This module concentrates on methods for connecting publish/subscribe queue managers.

Agenda

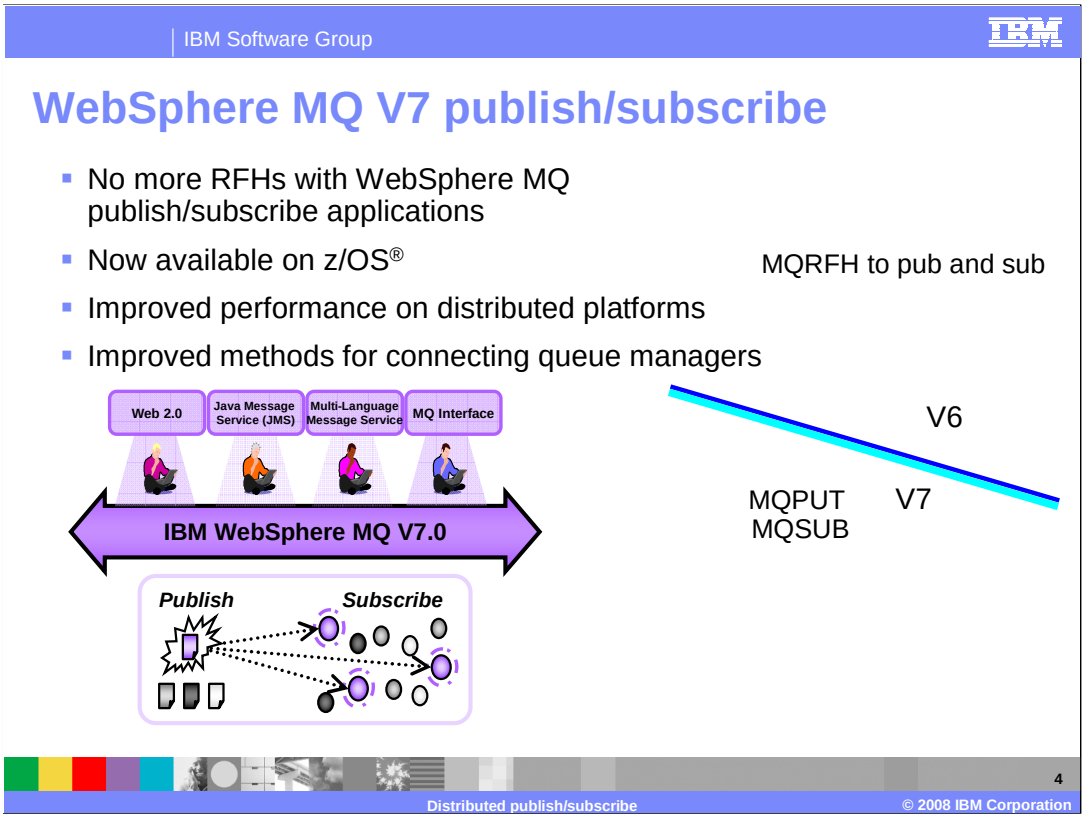
- Publish/subscribe in WebSphere MQ V7
- Distributed publish/subscribe approaches
  - Publish/subscribe clusters
  - Hierarchies
- Administration and migration
- Comparison and recommendations
- Summary

Distributed publish/subscribe approaches are discussed, including administration and a comparison of the approaches.

Section

Publish/subscribe in WebSphere MQ V7

This section covers the enhancements to publish/subscribe in WebSphere MQ V7.



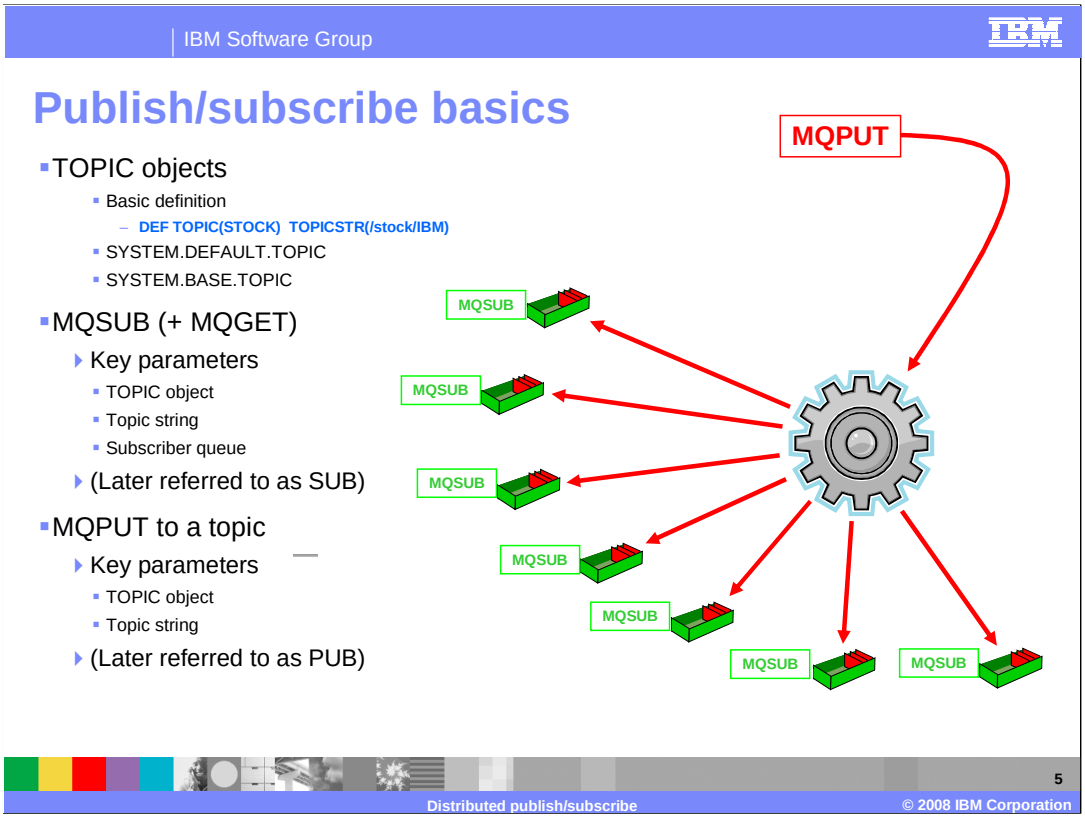
WebSphere MQ V7 now includes publish/subscribe built into the API. This replaces the publish/subscribe function offered in WebSphere MQ V6 which was built on an MQRFH message interface.

