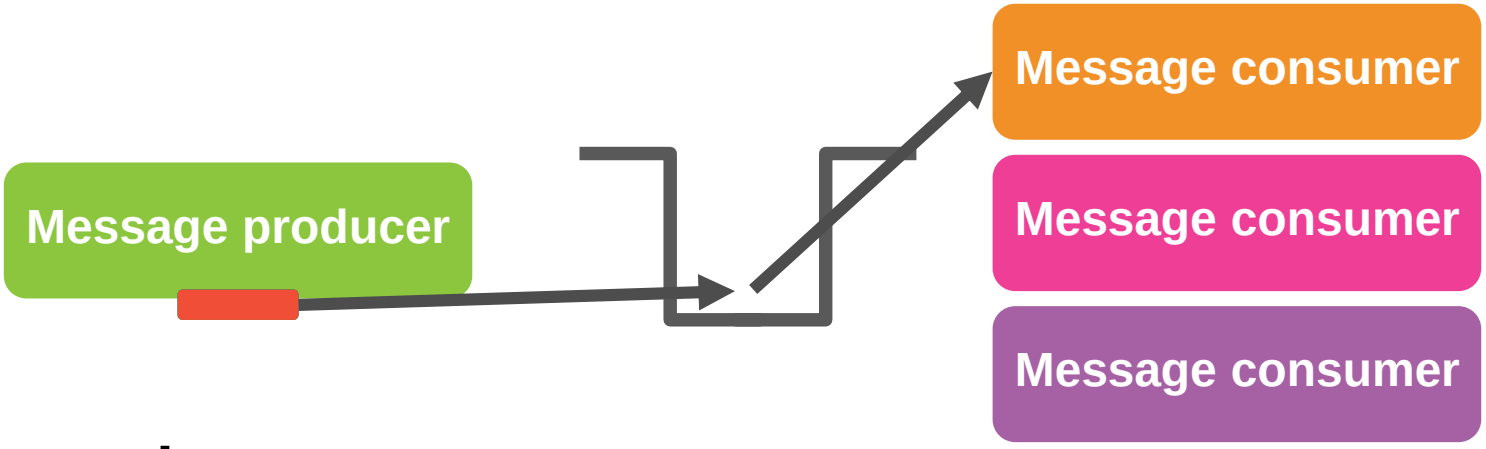
MQ Publish/Subscribe
an introduction to topic *objects, nodes and strings (among other things)*

David Ware
IBM MQ

Agenda

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

How does it compare to point-to-point?

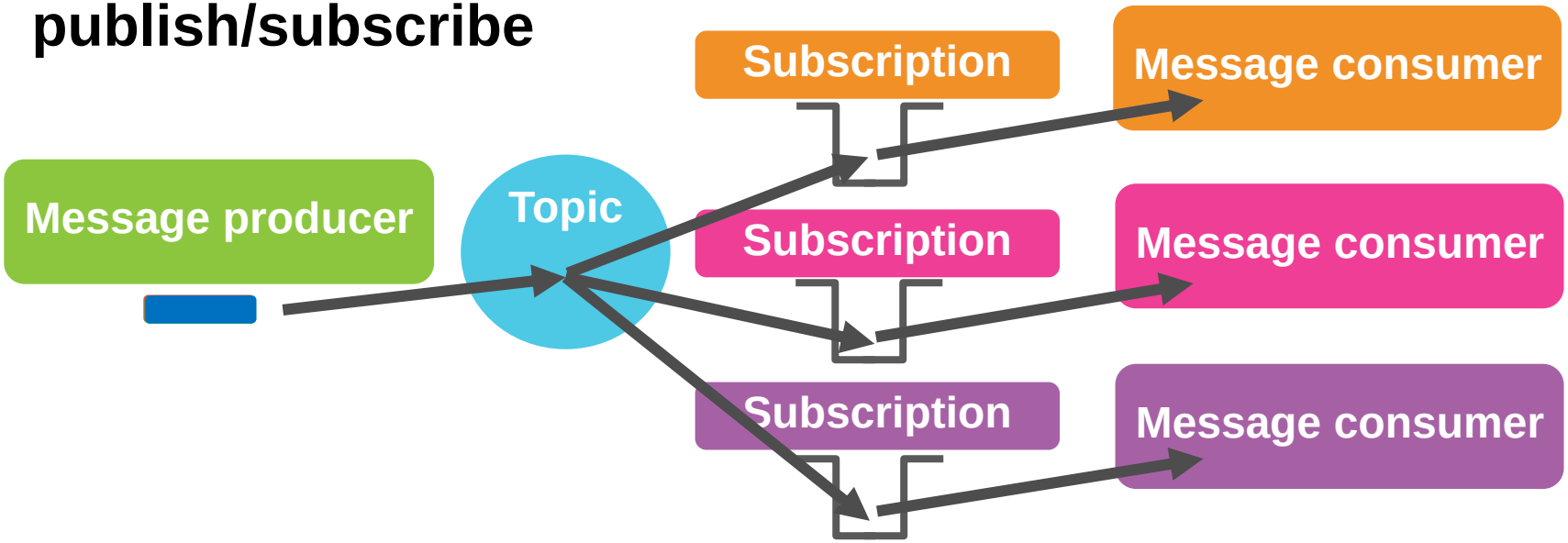


point-to-point

How does it compare to point-to-point?

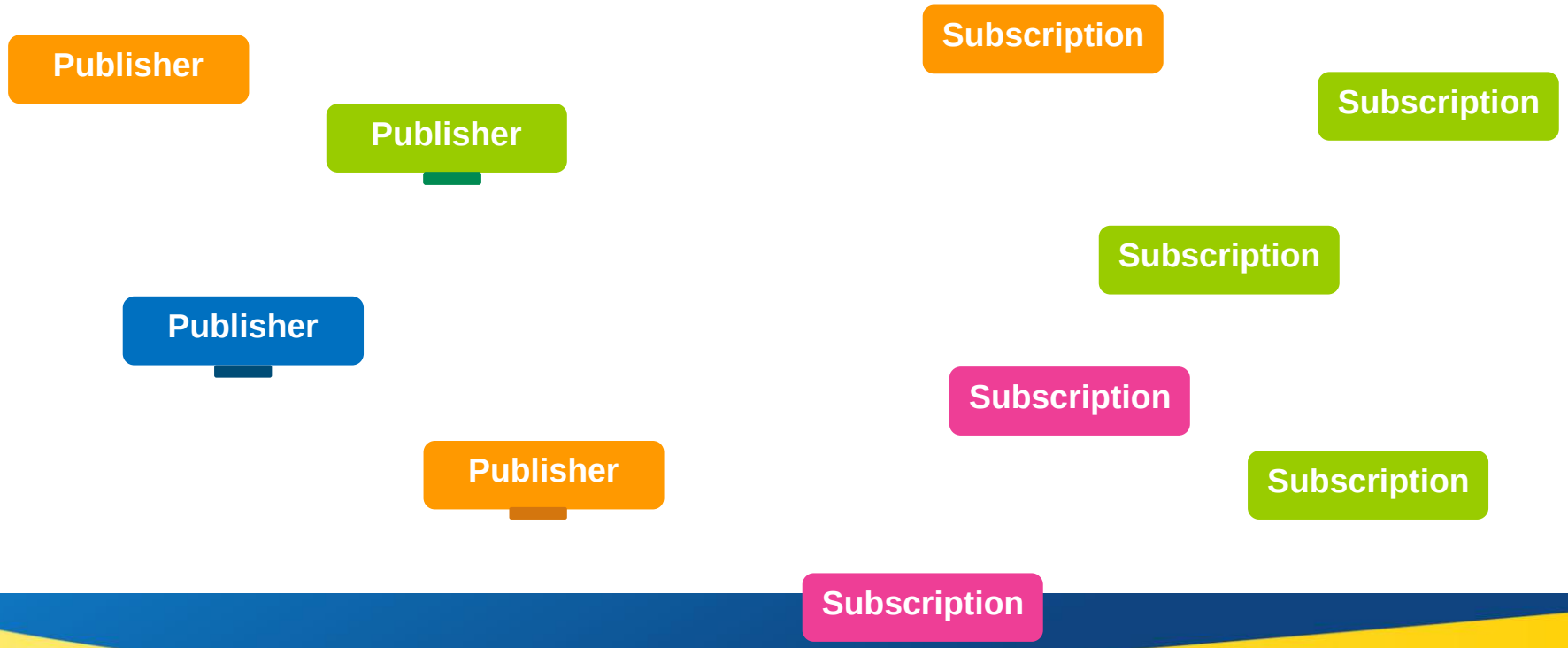
point-to-point

publish/subscribe



But which subscriptions receive the messages?

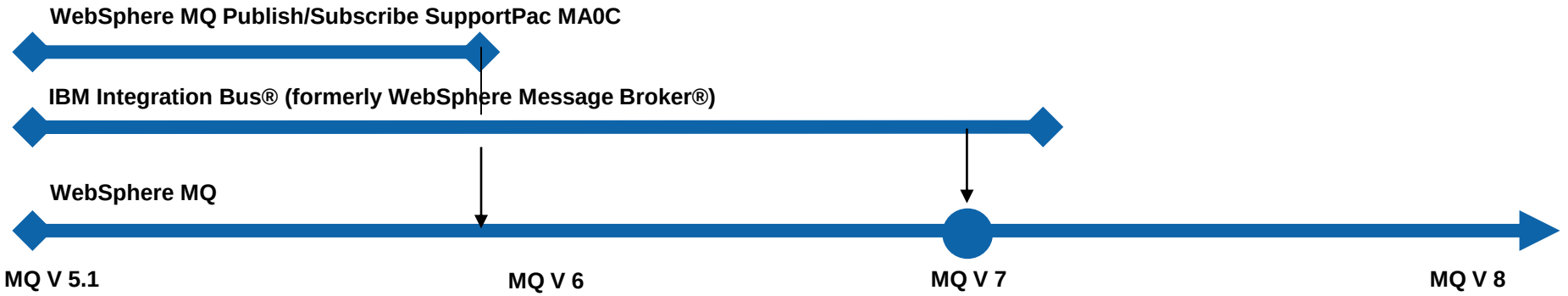
- Publishing and subscribing is based on '**topics**'
  - **Green** messages go to **green** subscribers
  - **Orange** messages go to **orange** subscribers
  - But nobody wants a ***blue*** message!



