

An introduction to MQ publish/subscribe

Carl Farkas
IBM Europe zWebSphere consultant
farkas@fr.ibm.com



The team

- Carl Farkas
farkas@fr.ibm.com

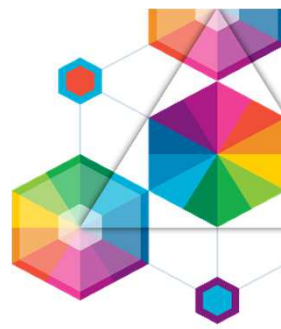


- Luc-Michel Demey
imd@demey-consulting.fr



(material based largely upon materials from David Ware, Morag Hughson and others)

Objectives of this MQ z/OS Proof Of Technology

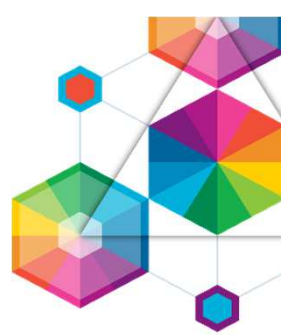


- This POT introduces the IBM MQ Pub/Sub feature. The objectives are to...
 - Familiarise you with the basic functions
 - Allow you to actually configure and test pub/sub, either on a Windows or a z/OS queue manager
 - *Warning: a PoT does not replace MQ education.* It is *not* expected that you'll have enough knowledge at the end of this PoT, for example, to fully master these new features. IBM provides via its partners formal education classes, and IBM also offers service engagements which can provide more in-depth information on MQ and its features.
- ▶ Prerequisites
- Be able to read some English
 - These exercises do assume that you are relatively familiar with *basic MQ concepts and administration*.
 - Above and beyond MQ, it is assumed that you have some experience with MQ in the particular operating system environment where your lab queue manager is running for the exercises, either Windows or z/OS. For z/OS, for example, it is assumed that you have a bit of experience executing SDSF commands, etc.

“I hear and I forget. I see and I remember. I do and I understand.” – Confucius



WebSphere Technical University



- **THE major IBM conference in Europe for WebSphere technology in 2015**
- **Hundreds of sessions on our favorite WebSphere topics: WAS, MQ, IIB (Broker), etc.**
- **Dublin, 13-16 October 2015**
- <http://www-304.ibm.com/services/learning/ites.wss/zz/en?pageType=page&c=G944977I79041O41>
- **Meet the developers and meet your peers throughout Europe**
- **A terrific investment in *your* skills**

- **Agenda provisoire dispo sur le web**
 - **52 sessions WAS + MQ**
 - **27 sessions « Cloud »**
 - **32 sessions Broker, Datapower, APIM...**
 - **10 sessions « Mobile »**
 - **24 sessions BPM, ODM, etc.**
 - **22 sessions « z » (CICS, WAS z/OS, MQ z/OS, etc.)**



IBM WebSphere Technical University 2015 13-16 October Dublin, Ireland

The IBM WebSphere Technical University will deliver in-depth technical content targeted at architects, developers, integrators and administrators by offering lectures and hands-on labs to expand your knowledge and expertise on the latest technologies, trends, tips and techniques.

Explore innovative capabilities and take a deep dive into growth-fueling technologies including WebSphere Application Server, WebSphere connectivity, such as WebSphere MQ and IBM Integration Bus, CICS, WebSphere on System z, DataPower and API Management, cloud computing, Mobile, BPM and Decision Management, collaboration, and much more.

This conference delivers over 120 in-depth technical sessions and hands-on labs across the following conference tracks:

- **Mobile Enterprise**
- **Smarter Process** Covering IBM Operational Decision Manager, IBM BPM, IBM Business Monitor and more.
- **Cloud** Covering IBM Pure Application Systems - Pure App as well as IBM Bluemix and SoftLayer, the highest performing cloud infrastructure available.
- **Application Infrastructure** Covering WebSphere Application Server, WebSphere MQ, MessageSight, Managed File Transfer, and MQ Advanced Security.
- **Integration** Covering IBM Integration Bus (formerly WebSphere Message Broker), DataPower, API Management, which runs on DataPower.
- **System z** Covering the newest CICS enhancements, the program



Top 5 reasons to attend:

Upgrade your IT skills: Training is the key to success. Get up to speed on the latest technologies, trends, tips and techniques - direct from IBM.

Select from over 120 in-depth technical sessions and labs. Expand your skills in the area of your choice with a variety of elective sessions for every level, from new user to expert.

Gain varied practical experience and get involved. Test-drive the latest technologies through hands-on labs. See leading-edge demos at the Solution Center and discuss the latest solutions. Gain exclusive insights into best practices projects and real-life applications by customer speakers.

Take advantage of the numerous networking opportunities. Exchange ideas and interact with peers, leading-edge

Bois-Colombes Tec

IBM

Les salles de conférence de l'IBM Client Center Paris

IBM **Client Center**

Rue des Minimes

Avenue de l'Europe

Cluny Sorbonne

Smart Data Center

Odéon Châtelet

Convention Accueil Picpus

Garnier Opéra Longchamp

Rivoli Louvre Tulleries

Concorde Etoile Vendôme

Accueil

Les Salles de A à Z

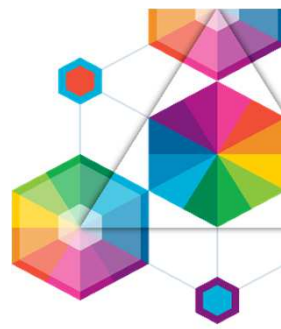
■ Auteuil	■ Garnier	■ Rivoli
■ Châtelet	■ Longchamp	■ Smart Data Center
■ Concorde	■ Odéon	■ Sorbonne
■ Convention	■ Opéra	■ Tulleries
■ Cluny	■ Picpus	■ Vendôme
■ Etoile		■ Espace de restauration

Besoin d'aide ?
Rendez-vous au point d'information **i**
ou appelez-nous au 01 58 75 01 90

iccparis@fr.ibm.com www.ibm.com/fr/clientcenter/

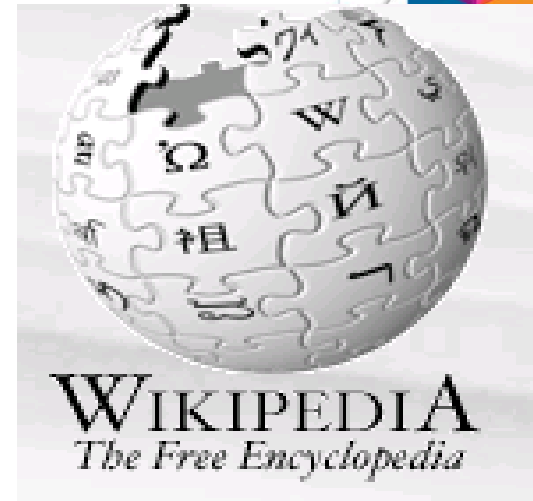
Agenda

- What is publish/subscribe?
- Publish/Subscribe in IBM MQ
- Topics
- Use cases
- Managing topics
- Subscriptions
- Publishing
- Application development
- Topologies



What is publish/subscribe?

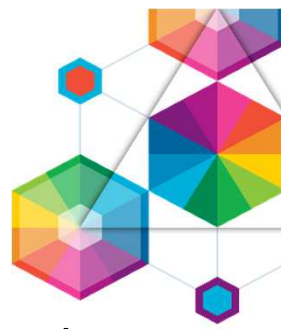




Publish/Subscribe

- **Publish/Subscribe** (or pub/sub) is an asynchronous messaging paradigm where senders (publishers) of messages are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into classes, without knowledge of what (if any) subscribers there may be.
- Subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what (if any) publishers there are. This decoupling of publishers and subscribers can allow for greater scalability and a more dynamic network topology.
- Pub/sub is a sibling of the Message Queue paradigm, and is typically one part of a larger Message-Oriented Middleware solution. Most messaging systems support in their API (e.g. JMS) both the pub/sub and Message Queue models.

“Topics”



Terminology in the Pub/Sub world

- **Topic**

This can mean *topic strings*, *topic objects* and even *topic nodes*, not to mention *JMS topics*!

- **Publisher (or sometimes called “producer” or “sender”)**

A publisher is an application that puts messages to a topic as opposed to a queue.

- A publisher can either open a topic object or a topic string.

- **Publication**

A publication is simply a message that is put to a topic rather than a queue

- **Subscription**

A subscription is an artefact that exists on a queue manager that describes where copies of messages published for a particular topic string are delivered to

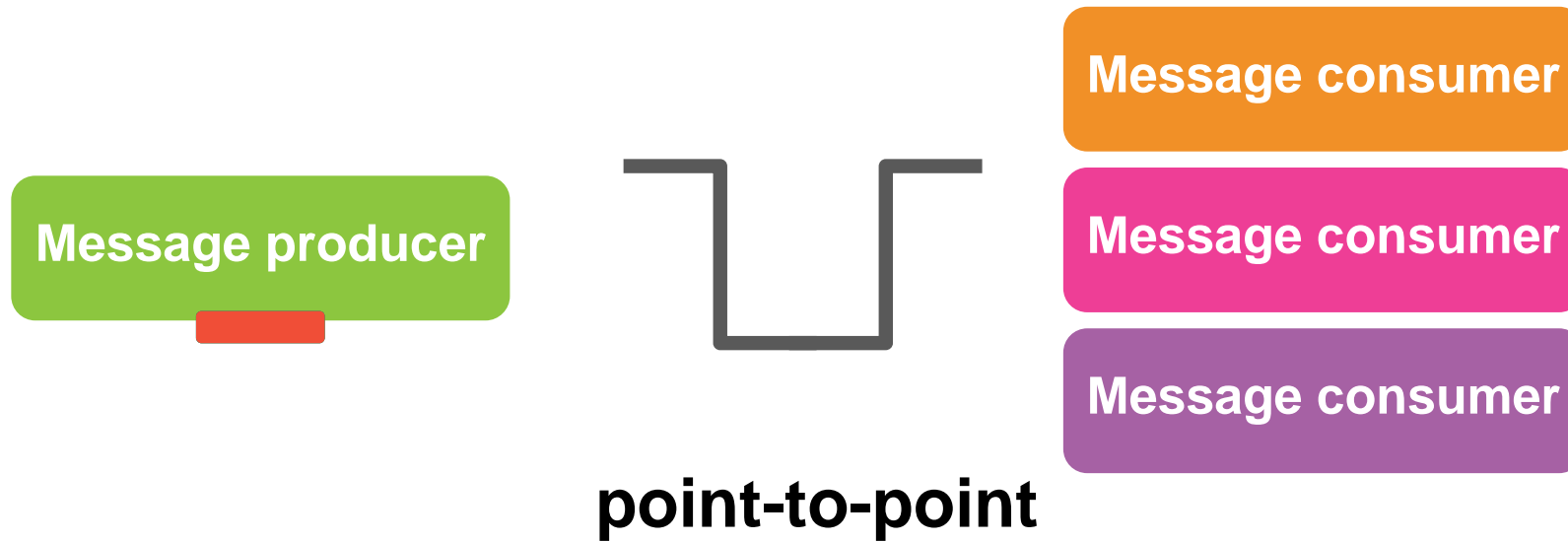
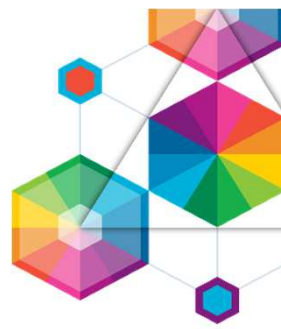
- A subscription identifies a queue for those messages

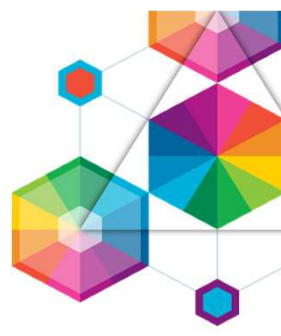
- **Subscriber (or sometimes called “consumer” or “receiver”)**

A subscriber is an application that consumes messages from a subscription.

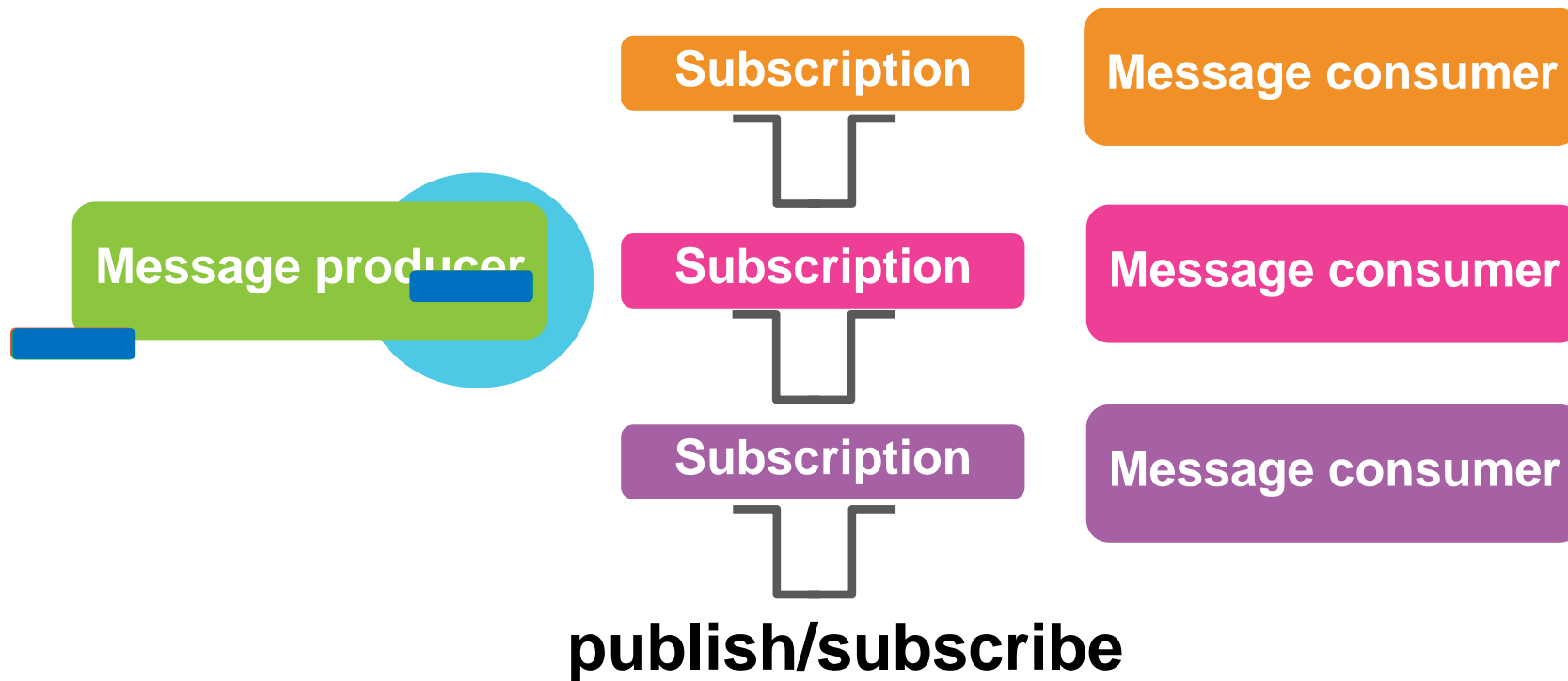
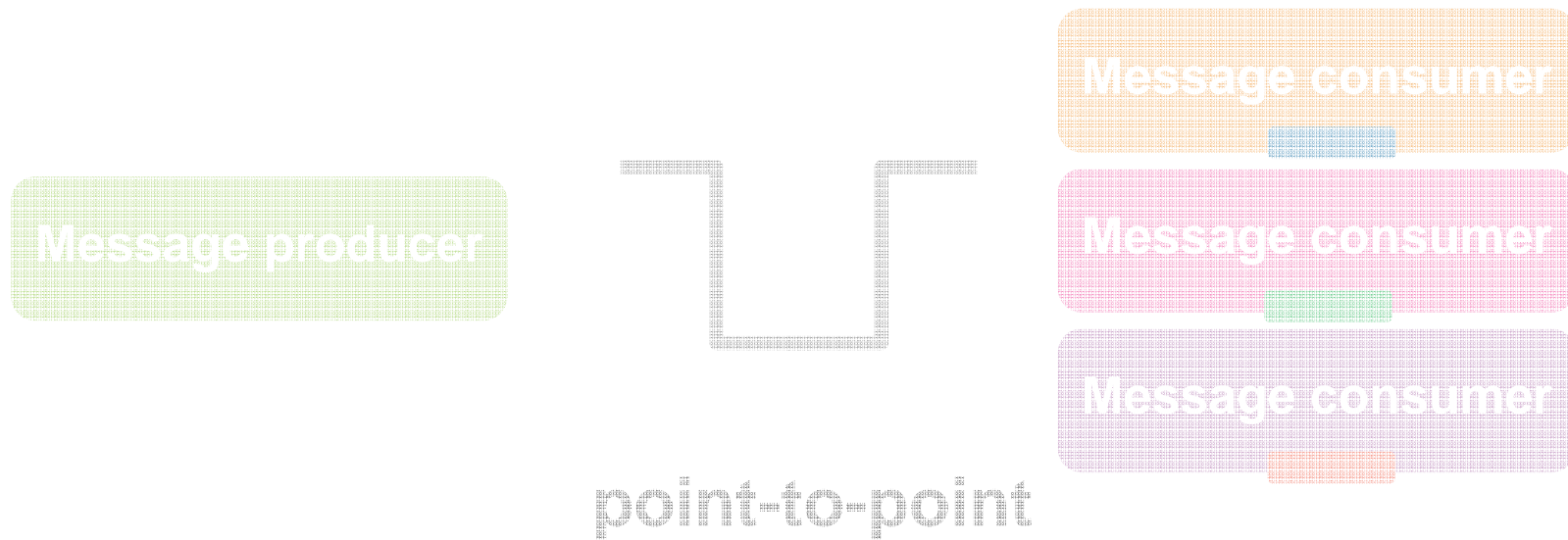
- Note also that sometimes the “subscriber” and “consumer” are different entities!