Publish/subscribe was not supported for z/OS in WebSphere MQ V6. WebSphere MQ V7 introduces publish/subscribe on the z/OS platform.

The new publish/subscribe available in V7 is easier to use and performs better than V6 publish/subscribe.

In addition to hierarchies, which were available in V6, V7 offers publish/subscribe clusters.

WebSphere MQ exposes interfaces to both server and client applications over Web 2.0, JMS, XMS, and MQI.



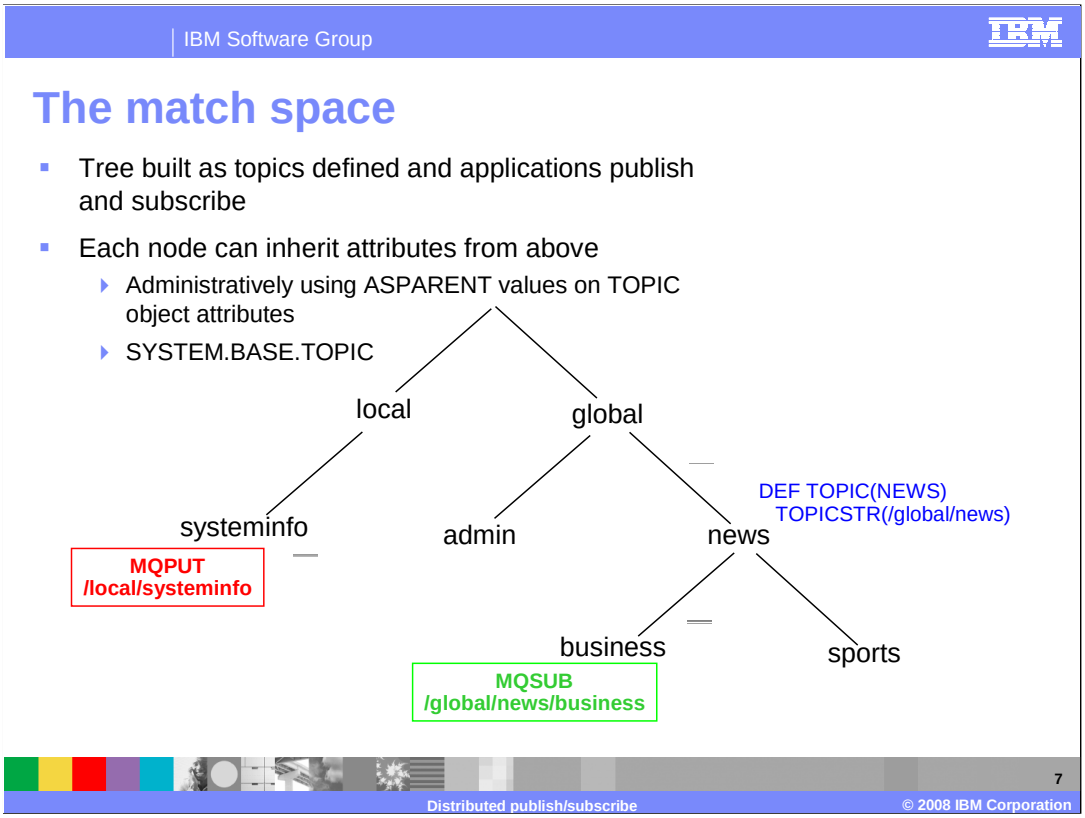
WebSphere MQ V7 introduces the TOPIC object. A topic is a character string that describes the subject of the information that is published in a publish/subscribe message. Instead of including a specific destination address in each message, a publisher assigns a topic to each message. The queue manager matches the topic with a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.

The new API MQSUB, and the existing MQGET API, provide the subscriber the capability to subscribe to a topic. The MQPUT API provides the publisher the capability to publish a topic.

Section

Distributed publish/subscribe approaches

This section describes various approaches to distributed publish/subscribe.



A *publish/subscribe topology* consists of queue managers and the connections between them, that support publish/subscribe applications.

A publish/subscribe application can consist of a network of queue managers connected together. The queue managers can all be on the same physical system, or they can be distributed over several physical systems. By connecting queue managers together, publications can be received by an application using any queue manager in the network.

A tree is built as topics are defined and applications publish and subscribe.

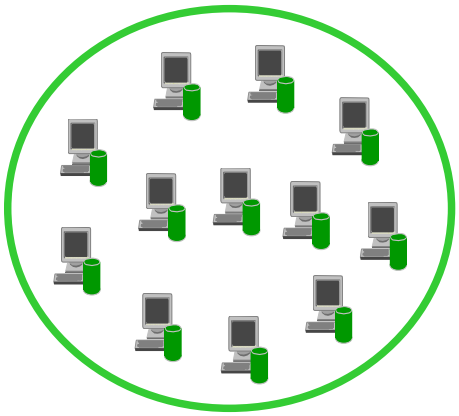
Manual subscriber

- Manual subscriber alternative
  - Direct one to one connectivity
  - Forward publications to a remote queue

Subscriptions can be created manually using an MQSC command or by applications. These subscriptions are issued to the local queue manager and contain information about the publications the subscriber wants to receive. Publishing to a remote queue which is local to the subscriber provides the distributed aspect. This is, however, a one to one connection.

Publish/subscribe clusters

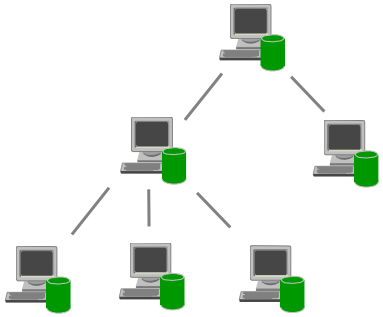
- Publish/subscribe clusters
  - Direct many to many connectivity
  - Based on WebSphere MQ clustering



A *publish/subscribe cluster* is a set of queue managers that are fully interconnected and form part of a multi-queue manager network for publish/subscribe applications. A cluster that is used for publish/subscribe messaging is no different from a standard WebSphere MQ cluster. As such, the queue managers within the publish/subscribe cluster can exist on physically separate computers and each pair of queue managers is connected together by a pair of channels.

