

Recording available at:
<http://www.slideshare.net/DavidWare1/ame-2271-mq-publish-subscribe-pdf>

An introduction to MQ Publish/Subscribe

David Ware

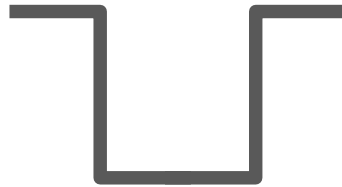
*Lead Architect, IBM MQ Distributed
dware@uk.ibm.com*

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

What is publish/subscribe?

How does it compare to point-to-point?

Message producer



Message consumer

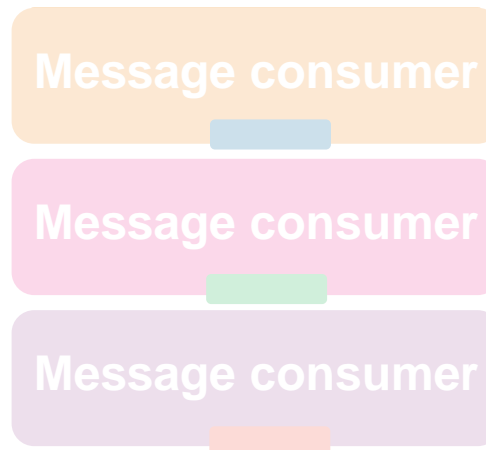
Message consumer

Message consumer

point-to-point

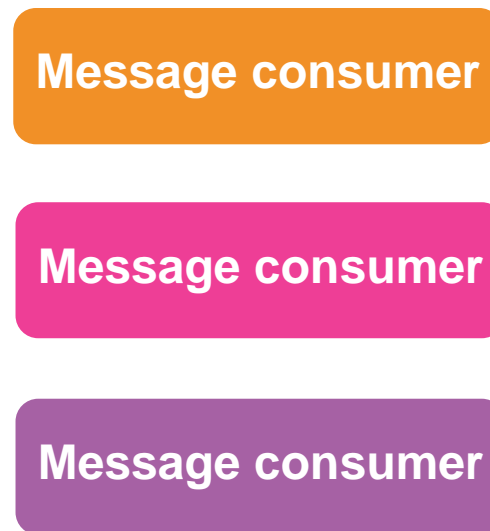
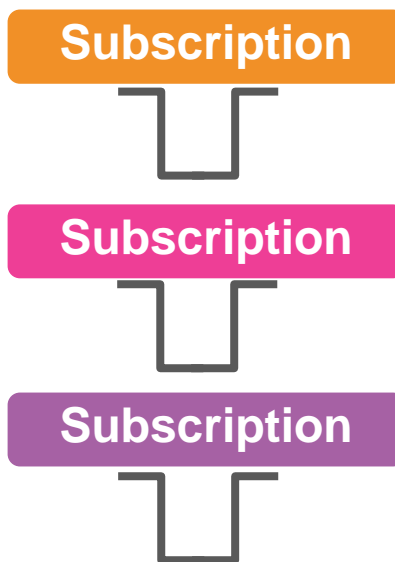
How does it compare to point-to-point?

Message producer



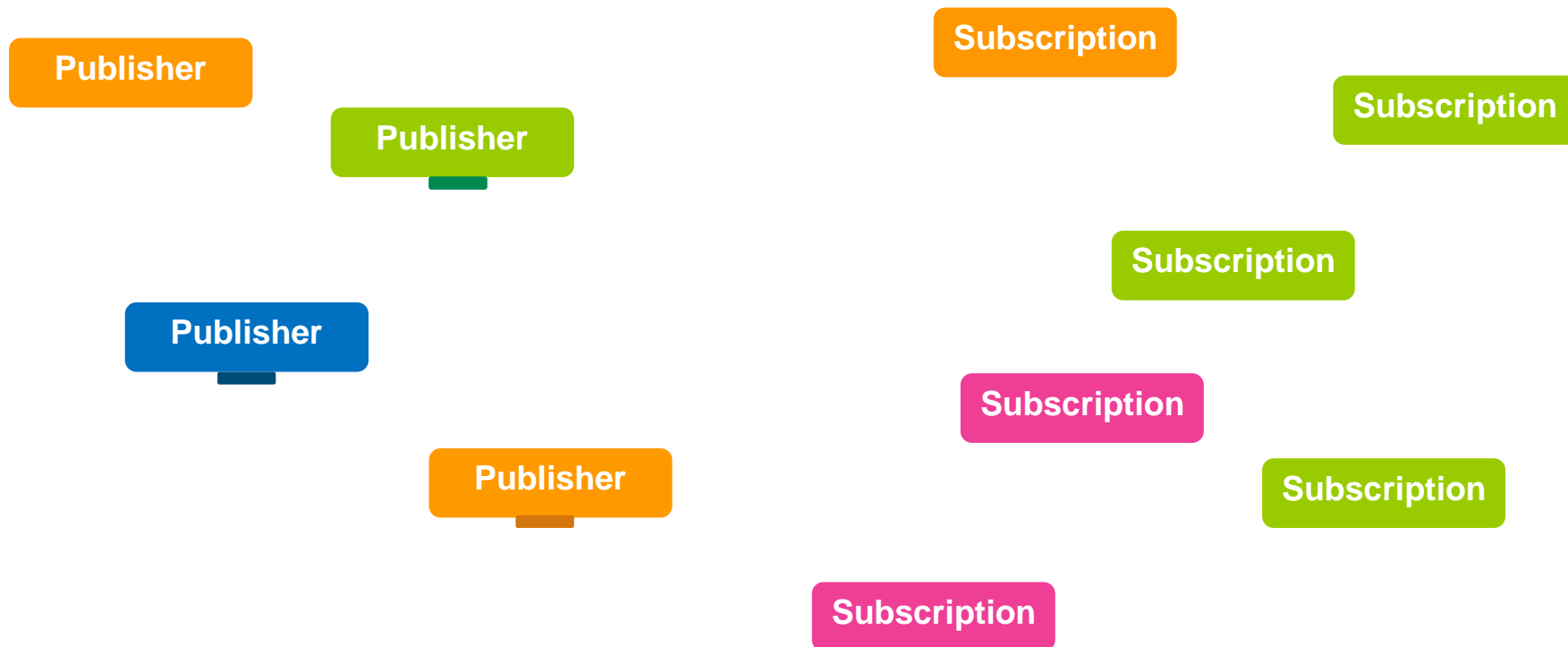
point-to-point

publish/subscribe



But which subscriptions receive the messages?

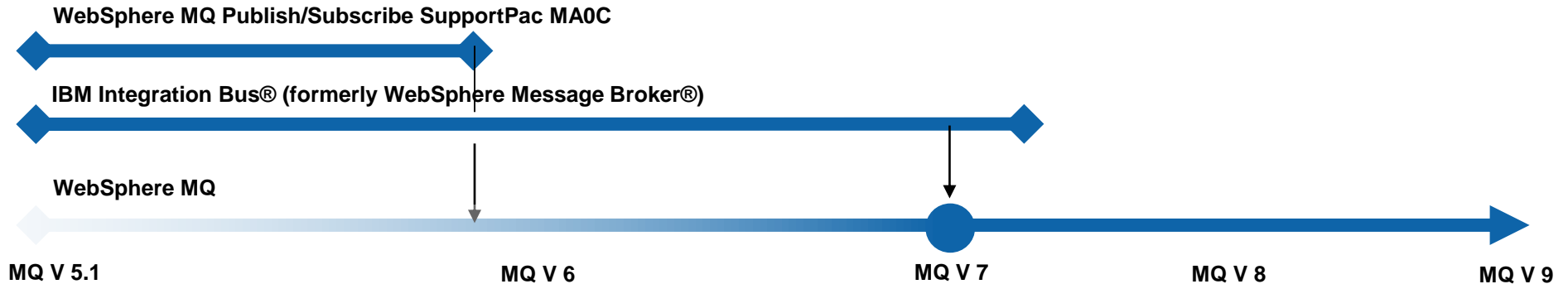
- Publishing and subscribing is based on **'topics'**
 - **Green** messages go to **green** subscribers
 - **Orange** messages go to **orange** subscribers
 - But nobody wants a **blue** message!



