Matthew B White (whitemat@uk.ibm.com) – MQ Integration Connectivity and Scale

September 2014

IBM MQ v8 and JMS 2.0 An Introduction

slideshare.net/calanais/ibm-mq-v8-and-jms-20

IBM MQ V8 delivering best in class enterprise messaging

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .Net and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple Cluster Transmit Queue on all platforms	SMF and shared queue enhancements

Agenda

- <u>Introduction</u> to the changes in the WMQ v8 JMS and Java Clients
- <u>Introduction</u> to JMS2.0 and how this has been implemented in v8

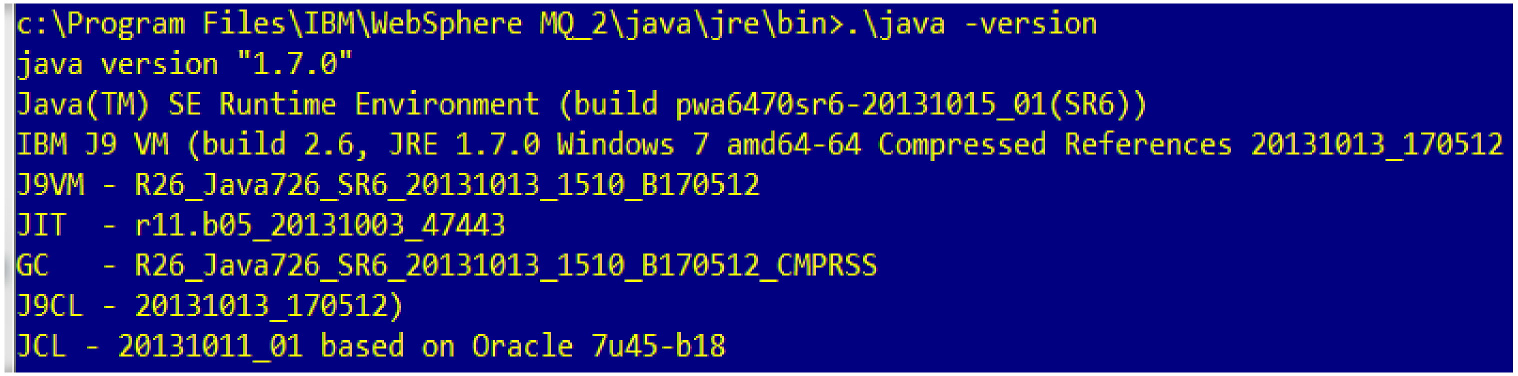
WMQ v8 – Summary of Changes

- New version of the Java Runtime – version 7
- Packaging Changes
  - Fewer JARs
  - Simpler and quicker access to the just the JARs
- Removal of DirectIP function
- Username and Password updates
- Improved Control of Tracing

New Java Runtime: Java 7

- JMS 2.0 – use of `java.lang.AutoCloseable` for main objects
- Interfaces in `jms.jar` also built using Java 7 class file format
- JRE now shipped with MQ v8 is IBM Java 7 (or hybrid depending on platform)

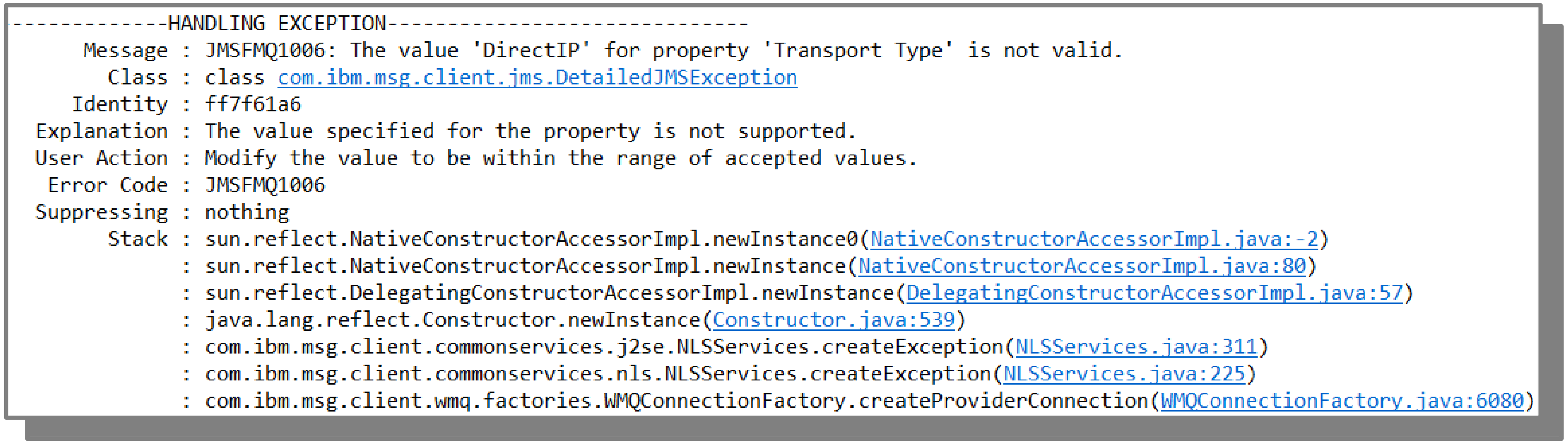
- **Features**:
  – http://radar.oreilly.com/2011/09/java7-features.html
  – Try-with-resources specifically as JMS objects are 'resources' now
  – Try-with-multiple-catch block of specific interest for coding exception handling
  – *not* adopted the new i-o classes



```
c:\Program Files\IBM\WebSphere MQ_2\java\jre\bin>.\java -version
java version "1.7.0"
Java(TM) SE Runtime Environment (build pwa6470sr6-20131015_01(SR6))
IBM J9 VM (build 2.6, JRE 1.7.0 Windows 7 amd64-64 Compressed References 20131013_170512
J9VM - R26_Java726_SR6_20131013_1510_B170512
JIT  - r11.b05_20131003_47443
GC   - R26_Java726_SR6_20131013_1510_B170512_CMPRSS
J9CL - 20131013_170512)
JCL - 20131011_01 based on Oracle 7u45-b18
```