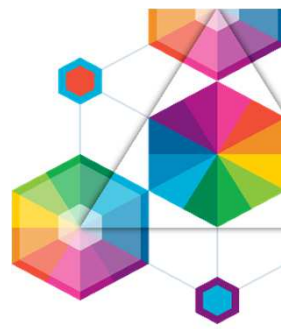
How does it compare to point-to-point?





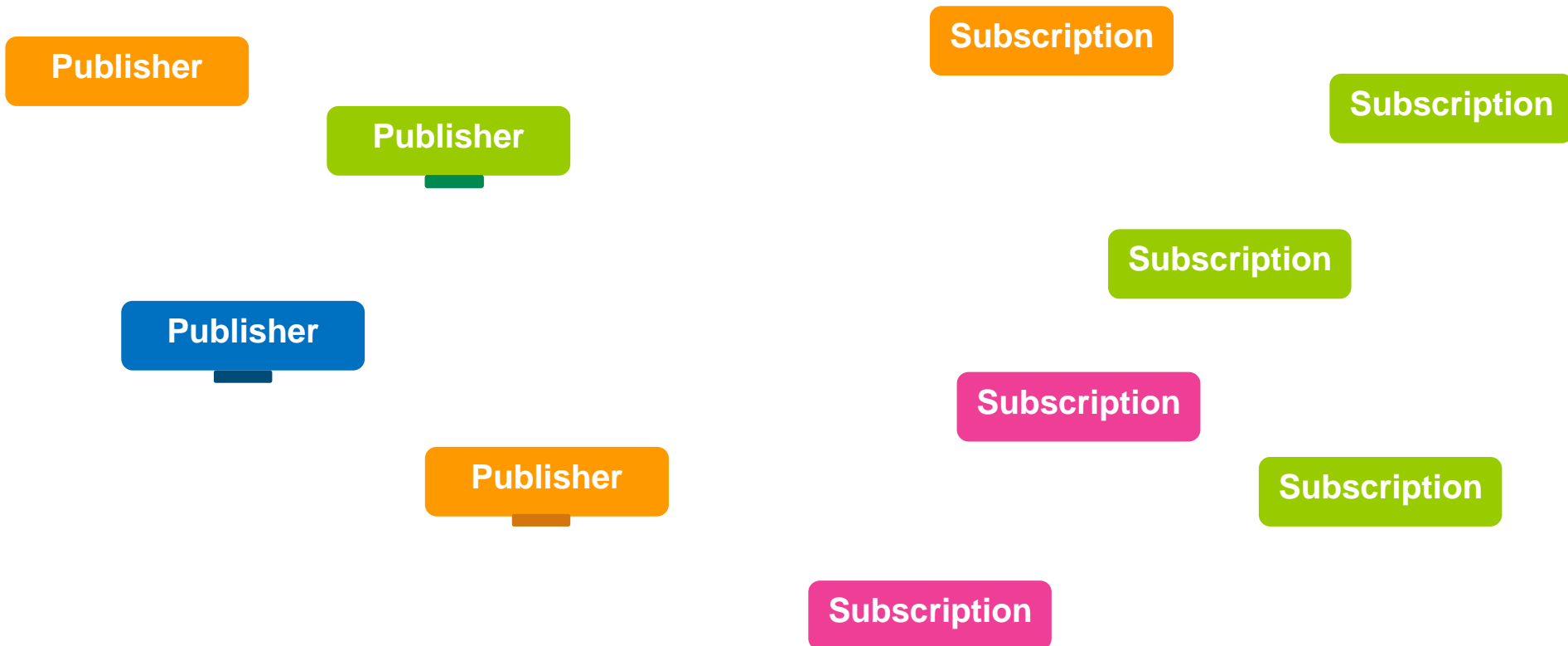
How does it compare to point-to-point?

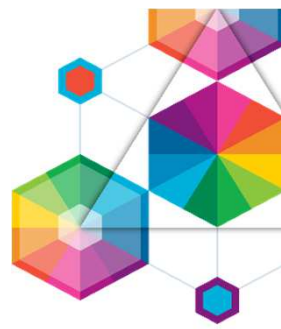




But which subscriptions receive the messages?

- Publishing and subscribing is based on ‘**topics**’
 - **Green** messages go to **green** subscribers
 - **Orange** messages go to **orange** subscribers
 - But nobody wants a **blue** message!

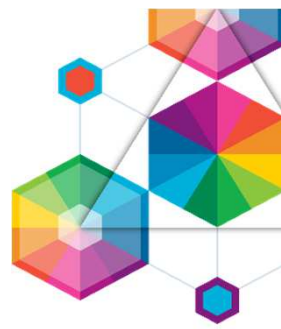




Point to Point Examples

- Post Card
 - Goes to just the person I send it to.
- eMail
 - Might go to lots of people but I get to choose exactly who gets it.
- Message Queuing
 - If I put a message it will go to one consumer.

With Point to Point, the sender (MQPUTer) explicitly defines the receiver (MQGETer) at MQOPEN time

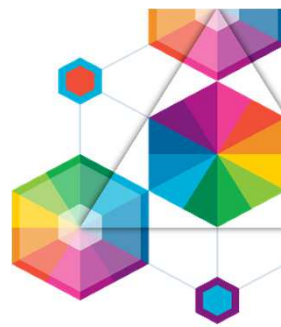


Publish / Subscribe Examples

- Magazine Publishing
 - In the US – almost 10,000 titles published in US (only 2,000 are considered ‘major publications’)
- Airline Departure Boards
 - Boards might display (subscribe to)
 - All departures
 - Departures at this Terminal
 - Departures by this Airline
- RSS News Feeds, Twitter, etc.
- Forums
- Facebook!

With Publish/Subscribe, the sender (MQPUTer) never explicitly defines the receiver; he never even knows if there are any receivers!

Moving Toward Loose Coupling

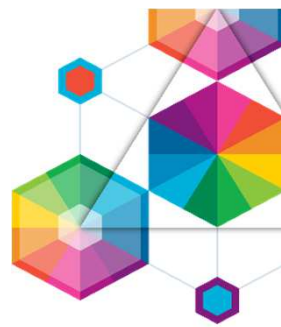


- Monolithic
 - Work is all on one computer!
- Client-Server
 - Work is distributed
 - Detailed knowledge of topology and connectivity needed
 - All components must be available for the application to work.
- Message Queuing
 - Work is distributed
 - Cooperating components can be available at different times
 - Work will queue at busy times or if a component is down
 - Connected by Queue Names.
- Publish/Subscribe
 - Cooperating components exchange data through an infrastructure that identifies the “subject” of the data.

Publish/Subscribe in IBM MQ



IBM MQ's publish/subscribe over the years



Publish/Subscribe brokers



IBM MQ Publish/Subscribe APIs

Command message based publish/subscribe API

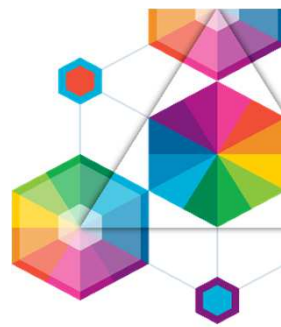
JMS publish/subscribe API

XMS publish/subscribe API

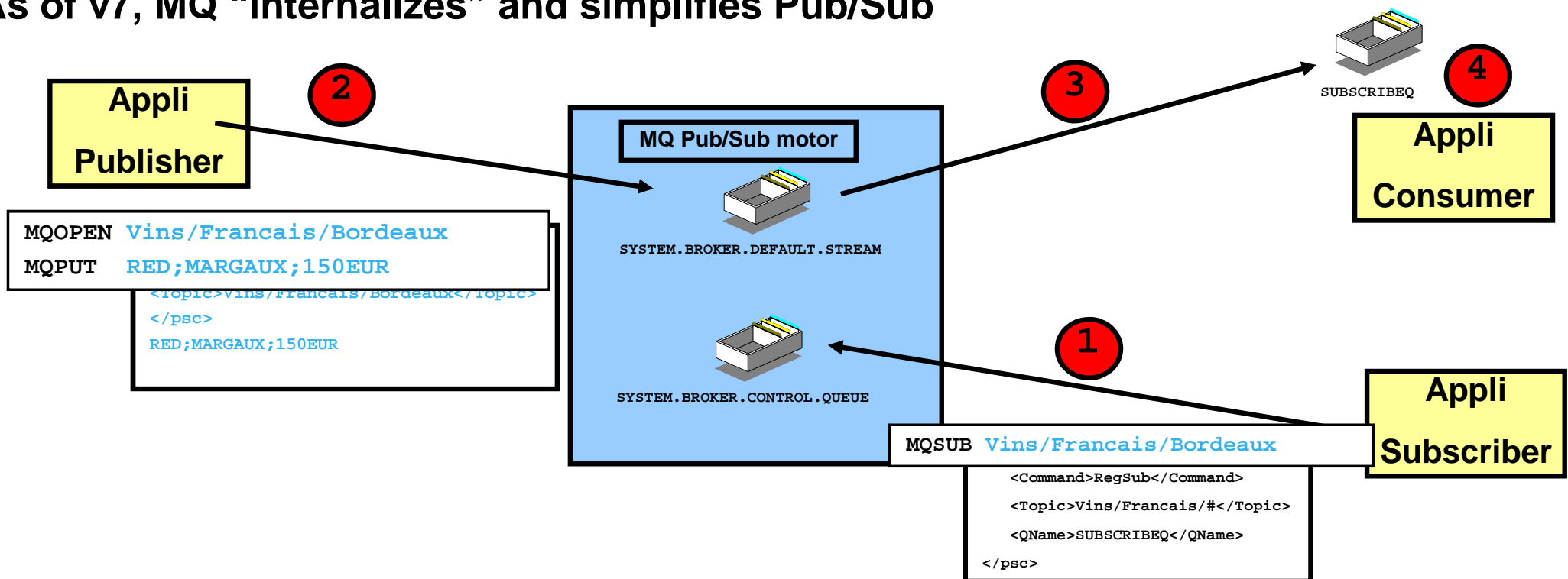
Native MQI publish/subscribe API

Note: IBM also proposes a publish/subscribe broker with MQTT and MessageSight. Conceptually this is the same as MQ publish/subscribe, although the API and many of the details differ. This is not covered in this lecture.

Publish/Subscribe – how does it work in MQ?



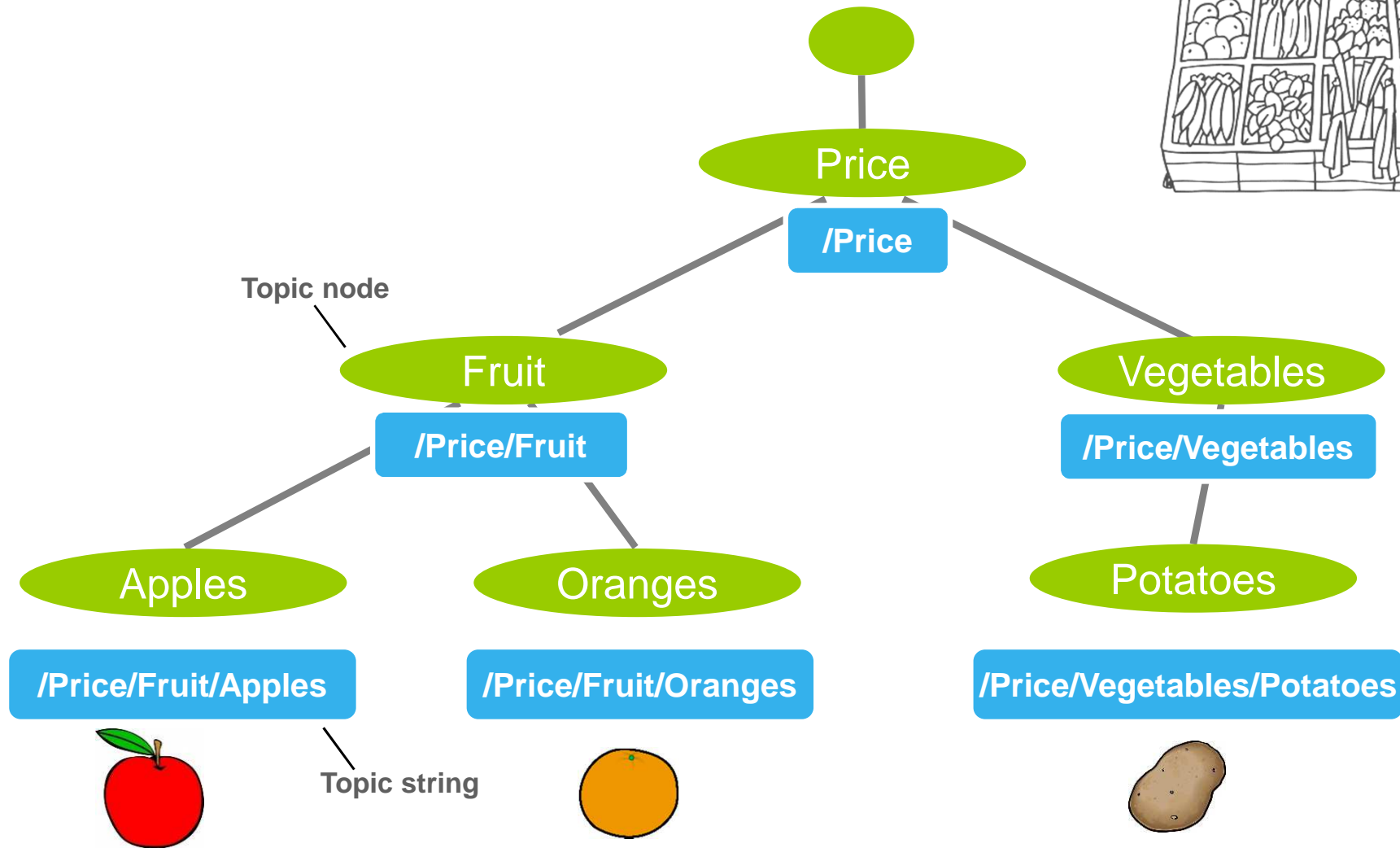
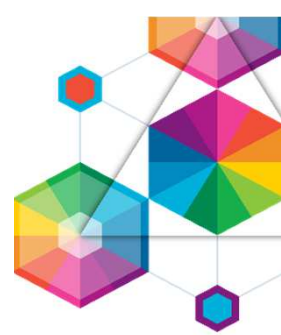
- **Pub/Sub is a function that allows:**
 - Publishing applications to make information (eg. Messages) available to a list of « interested » applications
 - Subscribing application can receive information
 - Fuller “decoupling” model of source (“Publisher”) and target (“Consumer”) applications
- **Typical uses of the Pub/Sub model:**
 - Document distribution, alert notification, newsgroups, and any application where the distribution list tends to be dynamic
- **As of v7, MQ “internalizes” and simplifies Pub/Sub**