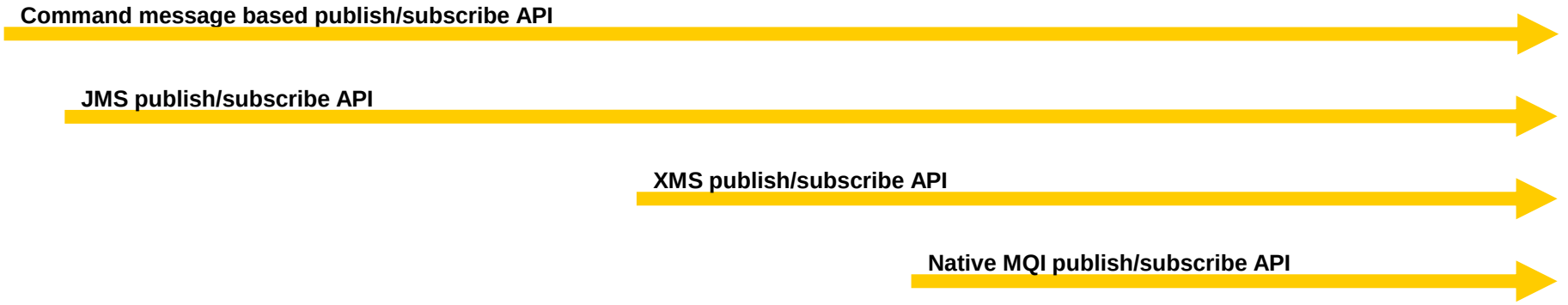
Publish/Subscribe in WebSphere MQ

WebSphere MQ's publish/subscribe over the years

Publish/Subscribe brokers

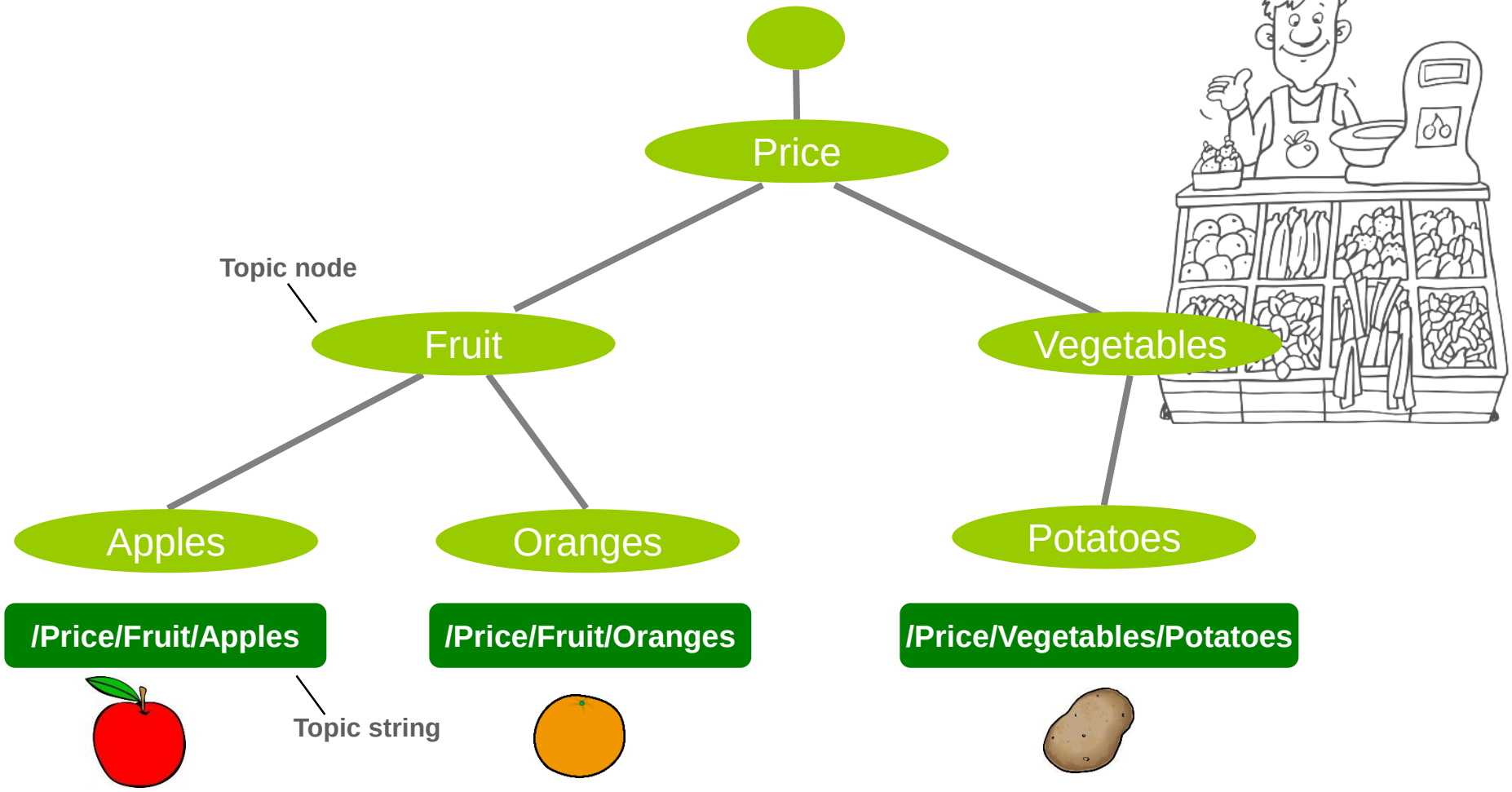


WebSphere MQ Publish/Subscribe APIs

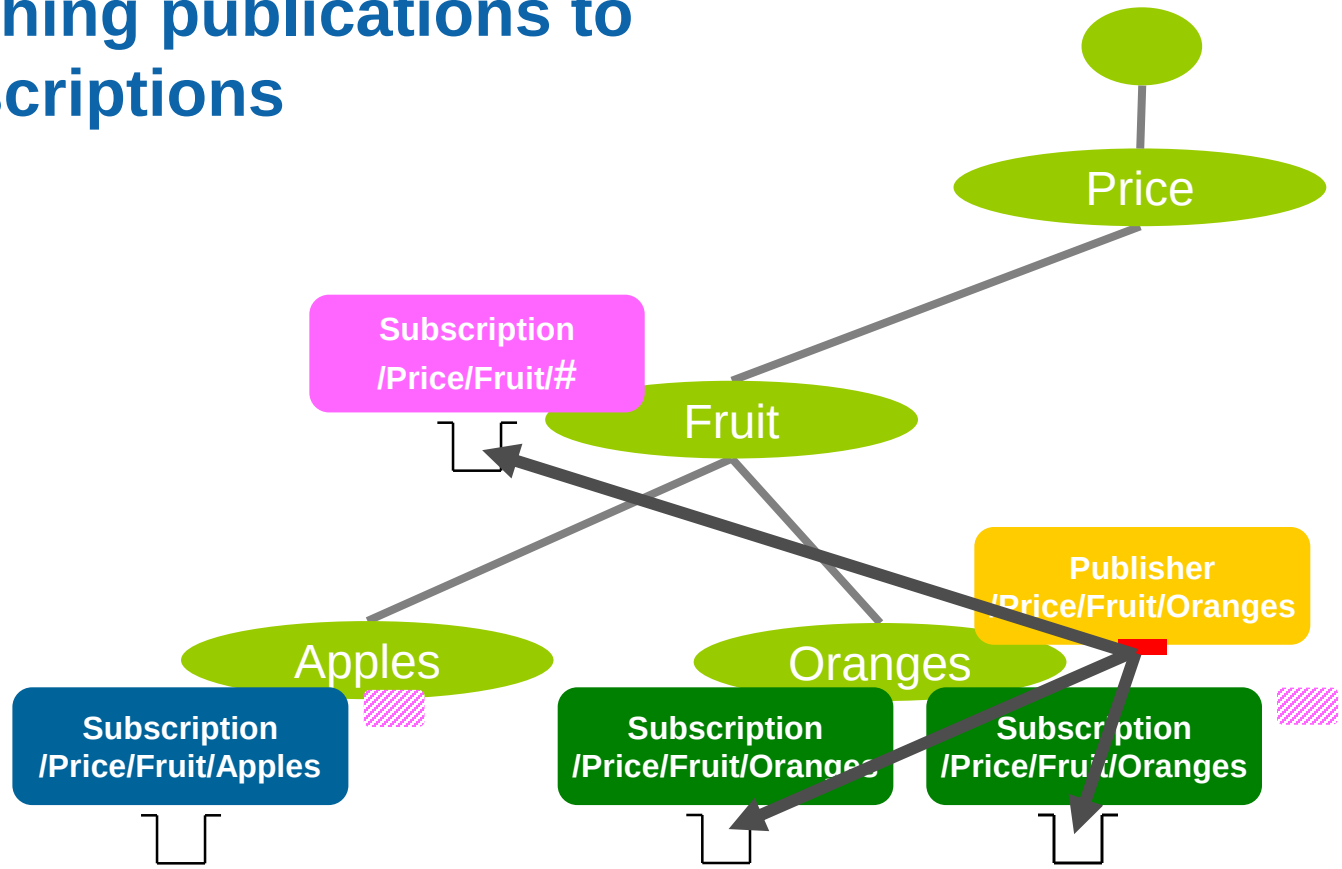


Publish/Subscribe since V7

It's all about the ***topic tree***



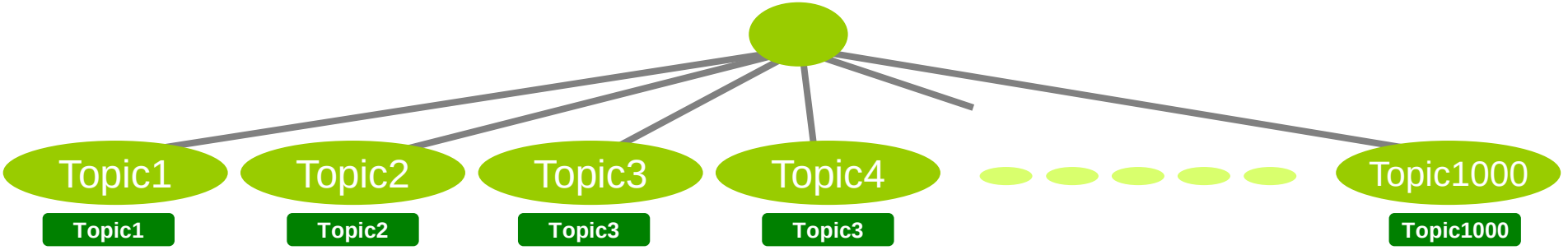
Matching publications to subscriptions



- 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

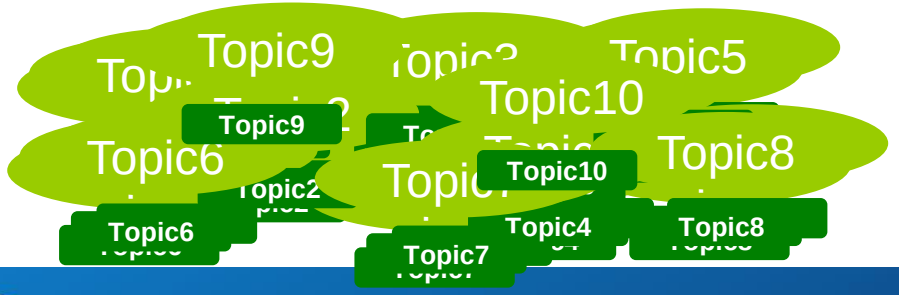
