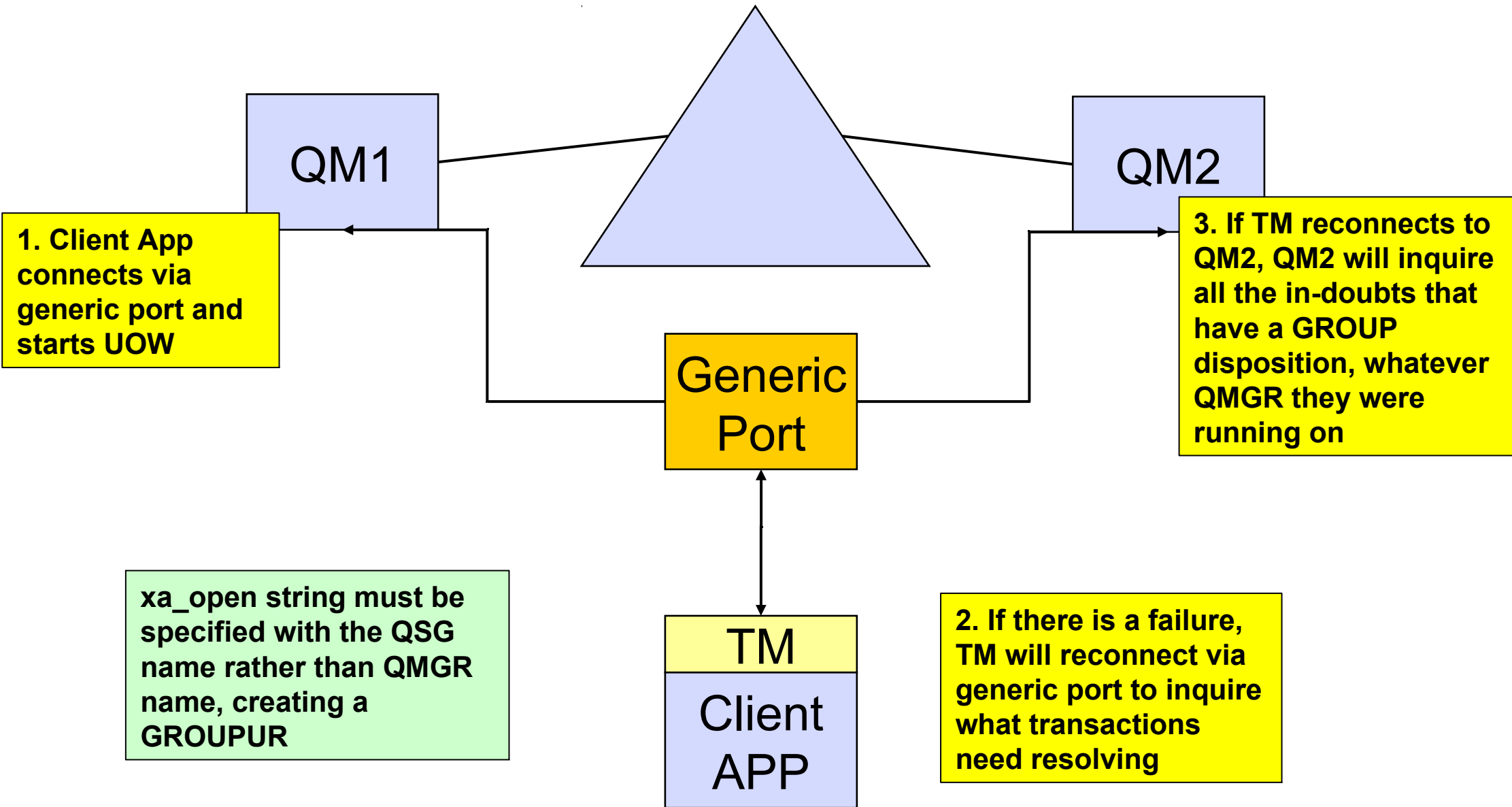
- These in-doubt transactions can be resolved on any QMGR in the QSG.
 - Required for support 2-phase commit resolution while connected to the QSG

- Requires use of the Extended Transactional Client
 - For example, from WAS
 - Configure the WAS client connection with the QSG name rather than the QMGR name

GROUPUR: The Problem



GROUPUR: The Solution (V7.0.1)