Topics

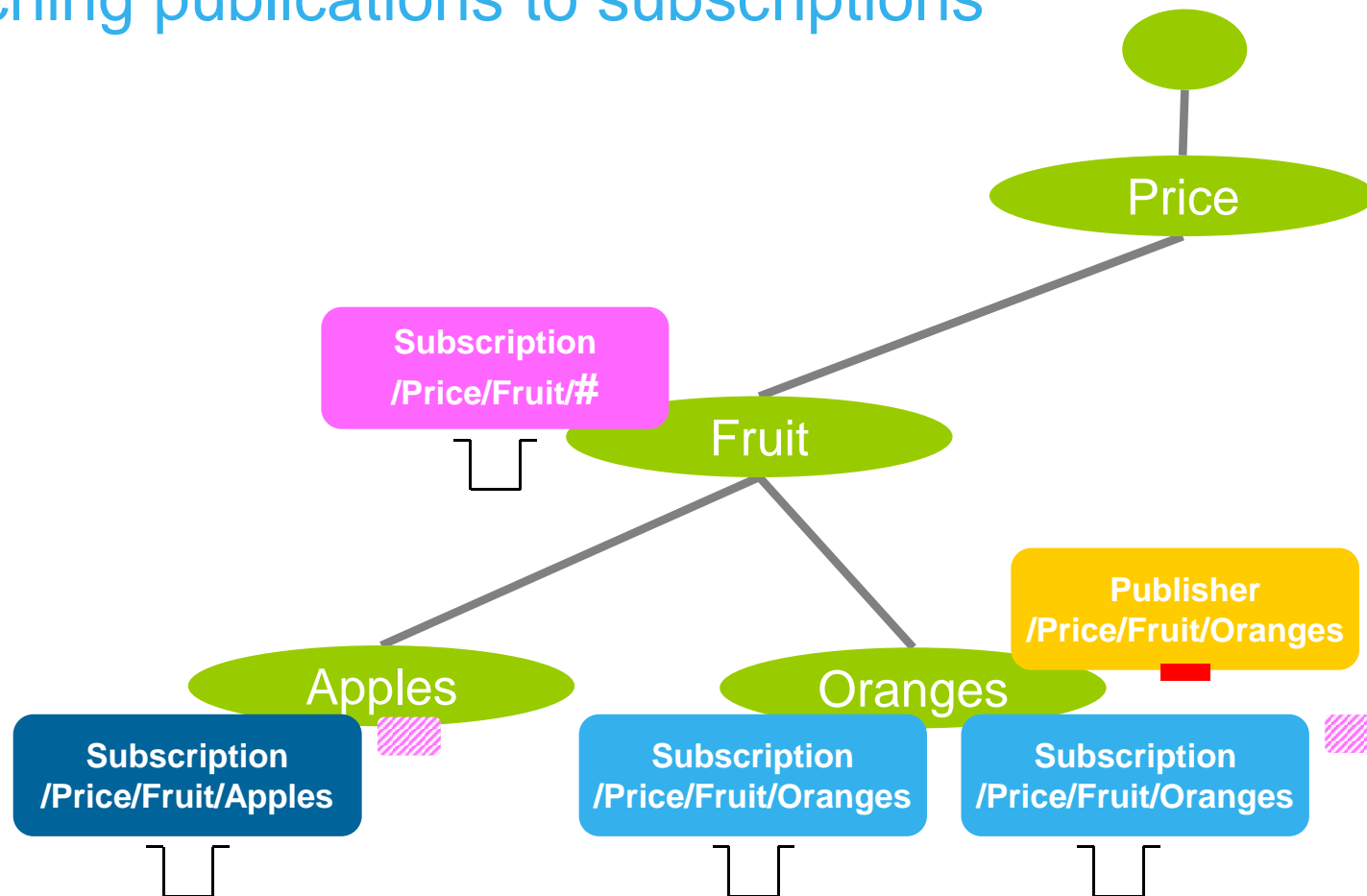
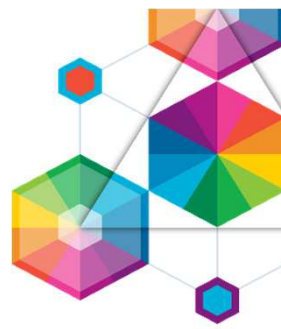


It's all about the *topic tree*



•The tree has “leaves” (topics nodes) added implicitly or explicitly, either administratively or by subscriptions

Matching publications to subscriptions

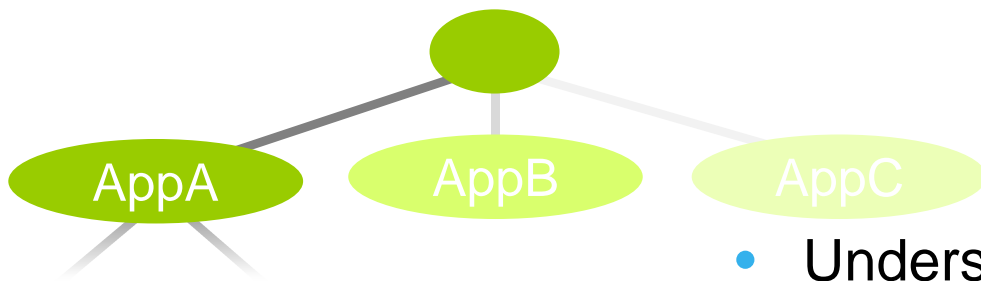


- Subscriptions are made up of (1) a topic and (2) a queue to receive the publications (plus other bits)
- Subscriptions are attached to matching nodes in the topic tree
- Publications identify the relevant topic node
- A copy of the publication is delivered to the queue identified by *each* matching subscription
- Wildcarding** subscriptions at the topic node level can receive messages from multiple topic strings

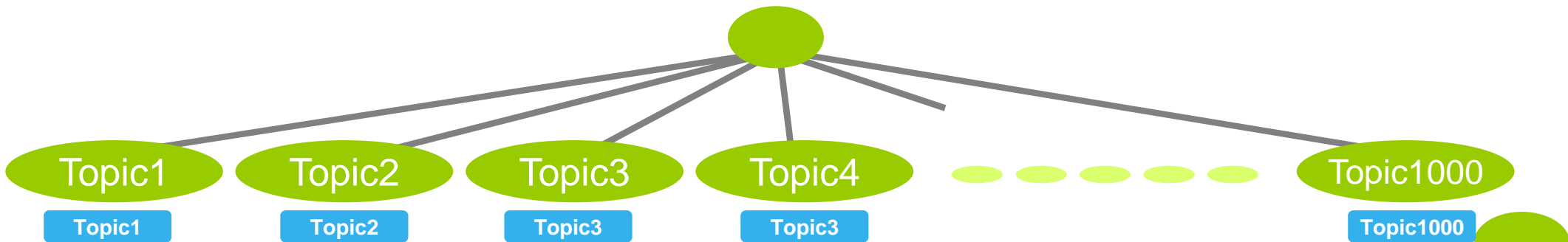
Designing your topic tree structure



- Make it extendable.



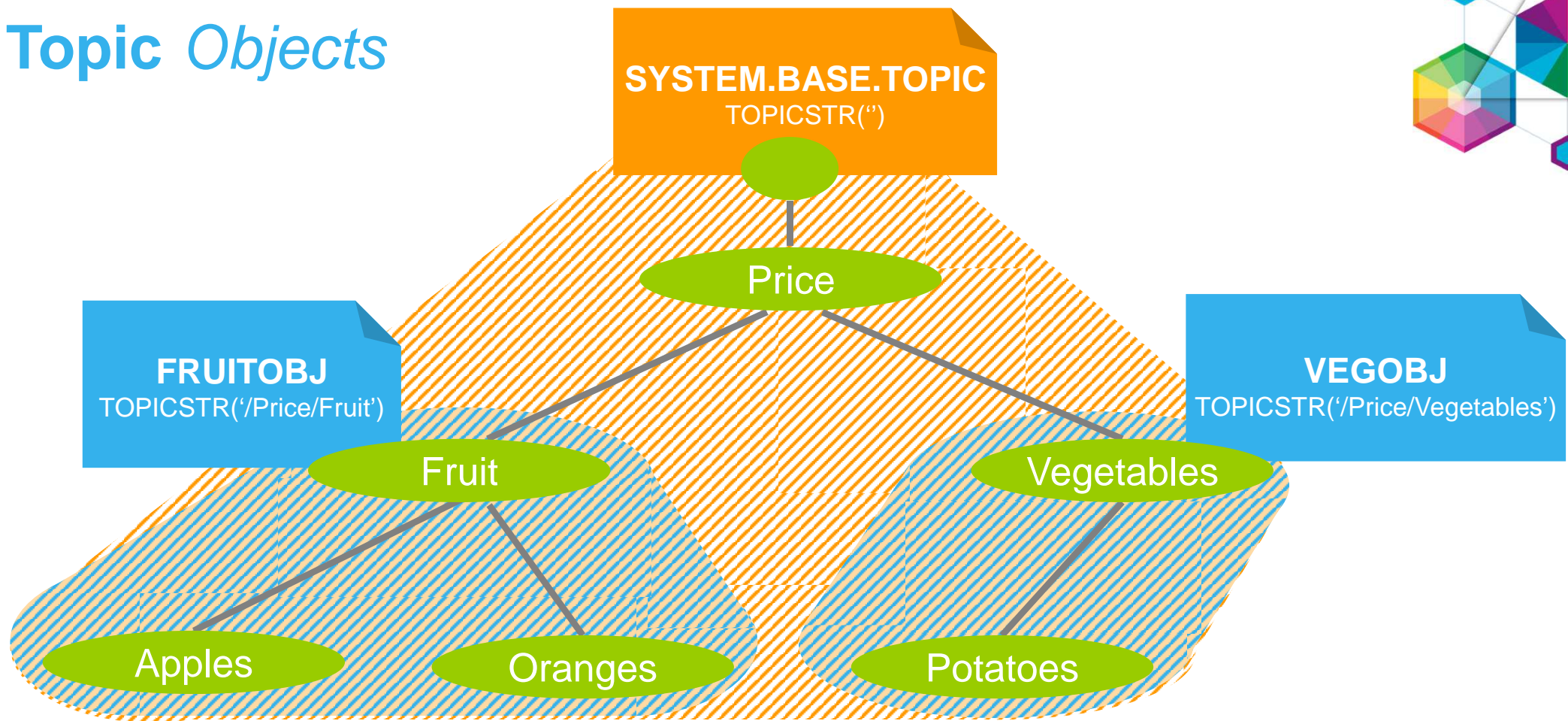
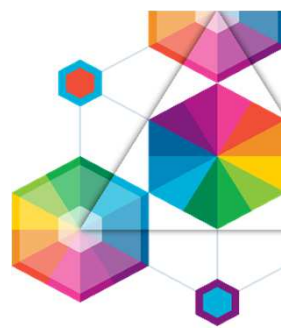
- Understand a rapidly changing set of topic strings.



- Avoid excessively **wide** or **deep** dynamic topic trees.
 - Use structure where appropriate.
 - Limit it to *subscribable* content.

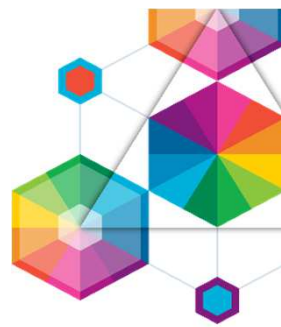


Topic Objects



- Topic objects are a point of administration associated with a node in the topic tree.
- You start with a base object defined for the ‘ ’ node ... the rest are *optional*.
- They provide hook points in the topic tree to configure specific pub/sub behaviour for a branch.
- A dynamically created topic node **inherits** its attributes from administered topic objects associated with topic nodes above it in the topic tree.