Designing your topic tree structure

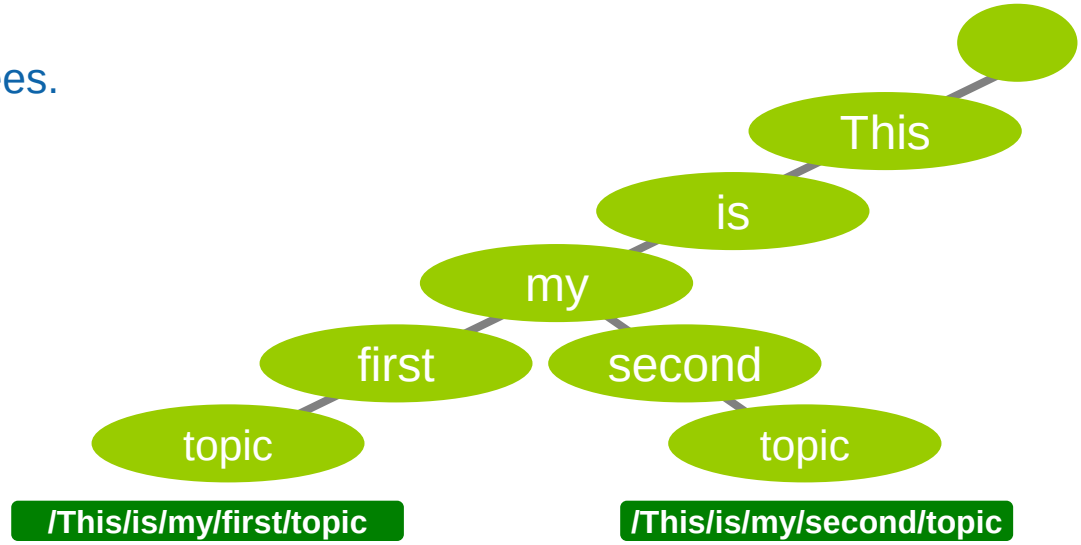
- Make it extendable, and understandable.



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

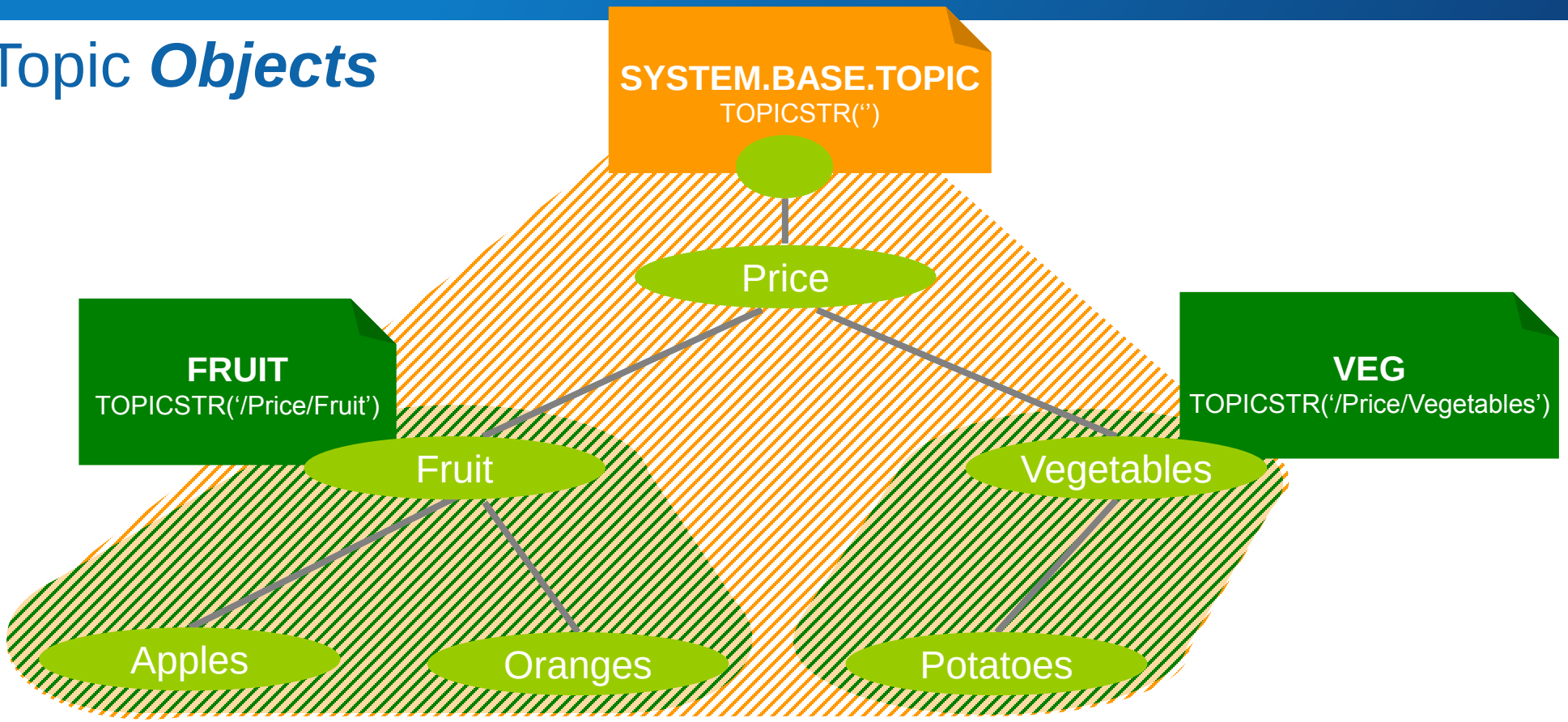
- Avoid a rapidly changing set of topics.





Publish/Subscribe configuration

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.

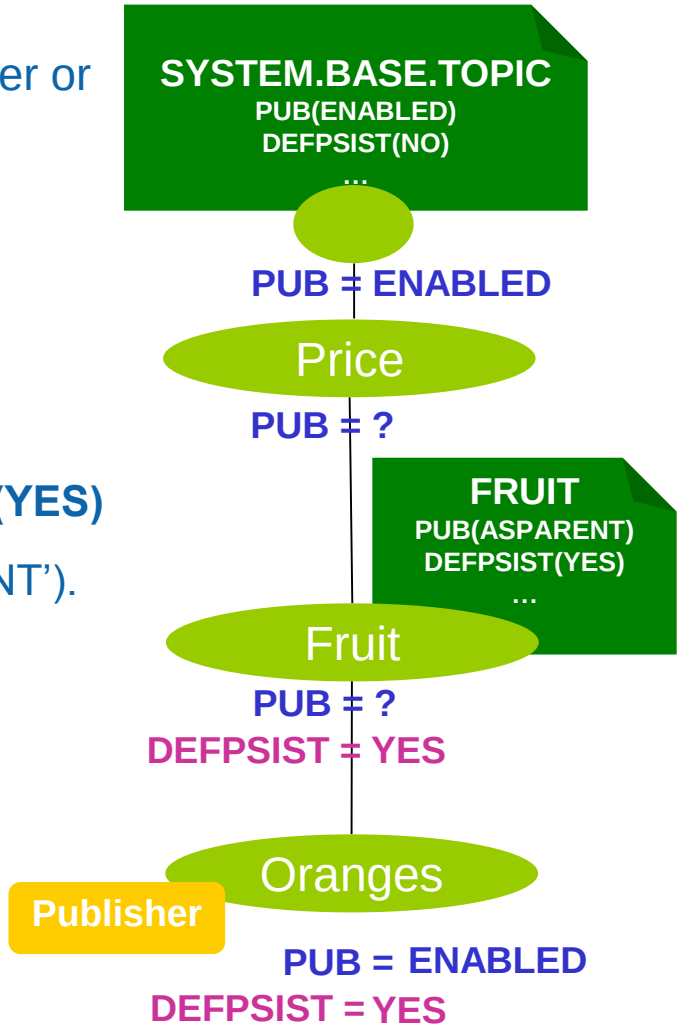
Topic object attributes

- Many attributes can be set on topic objects to effect a publisher or subscriber's behaviour.

