MQ Java, TLS Ciphers, Non-IBM JREs & APARs IT06775, IV66840, IT09423, IT10837 -- HELP ME PLEASE!

[Tom\_Leend](javascript:;" \o "Tom_Leend) |June 13 2016 Updated

**Introduction**

If you are using the MQ classes for JMS or the MQ classes for Java APIs in your Java application, which is running in a non-IBM Java Runtime, and are making secure TCP/IP connections to a queue manager, I'm sure you would have come across one or two of the APARs noted in the title of this blog post.  You might also be thinking, "what on Earth do these APARs actually do, do they concern the particular Java client version I'm using and why are all these APARs needed?".  If so, you're not the only one!

I have spent quite a lot of time in the past few weeks answering PMRs that contain instances of these APAR problems as well as questions, concerns, comments, interim fix build requests.  The APARs in question are as follows:

* [**IT06775** : WebSphere MQ Java / JMS Client CipherSuite to CipherSpec Mapping May Have Multiple Choices](https://www.ibm.com/developerworks/community/blogs/roller-ui/authoring/weblogEntryDetail.do?entryId=f8915851-f874-48d2-8cc7-42010c4c525e&method=view&rmik=tabbedmenu.weblog.archives&lnk=%2Froller-ui%2Fauthoring%2FweblogEntryManagement.do%3Fmethod%3Dquery%26rmik%3Dtabbedmenu.weblog.archives%26weblog%3Dmessaging%26status%3DPUBLISHED%26currentPage%3D2%26pageSize%3D50%26reverse%3D1&status=PUBLISHED&lang=en#APARIT06775)
* [**IV66840** : WMQ V7 Java/JMS Add Support For Selected TLS CipherSpecs When Running in a Non-IBM Java Runtime Environment](https://www.ibm.com/developerworks/community/blogs/roller-ui/authoring/weblogEntryDetail.do?entryId=f8915851-f874-48d2-8cc7-42010c4c525e&method=view&rmik=tabbedmenu.weblog.archives&lnk=%2Froller-ui%2Fauthoring%2FweblogEntryManagement.do%3Fmethod%3Dquery%26rmik%3Dtabbedmenu.weblog.archives%26weblog%3Dmessaging%26status%3DPUBLISHED%26currentPage%3D2%26pageSize%3D50%26reverse%3D1&status=PUBLISHED&lang=en#APARIV66840)
* [**IT09423**: WMQ Java/JMS Not Able to Connect to A SSLv3 Secured Channel When Using a Non-IBM JRE with IV66840 Activated](https://www.ibm.com/developerworks/community/blogs/roller-ui/authoring/weblogEntryDetail.do?entryId=f8915851-f874-48d2-8cc7-42010c4c525e&method=view&rmik=tabbedmenu.weblog.archives&lnk=%2Froller-ui%2Fauthoring%2FweblogEntryManagement.do%3Fmethod%3Dquery%26rmik%3Dtabbedmenu.weblog.archives%26weblog%3Dmessaging%26status%3DPUBLISHED%26currentPage%3D2%26pageSize%3D50%26reverse%3D1&status=PUBLISHED&lang=en#APARIT09423)
* [**IT10837** : WebSphere MQ Channel Start Fails with AMQ9637 When Using a TLS Cipher Running in an Oracle JRE](https://www.ibm.com/developerworks/community/blogs/roller-ui/authoring/weblogEntryDetail.do?entryId=f8915851-f874-48d2-8cc7-42010c4c525e&method=view&rmik=tabbedmenu.weblog.archives&lnk=%2Froller-ui%2Fauthoring%2FweblogEntryManagement.do%3Fmethod%3Dquery%26rmik%3Dtabbedmenu.weblog.archives%26weblog%3Dmessaging%26status%3DPUBLISHED%26currentPage%3D2%26pageSize%3D50%26reverse%3D1&status=PUBLISHED&lang=en#APARIT10837)

In this blog post, I'm hoping to bring together some of the discussions I've had with L2 Support teams, as well as with customers via PMRs, regarding these APARs and explain a little more about how they came about and why they are needed.