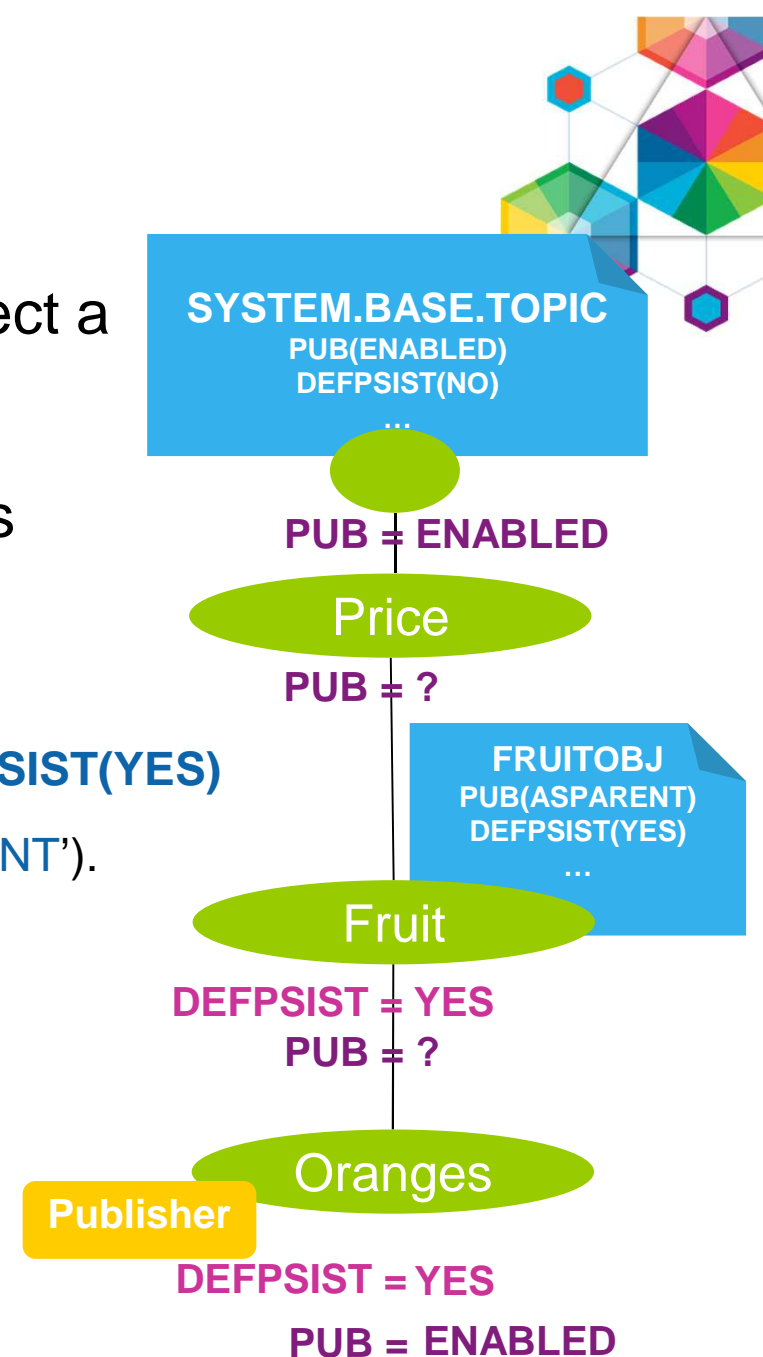
TOPIC administration syntax example



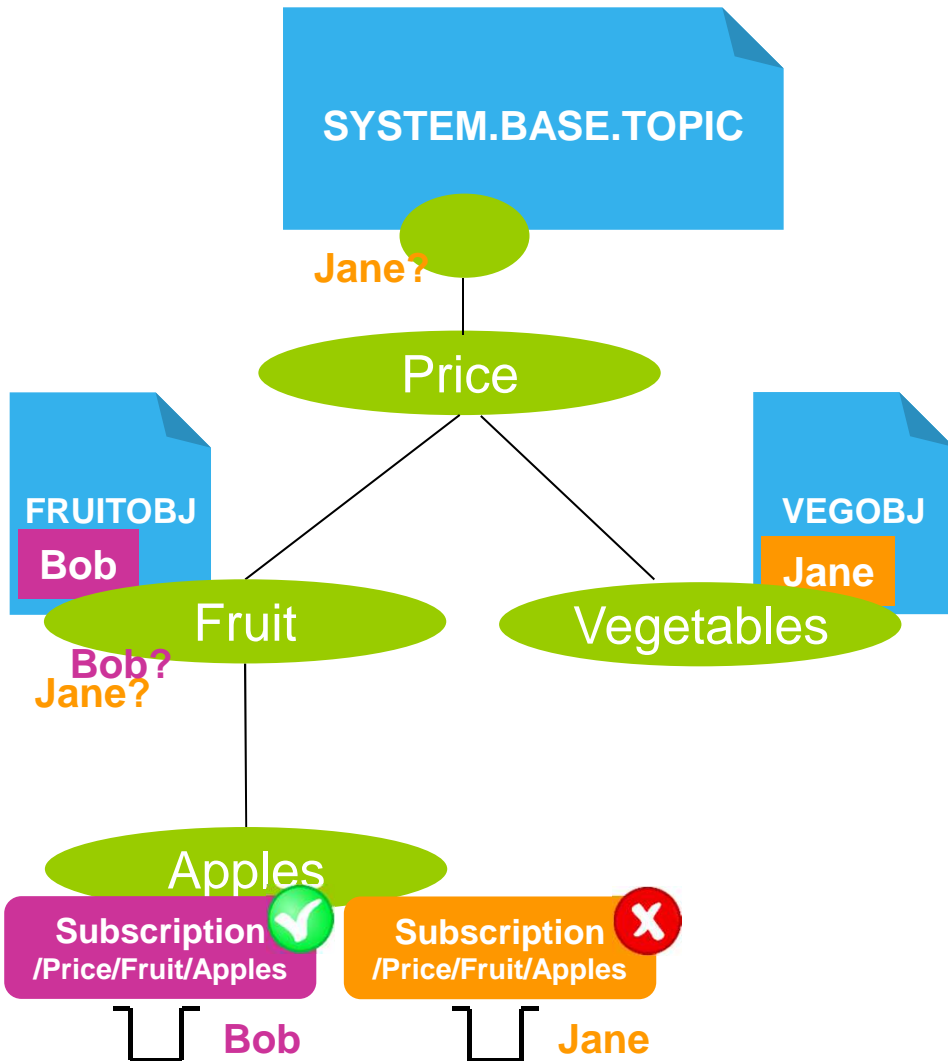
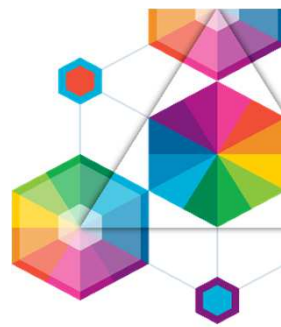
```
ALTER TOPIC( topic_name )
  [ CLUSTER( cluster_name ) ]          [ COMMINFO( comminfo_name ) ]
  [ CUSTOM( string ) ]
  [ DEFPRESP( ASPARENT | SYNC | ASYNC ) ]
  [ DEFPRTY( ASPARENT | integer ) ]
  [ DEFPSIST( ASPARENT | NO | YES ) ]  [ DESCR( string ) ]
  [ DURSUB( ASPARENT | YES | NO ) ]
  [ MCAST( ASPARENT | ENABLED | DISABLED | ONLY ) ]
  [ MDURMDL( q_name ) ]                [ MNDURMDL( q_name ) ]
  [ NPMSGDLV( ASPARENT | ALL | ALLDUR | ALLAVAIL ) ]
  [ PMSGDLV( ASPARENT | ALL | ALLDUR | ALLAVAIL ) ]
  [ PROXYSUB( FIRSTUSE | FORCE ) ]
  [ PUB( ASPARENT | ENABLED | DISABLED ) ]
  [ PUBSCOPE( ALL | ASPARENT | QMGR ) ]
  [ SUB( ASPARENT | ENABLED | DISABLED ) ]
  [ SUBSCOPE( ALL | ASPARENT | QMGR ) ]
  [ TYPE( LOCAL ) ]                   [ USEDQLQ( ASPARENT | NO | YES ) ]
  [ WILDCARD( BLOCK | PASSTHRU ) ]
```


Topic object attributes

- Many attributes can be set on topic objects to effect a publisher or subscriber's behaviour.
- Dynamic nodes inherit their behaviour from nodes above.
- Create a topic object for topic string **'/Price/Fruit'**
 - **DEFINE TOPIC(FRUITOBJ) TOPICSTR('/Price/Fruit') DEFPSIST(YES)**
 - Attributes default to *inherit settings from above* (e.g. 'ASPARENT').
 - (So by default, a new object does nothing)
- Publish a message to topic string **'/Price/Fruit/Oranges'**
 - **What message persistence to use?**
 - **Are publications enabled?**



Topic Security



- Access control is set as for queues, but for a defined **topic object**, not a topic string!
- Authority checks performed on the topic tree
 - Walk up the tree, just like attributes.
 - Keep checking until an authorisation is found or we run out of topic tree.

```

DEFINE TOPIC(FRUITOBJ)
TOPICSTR('/Price/Fruit')

setmqaut -m QMGR1 -t topic -n FRUITOBJ -
p BOB +sub
    
```

- Subscriber needs +sub authority on topic and +put on the associated queue
- Publisher needs +pub on the topic and +put on the associated queue
- Getter needs +get on the associated queue



Use cases (& break)



Use cases IBM products

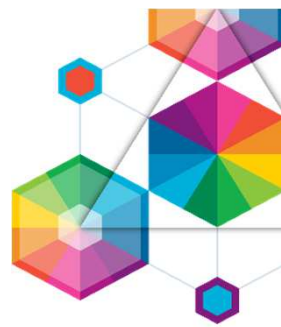
- IBM Managed File Transfer (MFT, ex-FTE)
 - Distribution of file transfer progress & log messages to “monitors”. The coordination queue manager publishes all progress and log messages. Any authorized user ids can request this information by subscribing.
 - Publications done, for example, to topic:
`SYSTEM.FTE/Log/agent_name/transfer_ID`
- IBM Integration Bus (IIB, ex-WebSphere Message Broker)
 - The Broker motor publishes statistics on the flow usage. Any authorized user ids can request statistics by subscribing.
 - Publications done, for example, to topic:
`$SYS/Broker/integrationNodeName/StatisticsAccounting/
recordType/integrationServerName/messageFlowName`

Managing topics



Managing topics

- Displaying topic object definitions
 - This shows how administered **topic objects** are configured



5724-H72 (C) Copyright IBM Corp. 1994, 2014.

DISPLAY TOPIC(FRUITOBJ)

2 : DISPLAY TOPIC(FRUITOBJ)

AMQ8633: Display topic details.

TOPIC(**FRUITOBJ**)

TOPICSTR(**/Price/Fruit**)

CLUSTER()

DURSUB(ASPARENT)

SUB(ASPARENT)

DEFPRTY(ASPARENT)

ALTDATE(2015-02-03)

PMSGDLV(ASPARENT)

PUBSCOPE(ASPARENT)

PROXYSUB(FIRSTUSE)

MDURMDL()

MCAST(ASPARENT)

USEDLQ(ASPARENT)

TYPE(LOCAL)

DESCR(Price of fruit)

CLROUTE(DIRECT)

PUB(**ASPARENT**)

DEFPSIST(**YES**)

DEFPRESP(ASPARENT)

ALTTIME(08.44.48)

NPMSGDLV(ASPARENT)

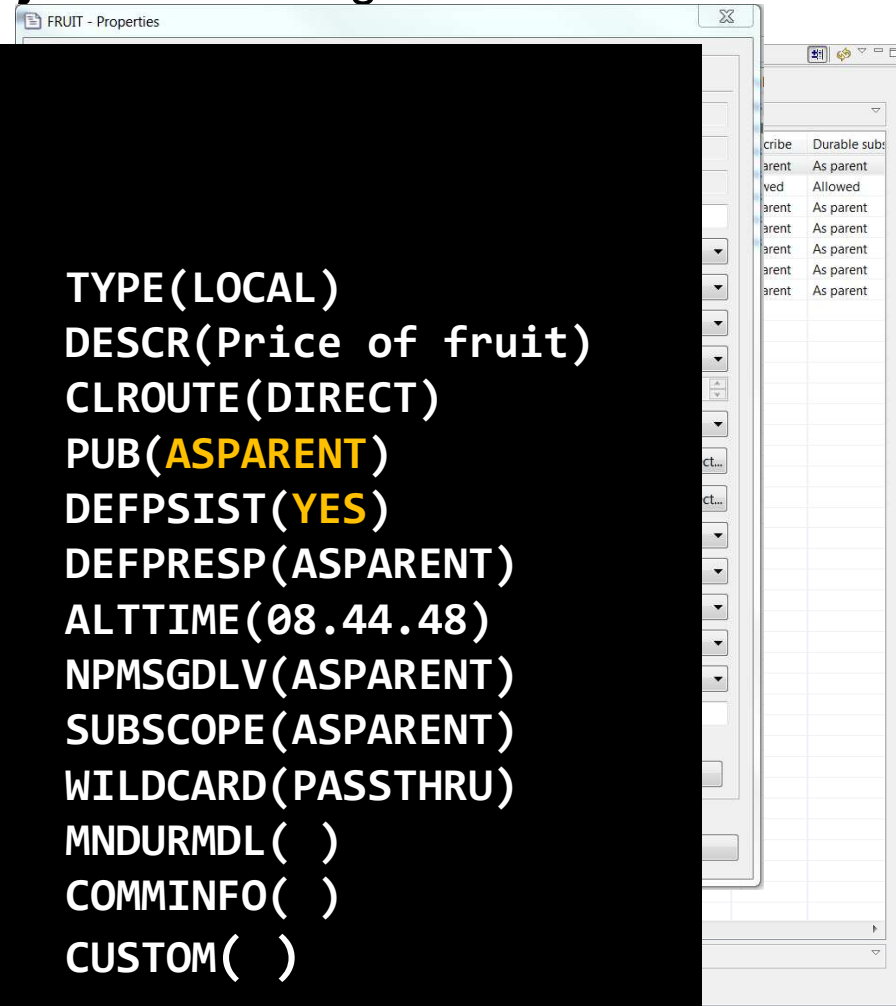
SUBSCOPE(ASPARENT)

WILDCARD(PASSTHRU)

MNDURMDL()

COMMINFO()

CUSTOM()



Managing topics

- Displaying the topic tree

```
DISPLAY TPSTATUS('/Price/Fruit/Apples')
```

```
23 : DISPLAY TPSTATUS('/Price/Fruit/Apples')
```

```
AMQ8754: Display topic status details.
```

```
TOPICSTR(/Price/Fruit/Apples) ADMIN( )
```

```
CLUSTER( )
```

```
COMMINFO(SYSTEM.DEFAULT.COMMINFO.MULTICAST)
```

```
MDURMDL(SYSTEM.DURABLE.MODEL.QUEUE)
```

```
MNDURMDL(SYSTEM.NDURABLE.MODEL.QUEUE)
```

```
CLROUTE(NONE)
```

```
DEFPSIST(YES)
```

```
DEFPRTY(0)
```

```
DEFPRESP(SYNC)
```

```
DURSUB(YES)
```

```
PUB(ENABLED)
```

```
SUB(ENABLED)
```

```
PMSGDLV(ALLDUR)
```

```
NPMSGDLV(ALLAVAIL)
```

```
RETAINED(NO)
```

```
MCAST(DISABLED)
```

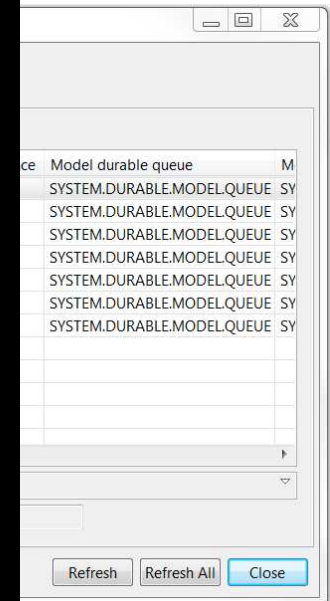
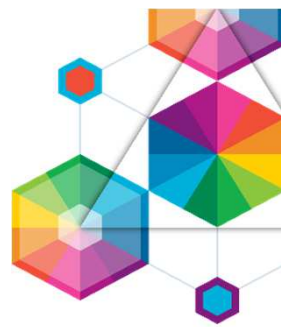
```
PUBCOUNT(0)
```

```
SUBCOUNT(1)
```

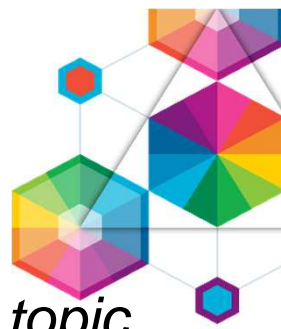
```
PUBSCOPE(ALL)
```

```
SUBSCOPE(ALL)
```

```
USEDLQ(YES)
```