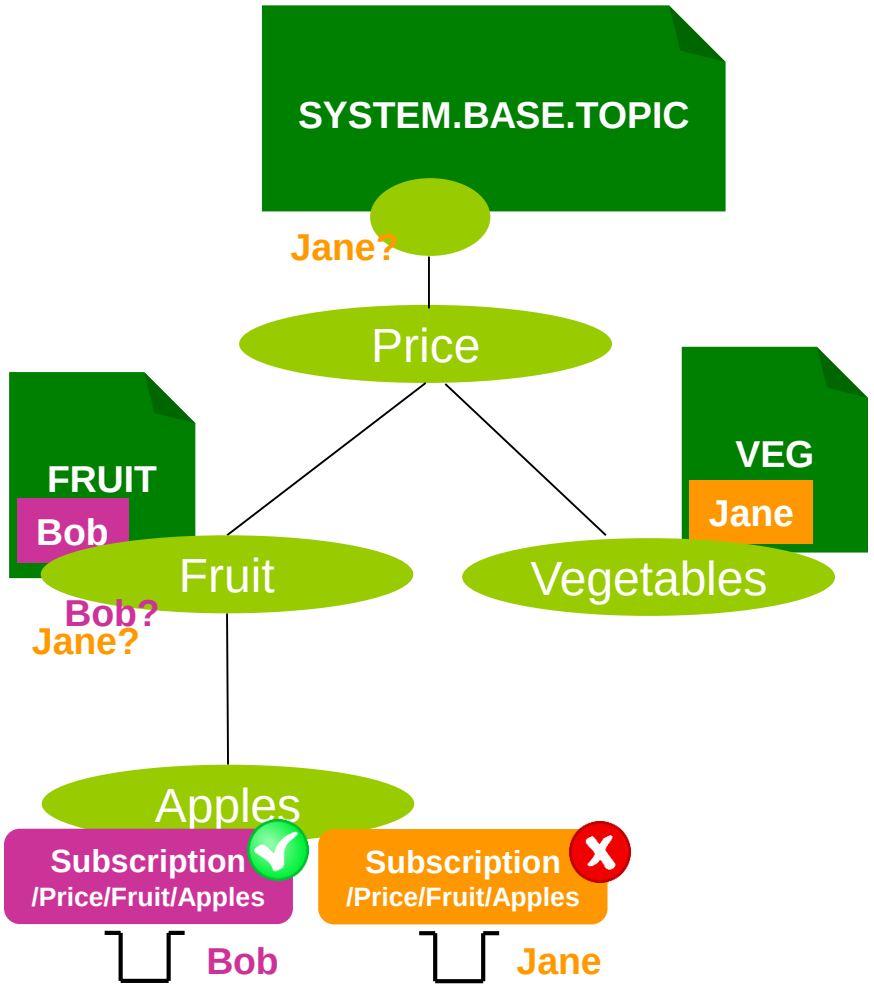
- Dynamic nodes inherit their behaviour from nodes above.

- Create a topic object for topic string **'/Price/Fruit'**
  - **DEFINE TOPIC(FRUIT) 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'**
  - **Are publications enabled?**
  - **What message persistence to use?**



Topic Security



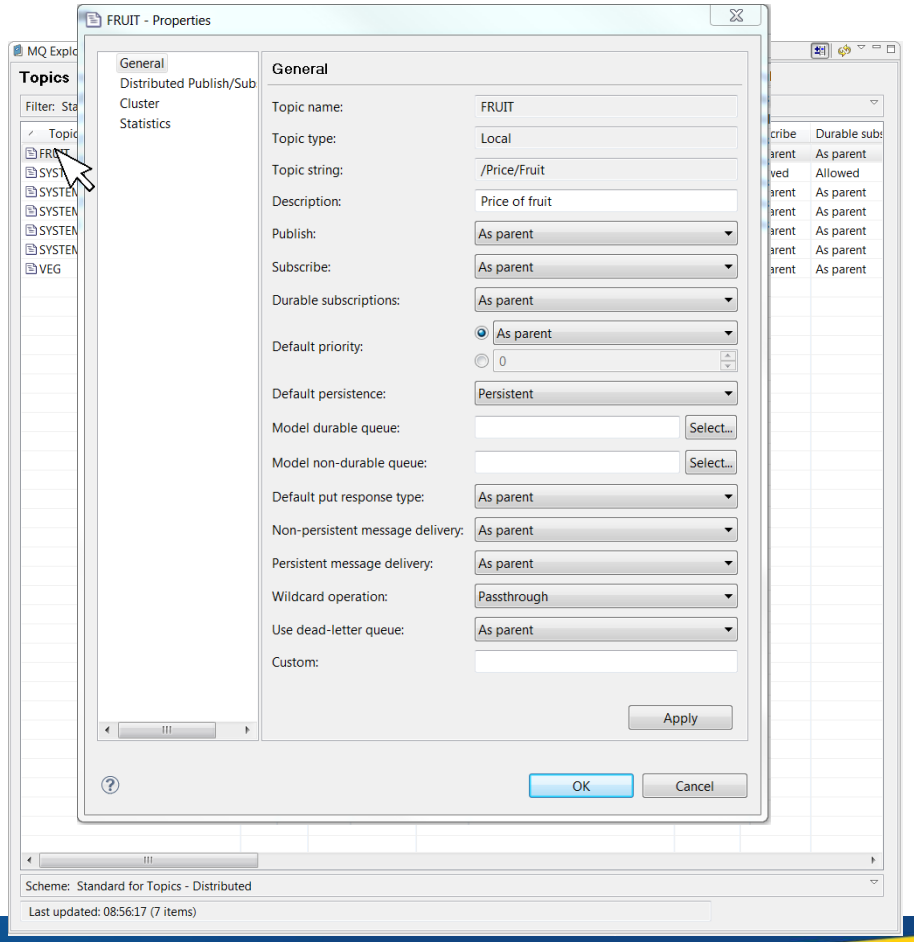
- Access control is set for a defined topic *object*
  - ***not a topic string!***
- Authority checks performed on the topic tree
  - Walk up the tree, just like attributes.
  - Keep checking until an authorisation is found or we run out of topic tree.
- Authority check on a subscription's destination queue
  - Check is for PUT to that queue
  - *(hang on for more on subscription queues)*
- ***Pick a suitable layer of the topic hierarchy and set access control at this point.***
- ***Think hard before adding additional access control at higher levels in the tree as this can cause confusion and grant very wide authorisations.***

Managing topics

Managing topics

- Displaying topic object definitions
  - This shows how the administered topic objects were configured

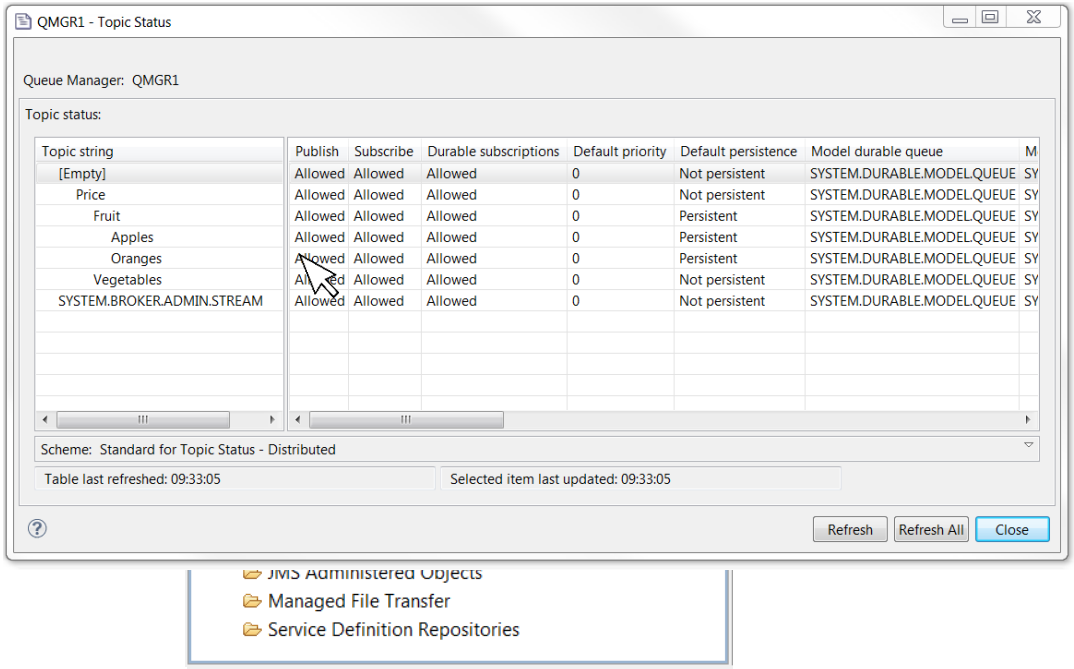
```
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager QMGR1. DISPLAY

DISPLAY TOPIC(*)
     1 : DISPLAY TOPIC(*)
AMQ8633: Display topic details.
   TOPIC(FRUIT)                            TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(SYSTEM.BASE.TOPIC)                TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(SYSTEM.BROKER.ADMIN.STREAM)       TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(SYSTEM.BROKER.DEFAULT.STREAM)     TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(SYSTEM.BROKER.DEFAULT.SUBPOINT)   TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(SYSTEM.DEFAULT.TOPIC)             TYPE(LOCAL)
AMQ8633: Display topic details.
   TOPIC(VEG)                              TYPE(LOCAL)

DISPLAY TOPIC(FRUIT)
     2 : DISPLAY TOPIC(FRUIT)
AMQ8633: Display topic details.
   TOPIC(FRUIT)                            TYPE(LOCAL)
   TOPICSTR(/Price/Fruit)                  DESCR(Price of fruit)
   CLUSTER( )                              CLROUTE(DIRECT)
   DURSUB(ASPARENT)                        PUB(ASPARENT)
   SUB(ASPARENT)                           DEFPSIST(YES)
   DEFPRTY(ASPARENT)                       DEFPRESP(ASPARENT)
   ALTDATE(2014-04-03)                     ALTTIME(08.44.48)
   PMSGDLV(ASPARENT)                       NPMSGDLV(ASPARENT)
   PUBSCOPE(ASPARENT)                      SUBSCOPE(ASPARENT)
   PROXYSUB(FIRSTUSE)                      WILDCARD(PASSTHRU)
   MDURMDL( )                              MNDURMDL( )
   MCAST(ASPARENT)                         COMMINFO( )
   USEDLQ(ASPARENT)                        CUSTOM( )
```