Publish/subscribe hierarchies

- Hierarchies
  - Indirect many to many connectivity
  - Direct one to many connectivity
  - Similar to WebSphere MQ V6 hierarchies



Queue managers can be grouped together in a hierarchy, where the hierarchy contains one or more queue managers that are directly connected. Queue managers are connected together using a connection-time parent and child relationship.

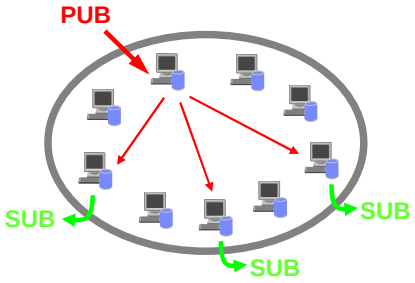
Section

Publish/subscribe clusters

This section gives details on publish/subscribe clusters.

Publish/subscribe clusters

- Similar to WebSphere Message Broker collectives
- Availability provided by direct links between qmgrs
- Benefits of traditional clusters
  - Simplified administration
    - Useful for publish/subscribe clusters
    - Non-disruptive addition and removal of qmgrs
  - Workload balancing
    - NOT useful for publish/subscribe clusters
- Cluster channels
  - Same CLUSSDR/CLUSRCVR model
  - Flexible connectivity
  - All to all connectivity
  - Overlapping clusters
    - No publish/subscribe through gateway queue managers
    - Use hierarchies for linking publish/subscribe clusters
- Cluster topics
  - Required for publishing to remote publish/subscribe cluster queue managers
- Cluster queues
  - Can be used in traditional way in a publish/subscribe cluster



Like traditional clusters, publish/subscribe clusters are designed for many to many queue manager connectivity. In traditional clusters, cluster objects are automatically defined based on usage, such as a put to a queue. The usage model is based on a putter using a set of cluster queues, not necessarily defined on all queue managers in the cluster. Therefore in traditional clusters it is unlikely that all queue managers will actually be connected to all other queue managers by auto-defined channels.

In publish/subscribe clusters, cluster objects are automatically defined before usage, at the time the first cluster TOPIC is defined in the cluster. This is because the usage model is different to that of traditional clusters. Channels are required from any queue manager to which a subscriber to a cluster topic is connected to all other queue managers so that a proxy-subscription can be fanned out to all the queue managers. Channels are also required back from any queue manager where a publisher of a cluster topic is connected to those queue managers which have a connected subscriber. Therefore in publish/subscribe clusters it is much more likely that all queue managers will actually be connected to all other queue managers by auto-defined channels. It is for this reason that, in publish/subscribe clusters, cluster objects are automatically defined before usage.

Cluster topics

- Required for publishing to remote publish/subscribe cluster QMs
- Publish/subscribe cluster exists when there are one or more cluster topics
  - Cluster topic is pushed to ALL queue managers in the cluster
  - Channels are automatically defined between ALL queue managers in the cluster
- Not to be confused with a subscriber

- Define
  - **DEFINE TOPIC(FOOTBALL) TOPICSTR(/global/sports/football) CLUSTER(SPORTS)**
- Display
  - **DISPLAY TOPIC(FOOTBALL)**
  - **DISPLAY TOPIC(FOOTBALL) TYPE(LOCAL)**
  - **DISPLAY TOPIC(FOOTBALL) TYPE(ALL)**
    - Local objects only
  - **DISPLAY TOPIC(FOOTBALL) TYPE(CLUSTER)**
  - **DISPLAY TOPIC(FOOTBALL) CLUSINFO**
  - **DISPLAY TCLUSTER(FOOTBALL)**
    - Cluster objects only
  - **DISPLAY TOPIC(FOOTBALL) TYPE(ALL) CLUSINFO**
    - Both local and cluster objects

Cluster TOPICS are regular TOPIC objects that are advertised to the cluster. When a cluster TOPIC is defined, the cluster in which it is defined becomes a publish/subscribe cluster.

To define a cluster TOPIC, you must provide a topic object name, topic string, and cluster name. The queue manager on which the topic is defined should be a member of the specified cluster.

There are two types of cluster topics displayed when displaying cluster topics. A display of type local shows directly administrable objects. A display of type cluster shows the cluster cache records, based upon local objects. A display of type all with CLUSINFO keyword shows both local and cluster objects.

Cluster topic hosts

- Cluster topics can be defined on any queue manager in the cluster
  - Full or Partial Repositories
- Cluster topics can be defined on more than one queue manager
  - QM1 - **DEF TOPIC(SPORTS) TOPICSTR(/global/sports) CLUSTER(DEMO)**
  - QM2 - **DEF TOPIC(SPORTS) TOPICSTR(/global/sports) CLUSTER(DEMO)**
  - Multiple definitions ok but not necessarily concurrent
    - If topic host is lost, cluster topic will remain usable for up to 30 days
    - RESET CLUSTER the lost queue manager
  - Definitions should be identical
    - Behavior is undefined where attributes conflict
    - Conflict reported
      - AMQ9465 / AMQ9466
      - CSQX465I / CSQX466I

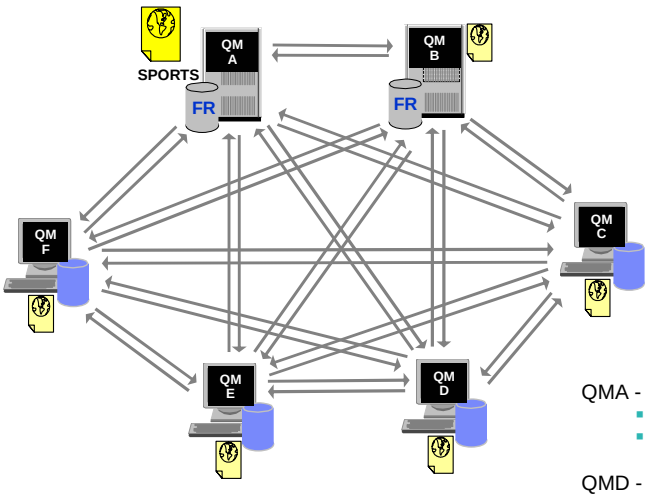
A cluster topic host is a queue manager with a clustered TOPIC object defined. Clustered TOPIC objects can be defined on any queue manager in the publish/subscribe cluster. When at least one clustered topic exists within a cluster the cluster is a publish/subscribe cluster. All clustered TOPIC objects should be identically defined on two queue managers and these machines should be highly available. If a single host of a clustered TOPIC object is lost, any cluster topic cache records based on the clustered topic object that exist in the cluster cache on other queue managers are usable within the cluster for a period of up to 30 days or until the cache is refreshed. The clustered TOPIC object can be redefined on a healthy queue manager. If a new object is not defined, up to 27 days after the host queue manager failure, all members of the cluster will report that an expected object update has not been received.