Applications and *topics*

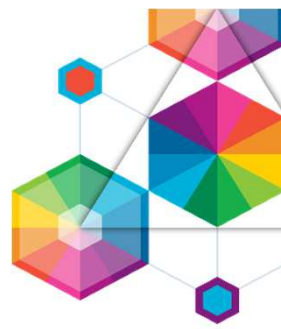


- *When creating subscriptions or opening topics to publish on, do I use a topic string or a topic object?*
 - A **topic string**, a **topic object** or **both**.
- *So which should I use?*
 - Using the topic string is probably the easiest, it's closest to what the application writer is expecting
 - `Sub(-, '/Price/Fruit/Apples')` → `/Price/Fruit/Apples`
 - Using a topic object maps the operation to the topic string of that topic object
 - `Sub(FRUITOBJ, ")` → `/Price/Fruit`
 - If you use both, you get both!
 - The topic string is appended to the topic string of the object
 - `Sub(FRUITOBJ, 'Apples')` → `/Price/Fruit/Apples`
- *If in doubt, check the topic tree for which nodes exist and are actually being used*

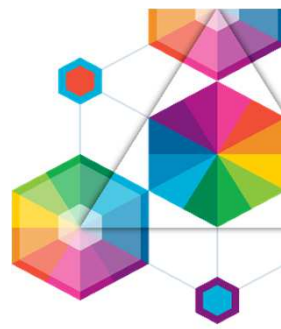
Subscriptions



Subscription types



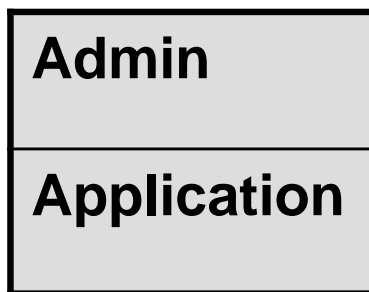
- There are many different *types* of subscriptions:
 - **Administered** or **application** created
 - **Durable** or **non-durable**
 - **Managed** or **unmanaged** subscription queues
- *These different aspects of a subscription can be combined, don't assume it's one or the other...*

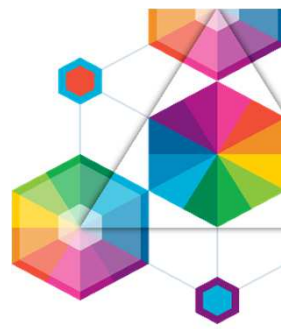


Subscription types

Subscription creation and deletion

- **Application created subscriptions**
 - Applications use an API to dynamically create and delete subscriptions
- **Administratively created subscriptions**
 - An administrator defines subscriptions that can be accessed by applications
 - Applications can either use the publish/subscribe APIs to access these subscriptions or access their associated queue using point-to-point APIs.







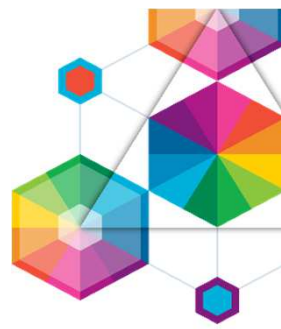


Subscription types

Subscription lifetime

- **Durable subscriptions**
 - The lifetime of the subscription is independent of any application
- **Non-durable subscriptions**
 - The lifetime of the subscription is bounded by the creating application
 - Subscriptions are automatically deleted when the application closes

	Durable	Non-durable
Admin		
Application		



Subscription types

Subscription queue management

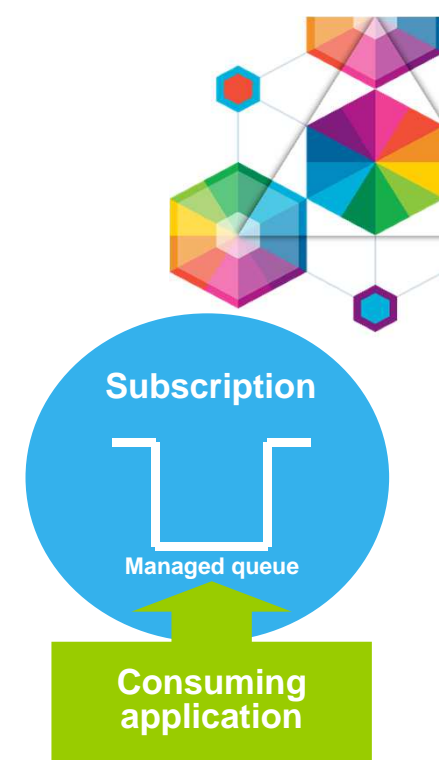
- **A subscription maps a topic to a queue. The queue relationship is either explicit or implicit...**
- **Managed subscription queue (implicit)**
 - The subscription automatically creates and deletes a queue for the use of queuing any matching publications.
- **Unmanaged subscription queue (explicit)**
 - When the subscription is created the name and location of an existing queue must be provided by you.

	Managed		Unmanaged	
	Durable	Non-durable	Durable	Non-durable
Admin				
Application			 (Not JMS)	 (Not JMS)

Accessing a subscription's messages

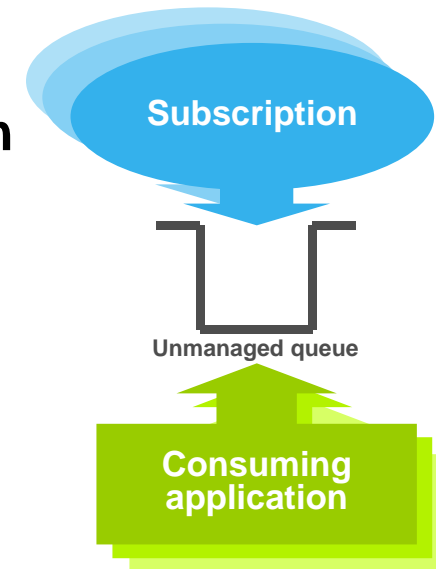
Via the *subscription*

- **An application opens the subscription**
 - *A 'true' pub/sub application*
- **Works with managed and unmanaged subscription queues**
- **Limited to one attached consuming application at a time**
 - Unless you're using JMS cloned/shared subscriptions
- **Generally better publish/subscribe status feedback**



Via the *queue*

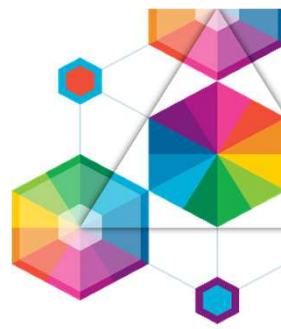
- **An application opens the queue associated with the subscription**
 - *This is really a point-to-point application*
- **Only works with unmanaged subscription queues**
- **Allows more freedom in what can be done**
 - For example, multiple concurrent consuming applications are possible or multiple subscriptions using a single queue.



Publishing

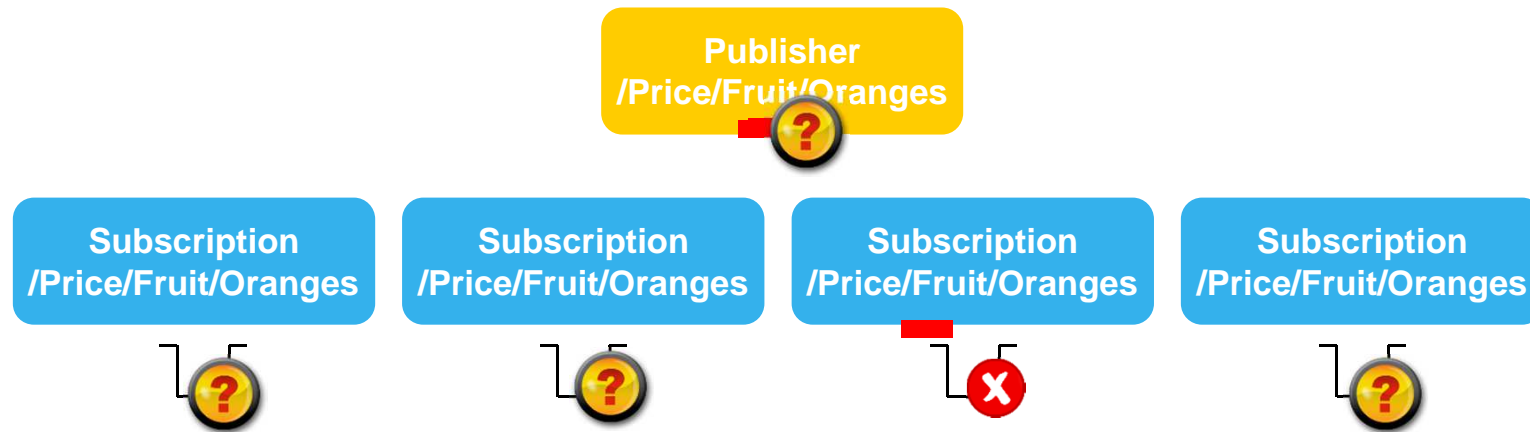
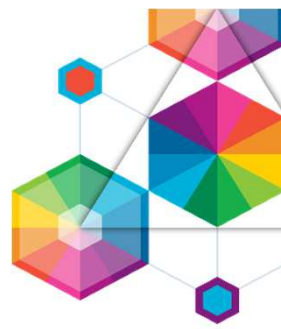


Publication, success or failure?



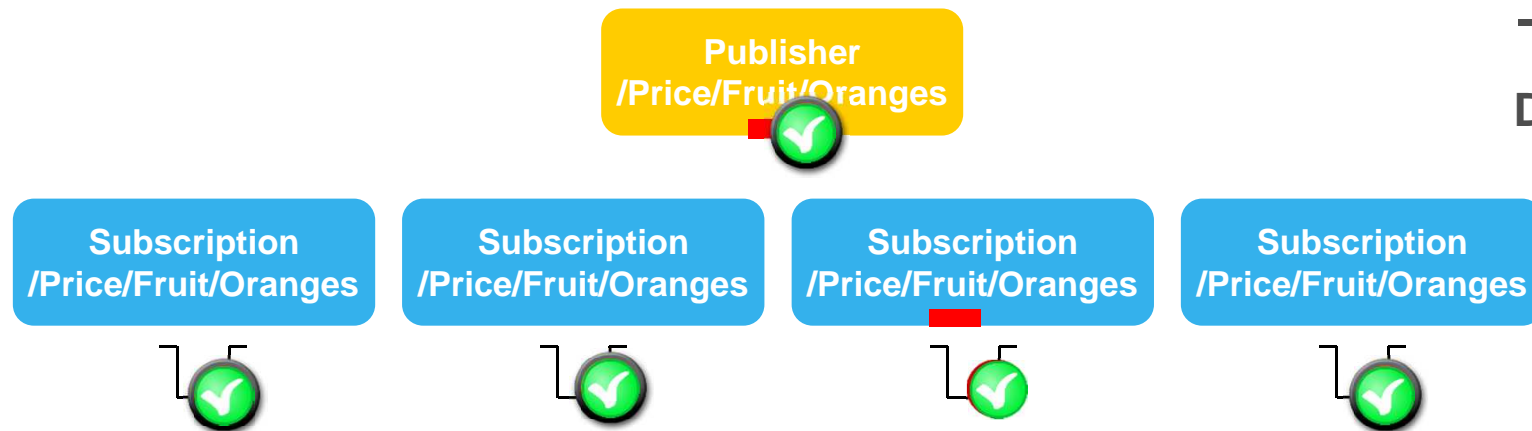
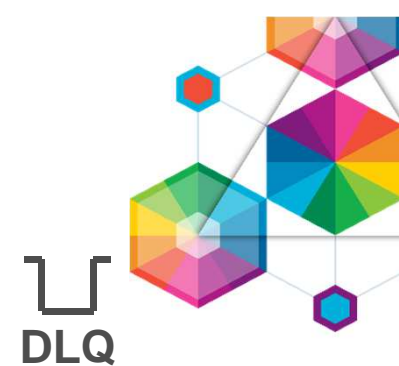
- Point-to-point is nice and simple:
 - Did the message get onto the queue?
 - Was it persistent and transacted?

Publication, success or failure?



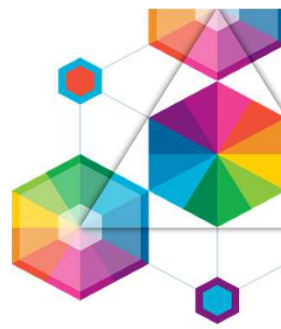
- Point-to-point is nice and simple:
 - Did the message get onto the queue?
 - Was it persistent and transacted?
- Publish/subscribe is not so clear cut...
 - **Persistence and transactions still ensures integrity of *successful* publications.**
 - But if one or more subscriptions can't receive the publication, ***should the publish fail?***

Publication, success or failure?

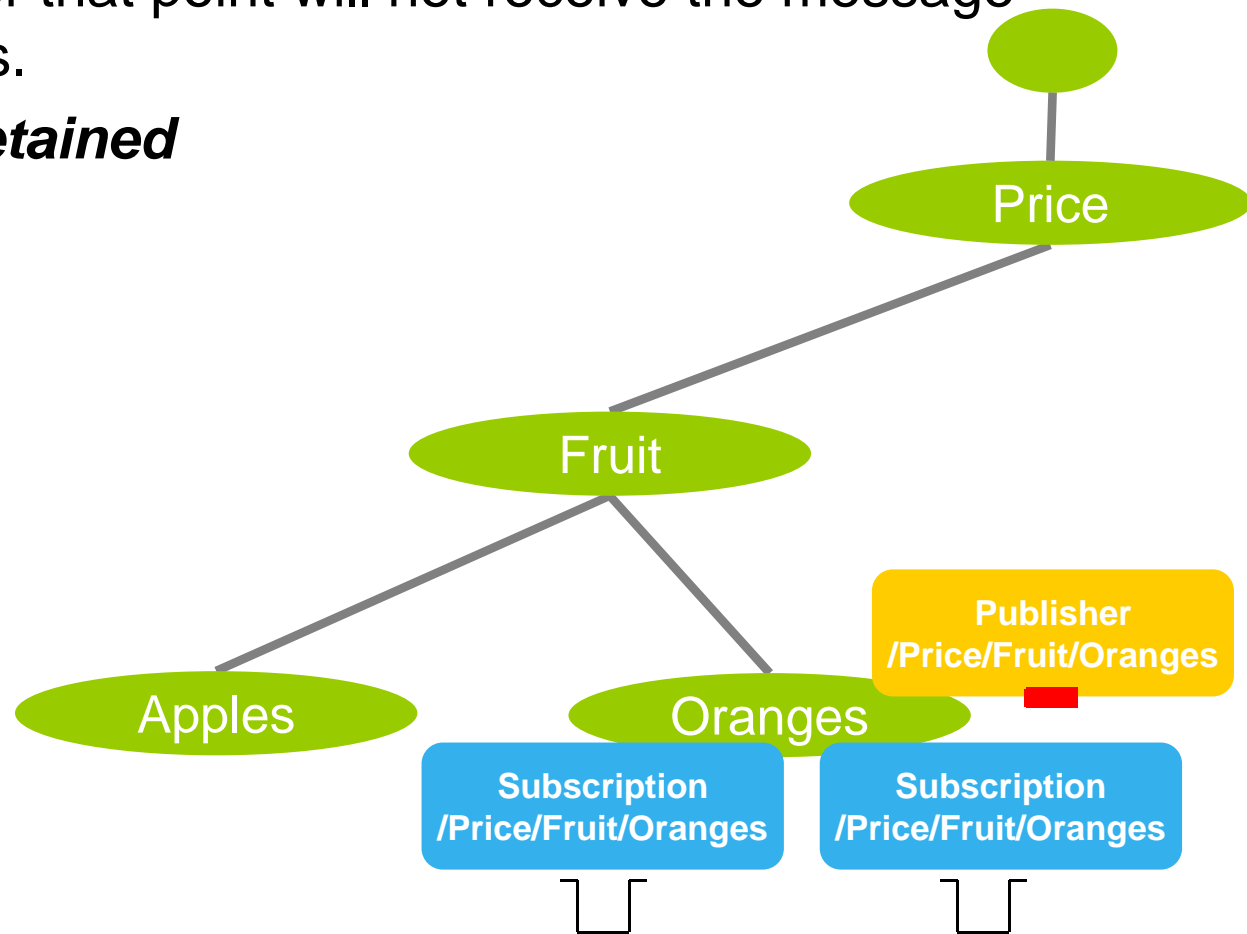


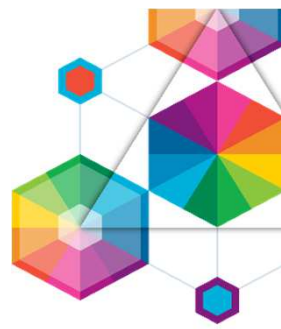
- *Should those subscriptions impact the others, should the publisher know?*
- *What if the subscriptions are **durable** and the publication is **persistent**?*
- Controlled at the **topic** level
 - Persistent Message Delivery (**PMSGDLV**) and Non-persistent Message Delivery (**NMSGDLV**).
 - **ALL** – all messages must be delivered to all subscribers, or roll-back
 - **ALLDUR** – all messages must be delivered to **DURABLE** subscribers, or roll-back
 - **ALLAVAIL** – in the case of any failures to deliver, no roll-back
- Don't forget that being able to DLQ a publication is still counted as a *success!*
 - **USEDLQ** on the **topic** to fine tune this behaviour. V7.1
- *And finally, remember – when there are no subscriptions, no-one gets it. That's still a successful publish!*