Managing topics

- Displaying the topic tree
  - This shows how the nodes in the topic tree behave

```
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager QMGR1. DISPLAY

DISPLAY PUBSUB ALL
     1 : display PUBSUB all
AMQ8723: Display pub/sub status details.
   QMNAME(QMGR1)                           TYPE(LOCAL)
   STATUS(ACTIVE)                          SUBCOUNT(5)      V8
   TPCOUNT(11)

DISPLAY TPSTATUS('#') TOPICSTR WHERE(SUBCOUNT GT 0)
 22 : DISPLAY TPSTATUS('#') TOPICSTR where(SUBCOUNT GT 0)
AMQ8754: Display topic status details.
   TOPICSTR(/Price/Vegetables/Potatoes)    SUBCOUNT(1)
AMQ8754: Display topic status details.
   TOPICSTR(/Price/Fruit/Oranges)          SUBCOUNT(1)
AMQ8754: Display topic status details.
   TOPICSTR(/Price/Fruit/Apples)           SUBCOUNT(1)

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

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**. No, a **topic object**. No, **both**. Actually, er, ***any of them!***
- *So which should I use?*
  - Using the topic string is probably the easiest, it's closest to what the application 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(**FRUIT**, '')* → **/Price/Fruit**
  - If you use both, you get both!
    - The topic string is appended to the topic string of the object
    - *Sub(**FRUIT**, '**Apples**')* → **/Price/Fruit/Apples**
- *If in doubt, check the topic tree for which nodes are actually being used*

Subscriptions

Subscription types

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

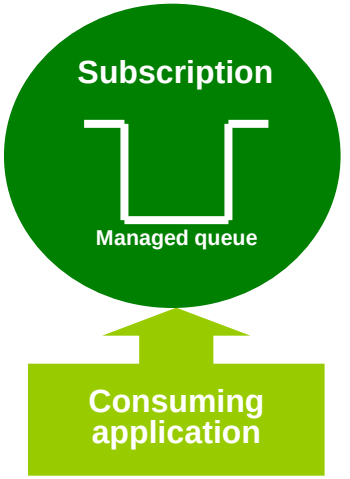
- **A subscription maps a topic to a queue. The queue relationship is either explicit or implicit…**
- Managed **subscription queue**
  - The subscription automatically creates and deletes a queue for the use of queuing any matching publications.
- Unmanaged **subscription queue**
  - When the subscription 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

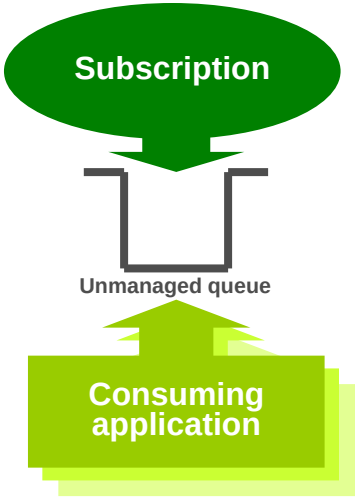
Via the *subscription*

- **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 pub/sub status feedback**



Via the *queue*

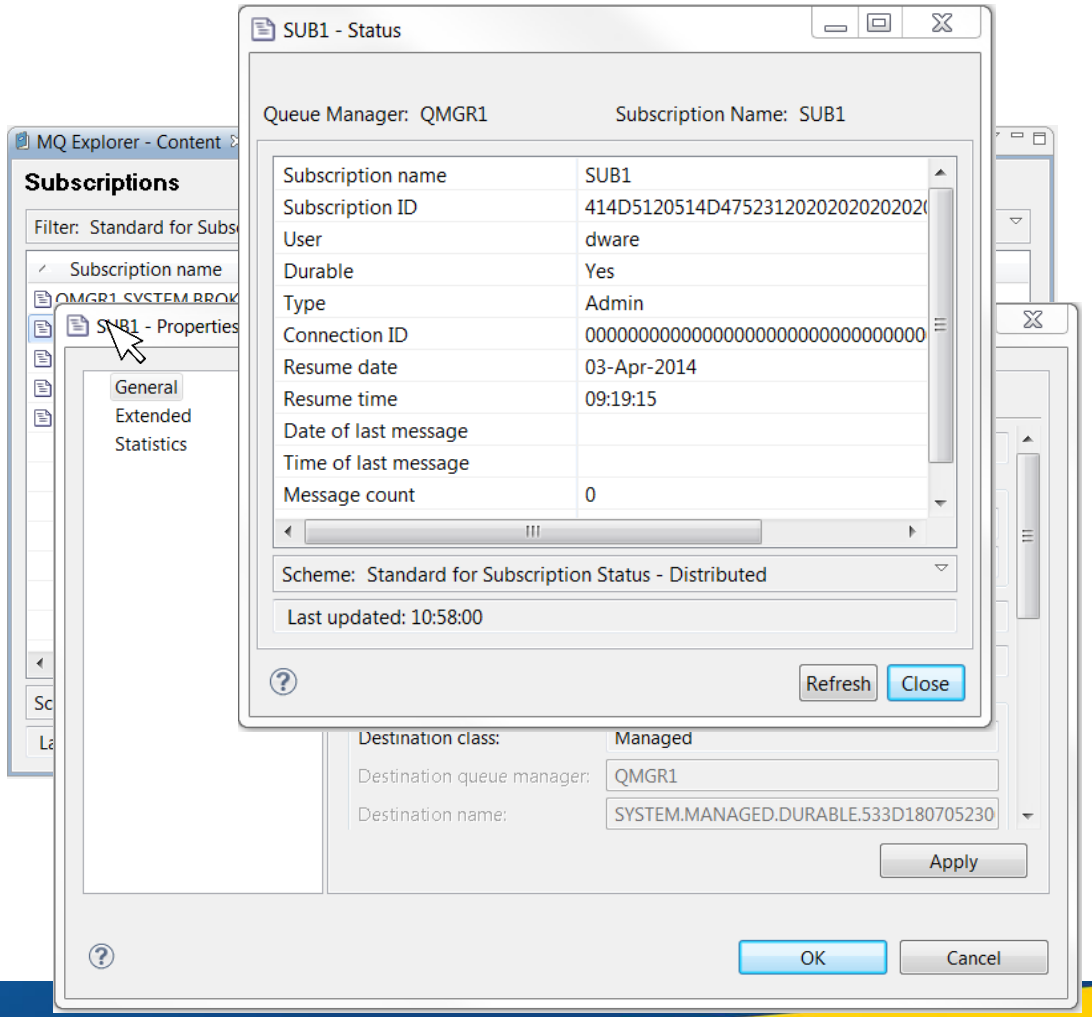
- **An application opens the queue associated with the subscription**
  - *This is really a point-to-point application*
- **Only works with unmanaged subscription queues**
- **Allows more freedom in what can be done**
  - For example, multiple concurrent consuming applications possible from any API



Managing subscriptions

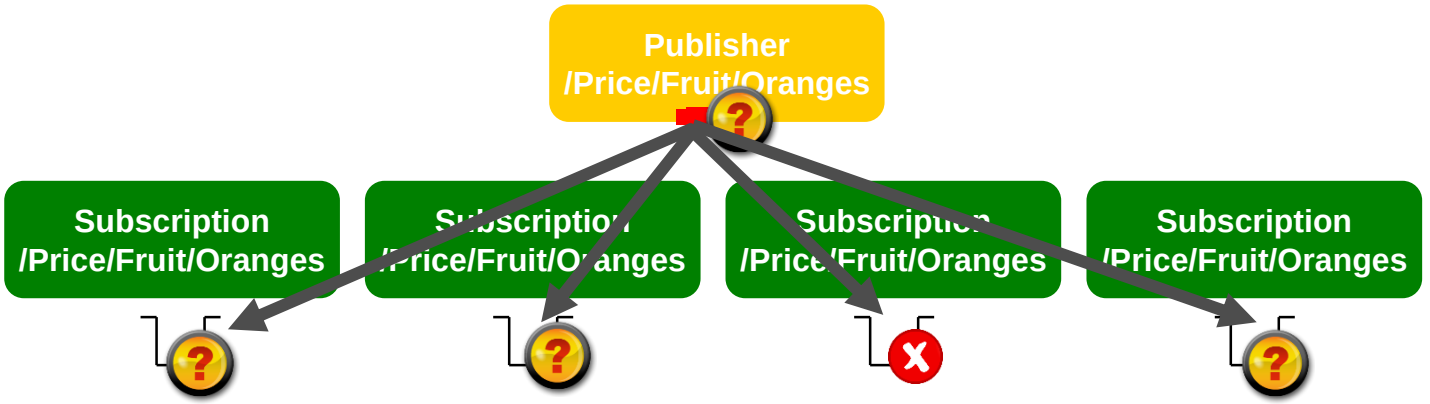
- Displaying subscriptions
  - This shows the subscriptions on a queue manager

```
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager QMGR1. DISPLAY
DISPLAY SUB(SUB1)
     2 : DISPLAY SUB(SUB1)
AMQ8096: WebSphere MQ subscription inquired.
   SUBID(414D5120514D4752312020202020202007183D5320002306)
   SUB(SUB1)                               TOPICSTR(/Price/Fruit/Apples)
   TOPICOBJ( )
   DEST(SYSTEM.MANAGED.DURABLE.533D180705230020)
   DESTQMGR(QMGR1)                         PUBAPPID( )
   SELECTOR( )                             SELTYPE(NONE)
   USERDATA( )
   PUBACCT(16010515000000DEA960DF651724E4B97C192FE803000000000B)
   DESTCORL(414D5120514D4752312020202020202007183D5320002306)
   DESTCLAS(MANAGED)                       DURABLE(YES)
   EXPIRY(UNLIMITED)                       PSPROP(MSGPROP)
   PUBPRTY(ASPUB)                          REQONLY(NO)
   SUBSCOPE(ALL)                           SUBLEVEL(1)
   SUBTYPE(ADMIN)                          VARUSER(ANY)
   WSCHEMA(TOPIC)                          SUBUSER(xxxx)
   CRDATE(2014-04-03)                      CRTIME(09:19:15)
   ALTDATE(2014-04-03)                     ALTTIME(09:19:15)

DISPLAY SBSTATUS(SUB1)
     3 : DISPLAY SBSTATUS(SUB1)
AMQ8099: WebSphere MQ subscription status inquired.
   SUB(SUB1)
   SUBID(414D5120514D4752312020202020202007183D5320002306)
   SUBUSER(xxxx)                           RESMDATE(2014-04-03)
   RESMTIME(09:19:15)                      LMSGDATE( )
   LMSGTIME( )
   ACTCONN(000000000000000000000000000000000000000000000000)
   DURABLE(YES)                            MCASTREL( , )
   NUMMSGS(0)                              SUBTYPE(ADMIN)
   TOPICSTR(/Price/Fruit/Apples)