There is no requirement that full repositories and topic hosts overlap, or that they are separated. In publish/subscribe clusters that have just two highly available machines among many machines, you should define both as full repositories and cluster topic hosts. In publish/subscribe clusters with many highly available machines you should define full repositories, and cluster topic hosts on separate highly available machines. Then the operation and maintenance of one function can be managed without affecting the operation of other functions.

Although there is nothing wrong with having multiple identical definitions for a topic within a cluster, it adds administration overhead when changing. If definitions are not kept consistent unpredictable behavior is expected, as indicated by warning messages in logs.

Publish/subscribe cluster architecture 1

QMA - **DEFINE TOPIC(SPORTS) TOPICSTR(/global/sports) CLUSTER(DEMO)**

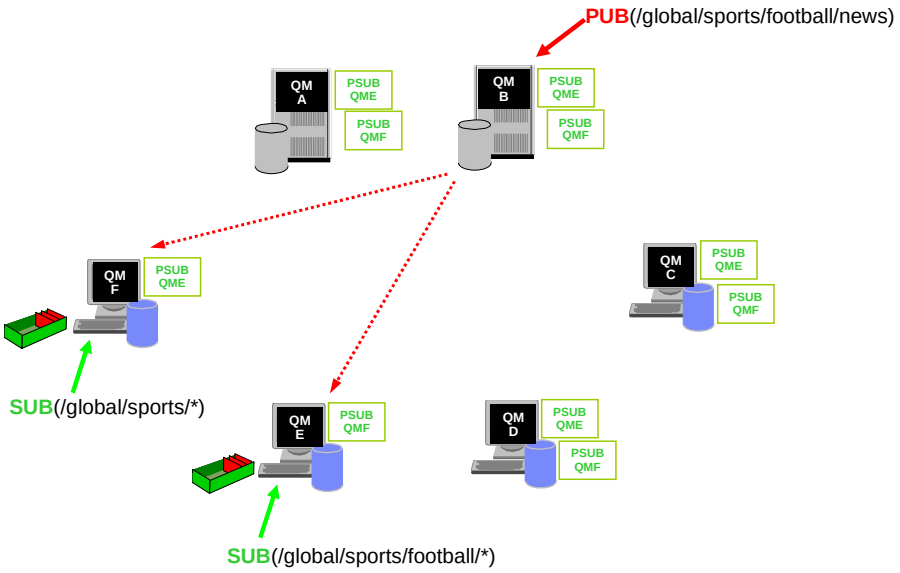


QMA - **DIS TOPIC(SPORTS) TYPE(ALL)**
- Local topic
- Cluster topic

QMD - **DIS TOPIC(SPORTS) TYPE(ALL)**
- Cluster topic

When a cluster topic is defined, the cluster topic object is published to the full repositories. The full repositories then push all cluster topic definitions to all queue managers within the cluster. Here queue manager QMA, in cluster DEMO, defines a cluster topic SPORTS. This topic definition is sent to all other queue managers in the cluster.

Publish/subscribe cluster architecture 2

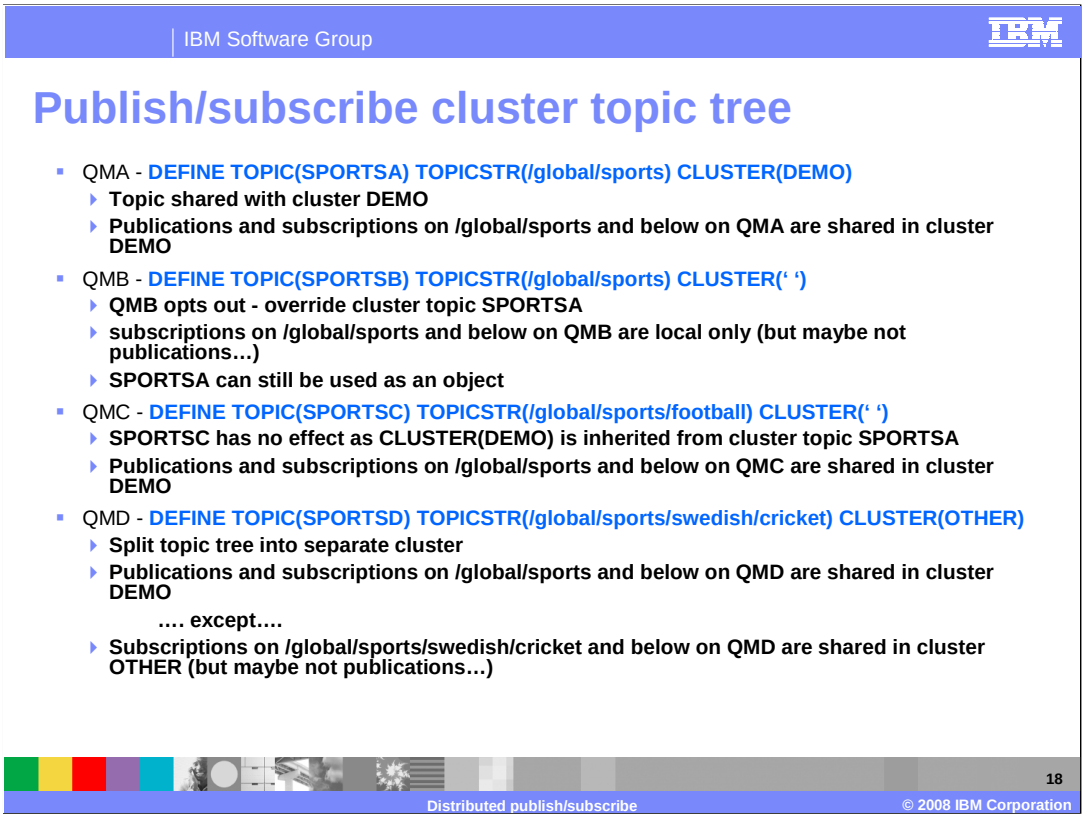


At each queue manager a single topic space is constructed from the local and cluster topic definitions. When an application subscribes to a topic that resolves to a clustered topic, WebSphere MQ creates a proxy subscription and sends it from the queue manager, to which the subscriber connected, to all members of the cluster in which the clustered topic object is defined. In the example here, the cluster topic string is /global/sports. The subscribers' topic strings are /global/sports/* and /global/sports/football/*. Proxy-subscriptions are sent to other queue managers in the publish/subscribe cluster for any topic string below the cluster topic string. The publisher's topic string is /global/sports/football/news; publications are sent based on local proxy-subscriptions.