Retained publications



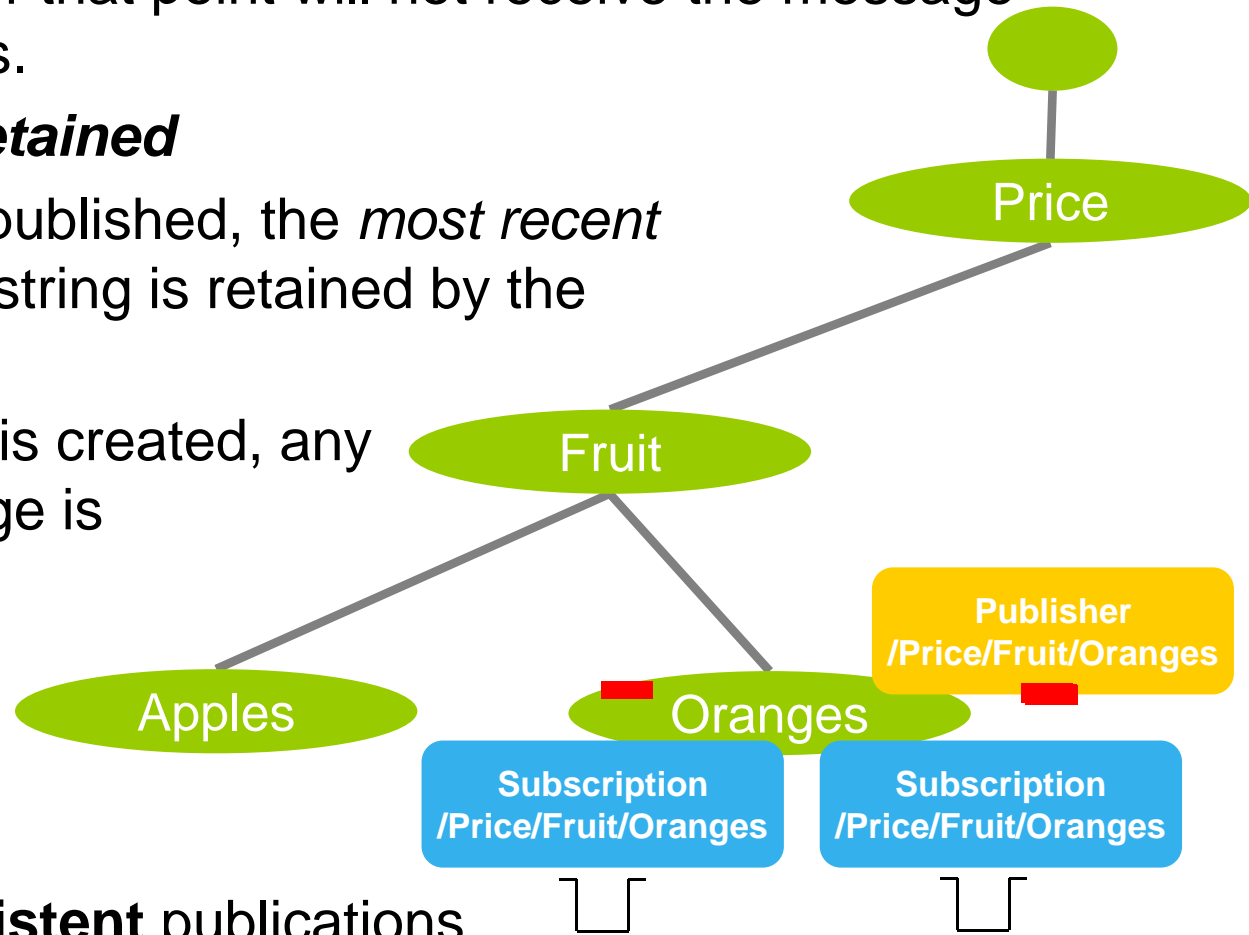
- When a message is published to a topic string, it is delivered to each matching subscription registered at that time.
- Subscriptions created after that point will not receive the message only newly published ones.
- Unless publications are ***retained***





Retained publications

- When a message is published to a topic string, it is delivered to each matching subscription registered at that time.
- Subscriptions created after that point will not receive the message only newly published ones.
- Unless publications are **retained**
- Every time a message is published, the *most recent* publication for each topic string is retained by the queue manager.
- When a new subscription is created, any matching retained message is delivered to it.
- Take care, using retained can be **subtle**
- Don't confuse it with **persistent** publications

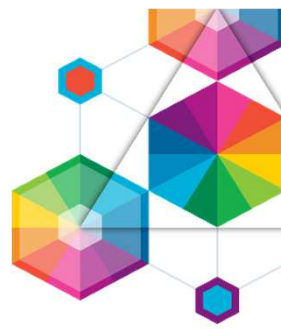


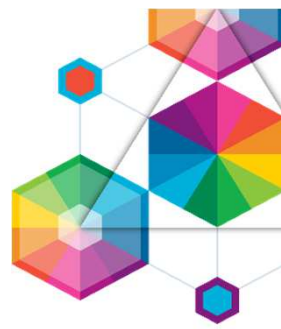
Application
development



Pub/Sub application development

- Support for all languages which use MQ (of course!)
- JMS provides standard objects (eg. Topic, Publisher, Subscriber) for Pub/Sub
- MQI added similar support (eg. Verbs, objects) beginning with MQ v7.0



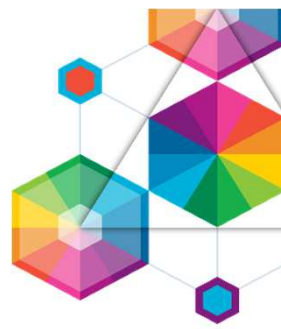


Publish/Subscribe using MQI - Summary

- The verbs used are:-
 - MQOPEN
 - MQPUT
 - MQSUB*
 - MQSUBRQ*
 - MQCLOSE
- New structures to accompany new verbs
 - MQSUB* – MQSD – Subscription Descriptor
 - MQSUBRQ* – MQSRO – Subscription Request Options

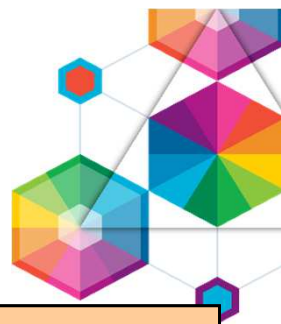
* New with MQv7

New structure for subscription



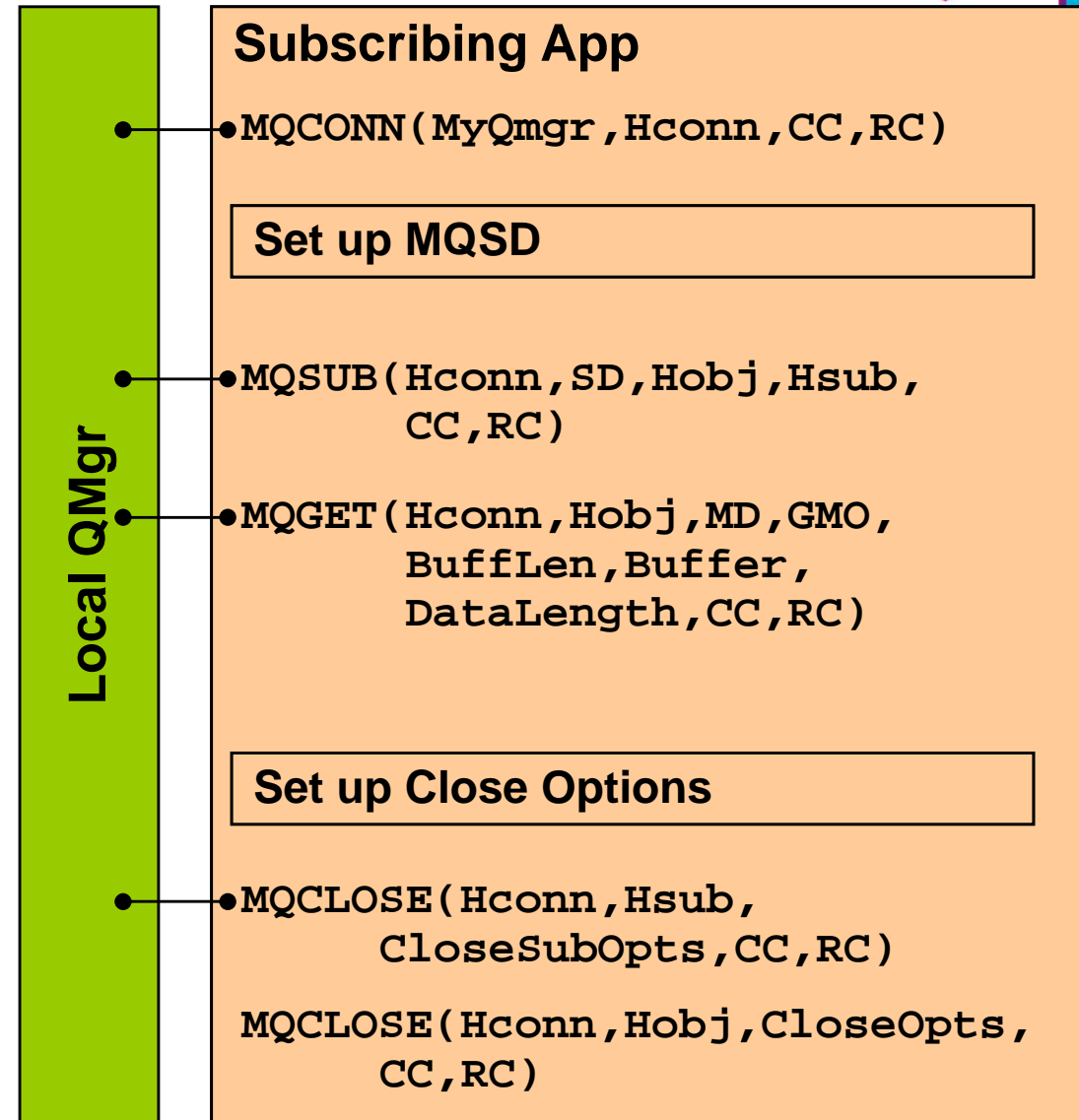
- MQSD – subscription description
 - StruclId – structure identifier
 - Version – structure version number
 - **Options** – options
 - **ObjectName** – object name
 - AlternateUserId – alternate user id
 - AlternateSecurityId – alternate security id
 - SubExpiry – subscription expiry
 - **ObjectString** – object string
 - **SubName** – subscription name
 - SubUserData – subscription user data
 - SubCorrelId – subscription correlation id
 - PubPriority – publication priority
 - PubAccountingToken – publication accounting token
 - PubAppIdentityData – publication application identity data
 - SubLevel – subscription level
 - ResObjString –

Non-durable managed subscription application



- **Connect to a queue manager**
- **Subscribe to a topic**
- **Get publications published to that topic**

- **Close subscription to unsubscribe**
- **Close queue**



Publishing application

- MQOPEN a topic
- MQOD describes a **topic to publish to**:
 - ObjectType
 - MQOT_Q for point-to-point
 - MQOT_TOPIC** for publish
 - ObjectString/ObjectName
 - Where to publish
- MQPUT a message

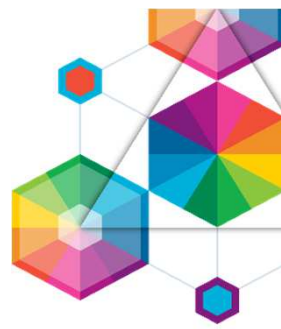
```
OpnOpts = MQOO_OUTPUT
          | MQOO_FAIL_IF_QUIESCING;
MQOPEN( hConn,
        &ObjDesc,
        OpnOpts,
        &hObj,
        &CompCode,
        &Reason);
MQPUT ( hConn,
        hObj,
        &MsgDesc,
        &pmo,
        strlen(pBuffer),
        pBuffer,
        &CompCode,
        &Reason);
```

```
MQOD    ObjDesc = {MQOD_DEFAULT};

ObjDesc.ObjectType      = MQOT_TOPIC;
ObjDesc.Version         = MQOD_VERSION_4;
ObjDesc.ObjectString.VSPtr = "Price/Fruit/Apples";
ObjDesc.ObjectString.VSLength = MQVS_NULL_TERMINATED;
```