Packaging: DirectIP Removal

- Ability to create a DirectIP (TCP or HTTP) connection has been removed.
- Function not supported exception thrown



```
-------------HANDLING EXCEPTION------------------------------
     Message : JMSFMQ1006: The value 'DirectIP' for property 'Transport Type' is not valid.
       Class : class com.ibm.msg.client.jms.DetailedJMSException
    Identity : ff7f61a6
 Explanation : The value specified for the property is not supported.
 User Action : Modify the value to be within the range of accepted values.
  Error Code : JMSFMQ1006
 Suppressing : nothing
       Stack : sun.reflect.NativeConstructorAccessorImpl.newInstance0(NativeConstructorAccessorImpl.java:-2)
             : sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:80)
             : sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:57)
             : java.lang.reflect.Constructor.newInstance(Constructor.java:539)
             : com.ibm.msg.client.commonservices.j2se.NLSServices.createException(NLSServices.java:311)
             : com.ibm.msg.client.commonservices.nls.NLSServices.createException(NLSServices.java:225)
             : com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createProviderConnection(WMQConnectionFactory.java:6080)
```

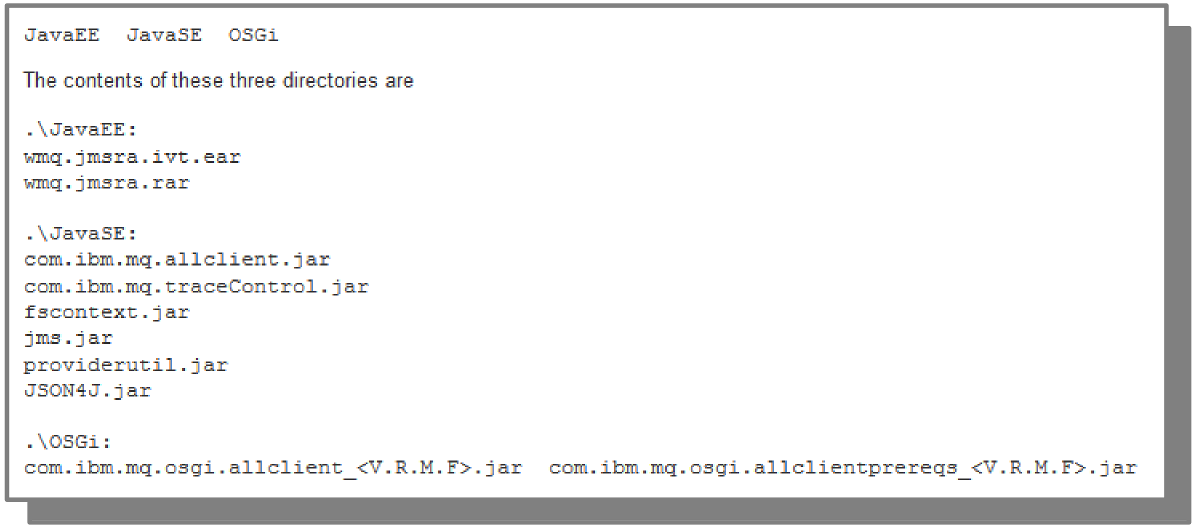
- APIs controlling properties retain their settings – specifically so administration of objects is possible via JMSAdmin or Explorer
  - Settings have no functional behaviour however
- Any IBM Message Broker that supported DirectIP is now out of normal support

Packaging: JAR files

- **Files that have gone:**
  - ldap.jar jndi.jar
    - *Both now included in the standard Java class libaries*
  - connector.jar
    - *Outdated dependency for MQ Java classes – not required*
  - CL3Export.jar CL3NonExport.jar dhbcore.jar
    - *DirectIP support - removed*
- **Files that are new**
  - com.ibm.mq.allclient.jar
    - *Base Java, JMS, Headers and PCF classes*
  - com.ibm.mq.traceControl.jar
    - *Remote trace control*
- **Changes**
  - jms.jar is the new JMS 2 version
  - no distinguishing features except the size – now 57kb