Publish/Subscribe in IBM MQ

WebSphere MQ's publish/subscribe over the years

Publish/Subscribe brokers



WebSphere MQ Publish/Subscribe APIs

Command message based publish/subscribe API

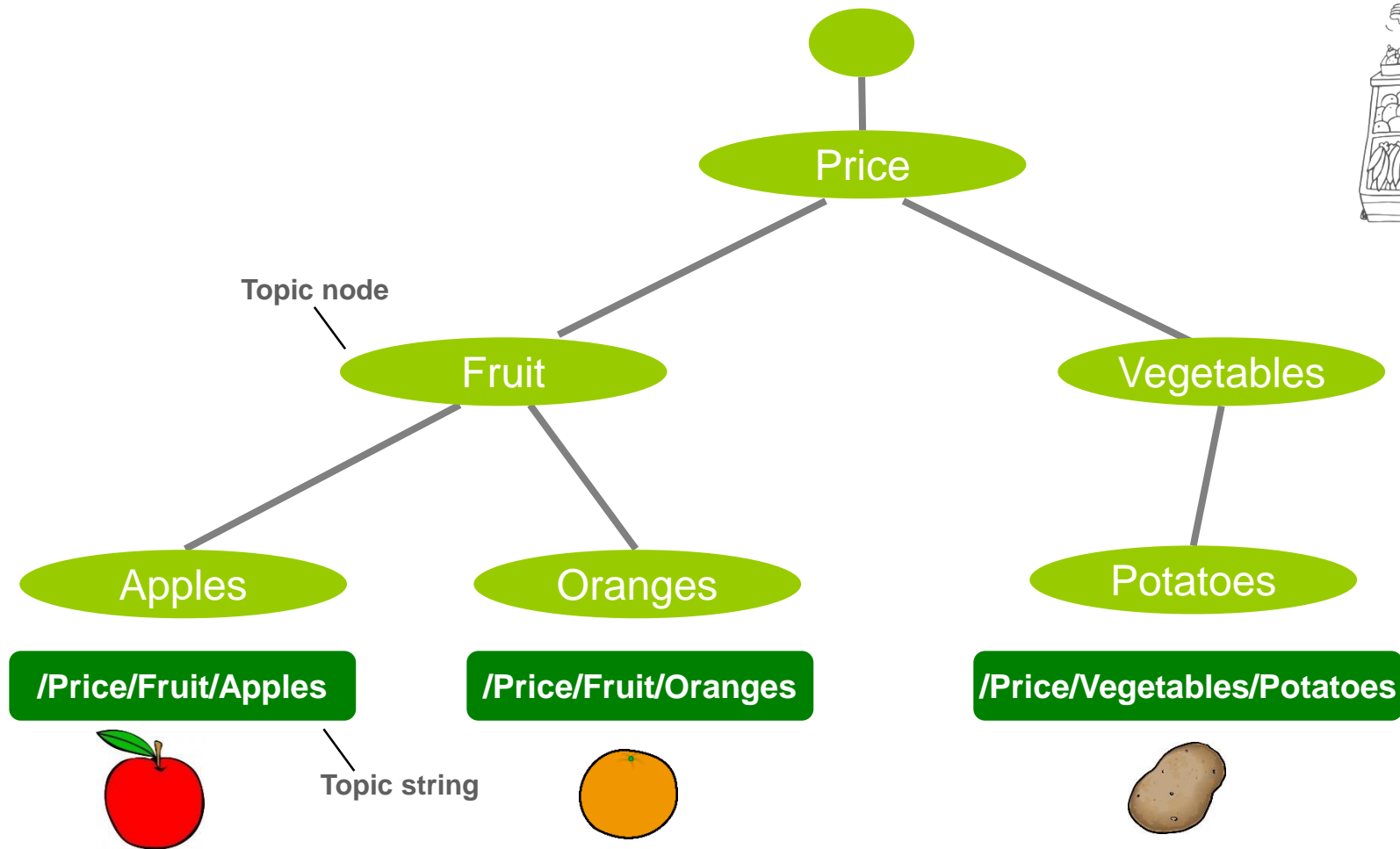
JMS publish/subscribe API

XMS publish/subscribe API

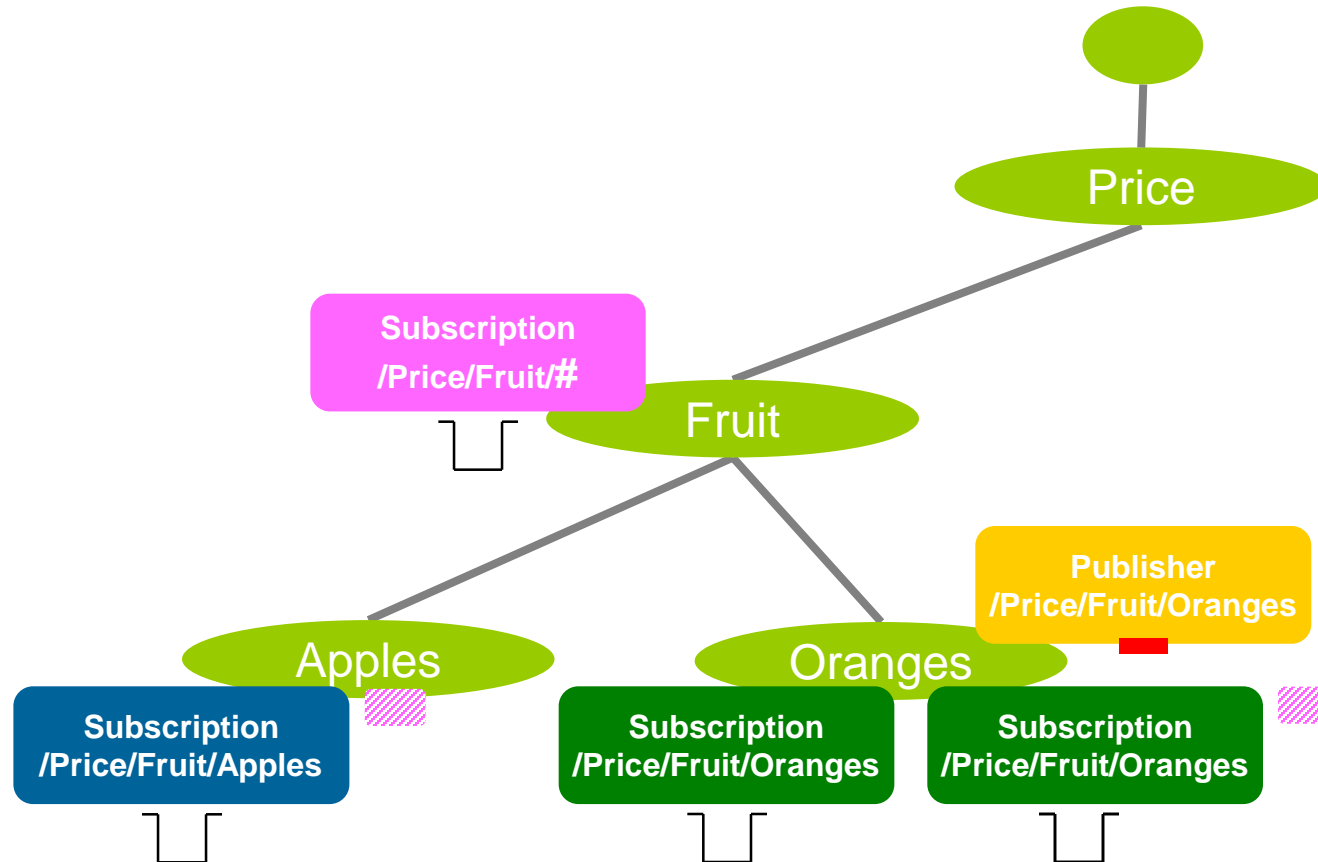
Native MQI publish/subscribe API

Topics

It's all about the *topic tree*



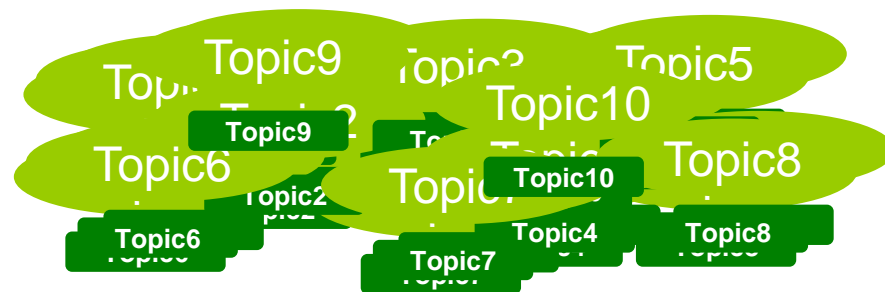
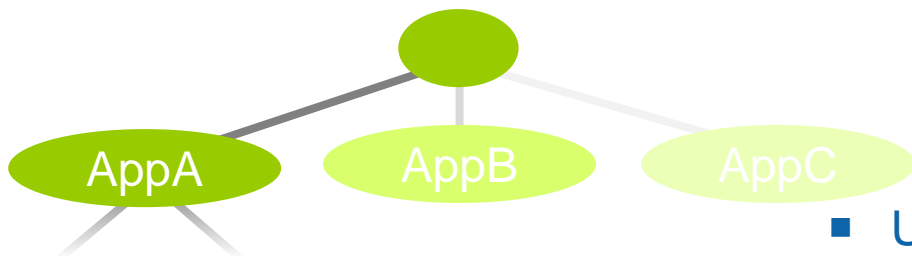
Matching publications to subscriptions