Before I start the main part of this blog post, I would suggest reading the following blog entry that was written by one of my colleagues from the MQ Java Development team:

<https://www.ibm.com/developerworks/community/blogs/messaging/entry/BiteSize_Blogging_MQ_Version_8_The_relationship_between_MQ_CipherSpecs_and_Java_Cipher_Suites?lang=en>

The blog entry provides a great description on the need for mapping between MQ CipherSpecs and Java CipherSuites and touches on the differences between IBM and Oracle Java Runtimes.  It specifically discusses MQ V8 Ciphers but the text will provide a grounding for the rest of my update here.

Okay, so let's begin!

By default, the MQ classes for JMS and classes for Java use the IBM Java CipherSuite names to map to corresponding MQ CipherSpecs.  These CipherSuite names are documented in Table 1 on the following linked MQ Knowledge Center pages:

IBM MQ V8: [SSL/TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS](http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q113220_.htm?lang=en)

WebSphere MQ v7.5: [SSL CipherSpecs and CipherSuites in JMS](http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.dev.doc/q032470_.htm?lang=en)

An application would specify one of the supported Java CipherSuites, configuring in on a JMS Connection Factory object for example, and the MQ Java client would map this to a corresponding MQ CipherSpec used during channel negotiation when a connection to a queue manager is established.

Prior to MQ V8 there were two IBM Java CipherSuites that could actually map to two distinct MQ CipherSpecs:

|  |  |  |
| --- | --- | --- |
| **IBM Java CipherSuite** | **MQ CipherSpec 1** | **MQ CipherSpec 2** |
| SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |

The MQ CipherSpecs listed under the "MQ CipherSpec 1" column of the table are SSLv3 CipherSpecs and so expect the SSLv3 protocol to be used for the handshaking when a secure socket is created between the JVM's Java Secure Socket Extension (JSSE) provider and GSKit (used by the MQ queue manager).  The CipherSpecs listed in the above with the column heading "MQ CipherSpec 2" expect the TLSv1.0 protocol to be used.  This means that a single IBM Java CipherSuite can be used to make a secure socket via the SSLv3 protocol **or** the TLSv1.0 protocol.

By default, the dual mappings Java CipherSuites map to the SSLv3 MQ CipherSpecs, not the TLSv1.0 variant.  Before any of the APARs mentioned in this blog post, to ensure the dual mapping CipherSuites were mapped to the TLSv1.0 MQ CipherSpecs Federal Information Processing Standard (FIPS) 104-2 compliant connections must have been enabled.  In the MQ classes for JMS, FIPS is enable by setting SSLFIPSREQUIRED String property to the value YES on JMS Connection Factory used by the application or using the setSSLFipsRequired(boolean) setter method on an MQConnectionFactory object:

cf.setStringProperty(WMQConstants.WMQ\_SSL\_FIPS\_REQUIRED, "YES");

mqcf.setSSLFipsRequired(true);

  For the classes for Java the "sslFipsRequired" variable on the MQEnvironment object can be set to true:

MQEnvironment.sslFipsRequired = true;

*[Note: FIPS 104-2 connections are only supported when the MQ Java client application is running within an IBM JRE.]*

Once FIPS 104-2 has been enabled, all secure connections established within the Java Runtime Environment **must**be FIPS 104-2 compliant.   There cannot be a mix of FIPS and non-FIPS connections.  This causes a bit of headache and it would be nice to enable the TLSv1.0 mapping for the SSL\_RSA\_WITH\_DES\_CBC\_SHA and SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA Java CipherSuites without the need to enable FIPS compliance.  This requirement resulted in [**APAR IT06775**](http://www-01.ibm.com/support/docview.wss?uid=swg1IT06775)**.**

This APAR is **not**applicable to the MQ V8 Java clients because dual mapping noted in the table has been removed.  The SSLv3 MQ CipherSpecs are not supported from MQ V8 and only the TLSv1.0 mapping is possible.