Publish/subscribe cluster benefits

- Messages do not need to pass through an intermediate queue manager
- No single point of failure
- Can optimize flow of publications and subscriptions through the network
- Can group clients according to the topics they publish and subscribe
- Topology highly scalable

Using clusters in a publish/subscribe topology provides several benefits. Messages destined for a specific queue manager in the same cluster are transported directly to that queue manager and do not need to pass through an intermediate queue manager. There is no single point of failure in this topology. If your clients are geographically dispersed, you can set up a cluster in each location and connect the clusters by joining a single queue manager in each cluster. This will optimize the flow of publications and subscriptions through the network. You can group clients according to the topics to which they publish and subscribe. A subscribing application can connect to its nearest queue manager, to improve its own performance. The number of clients per queue manager can be reduced by adding more queue managers to the cluster to share workload. This makes a publish/subscribe cluster topology highly scalable.



Topic definitions which inherit from others within the topic space is a powerful tool, but can cause unwanted side affects if used carelessly. Here are some of the possible combinations and possible dangers.

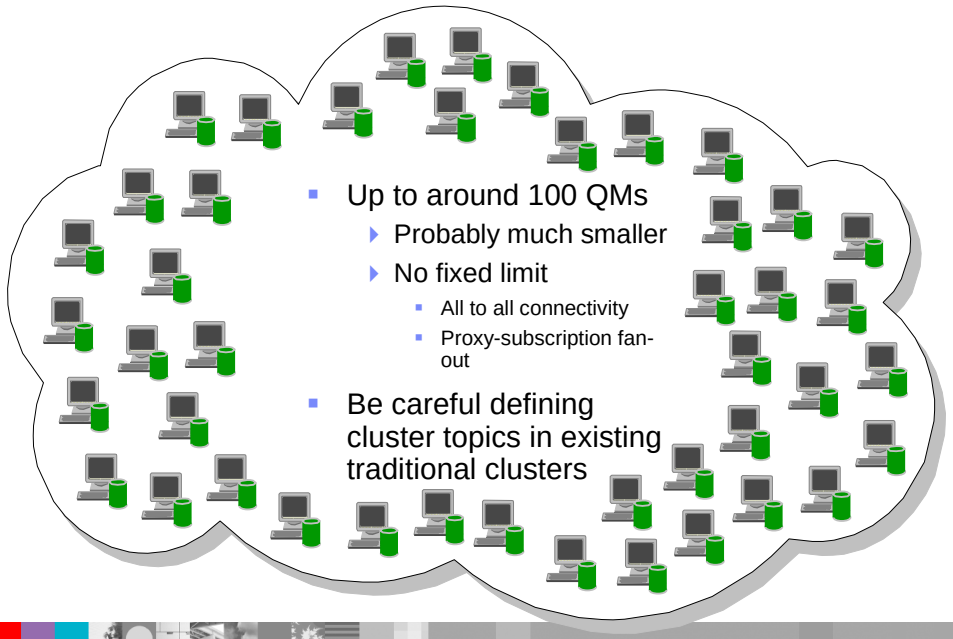
SportsA shows the simplest case of using a clustered topic definition to enable inter queue manager publish/subscribe activity in a cluster. This is the baseline 'normal' case.

SportsB describes a new definition on QMB which overrides SportsA. If the SportsA object is used on a publish or subscriber, as soon as the TopicString has been found in the match space it will actually be attributes from SportsB which affect processing. The cluster name of ' ' means it is inherited from any parent node in the topic tree. In this case there is no higher administration node defined, so the SportsB actually has a blank cluster name meaning no clustering. For the purposes of subscribers below this Topic, no proxy subscriptions are sent out. Note that as QMB still participates in cluster 'DEMO' it may receive proxy subscriptions on these topics, and publications will flow outwards. This may be counter intuitive! To stop this, pubscope(qmgr) and subscope(qmgr) can be used on the topic definition. Although you can use SportsA, any pubs or subs defined using it will be acting as though done on SportsB.

SportsC demonstrates the behavior when a cluster name **is** inherited. Because '/global/sports' has been linked to cluster 'DEMO', when a new topic is defined below this with cluster ' ', that value is inherited. Other attributes if defined on SportsC are still used; for example, the scope attributes described here could limit publish/subscribe beneath global/sports/football to the local queue manager.

SportsD looks complicated, but the only really non-intuitive behavior is again the situation where proxy subs exist for cluster 'DEMO' below the 'split' in the tree.

How large can a publish/subscribe cluster be?