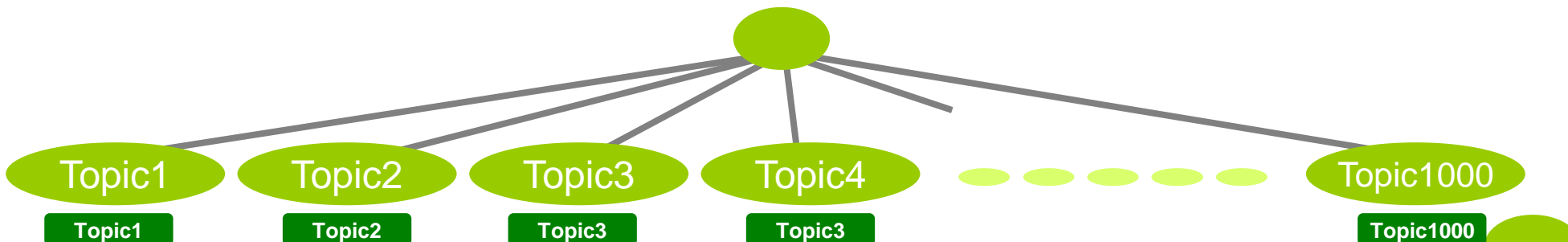
- Subscriptions are attached to matching nodes in the topic tree
- Publications identify the relevant topic node
- A copy of the publication is delivered to the queue identified by *each* matching subscription
- **Wildcarding** subscriptions at the topic node level can receive messages from multiple topic strings

Designing your topic tree structure

- Make it extendable.



- Understand a rapidly changing set of topic strings.

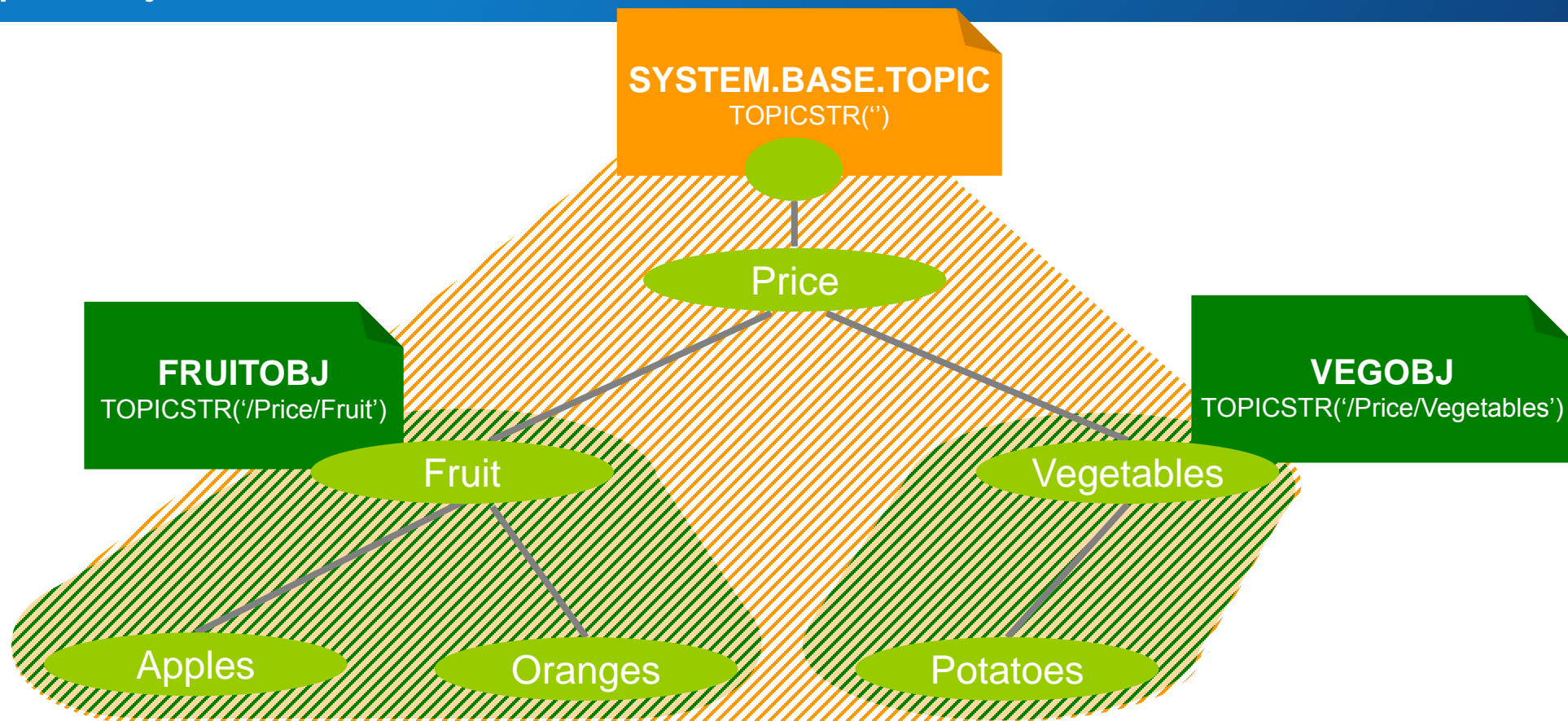


- Avoid excessively **wide** or **deep** dynamic topic trees.

- Use structure where appropriate.
- Limit it to *subscribable* content.