**[APAR IT06775](http://www-01.ibm.com/support/docview.wss?uid=swg1IT06775)**

APAR IT06775 (shipped in MQ V7.0.1.13, V7.1.0.7, V7.5.0.5 and V8.0.0.2) added a configuration property that, when set to true, mapped the dual mapping Java CipherSuites, SSL\_RSA\_WITH\_DES\_CBC\_SHA and SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA, to the MQ CipherSpec TLS\_RSA\_WITH\_DES\_CBC\_SHA and TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA respectively **without**the need to enable FIPS 104-2 compliant connections.

This configuration property is enabled via a Java System Property (-D option) with the name:

com.ibm.mq.cfg.preferTLS

when its value is set to:

true

For example:

  java -cp <MQ\_INSTALLATION\_PATH>/java/lib/com.ibm.mqjms.jar:$PWD -Dcom.ibm.mq.cfg.preferTLS=true MyApplication arg1 arg2 arg3

Below is table showing the default mappings in the pre-V8 MQ Java clients for the two dual mapping CipherSuites when com.ibm.mq.cfg.preferTLS is set to false and true.

|  |  |  |
| --- | --- | --- |
| **Value of**  **com.ibm.mq.cfg.preferTLS** | **IBM Java CipherSuite** | **MQ CipherSpec** |
| false | SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT |
| false | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US |
| true | SSL\_RSA\_WITH\_DES\_CBC\_SHA | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| true | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |

**What about Non-IBM JREs? The Blog Post Title says "Non-IBM JREs"**

Okay, okay... I did say at the start of this blog post I would discuss non-IBM Java Runtime Environments (Oracle JREs being the other supported environment for classes for JMS and classes for Java applications) so let's discuss that from here.

In the blog entry I linked to earlier in this post, IBM and Oracle Java Runtimes use a different naming convention for Java CipherSuites, primarily for TLS CipherSuites.  Therefore, the IBM Java Runtimes will name one particular CipherSuite using one convention whereas Oracle will name it something slightly differently.  Typically, the difference is in the start of the name, whether the name starts with the **prefix**"SSL\_" (in the IBM JRE case) or "TLS\_" (in the Oracle JRE case).

This difference in the TLS Java CipherSuite default naming conventions between the IBM and Oracle Java vendors was the reason for [**APAR IV66840**](http://www-01.ibm.com/support/docview.wss?uid=swg1IV66840).

**[APAR IV66840](http://www-01.ibm.com/support/docview.wss?uid=swg1IV66840)**

APAR IV66840 (shipped in MQ V7.0.1.13, V7.1.0.7, V7.5.0.5 and V8.0.0.2) provided a new internal mapping table for Java CipherSuites adhering to the Oracle naming convention to MQ CipherSpecs.  This allowed TLS CipherSuites to be used with the classes for JMS and classes for Java running in Oracle JREs.  A configuration property (again, set as a Java System Property when the JVM running the application is started) with the name:

com.ibm.mq.cfg.useIBMCipherMappings

was provided by this APAR, which accepts a Boolean (true / false) value.  The default value is **true**.

This property allows application administrators and developers to switch the MQ classes for JMS / classes for Java from using the default IBM Java CipherSuite to MQ CipherSpec mappings table (as documented in the Knowledge Center) to the Oracle CipherSuite to MQ CipherSpec mappings as described in the closing words of APAR IV66840.

Without this APAR and the configuration option it provides, MQ classes for JMS / classes for Java applications running in Oracle JREs would not be able to make use of TLS CipherSuites.  This would not have been acceptable given the recent, well documented, vulnerabilities uncovered in the SSLv3 protocol.

Having said that, as you might have guessed, there is a further complication (because these things can never be as straightforward as one hopes).  This concerns those pesky dual mapping Java CipherSuites we discussed earlier!