Obtaining and using the JAR files

- No longer need to install the server or full client SupportPac to get the RA or JMS/Java client JARs
- Self-extracting JARs available on Fix Central for
  - IBM MQ Resource Adapter
  - IBM MQ JMS and Java Clients
- Knowledge Centre updated to clarify ability about moving `com.ibm.mq.allclient.jar`
  - Supports the use of Maven repositories for example
- Full details in Technote 1683398
  - http://www-01.ibm.com/support/docview.wss?uid=swg21683398



```
JavaEE   JavaSE   OSGi
```

The contents of these three directories are

```
.\JavaEE:
wmq.jmsra.ivt.ear
wmq.jmsra.rar

.\JavaSE:
com.ibm.mq.allclient.jar
com.ibm.mq.traceControl.jar
fscontext.jar
jms.jar
providerutil.jar
JSON4J.jar

.\OSGi:
com.ibm.mq.osgi.allclient_<V.R.M.F>.jar  com.ibm.mq.osgi.allclientprereqs_<V.R.M.F>.jar
```

Security: Username and password passing

- **Motivation**: Start to properly use username and password for authentication in Client mode
- **Compatibility**: We haven't historically done this, so we make it optional
- Enabling the option
  - –Global boolean option
    - •Environment Variable *com.ibm.mq.jmqi.useMQCSPauthentication*
    - •System property *com.ibm.mq.cfg.jmqi.useMQCSPauthentication*
    - •Client ini file stanza *JMQI*, attribute *useMQCSPauthentication*
- Effect if option is set
  - –Flow the username and password data in the MQCSP as part of the CONAUTH flow
  - –Flow the "real" username in the UID flow

Security: Password Protection Configuration

- If you're concerned about security over tcp/ip connection use SSL.
- But to just 'obfuscate' the password, **Password Protection** can be used
- **Password Protection**
  - –Choices: "Compatible", "Always", "Optional"
  - –Environment Variable *com.ibm.mq.jmqi.PasswordProtection*
  - –System Property *com.ibm.mq.cfg.jmqi.PasswordProtection*
  - –Client ini file Stanza *Channels*, attribute *PasswordProtection*
  - –Only “Always” is really relevant to Clients
    - If set, then the list offered to the QM only allows for 3DES
    - If unset, the list allows for 3DES and NULL encryption
- **Random Number Type**
  - –Choices - “Standard", "Fast"
  - –Environment Variable *AMQ_RANDOM_NUMBER_TYPE*
  - –System Property *com.ibm.mq.cfg.jmqi.AmqRandomNumberType*
  - –Standard – use java.security.SecureRandom
  - –Fast – use java.util.Random
- This is only relevant if the previous “flow userid and password in the CONAUTH flow” options are in force, otherwise the userid and password is in the ID flow which will not be encrypted.

Dynamic Trace Control

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFKSJ_8.0.0/com.ibm.mq.tro.doc/q115930_.htm

- Java clients don't always alongside a QM install or even from a command line JVM
- May be embedded in some other system the admin has little admin control over

- Using JMX based administration
  – various properties of a running MQ JMS client can controlled
- Current Trace and basic information about running system can be found
  – Also includes some statistics on JMS2.0 async send queue

Dynamic Trace Control

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFKSJ_8.0.0/com.ibm.mq.tro.doc/q115930_.htm

```
C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -list
10008 : 'MQSample'
 9004 : 'MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -list'

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : false
User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
Trace File Name : mqjms.trc
Package Include/Exclude tree
root - Included

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -enable
Enabling trace
Tracing enabled : true

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : true
User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
Trace File Name : mqjms_10008.trc
Package Include/Exclude tree
root - Included
```

Introduction and Aims – JMS 2.0

- **Aims for you:**
  - Get some education on what's new and different in JMS 2.0
  - Additional information about Java related changes in MQ v8
- **Aims for us:**
  - Gauge adoption of JMS 2.0 and in what environments?
  - What business use cases for the new features can you think of?

References:

- JMS 2.0 – JSR 343 Java Message Service (JMS 2.0)
  - http://jcp.org/en/jsr/detail?id=343
- Final release on 21 May 2013.
  - https://java.net/projects/jms-spec/pages/JMS20FinalRelease

JMS 2.0 Headlines – What's new? *[Specification Reference: §1.2]*

- *Specification updates and clarifications*
  - Point to Point *AND* Publish/Subscribe domains required
- *New Messaging Features*
  - Delivery Delay
  - Asynchronous Send
  - Subscriptions can be shared across a messaging provider
- *API Changes*
  - Use of Java7's `java.lang.AutoCloseable`
  - JMS Simplified API
  - Session doesn't need parameters (for JavaEE)
- *JavaEE Updates*
  - *Recommendation* on provision of a Resource Adapter
  - Specification clarifications
  - Part of Java EE 7