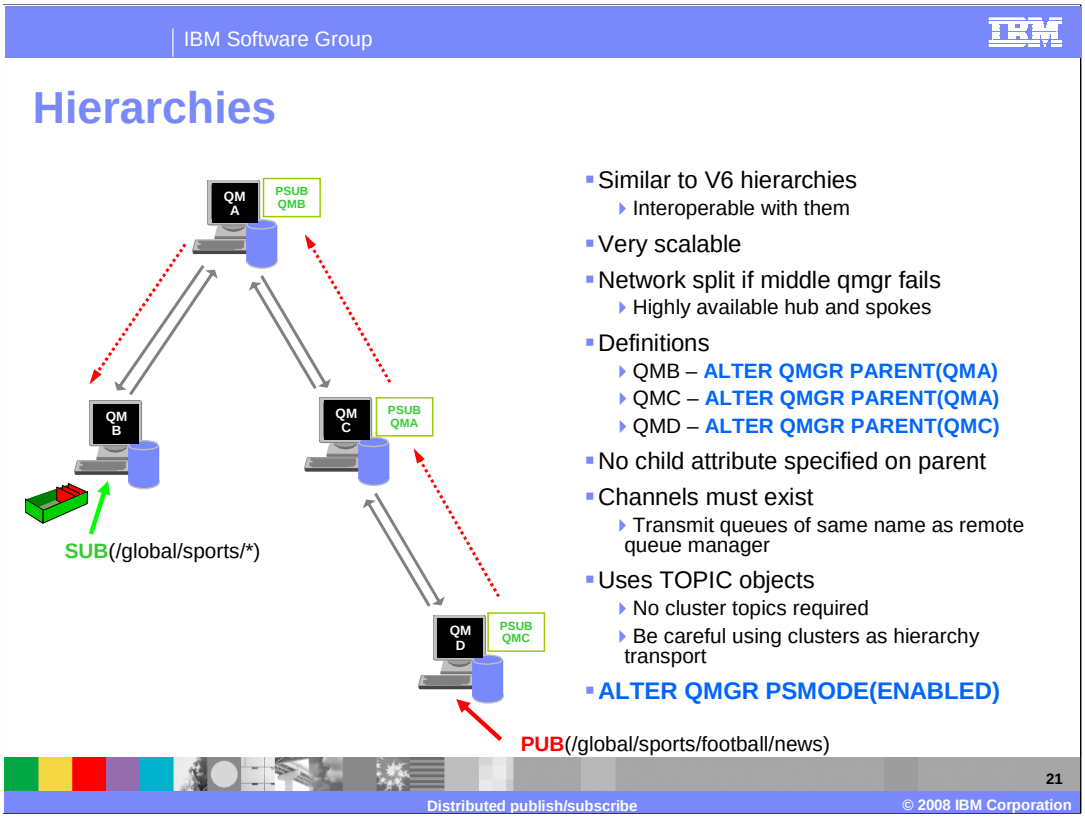


Although there is no fixed limit on the number of queue managers in a publish/subscribe cluster, one hundred queue managers is a reasonable limit and probably much smaller.

Section

Publish/subscribe hierarchies

This section provides details on publish/subscribe hierarchies.



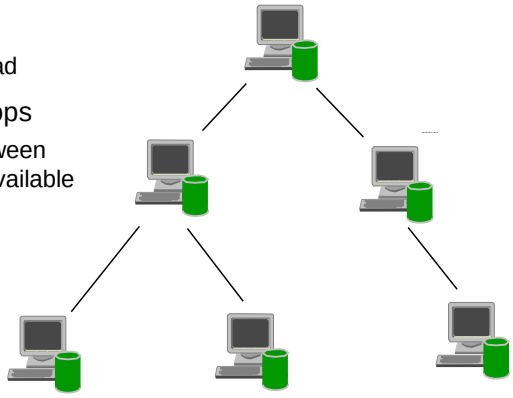
You can connect a local queue manager to a parent queue manager to create a hierarchy.

Before you can add a queue manager to a hierarchy, channels in both directions must exist between the prospective parent queue manager and child queue manager. Queue managers use explicit addressing when sending messages to queues that reside on another queue manager.

WebSphere MQ hierarchical connections require that the queue manager attribute PSMODE is set to ENABLED.

How large can a hierarchy be?

- More scaleable than a single publish/subscribe cluster
  - Less direct connectivity
  - Less proxy-subscription overhead
- But availability dependent on hops
  - Relies on all channels/QMs between publisher and subscriber being available



A hierarchy is very scalable and has less overhead of direct connectivity and proxy-subscriptions. But availability is dependent on all channels and queue managers between publisher and subscriber being available.

Section

Administration and migration

Distributed publish/subscribe administration is discussed in this section.

TOPIC object parameters

- TOPICSTR
  - Don't put the root node (/) into a cluster.
  - Split the tree into /global and /local
- PUB and SUB
  - Controls whether messages can be published or subscriptions made on the topic
    - ASPARENT
    - ENABLED
    - DISABLED
- PUBSCOPE and SUBSCOPE
  - Determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster.
    - ASPARENT
    - QMGR
    - ALL
  - You can restrict to QMGR programmatically using MQPMO_SCOPE_QMGR / MQSO_SCOPE_QMGR
- PROXYSUB
  - Controls whether a proxy subscription can be sent for this topic to directly connected queue managers, even if no local subscriptions exist.
    - FIRSTUSE
    - FORCE!

When defining a topic, do not put the root node, /, in the TOPICSTR parameter if this is a cluster. The PUB and SUB parameters control whether messages can be published or subscriptions made on the topic. PUBSCOPE determines whether this queue manager propagates publications to queue managers as part of a hierarchy or as part of a publish/subscribe cluster. SUBSCOPE determines whether this queue manager subscribes to publications in this queue manager or in the network of connected queue managers. PROXYSUB controls whether a proxy subscription can be sent for this topic to directly connected queue managers, even if no local subscriptions exist.

Distributed display information

- Displays status of the match space
  - **DISPLAY TPSTATUS('#')**
- Displays proxy-subscriptions
  - **DIS SUB(*) SUBTYPE(PROXY)**
- Displays publish/subscribe local and hierarchy status
  - **DIS PUBSUB TYPE(LOCAL / CHILD / PARENT / ALL)**
    - Displays QMNAME + STATUS
      - LOCAL -> ACTIVE / COMPAT / ERROR / INACTIVE / STARTING / STOPPING
      - CHILD -> ACTIVE / ERROR / REFUSED / STARTING / STOPPING
      - PARENT -> ACTIVE / ERROR / STARTING / STOPPING
      - ALL
- Displays cluster status
  - **DIS CLUSQMGR(*)**

The DISPLAY TPSTATUS command displays the status of one or more topic nodes in a topic tree, as specified by the parameters supplied. DISPLAY SUB with SUBTYPE(PROXY) provides information on internally created subscriptions used for routing publications through a queue manager.

DISPLAY PUBSUB provides the publish/subscribe status for this queue manager and for parent and child hierarchical connections.

Use the DISPLAY CLUSQMGR command to display cluster information about queue managers in a cluster. If you issue this command from a queue manager with a full repository, the information returned pertains to every queue manager in the cluster. If you issue this command from a queue manager that does not have a full repository, the information returned pertains only to the queue managers in which it has an interest.