DISPLAY QLOCAL(SYSTEM.MANAGED.DURABLE.533D180705230020)
```



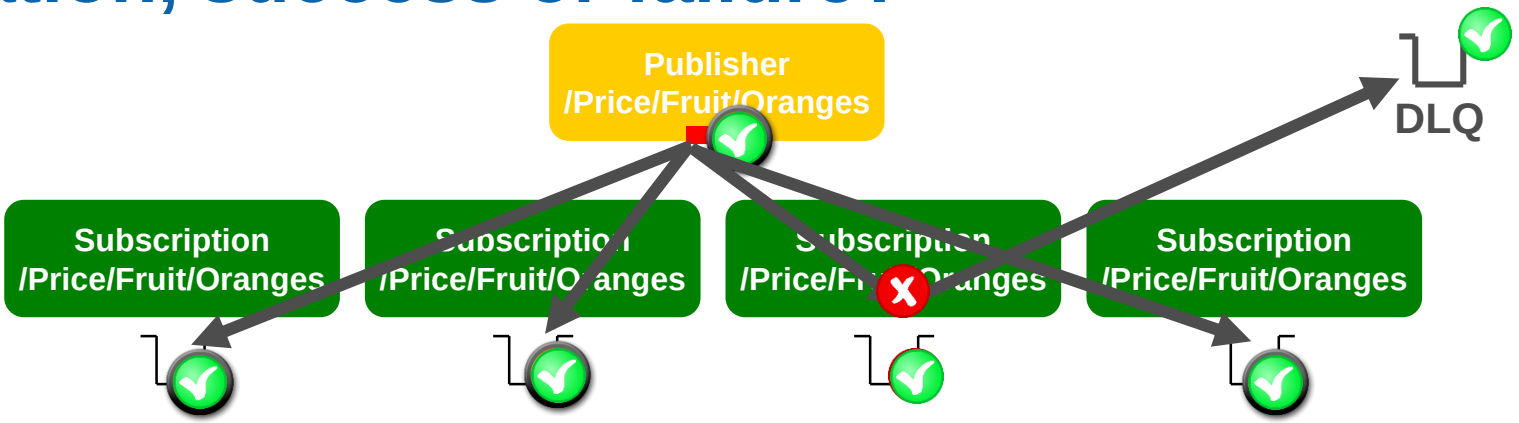
Publishing

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 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?*
- *What if the subscriptions are non-durable, or the publication is non-persistent?*

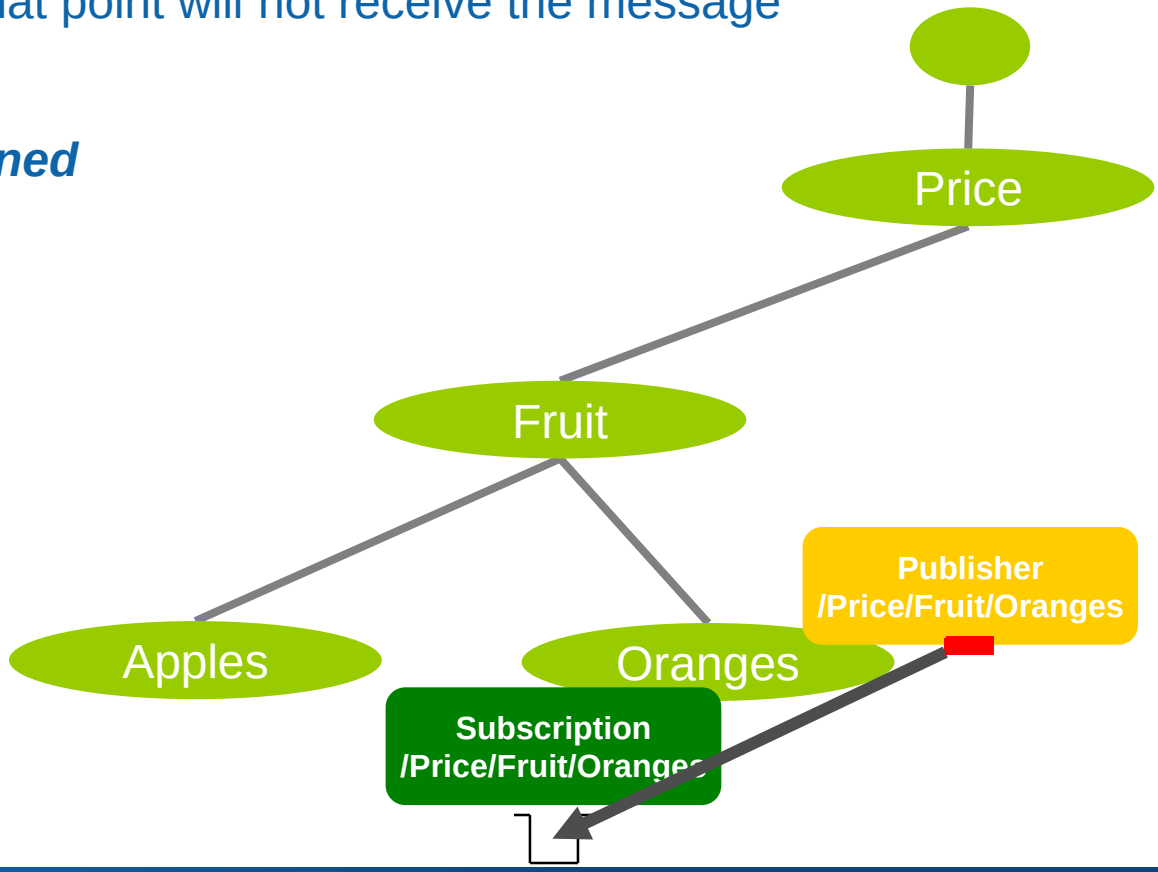
- Controlled at the topic level
  - Persistent Message Delivery (**PMSGDLV**) and Non-persistent Message Delivery (**NPMSGDLV**).
    - **ALL, ALLDUR, ALLAVAIL**

- 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!*

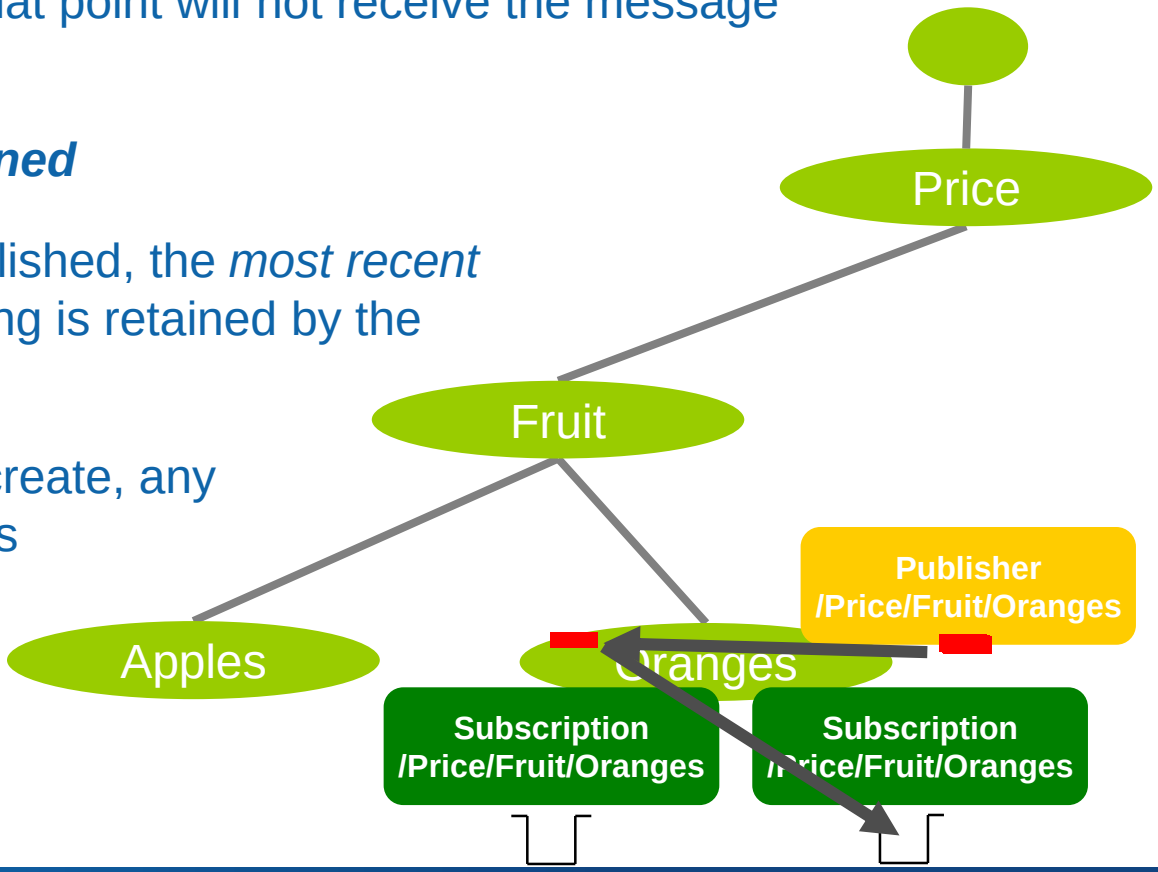
Retained publications

- 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

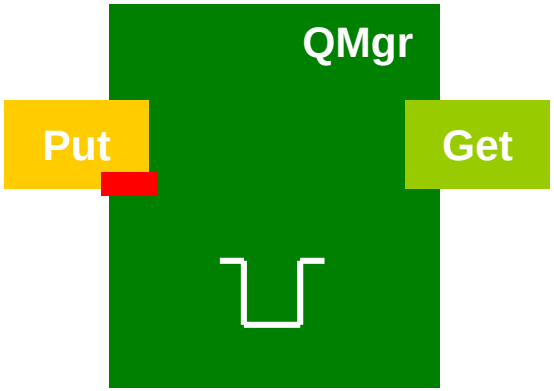
- When a message is published to a topic string, it is delivered to each matching subscription registered at that time.
- Subscriptions created after that point will not receive the message only newly published ones.