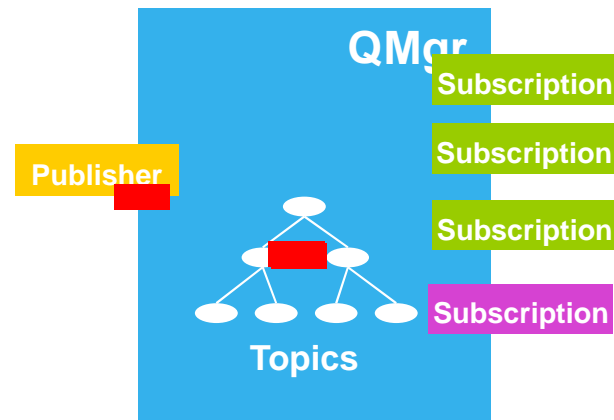
Topologies

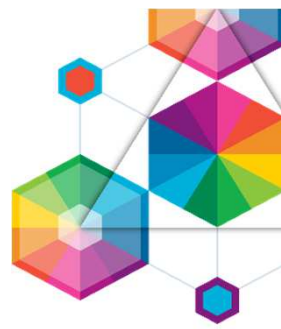




Distributed publish/subscribe

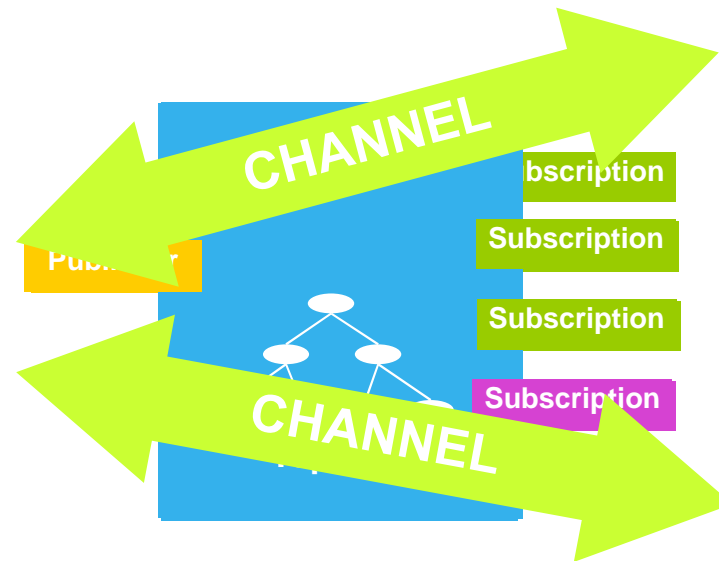
- We know how everything revolves around the topic tree, dynamically built up in a queue manager

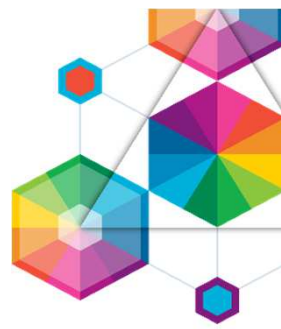




Distributed publish/subscribe

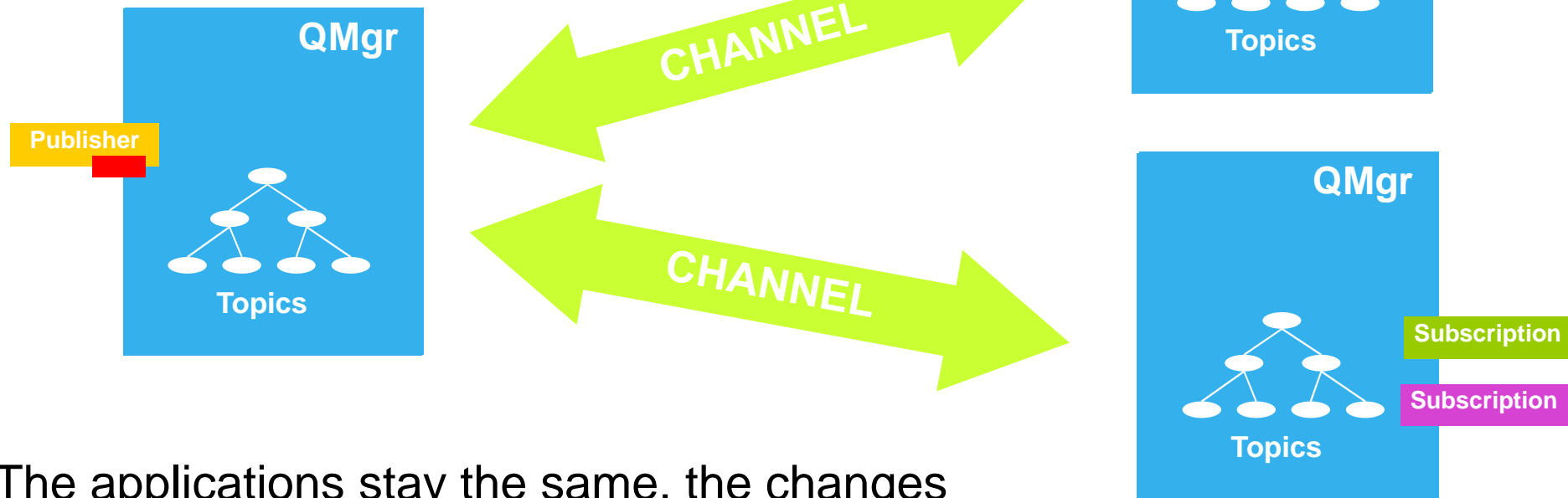
- We know how everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them



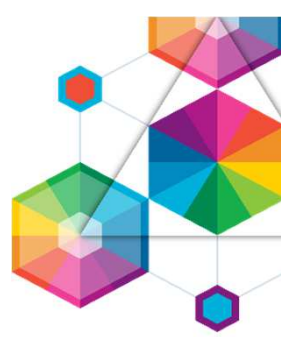


Distributed publish/subscribe

- We know how everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them
- Enabling publications to be propagated to subscriptions on different queue managers

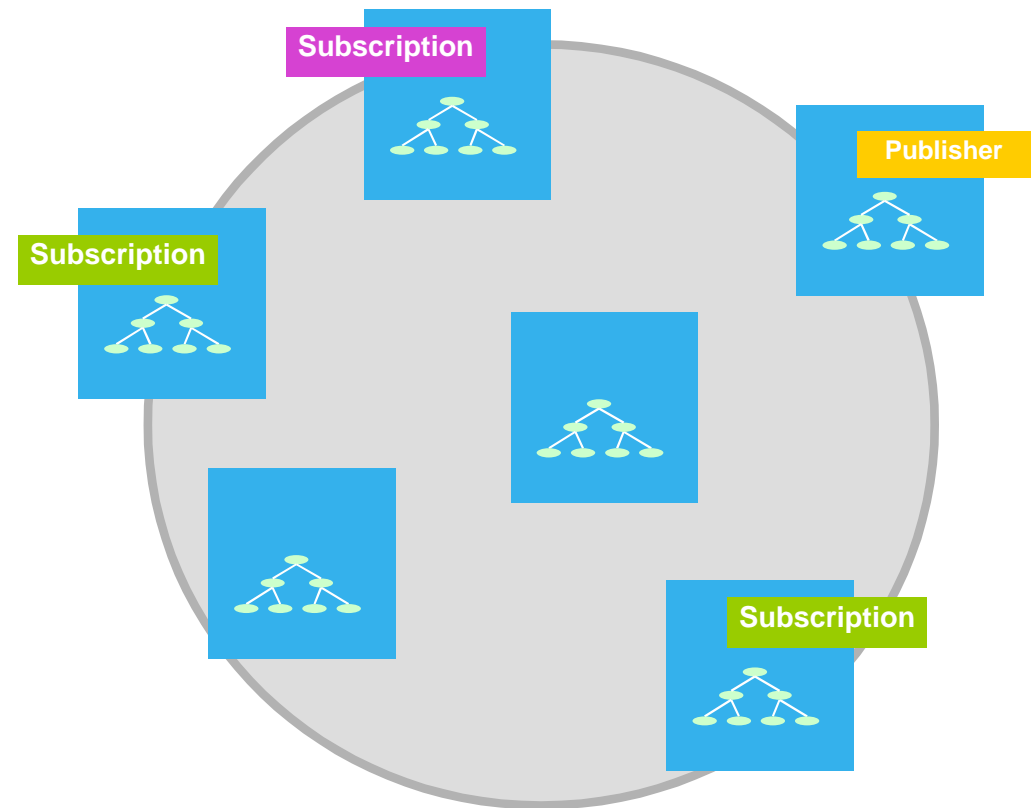
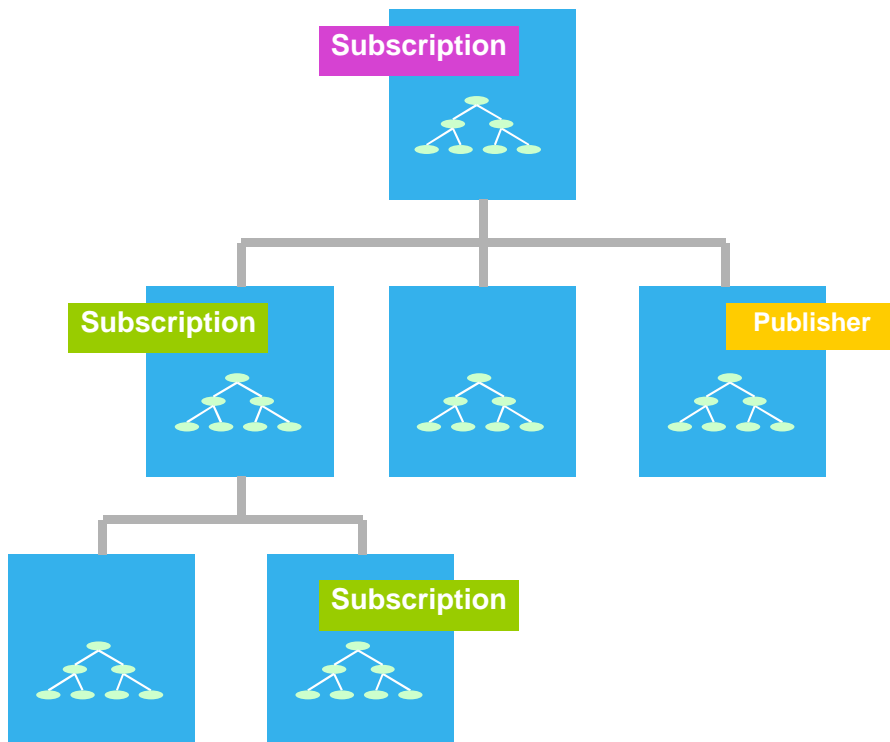


- The applications stay the same, the changes are at the configuration level.



Distributed publish/subscribe topologies

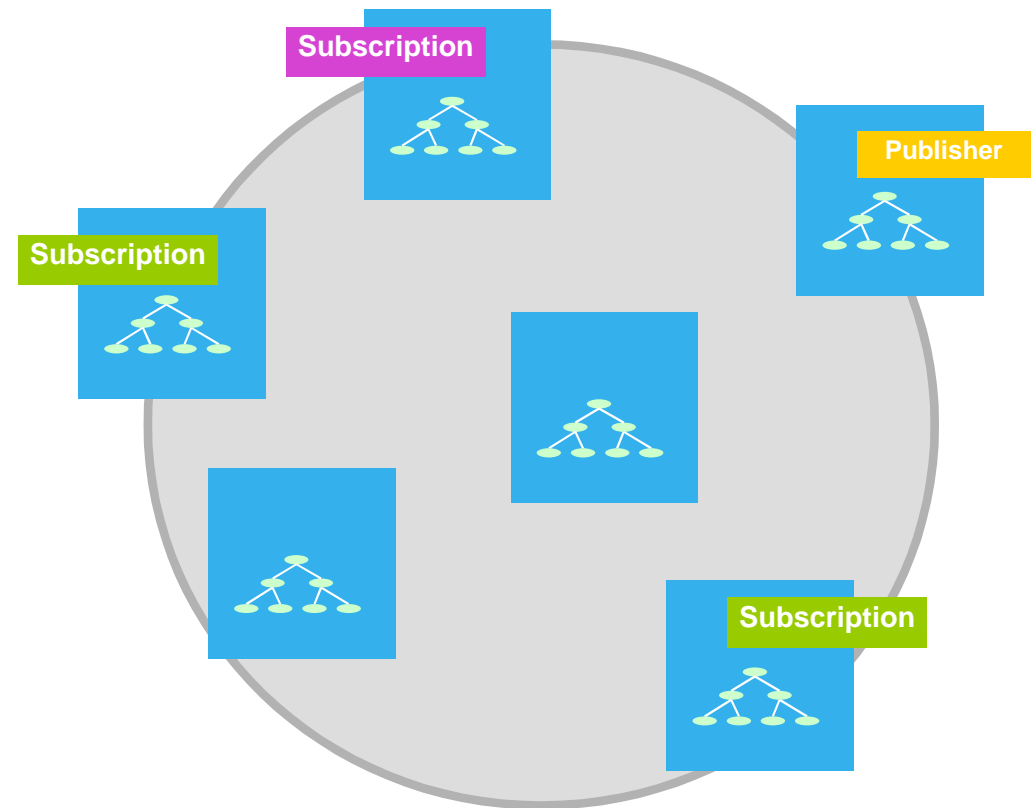
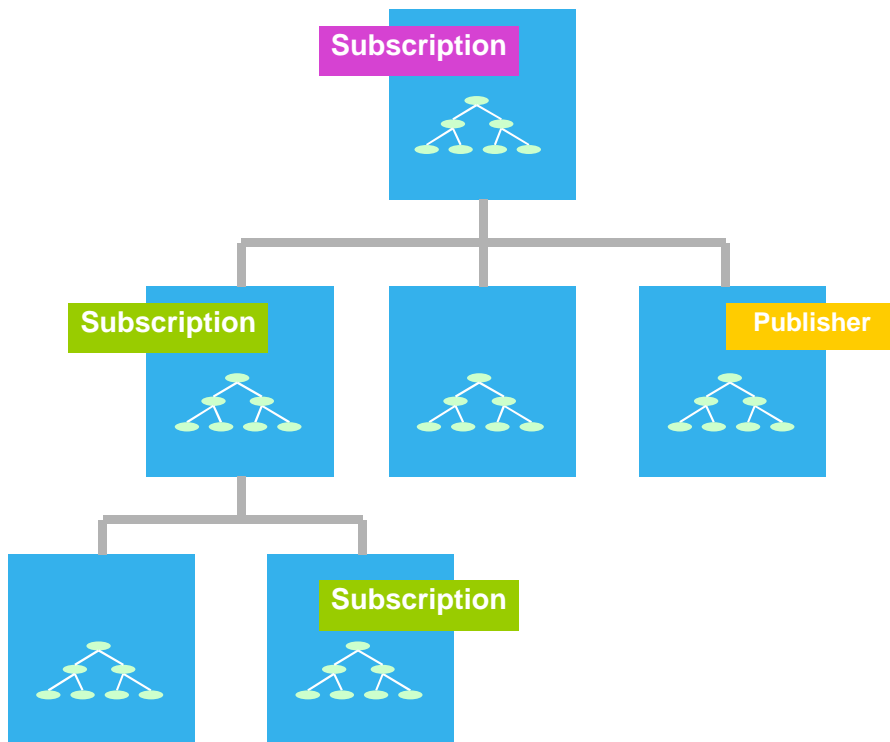
- Publish/subscribe topologies can either be created as a defined ***hierarchy*** or more dynamically as a ***cluster***





Distributed publish/subscribe topologies

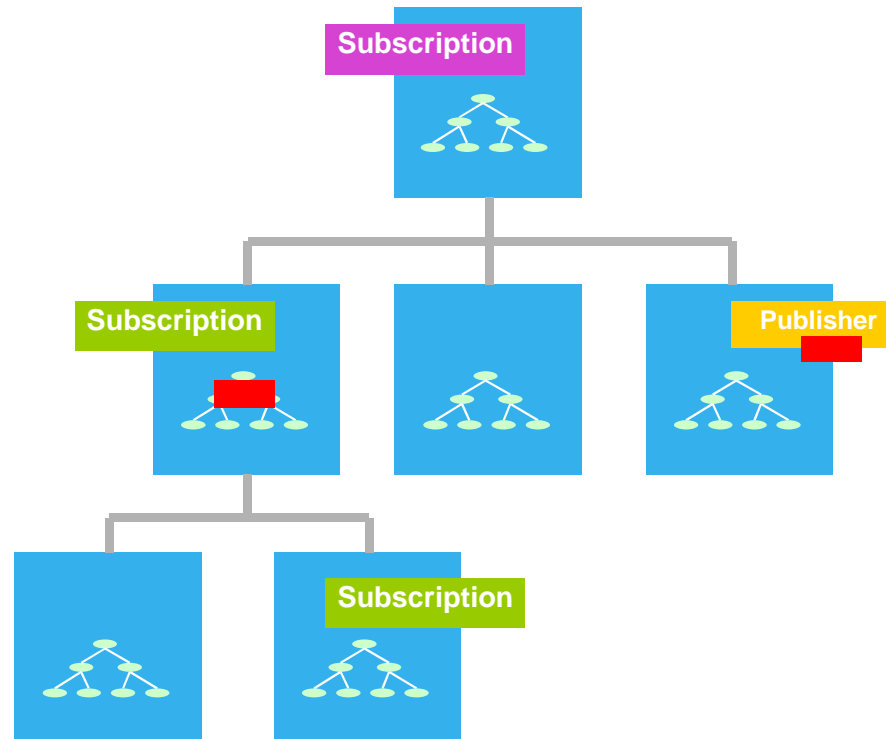
- A hierarchy manually defines the relationship between each queue manager.
- Messages are flowed via those relationships.