Administrative commands

- Remove a queue manager from a cluster
  - **RESET CLUSTER** - topics always reset
- Monitor health of channels and health of transmit and subscriber queues
  - **DIS CHSTATUS(*)**
  - **DIS QL(*) CURDEPTH**

Use the MQSC command RESET CLUSTER to force a queue manager to leave a cluster. Use the channel status display to monitor the health of the channels. Check the current depth of the transmit and subscriber queues to ensure a healthy distributed environment.

Administrative considerations

- Security
  - Security by name rather than object
  - setmqaut -m QM2 -n MYTOPIC1 -t topic -p User +pub
  - setmqaut -m QM3 -n MYTOPIC1 -t topic -p User +sub
    - On each hierarchy queue manager with queue manager name
- Proxy-subscription timing
  - Proxy-subscriptions must flow before a publication can be received
  - Can use PROXYSUB(FORCE) on the topic object

Set security by name rather than object for topic publication and subscription. Set for each hierarchy queue manager by queue manager name as shown here.

Spreading the publish/subscribe work over several queue managers does not change the programming model. However the timing can change because proxy-subscriptions must flow before a publication can be received.

Publish everywhere is not supported in publish/subscribe clusters or hierarchies, but a similar technique is available by using the PROXYSUB attribute for a high-level topic object.

Loops in distributed publish/subscribe

- Loops in the publish/subscribe network are bad news
  - Subscribers receive multiple copies of the same publication
  - Detrimental to other workloads
  - Use system resources: processor, network, WMQ processes
- Messages are finger printed
  - Outbound of publish/subscribe cluster with cluster name
  - On each hierarchy queue manager with queue manager name

Loops in distributed publish/subscribe can cause publishers to receive multiple copies of the same subscription. They are also detrimental to other workloads and use system resources. A loop could be caused by an invalid configuration of a hierarchy and a pub/sub cluster or a parent queue manager not found in a hierarchy.

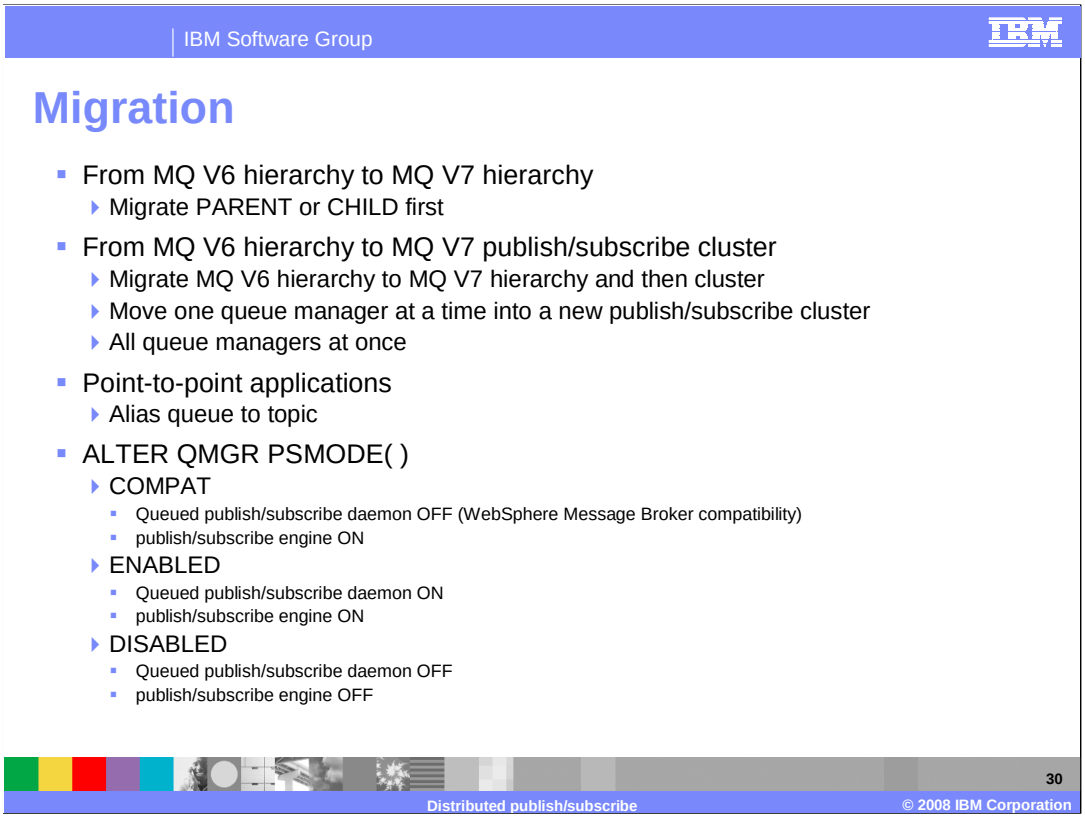
Stopping a queue manager

- Timing is important from a disconnect perspective
- Aim is to avoid transmit queue build up
  - SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - Hierarchy transmit queues
- Disconnect subscribers before stopping queue manager or channel initiator
  - Use DELETE SUB command
- Results when subscribers are not disconnected first
  - Durable subscribers
    - Transmit queue build up required
  - Non-durable subscribers
    - Transmit queue build up occurs; not desired
- Resynchronize when restarting the queue manager
  - Use REFRESH QMGR TYPE(PROXYSUB)

Timing is important when stopping a queue manger in a distributed environment. A disconnect can cause build up in the transmit queues of associated queue managers. You should disconnect subscribers before stopping a queue manager or stopping a channel initiator on z/OS. When you stop a queue manager local subscribers are disconnected, but other queue managers in the publish/subscribe cluster or hierarchy are not notified. When you stop a z/OS channel initiator local subscribers remain connected.

Subscriptions are configured to be durable or non-durable. Subscription durability determines what happens to subscriptions when subscribing applications disconnect for a queue manager. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. This is the desired result. Non-durable subscriptions exist only as long as the subscribing application's connection to the queue manager remains open.

Using REFRESH of the queue manger with type PROXYSUB resynchronizes the proxy subscriptions that are held with, and on behalf of, queue managers that are connected in a hierarchy or publish/subscribe cluster.