Simplified API

Messaging Features

JavaEE

Updates

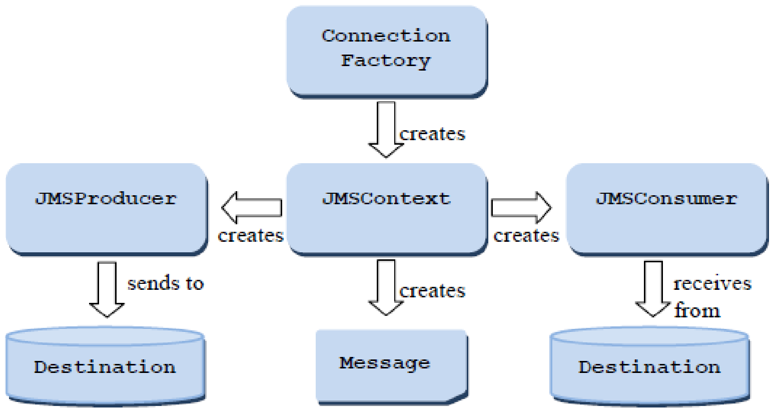
Architecture & Simplified API [§2]

- Overall messaging architecture still the same
- Best practices all the same
- Still a specification for a messaging and state model
  - Not a transport protocol

- *Simplified API*
  - New set of interfaces to reduce number of objects needed
  - Aim to make it easier to use within JavaEE <u>and</u> JavaSE
  - Faster access to data when creating and manipulating messages
  - Error handling changes

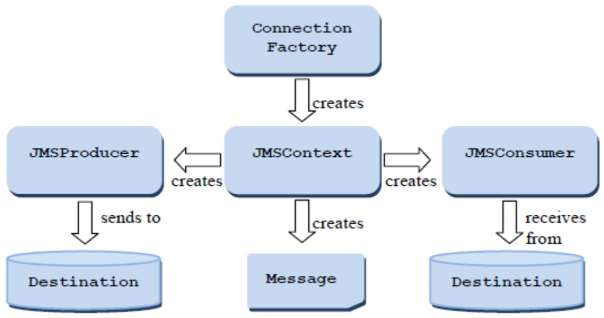
Simplified API
[§2.8]

- **ConnectionFactory**
  an administered object to create a Connection. As used by the classic API.
- **JMSContext**
  an active connection to a JMS provider and a single-threaded context for sending and receiving messages
- **JMSProducer**
  created by a **JMSContext,** used for sending messages to a queue or topic
- **JMSConsumer**
  created by a **JMSContext,** used for receiving messages sent to a queue or topic



All based on JMS1.1
Unified Domain concepts

JMSContext Features [§6]

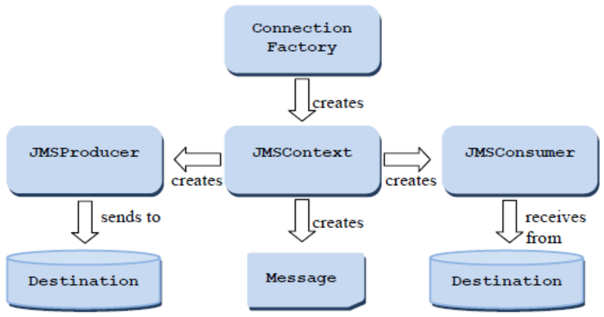


- Connection and Session abstraction
- Retains the same semantics, e.g. Temporary destinations for JMSContext scoped by concept of underlying connection

- AutoStart of underlying connection – i.e. can forget to forget starting the connection
- Can create duplicate JMSContext, but using same underlying connection

- Application Managed – from `createContext()` on ConnectionFactory
- Container Managed - @Inject annotation

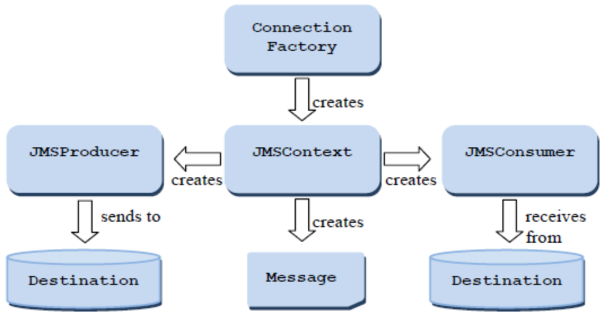
Producing Messages [§7]



- **JMSProducer**
  - –Can take on role of being 'proxy' message object.
  - –Message properties set on the producer object prior to sending a 'body'
  - –Existing MessageProducers can't do that; though have been extended for new messaging styles
  - –Method chaining
  - –'lightweight object' therefore no close

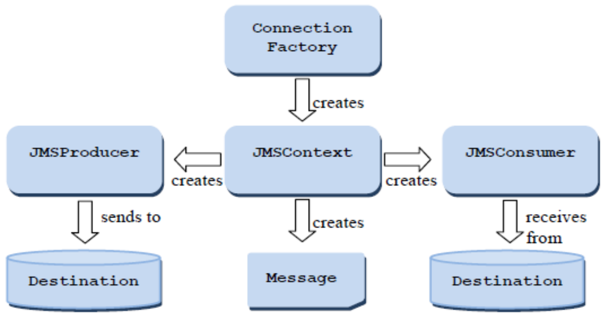
```
    producer.setProperty("MyProperty", "JMS2.0").send(destination, "SimplePTP: ");

    context.createProducer().setTimeToLive(1000).setDeliveryMode(NON_PERSISTENT).send(dataQueue, body);
```

Consuming messages [§8]