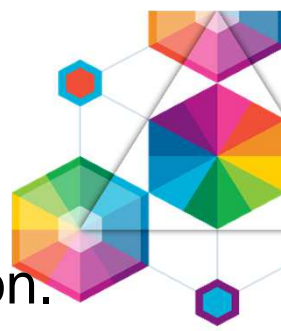




Distributed publish/subscribe topologies

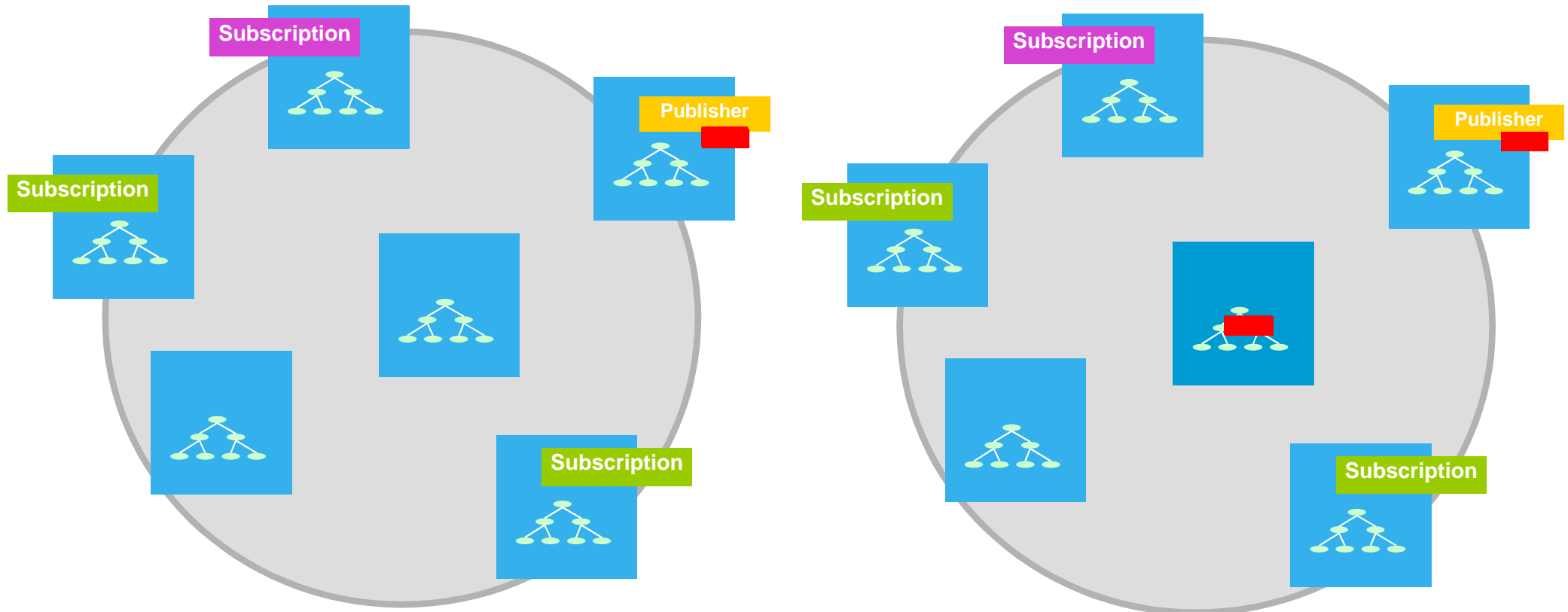
- A hierarchy manually defines the relationship between each queue manager.
- Messages are flowed via those relationships.





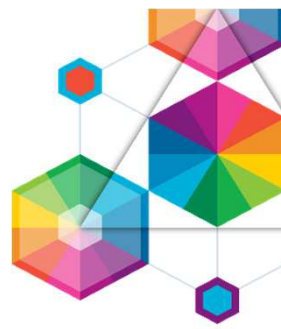
Distributed publish/subscribe topologies

- A cluster can be used for publish/subscribe through simple configuration.
- Any queue manager can publish and subscribe to topics
- Published messages can go **direct** between queue managers



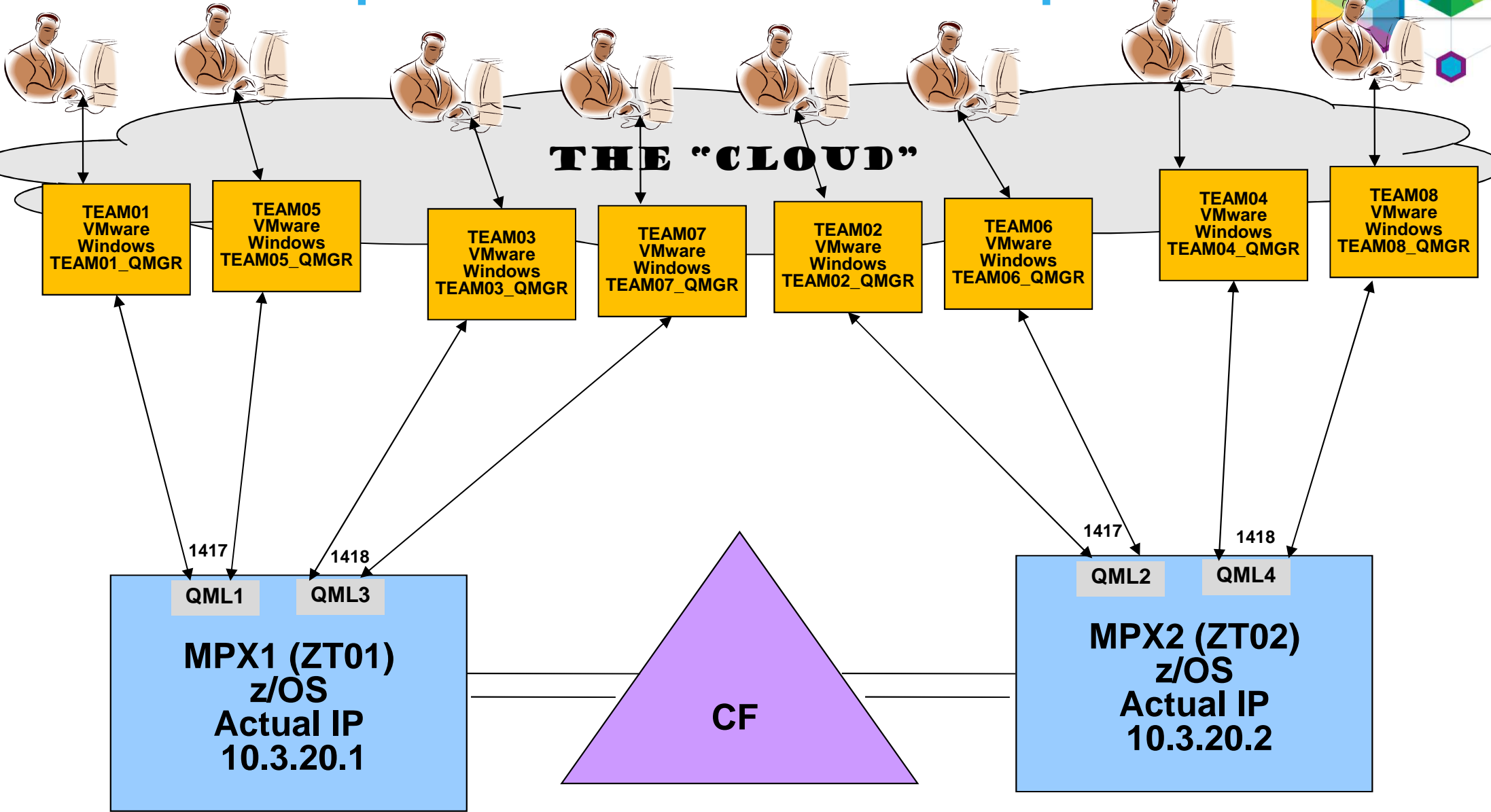
V8 Or be **routed** via specific queue managers

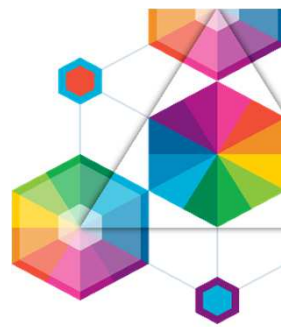
Summary



- Publish/Subscribe in IBM MQ
- Administration of publish/subscribe
- Management of publish/subscribe
- Subscriptions and publications
- Topologies

The Workshop Environment - LPAR Map





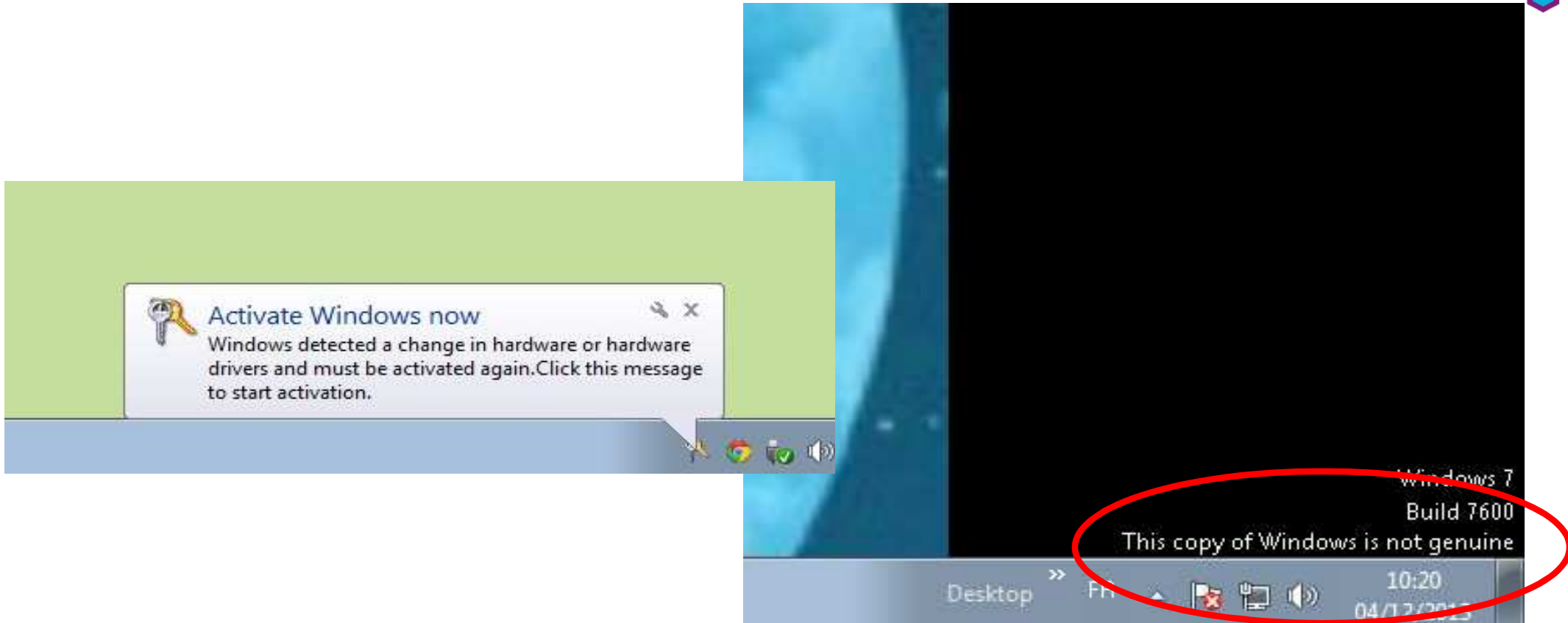
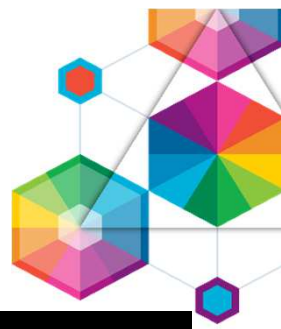
Rules of engagement

- Each lab includes detailed instructions.
- You're strongly advised to follow the instructions exactly.
- There are 8 "teams" (TSO userids): TEAM01, TEAM02....
 - Password: PUB09SUB

Team	Primary QM	LPAR
TEAM01, TEAM05	QML1	MPX1 (ZT01)
TEAM02, TEAM06	QML2	MPX2 (ZT02)
TEAM03, TEAM07	QML3	MPX1 (ZT01)
TEAM04, TEAM08	QML4	MPX2 (ZT02)

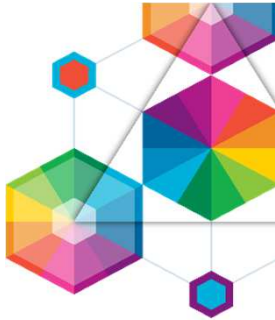
- Given the nature of the POT, and the limited time for each exercise, it's probably not a good idea to deviate, especially in the beginning.
- If you're not sure if a step worked correctly, ask the instructor. Better to ask too much, than not enough.
- These LPARs are only "sandbox" systems.
 - You are given enough privilege on these z/OS systems to do the exercises, see results, etc.
 - Please try to resist all temptation to modify the queue manager settings, change the z/OS userid "settings", eg. PF keys, etc. from the default values. Other classes/students use these ids!

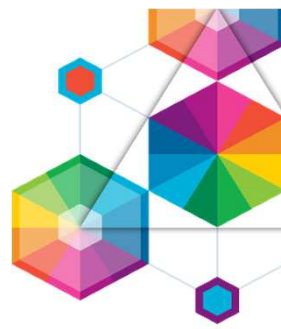
Warning: you may see the following in your VMware.....



Don't worry..... be happy..... This IS a genuine Windows, but running on a Cloud environment and I just don't bother re-activating each time it's deployed. Please do NOT go to the activation menus... just ignore this.

Backup





Bibliography

- [IBM MQ Knowledge Center](#)

(http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q001010_.htm?cp=SSFKSJ_8.0.0%2F1&lang=en)

- SC34-6950 MQv7 Publish/Subscribe User's Guide
- WSG24-7583 WebSphere MQ V7.0 Features and Enhancements