Windows Communication Framework

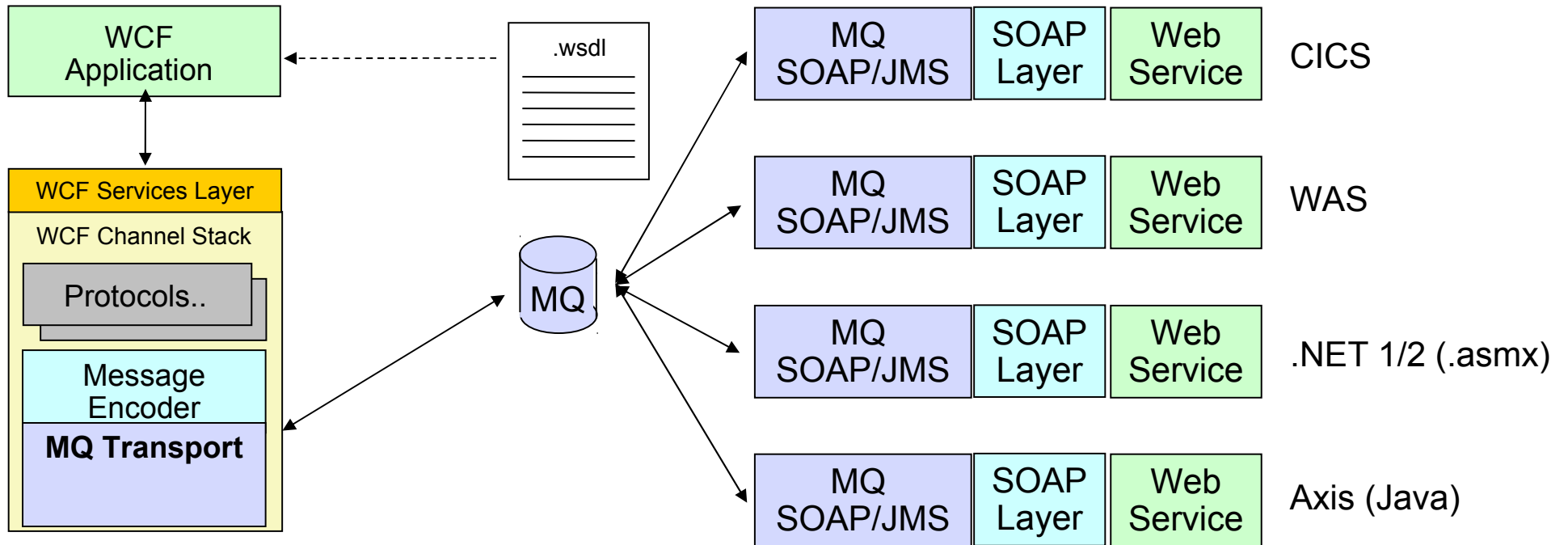
- Windows Communication Framework (WCF)
 - Underpins .NET Web Services and Messaging
 - Built-in Transports – e.g. MSMQ, HTTP(S), Named Pipes, TCP/IP, etc.
 - Transports can be extended with “custom channels”
 - Allows alternative transports (like MQ) to be slotted into WCF seamlessly

- WCF support now included in product
 - Previously an Alphaworks prototype

- Built on XMS .Net classes
 - Now shipped as part of the product instead of SupportPac
 - Internally, it exploits V7 API, in same way as JMS implementation

WCF: Usage within MQ

- Interface to web services hosted over MQ using SOAP/JMS
- Extending to other application servers with adoption of W3C SOAP/JMS standard (BEA, Sonic, TIBCO, Axis)



WCF: Usage within MQ

N**O****T****E****S**

- What is being provided here is the first phase of features and architectural possibilities for use of WCF.
- Over time we would expect to see more features, such as better transactional support and additional security configuration. There are also options such as exploiting a WSDL definition of an application that are being considered.
- But this component provides the core piece that is needed, based on highest priority customer requirements.
- The XMS .Net implementation has been redesigned to follow the same model internally as the V7 JMS classes. These classes are now included in the WMQ product instead of being a SupportPac. They are installed automatically on upgrade to 7.0.1 if the Java Messaging and Web Services component was previously installed.
- The C and C++ XMS classes are, for now, still a separate SupportPac.

Simplified Administration

- Remote runmqsc no longer requires default qmgr
 - New optional “-m” flag to specify a local qmgr when “-w” also used
 - Useful in active/active HA configurations where no default qmgr may exist

- Can set dsp permissions for any user on SYSTEM.AUTH.DATA.QUEUE
 - Making it easier to have “read-only” administration
 - No more Authorisation Failure Events if someone tries to access this queue
 - Also being retro-fitted to V6 via APAR IZ52608: expected in 6.0.2.8

Publish/Subscribe Enhancements

- Option to discover if no subscribers (user or proxy) during MQPUT/PUT1
 - MQPMO_WARN_IF_NO_SUBS_MATCHED
 - MQRC_NO_SUBS_MATCHED
- Will guarantee that no one has received the publication
 - But does NOT guarantee that anyone will definitely receive the publication
 - For example, it is not returned if the target queue is full

- Publish Exit
 - When a publication is made, this exit is invoked for each valid subscriber
 - Runs “inside” the queue manager
 - Can change routing destination
 - Can change contents of message
 - Can change contents of message descriptor
 - Can inhibit publication

Publish/Subscribe Enhancements for Message Broker

- Message Broker will exploit and integrate with WMQ's pub/sub engine
 - Instead of having an independent implementation
 - Easily connects MB's wide range of connectivity and format support to WMQ backbone
 - Common topic domain

- V7.0.1 assists with migration for current users of MB's pub/sub
 - Both tools and documentation provided
 - Extracts lists of topics and subscriptions to automatically populate WMQ objects
 - Help also provided with authorisations
 - Purely administrative so that developers of message flows will not see any change