Now, the closing words for APAR IV66840 do indeed state that each of those (previously) dual mapping CipherSuites map to a single MQ CipherSpecs like so:

|  |  |  |
| --- | --- | --- |
| **Value of com.ibm.mq.cfg.useIBMCipherMappings** | **Java CipherSuite** | **MQ CipherSpec** |
| false | SSL\_RSA\_WITH\_DES\_CBC\_SHA | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| false | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |

which **was**true when this APAR shipped in the WebSphere MQ v7.5.0.5 Fix Pack.  This is because the APAR focused solely on allowing TLS capable Java CipherSuite to MQ CipherSpec mappings in Oracle JREs.

However, this caused a problem because some customers who used Oracle JREs wanted to make use of **both**SSLv3 and TLS CipherSuites in the same JVM instance!  In hindsight, this should have really been anticipate.  But after IV66840. customer's who enabled the Oracle TLS CipherSuite mappings table were not able to use any SSLv3 Cipher mappings (i.e., when the property provided by this APAR had been set to the value false).  Essentially IV66840 added TLS Cipher capability for Oracle JREs but removed SSLv3 capability at the same time.  Customers reported to IBM Support that they wanted to continue using mappings of:

|  |  |
| --- | --- |
| **Java CipherSuite** | **MQ CipherSpec** |
| SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT |
| SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US |

which worked prior to IV66840 in Oracle and IBM JREs (because the names are equivalent in both mapping tables) but did not work after they had set the useIBMCipherMappings property to the value false.  This caused problems for the migration of applications because the switch from using SSLv3 Ciphers to newer TLS Ciphers could not be staged.

The need to run applications in a single Oracle JRE which used a combination of both SSLv3 and TLSv1.0 (or TLSv1.2) Ciphers resulted in another [APAR IT09423](http://www-01.ibm.com/support/docview.wss?uid=swg1IT09423).

**[APAR IT09423](http://www-01.ibm.com/support/docview.wss?uid=swg1IT09423)**

APAR IT09423 shipped in the WebSphere MQ V7.1.0.7 and V7.5.0.6 Fix Packs to reinstated the SSLv3 **default**mappings of:

|  |  |  |
| --- | --- | --- |
| **Java CipherSuite** | **MQ CipherSpec 1 (now default)** | **MQ CipherSpec 2 (Configurable)** |
| SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |

even when the Java System Property, com.ibm.mq.cfg.useIBMCipherMappings was set to the value false.  This therefore allows the use of both SSLv3 and TLS Ciphers in Oracle JREs, as the mapping table contains both.

To enable the TLSv1.0 only mappings in an Oracle JRE for the dual mappings CipherSuites, the "com.ibm.mq.cfg.preferTLS" property (mentioned earlier and introduced by APAR IT06775) must be used and set to the value true.

Let's try and put that into a table as a summary... (wish me luck):

|  |  |  |  |
| --- | --- | --- | --- |
| **Value of**  **com.ibm.mq.cfg.**  **useIBMCipherMappings** | **Value of**  **com.ibm.mq.cfg.**  **preferTLS** | **Java CipherSuite** | **MQ CipherSpec** |
| true | true | SSL\_RSA\_WITH\_DES\_CBC\_SHA | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| true | false | SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT *(unless FIPS is enabled)* |
| false | true | SSL\_RSA\_WITH\_DES\_CBC\_SHA | TLS\_RSA\_WITH\_DES\_CBC\_SHA |
| false | false | SSL\_RSA\_WITH\_DES\_CBC\_SHA | DES\_SHA\_EXPORT |
| true | true | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |
| true | false | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US *(unless FIPS is enabled)* |
| false | true | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA |
| false | false | SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA | TRIPLE\_DES\_SHA\_US |

*Now, you might have observed that the Java CipherSuites "SSL\_RSA\_WITH\_DES\_CBC\_SHA" and "SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA" exist in both the IBM and Oracle CipherSuite to MQ CipherSpec mapping tables.  Therefore, from WebSphere MQ V7.0.1.7 and V7.5.0.6 (due to APAR IT09423) the default mappings of these are the same in both cases, given by the table directly above.  Therefore, if using either of these Java CipherSuites, it actually does not matter whether the com.ibm.mq.cfg.useIBMCipherMappings Java System Property is set to false or true because the mappings are the same in MQ V7.5.0.6 for both.*