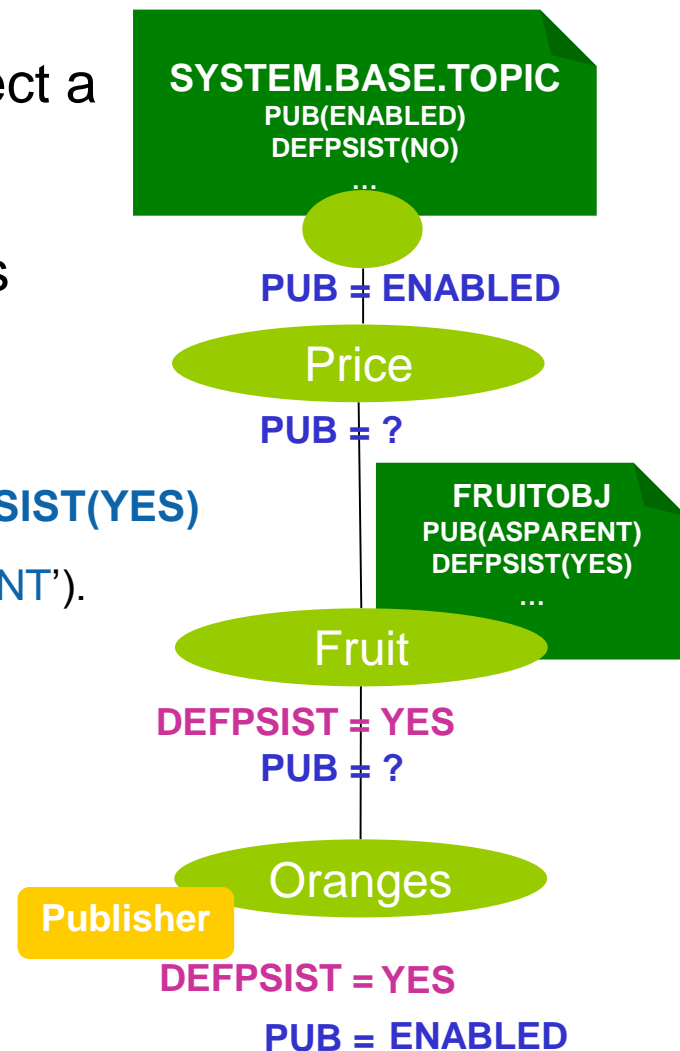


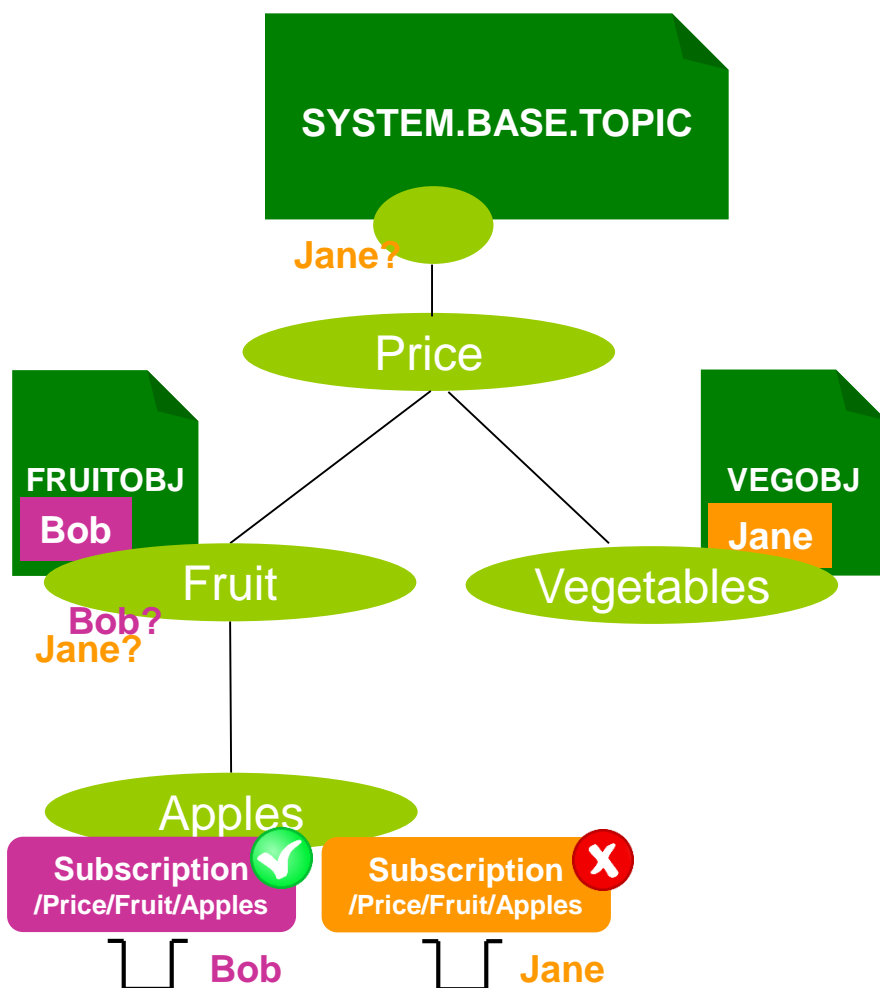
configuration



- 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.

- 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?**





- 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.

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)

DEFPERSIST(YES)

DEFPRESP(ASPARENT)

ALTTIME(08.44.48)

NPMSGDLV(ASPARENT)

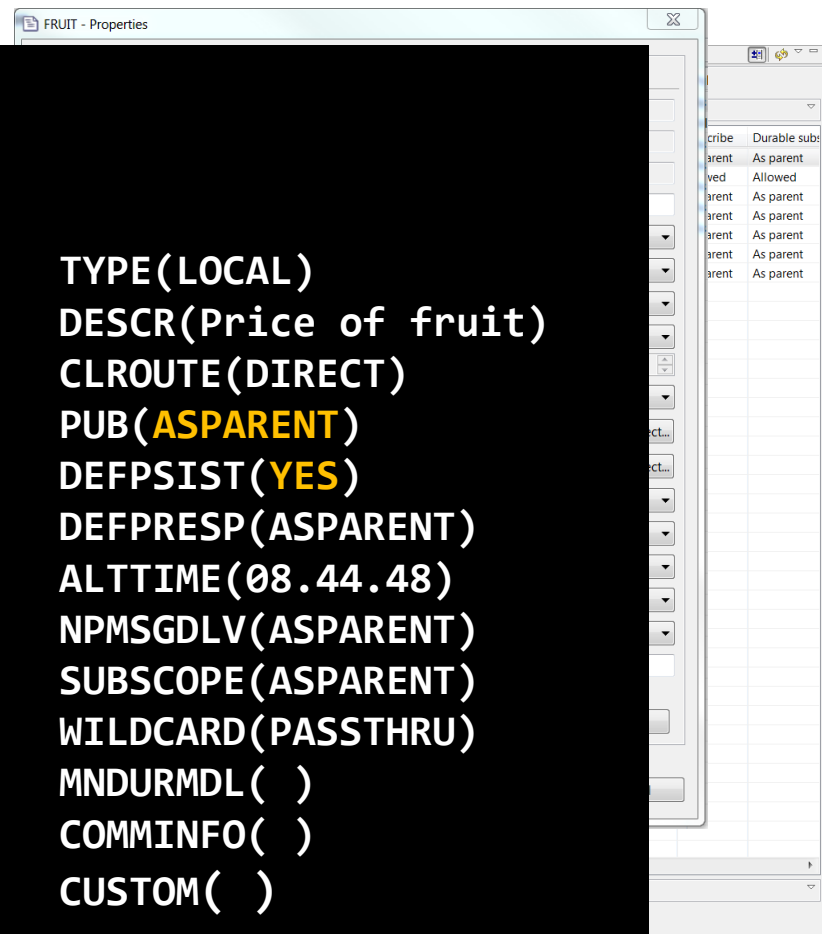
SUBSCOPE(ASPARENT)

WILDCARD(PASSTHRU)

MNDURMDL()

COMMINFO()

CUSTOM()



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

```
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)
```

```
DEFPRTY(0)
```

```
DURSUB(YES)
```

```
SUB(ENABLED)
```

```
NPMSGDLV(ALLAVAIL)
```

```
MCAST(DISABLED)
```

```
SUBCOUNT(1)
```

```
SUBSCOPE(ALL)
```

```
DEFPSIST(YES)
```

```
DEFPRESP(SYNC)
```

```
PUB(ENABLED)
```

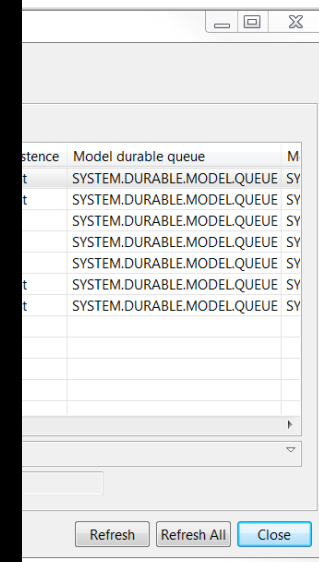
```
PMSGDLV(ALLDUR)
```

```
RETAINED(NO)
```

```
PUBCOUNT(0)
```

```
PUBSCOPE(ALL)
```

```
USEDLQ(YES)
```



Applications

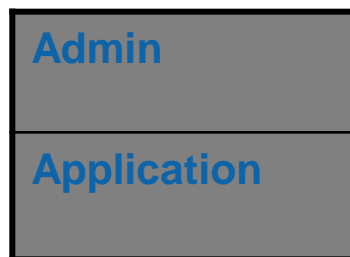
- *When creating subscriptions or opening topics to publish on, do I use a topic string or a topic object?*
 - ▶ A **topic string**. No, a **topic object**. No, **both**. Actually, er, **any of them!**
- *So which should I use?*
 - ▶ Using the topic string is probably the easiest, it's closest to what 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(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 are actually being used*

Subscriptions

- 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 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 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 queue management