- **JMSConsumer**
  - `ReceiveBody(...)` methods - message bodies only.
  - Not completely symmetrical with sending
    - i.e. if want properties still need a message object
- Async and Sync consumption as before
- Can closed from another thread
  - Waits on in-progress consumption to finish

Messages [§3]



- Message Body only
  - JMSProducer & JMSConsumers can now work with the message bodies without message objects
  - Messages have a `getBody()` method

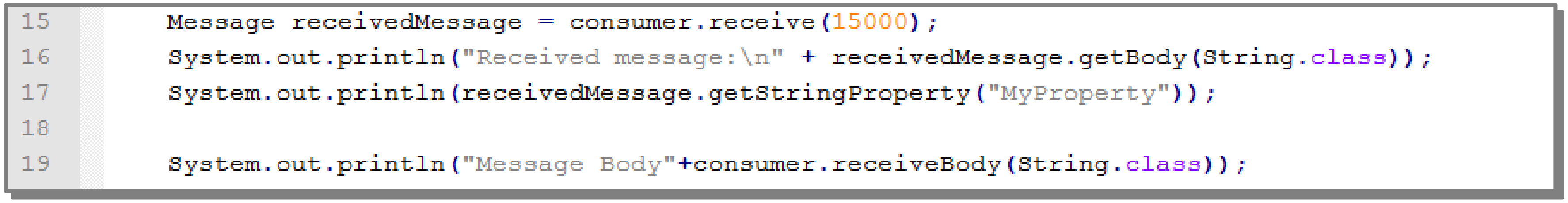
- After sending message, application free to modify message

- JMSXDeliveryCount is now mandatory [§3.5.11]
  - Doesn't have to be exactly correct
    - i.e. no persistence of value required
  - If JMSRedelivered=true, then JMSXDevlieryCount>=2
  - MQ JMS has always set this

Receiving Message Bodies Synchronously [§8.6]

- `<T> T receiveBody( Class<T> c)`
  - Receives the next message produced for this JMSConsumer and returns its body as an object of the specified type
- `<T> T receiveBody( Class<T> c, long timeout)`
  - Receives the next message produced for this JMSConsumer that arrives within the specified timeout period, and returns its body as an object of the specified type
- `<T> T receiveBodyNoWait( Class<T> c)`
  - Receives the next message produced for this JMSConsumer if one is immediately available and returns its body as an object of the specified type
- Can throw a MessageFormatRuntimeException if wrong class used
  - If AUTO_ACK, will be as if method never called
    ..else will be as if method failed, app must force rollback

Message Body Conversions



```
        Message receivedMessage = consumer.receive(15000);
        System.out.println("Received message:\n" + receivedMessage.getBody(String.class));
        System.out.println(receivedMessage.getStringProperty("MyProperty"));

        System.out.println("Message Body"+consumer.receiveBody(String.class));
```

Message Type	**Parameter**
TextMessage	String.class
ObjectMessage	Java.io.Serializable
MapMessage	Java.util.Map or java.lang.Object
BytesMessage	byte[].class
Message	*Always returns null*
<nobody>	*Always returns null*

Message Body Conversions: MQ v8 Notes

- Applications needs to be careful how these are used, especially with Message Consumers
- Consumer's using `receiveBody` if AUTO_ACK or DUPS_OK needs lock message on queue so unavailable to anybody else
  - Message is locked on the queue using GMO_LOCK
  - Body conversion is attempted
  - Failure means message is unlocked – with the potential to be delivered again
  - Success requires message to be removed from queue destructively

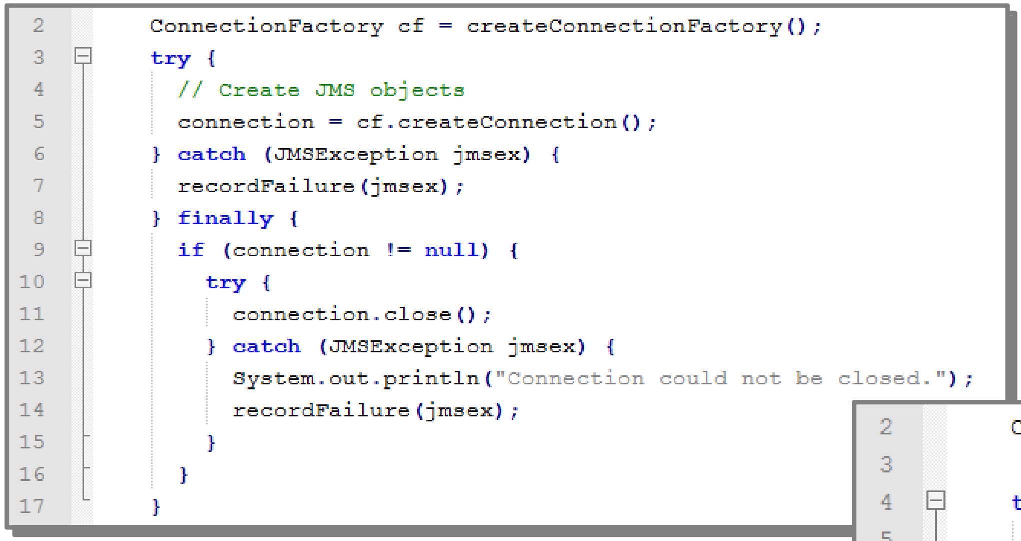
  - Note this is NOT redelivery – messages are not marked as with a re-delivered message
- Transactional `receiveBody` – gets message and then examines the message body
  - Failure of this is returned to application
  - Application expected to rollback transaction
- Closing the session will close the open handle to the queue and the locks will be released