- MB still used for selection based on message content
 - WMQ does the filtering of subscribers based on topic and message properties
 - MB parses message content to do further level of filtering



Publish/Subscribe Enhancements

NOTES

- Full Message Broker integration and exploitation for the administration of pub/sub requires a post-V6.1 Message Broker.
- In preparation for that, WMQ is providing tools and documentation to help migration. State is read from an existing Message Broker (or Event Broker) and – as far as possible – replicated in WMQ’s definitions. Partly because of the different security models that exist, not everything can be precisely moved, but these are called out during the migration process.
- The migration tools are not on z/OS initially, but will be enabled via PTF soon after V7.0.1 is available.
- The MB application development and “runtime” aspects are essentially unchanged – existing message flows continue to work with no indication that WMQ is now doing some of the work that MB used to do.
- This will join WMQ and MB into a common pub/sub domain, with a common topic space and a common security model.

Security & Monitoring

- Command and Configuration Events for Distributed platforms
 - Matching already-available z/OS function
 - Successful commands – MQSC or PCF – recorded as event messages
 - Configuration changes report “before” and “after” definitions of objects
 - Provides a record of who changed what and when

```
display qmgr event
AMQ8408: Display Queue Manager details.
      QMNAME (V7)  CMDEV (ENABLED)  CONFIGEV (ENABLED)  ...
```

- SSL OCSP Support
 - Now commonly used as alternative to LDAP-based CRLs
 - Simpler to manage as no need to have an LDAP server
 - Can use details provided in inbound certificate
 - Also can be configured within queue manager or by application code



Security & Monitoring

N**O****T****E****S**

- Command events do not currently record the security commands like setmqaut. They also do not record other command line programs such as strmqm.
- The formats of these events are common across z/OS and Distributed platforms.
- Many monitoring tools will already understand the formats; they may just have to be pointed at the appropriate EVENT queue.
- Remember that the event queues can be redefined as aliases to topics, so events can be delivered to multiple consumers via pub/sub.
- As on z/OS, only SUCCESSFUL commands are reported. Failures due to security problems will be recorded in the usual way via Authorisation events. Command Events can be configured to not record DISPLAY/Inquire commands.
- Use of OCSP can be driven purely from information found within the incoming certificate.
- Alternatively it can be configured on queue manager using new AUTHINFO subtype and SSLCRLNL definitions (Distributed platforms only)
- Configurable in CCDT using new AUTHINFO subtype (all platforms)
- Client applications can directly specify OCSP servers using the MQAIR structure in the MQI.

z/OS Constraint Relief

- In V7.0, the queue manager started to exploit 64-bit addressing
 - New Pub/Sub features

- In V7.0.1 more Queue Manager storage moves to 64-bit
 - 64-bit Queue Indices
 - 64-bit Lock Manager
 - 64-bit Security Profile Cache

- Can have more open queues, more messages on indexed queues etc

- Small message storage changed
 - One message per page instead of fitting multiples into single page
 - May increase DASD use, especially if messages not quickly retrieved
 - But typical workloads show improved performance and reduced CPU



z/OS Constraint Relief

NOTES

- In V7.0, MQ took the first step towards having a 64-bit implementation on z/OS. New elements for the publish/subscribe components were written to use this addressing.
- In V7.0.1, the next step is to move some more of the queue manager's internal tables using the new infrastructure. This is invisible to users, except it frees space for the other internal tables which can now be larger. For example, it should now be possible to have more messages on indexed queues or have more open queues without running out of storage.
- The overall effects will of course be heavily dependent on workload.
- The way messages are stored on disk has changed to put small messages (<2K) in their own page instead of trying to fit as many messages as possible into a page. While this has the potential to increase DASD requirements (more "wasted" space per message), measurements have shown improvements in throughput, and reduced steady-state pageset, because the page becomes available for reuse faster. The CPU costs can also go down because there is less processing working out where to store the message.

z/OS Log Compression

- Can increase the throughput possible for persistent messages

- May reduce the size of your logs
 - Dependent on your message content
 - Useful if you are DASD constrained.

- RLE (run-length encoding) of “insert” log records for private queue messages
 - Will not compress shared queue log records
 - SMF 115 records updated to show compression rates achieved etc
- Controlled via zPARM option at queue manager level.
 - COMPLOG(NONE) or COMPLOG(RLE) in CSQ6LOGP
 - Can also be viewed/controlled via DISPLAY LOG / SET LOG



z/OS Log Compression

N**O****T****E****S**

- For log compression, this improves the I/O processing, reducing the bandwidth required to log persistent messages. Clearly there is some CPU cost associated with the compression (and, if required during recovery, decompression). But this may be offset by the I/O savings – and for some customers, it's the I/O usage that was most critical in limiting their workload.
- Compression is done using the same run-length encoding algorithm that is an option on channels.

WebSphere MQ V7.0.1: Summary

- Continued enhancements to the messaging backbone
- Early delivery of new function – no need to wait for the next full version
- Simplification for administrators and application developers
- Improved system exploitation