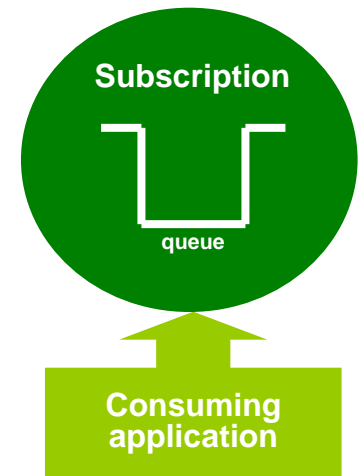
- **A subscription maps a topic to a queue. The queue relationship is either explicit or implicit...**
- **Managed subscription queue**
 - ▶ The subscription automatically creates and deletes a queue for the use of queuing any matching publications.
- **Unmanaged subscription queue**
 - ▶ When the subscription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

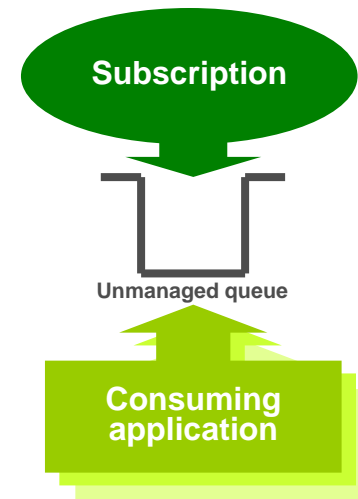
Via the *subscription*

- **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

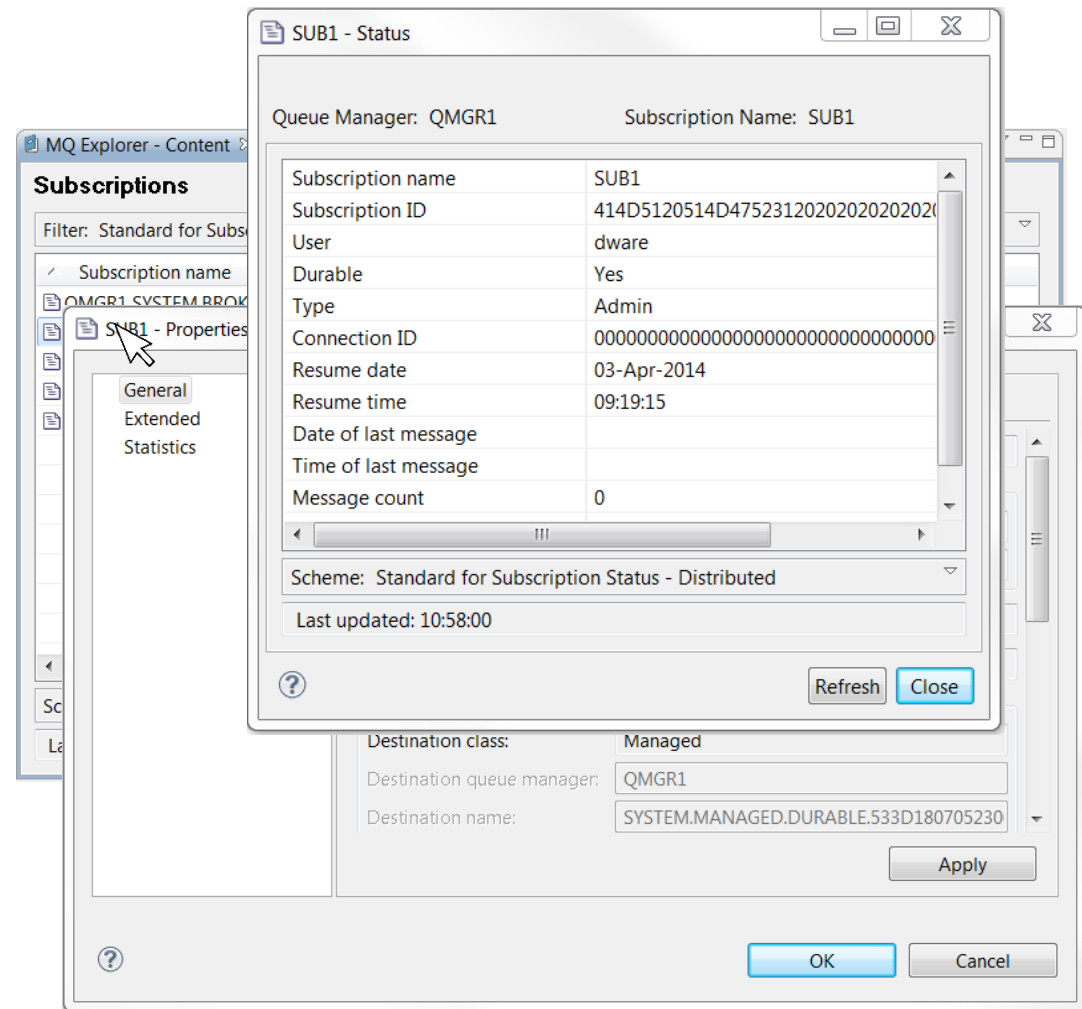
- **Displaying subscriptions**
 - ▶ This shows the subscriptions on a queue manager

```
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(414D5120514D47523120202020202007183D5320002306)
SUB(SUB1) TOPICSTR(/Price/Fruit/Apples)
TOPICOBJ( )
DEST(SYSTEM.MANAGED.DURABLE.533D180705230020)
DESTQMGR(QMGR1) PUBAPPID( )
SELECTOR( ) SELTYPE(NONE)
USERDATA( )
PUBACCT(16010515000000DEA960DF651724E4B97C192FE803000000000B)
DESTCORL(414D5120514D475231202020202007183D5320002306)
DESTCLAS(MANAGED) DURABLE(YES)
EXPIRY(UNLIMITED) PSPROP(MSGPROP)
PUBPRTY(AS PUB) REQONLY(NO)
SUBSCOPE(ALL) SUBLEVEL(1)
SUBTYPE(ADMIN) VARUSER(ANY)
WSHEMA(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(414D5120514D475231202020202007183D5320002306)
SUBUSER(XXXX) RESMDATE(2014-04-03)
RESMIME(09:19:15) LMSGDATE( )
LMSGTIME( )
ACTCONN(0000000000000000000000000000000000000000000000000000000)
DURABLE(YES) MCASTREL( , )
NUMMSGS(0) SUBTYPE(ADMIN)
TOPICSTR(/Price/Fruit/Apples)

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

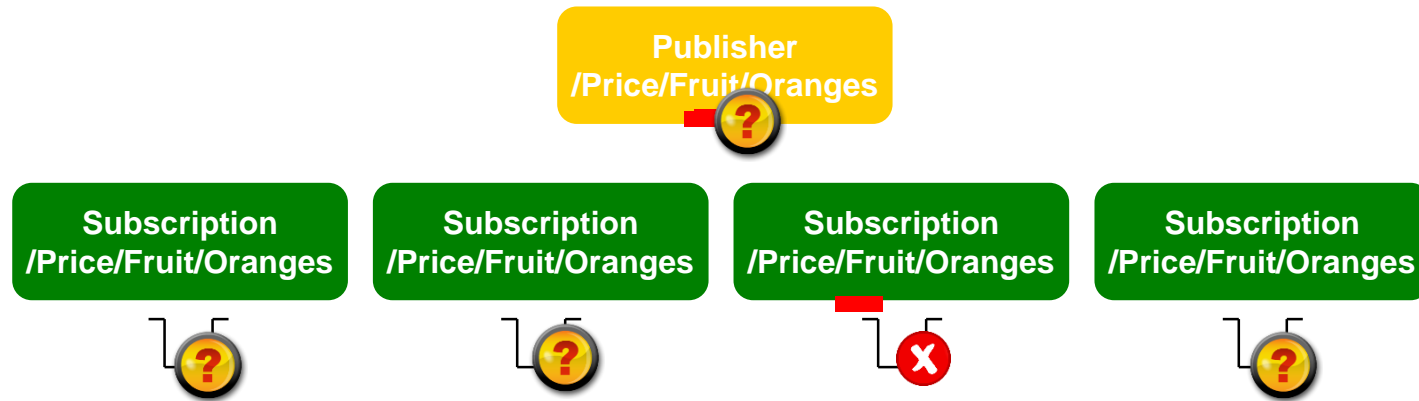


Publishing



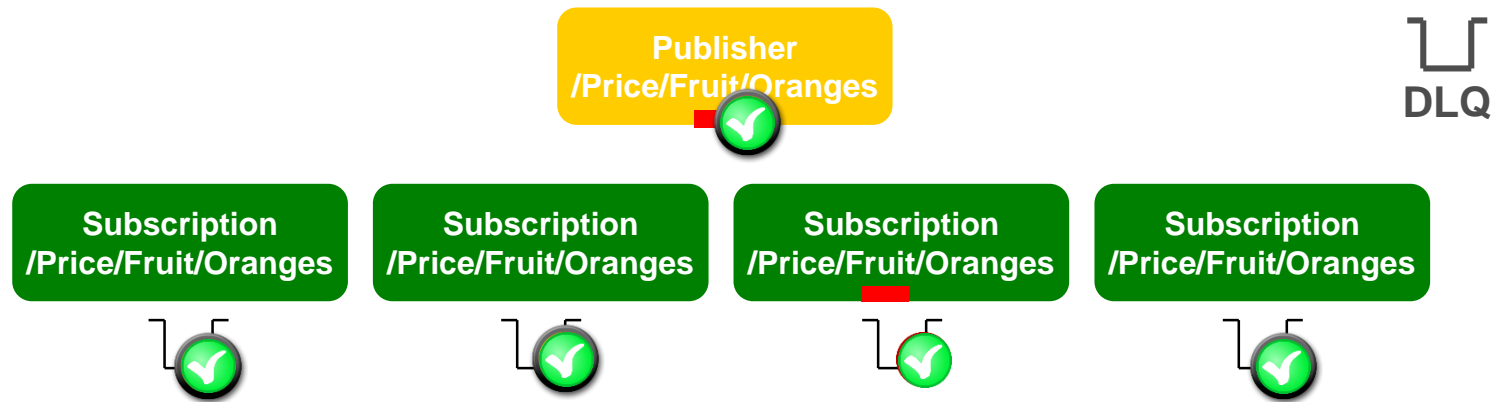