JMS 1.1 Interfaces Updates [§6.1]

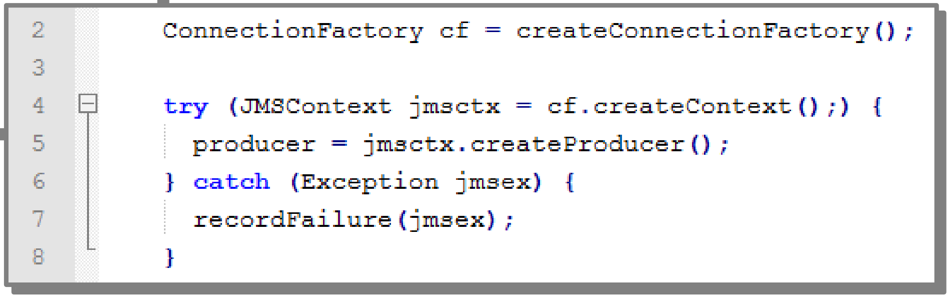
- **Connections**
  - –New methods for creating session with no transaction arguments
  - –Shared Connection Consumers
- **Sessions**
  - –Creating share consumers
  - –Creating JMS1.1 MessageConsumer for durable subscriptions
- **Message Producer**
  - –New set/get for Delivery Delay
  - –Send methods extended to supply Completion Listener
- **Message Consumer**
  - –No change
- **Message**
  - –For delivery delay – `getJMSDeliveryTime()`
  - –New methods to directly access body data, `getBody()` `isBodyAssignableTo()`

Java 7 - Auto-Closeable

- Java 7 gives large variety of
  - new functions – class libraries
  - Java Language enhancements
  - New class file version
- JMS2.0 drives Java 7 specifically because of the try-with-resources



```java
ConnectionFactory cf = createConnectionFactory();
try {
  // Create JMS objects
  connection = cf.createConnection();
} catch (JMSException jmsex) {
  recordFailure(jmsex);
} finally {
  if (connection != null) {
    try {
      connection.close();
    } catch (JMSException jmsex) {
      System.out.println("Connection could not be closed.");
      recordFailure(jmsex);
    }
  }
}
```



```java
ConnectionFactory cf = createConnectionFactory();

try (JMSContext jmsctx = cf.createContext();) {
  producer = jmsctx.createProducer();
} catch (Exception jmsex) {
  recordFailure(jmsex);
}
```

Exceptions and Exception Listeners [§10]

- New exception **`JMSRuntimeException`** thrown from new API calls
- JavaDoc shows mandatory cases
  - ...but... provider may thrown these for other cases as well

- Exceptions thrown on a JMS calls, must not be delivered to an ExceptionListener

- Compiler will not force you to do error checking
- Must take responsibility for doing it at suitable points

Simplified API

Messaging Features

JavaEE

Updates

New messaging features - recap

- **Asynchronous Send**
  - Application sends messages via API that returns before server has processed messages
  - Confirmation via listener
- **Delayed Delivery**
  - Allow delivery at a later point in time
  - (remember this is not a database)
- **Shared Subscriptions**
  - Subscription which can opened by multiple consumers
  - Messages shared amongst consumers (no fairness rules provided)

Subscriptions [§8.3 §4.2]

- **Unshared non-durable subscriptions**
  - As per JMS 1.1. non-durable subscription
- **Shared non-durable subscription**
  - Identified by 'sharedSubscriptionName' and 'clientId' if set
  - If clientId is set, all consumers must share the same clientId
  - Subscription/undelivered messages deleted when last consumer is closed
- **Unshared Durable Subscriptions**
  - As per JMS 1.1 Durable subscriptions
- **Shared Durable Subscriptions**
  - Has the features of a durable subscription but pulls in multiple consumers aspect
  - ClientId is optional
- Same 'sharedSubscriptionName' can be used for durable and non-durable

Asynchronous Send [§7.3]

- *Synchronous Send* means
  - –Send message and wait for a response before returning from send call
- *Asynchronous Send* means
  - –Send message and return from the send call before response from server

- Closing must wait for failure or completion of async sends
- Callbacks made in the same order the messages where sent
- Message objects not multi-threaded – don't modify until onCompletion
- `CompletionListener` called when response has been received.
  - –Does not work like message listener in terms of thread of control
- Quality of Service
  - –After `onCompletion` equal to sync send(...)

Message Delivery Delay [§7.9]

- Earliest Time JMS provider may give message to consumer
- Sets the <u>minimum</u> length of time (in ms) that must elapse after a message is sent before the JMS provider may deliver the message to a consumer

- For transacted sends, this time starts when the client sends the message, not when the transaction is committed.