- Unless publications are ***retained***
- Every time a message is published, the *most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



Topologies

Distributed queuing

- We all know how point-to-point works…

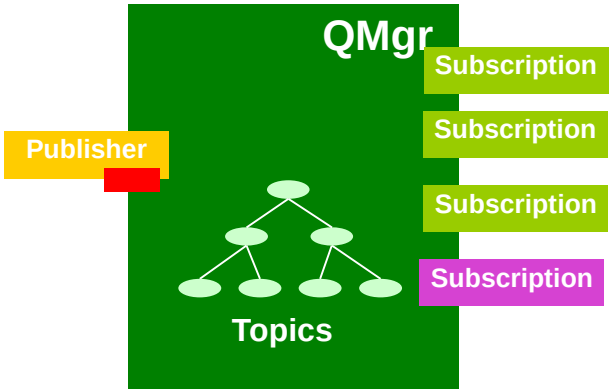


Distributed queuing

- We all know how point-to-point works
- Even across multiple queue managers

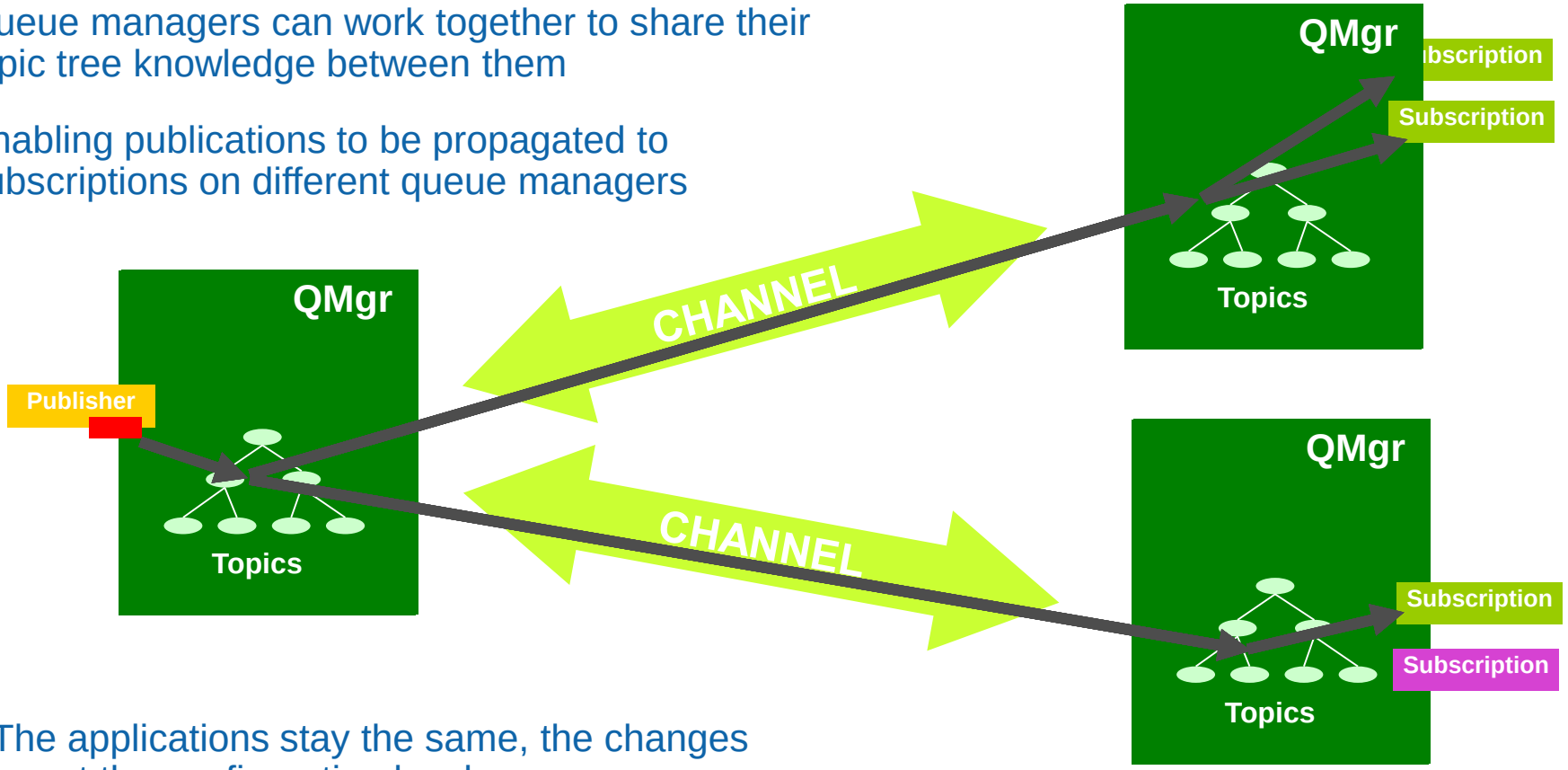
Distributed publish/subscribe

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



Distributed publish/subscribe

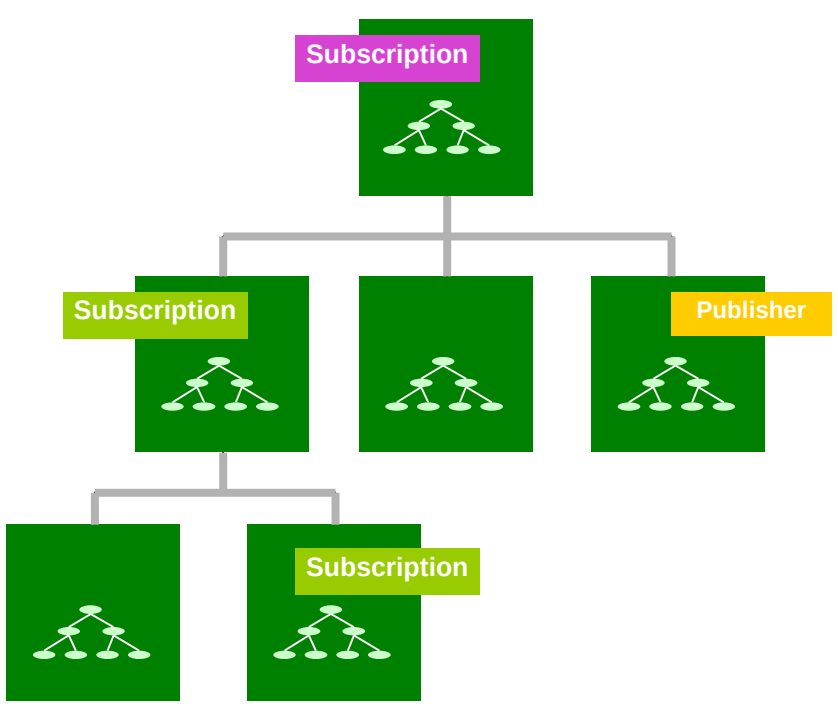
- And we know how everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them
- Enabling publications to be propagated to subscriptions on different queue managers

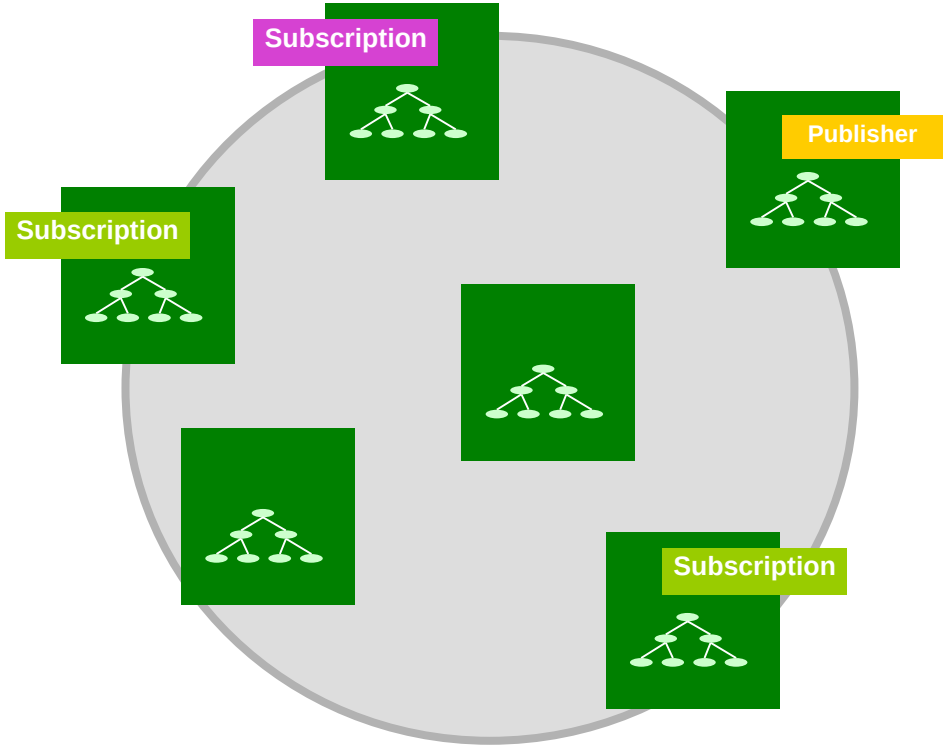


- The applications stay the same, the changes are at the configuration level.

Distributed publish/subscribe topologies