- Point-to-point is nice and simple:
 - ▶ Did the message get onto the queue?
 - ▶ Was it persistent and transacted?

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?***

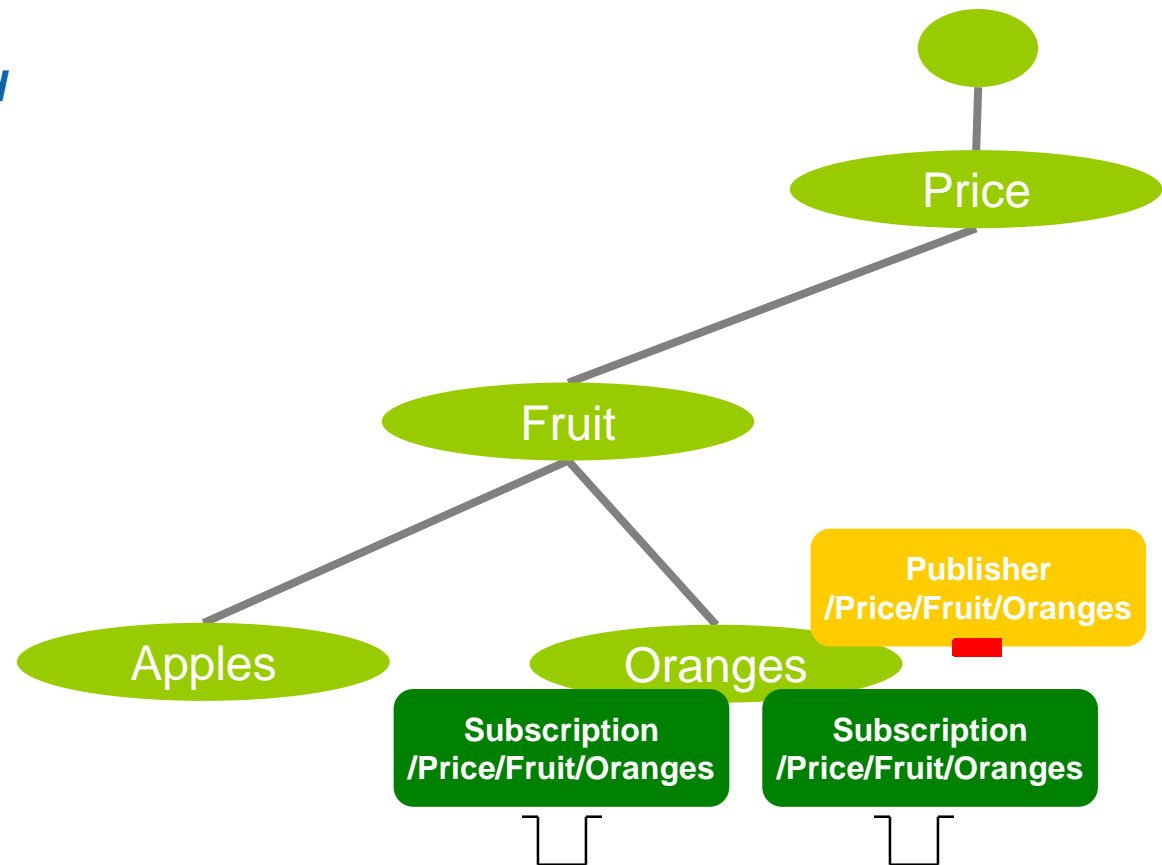
Publication, success or failure?



- *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 (NPMSGDLV)** : **ALL, ALLDUR, ALLAVAIL**
- *Don't forget that being able to DLQ a publication is still counted as a **success!***
 - ▶ **USEDQLQ** on the topic to fine tune this behaviour.
- *And finally, **remember** – when there are no subscriptions, no-one gets it. That's still a **successful publish!***

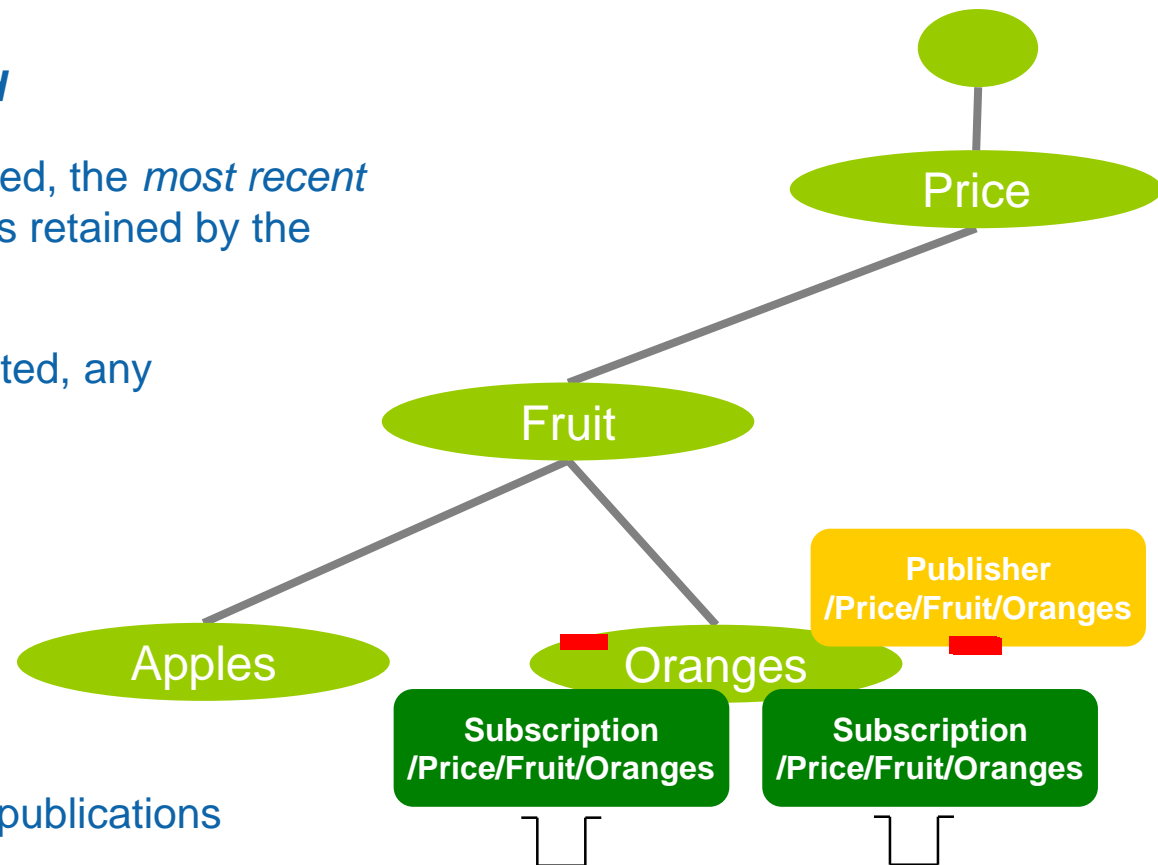
Retained publications

- 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*



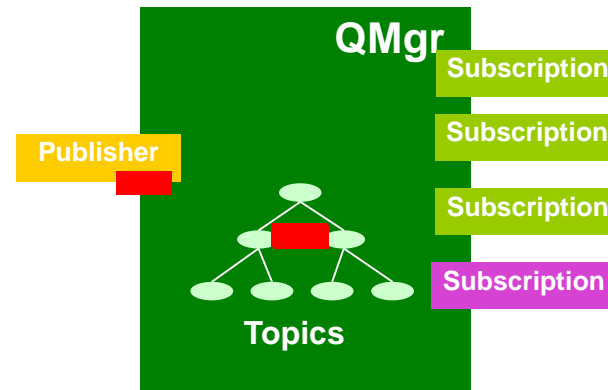
Retained publications

- 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



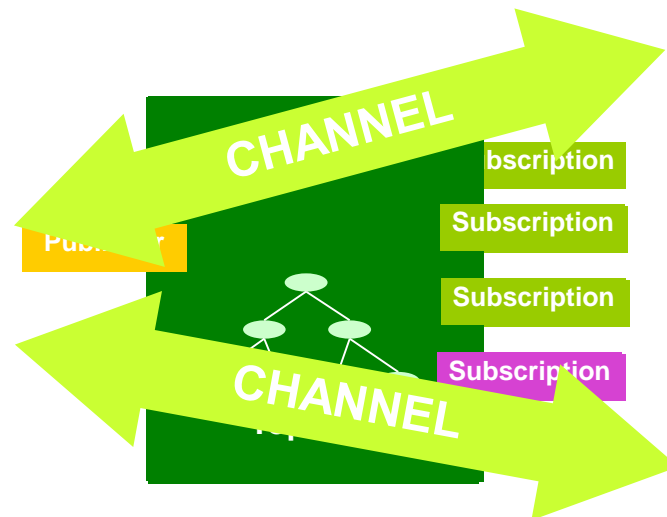
Topologies

- Everything revolves around the topic tree, dynamically built up in a queue manager



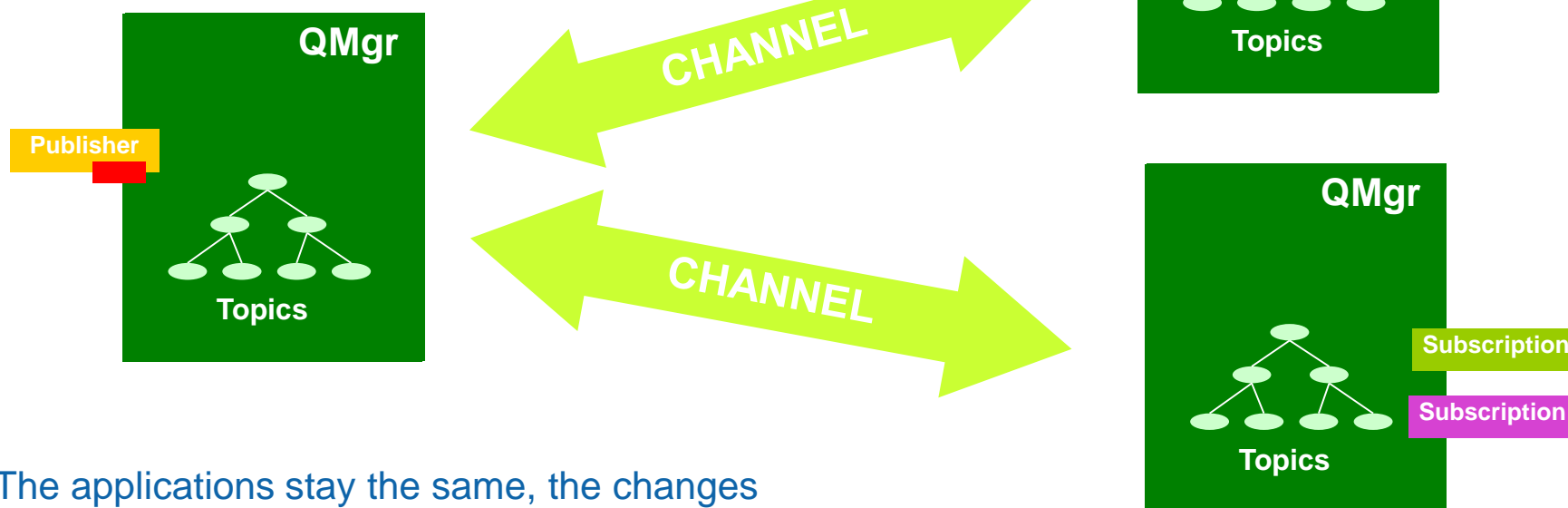
Distributed publish/subscribe

- 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

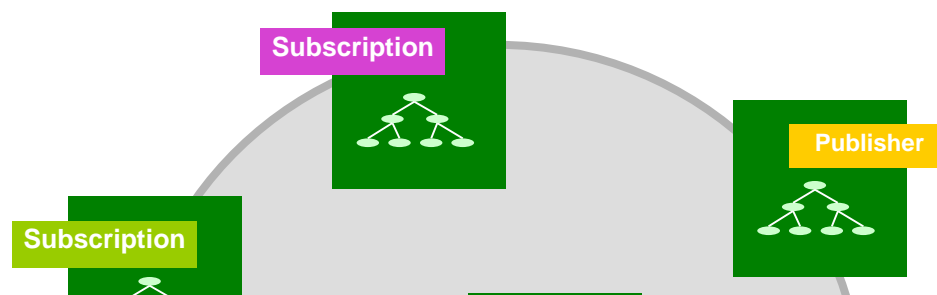
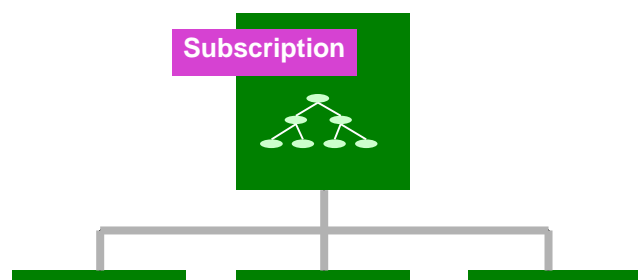
- Everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them
- Enabling publications to be propagated to subscriptions on different queue managers



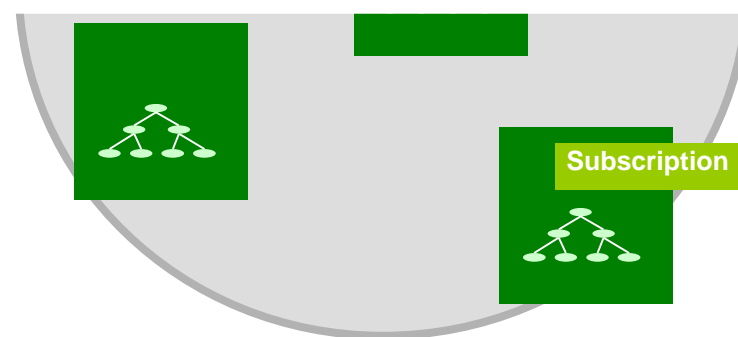
- The applications stay the same, the changes are at the configuration level.

Distributed publish/subscribe topologies

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



Come to the pub/sub network session for much more on topologies...