- Delivery Delay set longer than expiry is an error!
  – Will throw an exception

Message Delivery Delay – MQ v8 Notes [§7.9]

- Implemented using a single internal staging queue on the Qmgr
- Header added to messages placed on this queue.
- Qmgr component 'delayed delivery processor' monitors
  - Delivery performed when delay completes

- Only available for use by JMS
- Queue is created by default on Distributed, needs creating manually on zOS
- Full details at Knowledge Center topic 'q119200_'

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q119200_.htm

Simplified API

Messaging Features

JavaEE

Updates

JMS and JavaEE
[§12 §13]

- JMS one of the constituent specifications of JavaEE
- JavaEE 7 brings in JMS 2.0

- ASF Mode still present – extended for Shared Subscriptions
- Context APIs also include XA variants
- MDB Activation Specifications main way of driving messages into JavaEE

- Strongly *Recommended* to provide a JCA Resource Adapter

JMS and JavaEE – API updates [§12 §13]

- Prescriptive list of APIs that can not be called in the containers
  - No message listeners
  - Single session / connection
  - No connection consumers
  - No Asynchronous Send
  - Transaction control etc.

- Session/Context APIs - No local transactions or client acknowledgement
  - `javax.jms.Connection.createSession()`
  - `javax.jms.ConnectionFactory.createContext()`

Resource Adapter
[§13]

- Specification assumption that one will be provided

- Two additional properties for an activation specification
- *destinationLookup*
  - the JNDI name of javax.jms.Queue or javax.jms.Topic that defines the JMS queue or topic for the MDB
- *connectionFactoryLookup*
  - the JNDI name of javax.jms.ConnectionFactory, used to connect to the JMS provider

MQ v8 Resource Adapter

- Resource Adapter is specifically for JavaEE 7 – therefore JCA 1.7
- Stand alone applications use JMS 2.0 JAR supplied with MQ
- JavaEE servers have their own – the RA doesn't have one...
- **WMQ v8 Resource Adapter only deployed in JavaEE7**

- **WMQ Resource Adapters deployed in WAS 8.5.5/8.5/7 can connect to a WMQ v8 QueueManager in 'bindings' or 'client'**
-

- *WebSphere MQ resource adapter v8.0 statement of support http://www-01.ibm.com/support/docview.wss?uid=swg27041968*

Simplified API

Messaging Features

JavaEE

Updates

Migration of applications

- JMS 2.0 supports all the JMS1.1 APIs
  - The JMS2.0 version of the specification though defines the JMS1.1 usage
- Existing applications, with recompile, can run with a JMS2.0 implementation
- JMS1.1 apps though written against a JMS2.0 implement will not work
  - Existing interfaces reference some new JMS2.0 interfaces
- Note that the JVM for JMS2.0 implementations and Java7 runtime is needed
- Some API behavior changes...

JMS and the compliance test suite

- To ensure that the JMS provider has implemented the specification correctly there is a compliance test suite that has to be run.
- This has been updated together with the JMS 2.0 specification
- To meet compliance with this new spec – some minor functional behavior will change from previous implementations.
- 'out of the box' MQ JMS needs to comply but old behavior can be switched in.
  - the property `com.ibm.mq.jms.SupportMQExtensions`, which can be set to TRUE, to revert these changed behaviors back to previous implementations.

SuportMQExtensions

- Three areas of functionality are reverted by setting SupportMQExtensions to True:
- **Message priority**
- Messages can be assigned a priority, 0 - 9. Before JMS 2.0, messages could also use the value -1, indicating that a queue's default priority is used. JMS 2.0 does not allow a message priority of -1 to be set. Turning on SupportMQExtensions allows the value of -1 to be used.
- **Client id**
- The JMS 2.0 specification requires that non-null client ids are checked for uniqueness when they make a connection. Turning on SupportMQExtensions, means that this requirement is disregarded, and that a client id can be reused.
- **NoLocal**
- The JMS 2.0 specification requires that when this constant is turned on, a consumer cannot receive messages that are published by the same client id. Before JMS 2.0, this attribute was set on a subscriber to prevent it receiving messages that are published by its own connection. Turning on SupportMQExtensions reverts this behavior to its previous implementation.