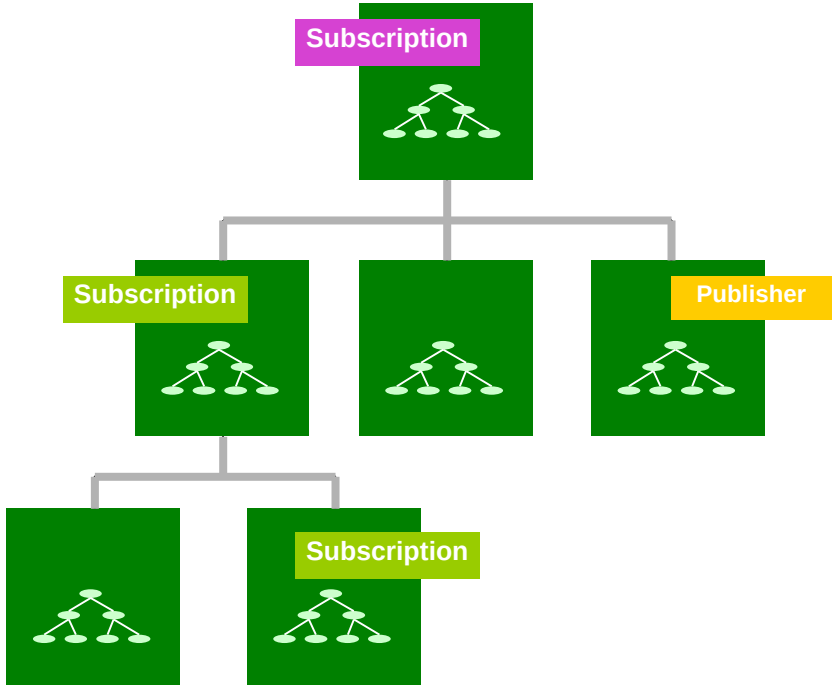
- Publish/subscribe topologies can either be created as a defined ***hierarchy*** or more dynamically as a ***cluster***

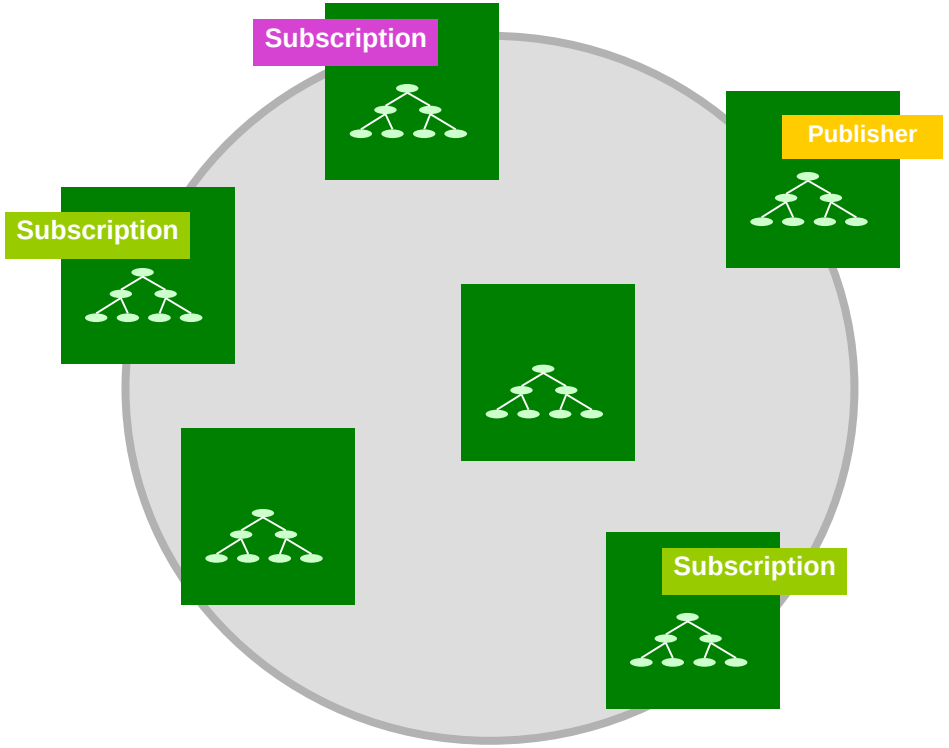




Distributed publish/subscribe topologies

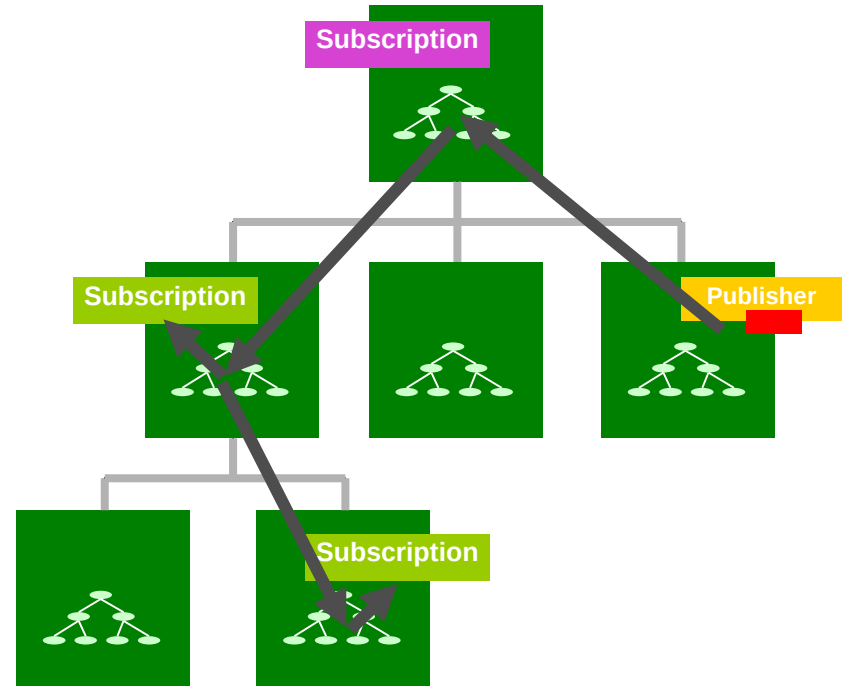
- A hierarchy manually defines the relationship between each queue manager.
- Messages are flowed via those relationships.





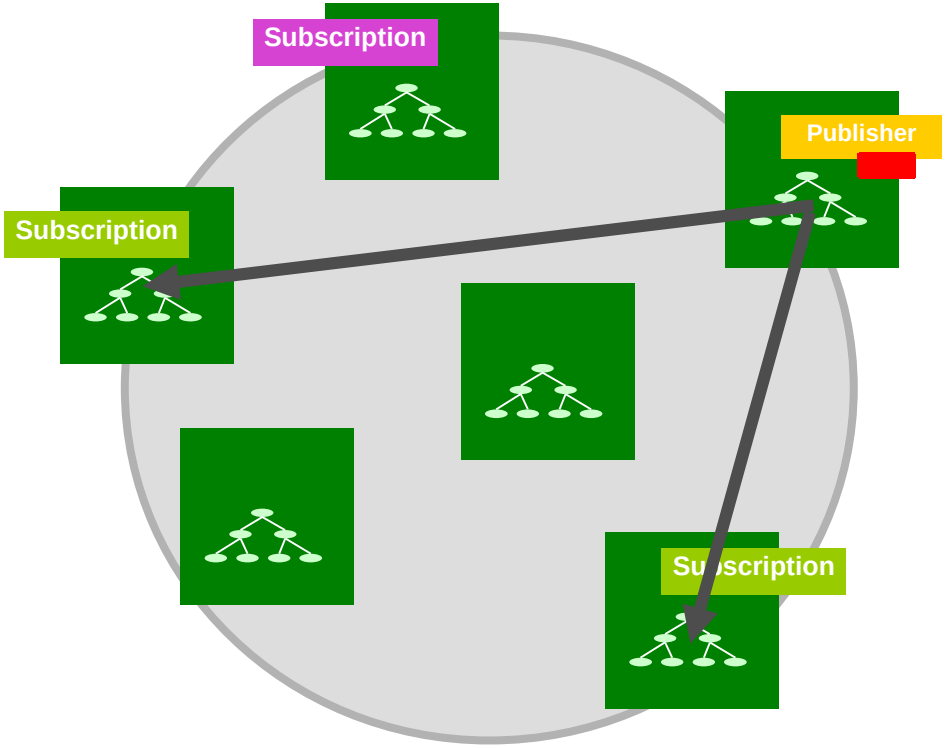
Distributed publish/subscribe topologies

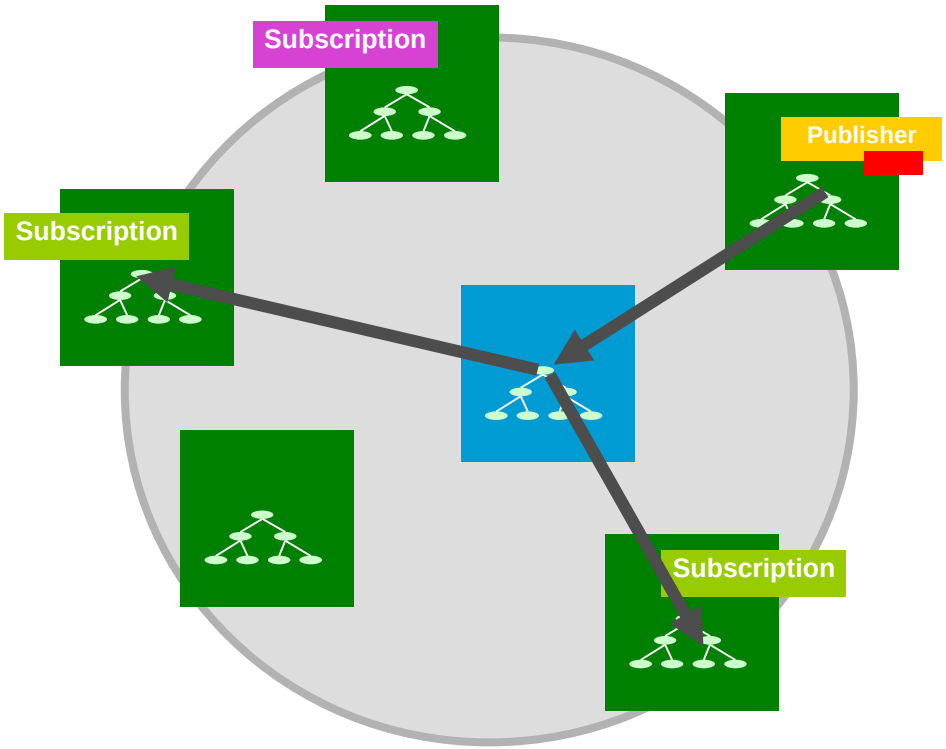
- A hierarchy manually defines the relationship between each queue manager.
- Messages are flowed via those relationships.



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 WebSphere MQ
- Administration of publish/subscribe
- Management of publish/subscribe
- Subscriptions and publications
- Topologies

Questions & Answers

Legal Disclaimer