System topics (MQ v9)

System topics – new for MQ V9 and the MQ Appliance

IBM MQ V9 (and the M2000 Appliance) has created its own branch of the topic tree

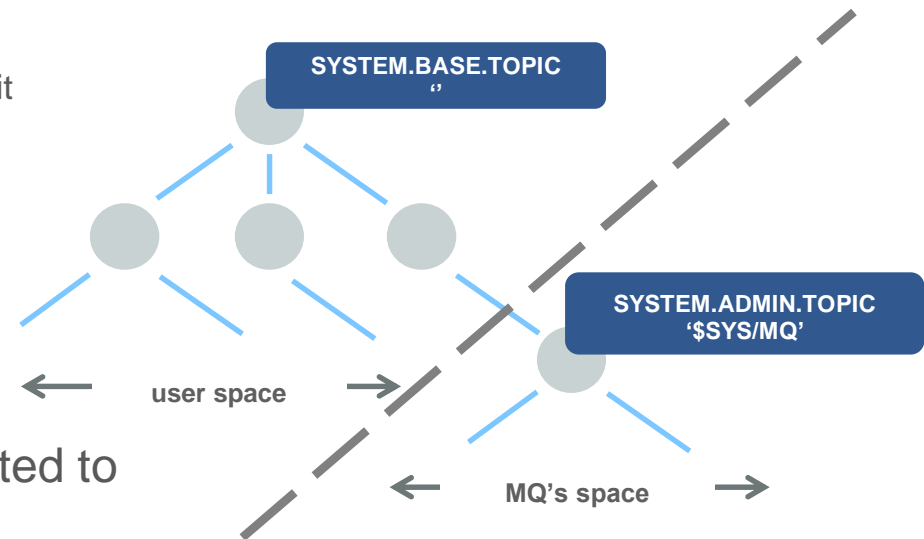
- “\$SYS/MQ”

This is used for publishing and subscribing to data from the queue manager

A new default object sits at the root of this branch, **SYSTEM.ADMIN.TOPIC**

It sits below **SYSTEM.BASE.TOPIC** but consider it a *complete new tree*

- No attributes are inherited from above
- No authorities are inherited from above
- The topic nodes are not matched by wildcards above it
 - E.g. Subscribing to ‘#’ will not match anything starting “\$SYS/MQ”

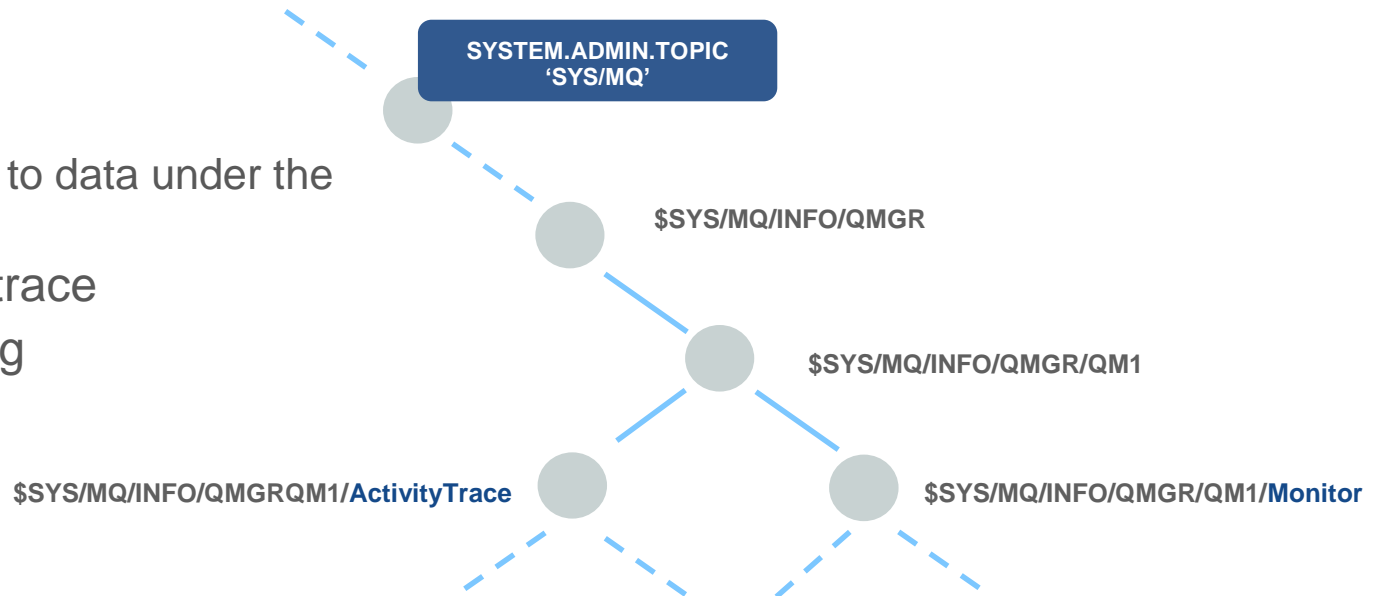


Out of the box, access to the branch is restricted to administrators only

- Access can be granted to other users through the definition of admin topics and access control at key points below \$SYS/MQ

MQ V9 publishes two types of data under the system topic branch

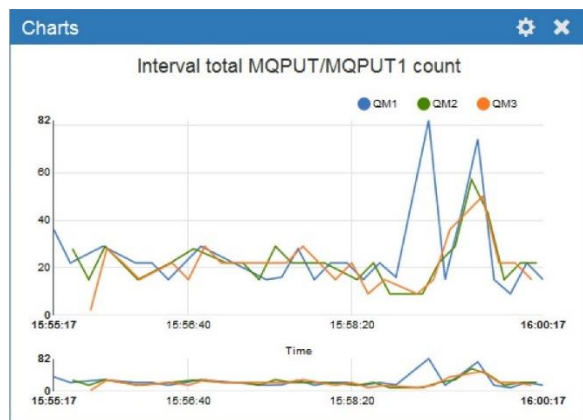
- Application activity trace
- Resource monitoring



Each queue manager *listens* for subscriptions to recognised topic strings and *only publishes when subscriptions exist*

Subscribe to MQ and system monitoring information

- Subscribe for a regular feed of performance monitoring information
- Familiar **queue manager** wide and **queue** level statistics
- Plus new **CPU, memory** and **disk** usage
 - For example, disk reads/writes, disk latency
- Enables integration into multiple monitoring solutions



This capability already underpins the charting in the MQ Appliance WebUI



Simplifies integration into off the shelf tooling