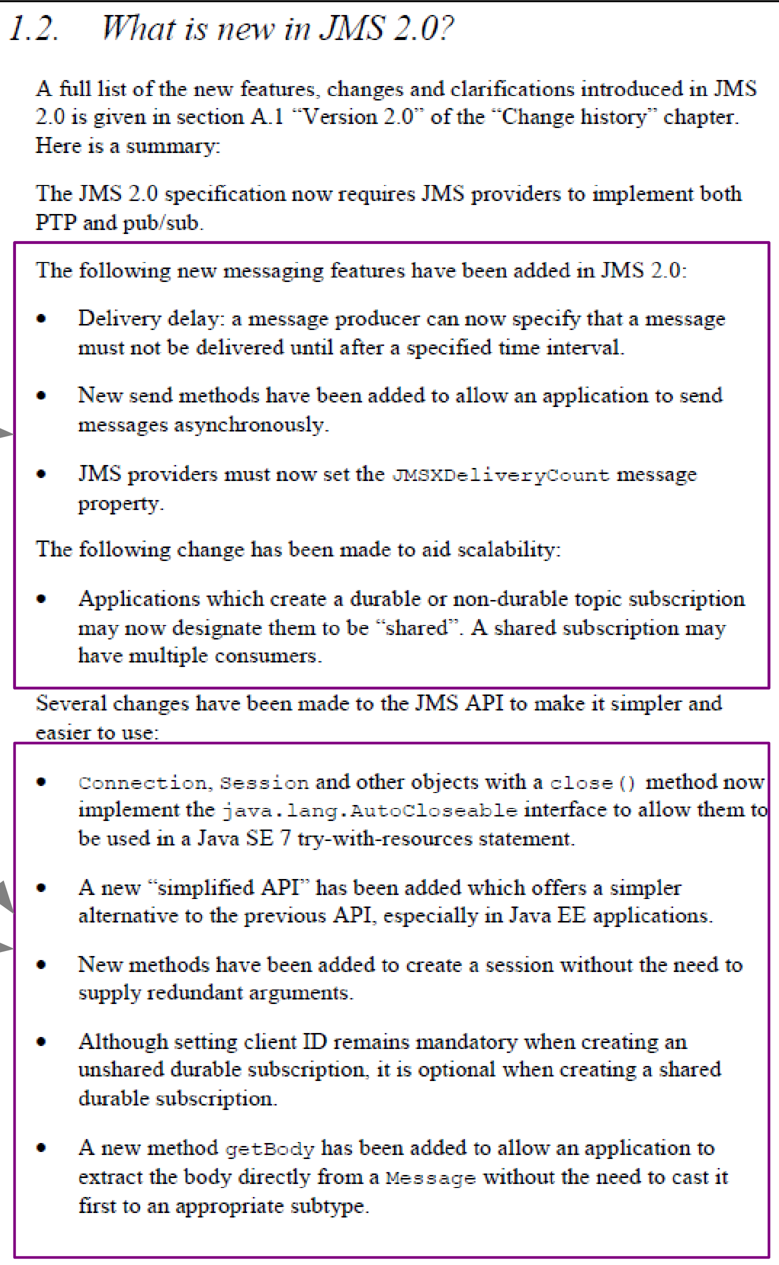
Provider Version

http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.mig.doc/q001850_.htm

- We've introduced a new property for the ProviderVersion field -
  - This is for 'Normal Mode'.
  - This is represented by a PROVIDERVERSION=8.
- This means that we need to connect to a QM that has a cmd level of 800 (WMQ v8). When this is achieved you can use all the JMS 2.0 features.
- Connecting via 'Normal Mode with Restrictions'
  - This is how PROVIDERVERSION=7 is defined
  - This can connect to a queue manager with cmd level greater than 700
  - You can use the JMS2.0 API but not Ayscn Send, Delayed Delivery or Shared Subs.
- Connecting via the 'Migration mode' means that everything is JMS1.1
  - No matter the QM connected to
- Leaving PROVIDERVERSION unset means
  - 'Normal Mode' is attempted first
  - If the command level is less than 800 then 'Normal Mode with Restrictions' is used (established hConn is re-used)
  - If the command level is less than 700, the connection is closed and recreated with 'Migration Mode'

ProviderVersion=8
CMDLvl=800

ProviderVersion=7
CMDLvl=700+



1.2. *What is new in JMS 2.0?*

A full list of the new features, changes and clarifications introduced in JMS 2.0 is given in section A.1 "Version 2.0" of the "Change history" chapter. Here is a summary:

The JMS 2.0 specification now requires JMS providers to implement both PTP and pub/sub.

The following new messaging features have been added in JMS 2.0:

- Delivery delay: a message producer can now specify that a message must not be delivered until after a specified time interval.
- New send methods have been added to allow an application to send messages asynchronously.
- JMS providers must now set the `JMSXDeliveryCount` message property.

The following change has been made to aid scalability:

- Applications which create a durable or non-durable topic subscription may now designate them to be "shared". A shared subscription may have multiple consumers.

Several changes have been made to the JMS API to make it simpler and easier to use:

- `Connection`, `Session` and other objects with a `close()` method now implement the `java.lang.AutoCloseable` interface to allow them to be used in a Java SE 7 try-with-resources statement.
- A new "simplified API" has been added which offers a simpler alternative to the previous API, especially in Java EE applications.
- New methods have been added to create a session without the need to supply redundant arguments.
- Although setting client ID remains mandatory when creating an unshared durable subscription, it is optional when creating a shared durable subscription.
- A new method `getBody` has been added to allow an application to extract the body directly from a `Message` without the need to cast it first to an appropriate subtype.

Error messages when using Provider Version

Error Code	Message
JMSCC5007	Use of the JMS2.0 API "{0}" is not supported with this instance of this connection
	Only connections with a correct type of connection can support using the JMS2.0 API
JMSCC5008	Use of the JMS2.0 Function "{0}" is not supported with this instance of this connection
	The use of the JMS2.0 functionality mentioned in the message is only supported when connecting to a WebSphere MQ V8 queue manager using WebSphere MQ messsaging provider V8 mode