A migration tool is provided to migrate V6 subscriptions and configuration on distributed platforms. The Information Center provides detailed instructions on how to migrate from V6 hierarchies to V7 hierarchies and from V6 hierarchy to V7 cluster. Additionally, WebSphere MQ Version 7.0 introduces an extension to the alias queue object that allows an alias queue to be mapped to a topic object.

The PSMODE parameter on the queue manager controls whether the publish/subscribe engine and the queued publish/subscribe interface are running. It therefore controls whether applications can publish or subscribe by using the application programming interface and the queues that are monitored by the queued publish/subscribe interface.

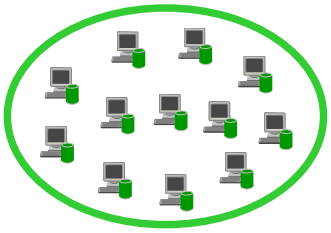
Section

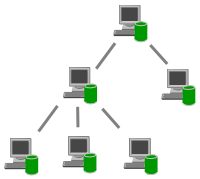
Comparison and recommendations

This section compares the various approaches and considerations for selecting a publish/subscribe architecture for your installation.

Comparison

- Publish/subscribe clusters
  - Available
    - Direct connections
  - Flexible, low admin
    - Cluster auto-definition





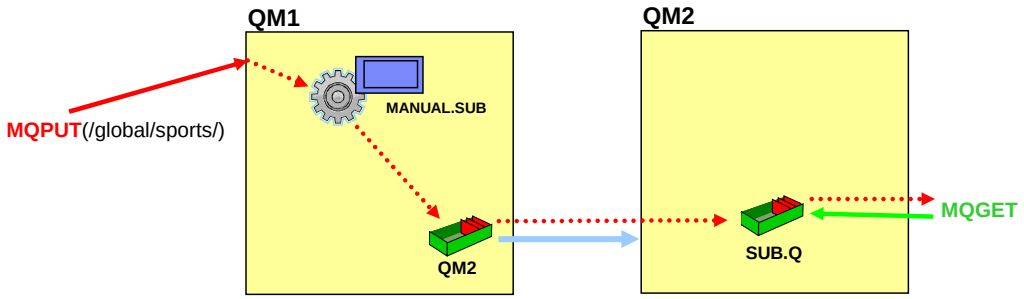
- Manual subscriber
  - Quick and dirty
  - Very inflexible, high admin

- Hierarchies
  - Very scalable
    - Proxy-subscriptions
    - Low fan-out cost
  - Lack of availability in multi-hop
  - Inflexible

Publish/subscribe clusters are highly available and flexible. However, the overhead of many connections can cause problems. Hierarchies are very scalable, but have lower availability. A manual subscriber is easy to set up, but is very inflexible and changes incur high administration costs.

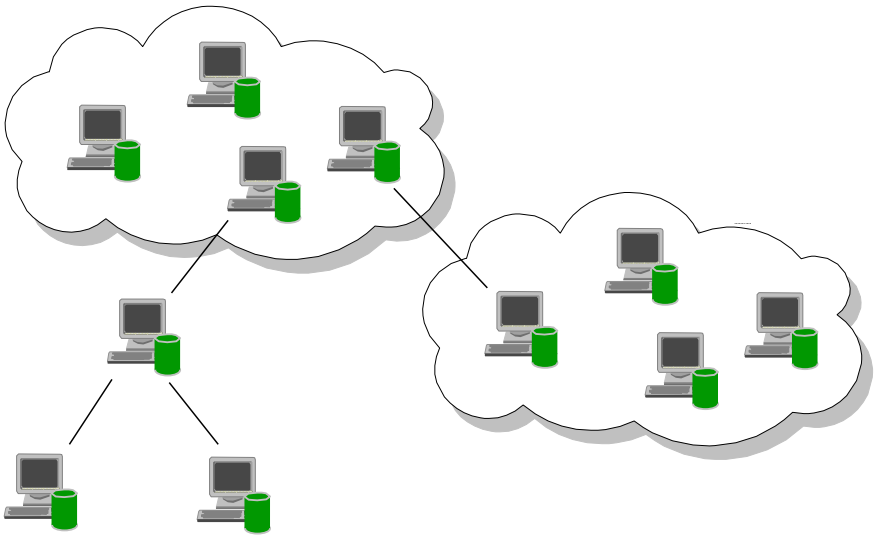
Manual subscriber

- Forward publications to a remote queue
  - QM1
    - DEF SUB(MANUAL.SUB) TOPICSTR('/global/sports/*') DESTQMGR(QM2) DEST(SUB.Q)
  - QM2
    - Application gets message from SUB.Q
- Inflexible



The manual subscriber method becomes inflexible if adding subscriptions.

Mixture of topologies



A mixture of clustering and hierarchies in V7 publish/subscriber is possible.

Recommendations

- Start small
- Do not put the root node (/) into a cluster
  - Make global topics obvious (for example /global or /cluster)
- Be careful architecting of deep hierarchies
  - A very shallow hierarchy with very highly available parent works well
- Be careful mixing traditional clusters
  - Large clusters and cluster topic objects leads to large number of channels
  - Admin controls on topic objects
- Monitor depth of transmit queues
  - SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - Hierarchy transmit queues
  - Do you need durable subscribers?
  - Did you stop subscribers before stopping queue manager

You should start small in your distributed design. Putting the root node into a cluster can lead to confusion; make global topics obvious. A shallow hierarchy with a highly available parent works well. Large clusters and cluster topic objects exacts him administrative costs and leads to a large number of channels. Monitor the depth of the transmit queues; don't let them get full.

Section

Summary

This section provides a summary of distributed publish/subscribe.

Summary

- Publish/subscribe in WebSphere MQ V7
- Distributed publish/subscribe approaches
  - publish/subscribe Clusters
  - Hierarchies
- Administration and migration
- Comparison and recommendations

This module provided an overview of publish/subscribe in V7. The various approaches to distributed publish/subscribe were discussed and administration and migration considerations provided. A comparison of the approaches was provided.

References

- Information center
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- Manuals
  - Publish/Subscribe User's Guide
  - Queue Manager Clusters
  - Script (MQSC) Command Reference

Additional information is available from the WebSphere MQ Web site.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_PubSub_Distributed.ppt

This module is also available in PDF format at: ../PubSub_Distributed.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere z/OS

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.