```
$ amqsrua -m V9000_A
CPU : Platform central processing units
DISK : Platform persistent data stores
STATMQI : API usage statistics
STATQ : API per-queue usage statistics
Enter Class selection
CPU
SystemSummary : CPU performance - platform wide
QMGrSummary : CPU performance - running queue manager
Enter Type selection
SystemSummary
Publication received PutDate:20160411 PutTime:10465573
User CPU time percentage 0.01%
System CPU time percentage 1.30%
CPU load - one minute average 8.00
CPU load - five minute average 7.50
CPU load - fifteen minute average 7.30
RAM free percentage 2.02%
RAM total bytes 8192MB
Publication received PutDate:20160411 PutTime:10466573
User CPU time percentage 0.01%
System CPU time percentage 1.30%
...
```

Sample tool demonstrates the capabilities and coding

Subscribe to Application Activity Trace

- See exactly what applications are doing with a live stream of data on their API calls
- Useful for tracking application usage and diagnosing application errors
- Subscribe to an application's name, a channel being used or a specific connection
- Sample tool for dynamic monitoring and formatting of trace

```
$ amqspout QUEUE1 QMGR1
Sample AMQSPUTO start
target queue is Q1
Hello
World
Sample AMQSPUTO end
```



```
$ amqsact -m QMGR1 -a amqspout -w 60
Subscribing to the activity trace topic:
'$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/AppName/amqspout'

MonitoringType: MQI Activity Trace
...
QueueManager: 'QMGR1'
ApplicationName: 'amqspout'
Application Type: MQAT_UNIX
...
=====
=
Tid Date      Time      Operation      CompCode      MQRC      HObj
(ObjName)
-----
001 2016-04-14 09:56:53  MQXF_CONNX    MQCC_OK       0000      -
001 2016-04-14 09:56:53  MQXF_OPEN     MQCC_OK       0000      2 (QUEUE1)
001 2016-04-14 09:56:53  MQXF_PUT      MQCC_OK       0000      2 (QUEUE1)
001 2016-04-14 09:56:53  MQXF_PUT      MQCC_OK       0000      2 (QUEUE1)
001 2016-04-14 09:56:53  MQXF_CLOSE    MQCC_OK       0000      2 (QUEUE1)
001 2016-04-14 09:56:53  MQXF_DISC     MQCC_OK       0000      -
=====
=
```

Summary

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



Legal Disclaimer

- © IBM Corporation 2015. All Rights Reserved.
- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.