**[APAR IT10837](http://www-01.ibm.com/support/docview.wss?uid=swg1IT10837)**

I've got one final APAR to mention and that is **IT10837**(targeted for V7.1.0.8 and V7.5.0.7 and shipped in V8.0.0.5).  This APAR affects applications running within Oracle JREs that use TLS CipherSuites to connect to a queue manager where the server-connection channel being used has the SSLCAUTH attributed set to "REQUIRED" (the default value).  This means that the client should pass a certificate to the queue manager such that the connecting client can be authenticated by the MQ server.

When the application was running in an Oracle JRE, the SunJSSE provider was not creating a default internal Key Manager object for TLS socket connections, meaning that the client's signed personal certificates were not available for client authentication during the handshake. The IBM JSSE provider does do this based off the information passed via the Java System Properties:

* javax.net.ssl.keyStore

and

* javax.net.ssl.keyStorePassword

Because a KeyManager object was not created by default, the client certificate was not passed to the queue manager (GSKit) for authentication.  As such, the connection from the application would failed.  In this scenario, the queue manager would write the following error message into its error log file:

AMQ9637 (Channel is lacking a certificate)

The fix for this APAR is for the MQ classes for JMS and classes for Java to read a certificate keystore, based on the information in the two Java System Properties noted above, and create a KeyManager based on that information in the case when com.ibm.mq.cfg.useIBMCipherMappings is set to the value false in the JVM .  This can then be used when the SSLContext is created (which is subsequently used to create an SSLSocketFactory and eventually a secure socket object).

There is a local workaround which is for the application itself to create TrustManagerFactory and KeyManagerFactory factory objects for the appropriate certificate stores and to initialise an SSLContext object these objects.  From this SSLContext object and SSLSocketFactory can be created and passed to the MQ classes for JMS (by setting it on the JMS Connection Factory) or to the classes for Java (by setting it on the MQEnvironment or in a Hashtable passed to the MQQueueManager constructor).  For example:

---- Code Snippet Start ----

KeyStore keyStore = KeyStore.getInstance("JKS");  
java.io.FileInputStream keyStoreInputStream = new java.io.FileInputStream("/home/tom/myKeyStore.jks");  
keyStore.load (keyStoreInputStream, password\_char\_array);

KeyStore trustStore trustStore = KeyStore.getInstance ("JKS");  
java.io.FileInputStream trustStoreInputStream = new java.io.FileInputStream("/home/tom/myTrustStore.jks");  
trustStore.load (trustStoreInputStream, password\_char\_array);

keyStoreInputStream.close();  
trustStoreInputStream.close();

KeyManagerFactory keyManagerFactory =   
  KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());  
TrustManagerFactory trustManagerFactory =   
  TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());

keyManagerFactory.init(keyStore,password);  
trustManagerFactory.init(trustStore);

SSLContext sslContext = SSLContext.getInstance("TLSv1");

sslContext.init(keyManagerFactory.getKeyManagers(),   
  trustManagerFactory.getTrustManagers(),   
  null);

SSLSocketFactory sslSocketFactory = sslContext.getSocketFactory();

// classes for JMS  
myJmsConnectionFactory.setObjectProperty(  
  WMQConstants.WMQ\_SSL\_SOCKET\_FACTORY, sslSocketFactory);

// classes for Java  
MQEnvironment.sslSocketFactory = sslSocketFactory;

---- Code Snippet End ----

**Conclusion**

Firstly, congratulations if you've made it to end of this blog post!  As you can see, configuring your MQ classes for JMS and classes for Java applications for secure connections in different Java Runtime Environments is not as easy as you might haved expected, or have hoped!  But, hopefully you'll be able to use this blog post to help diagnose issues you might be seeing while writing your MQ Java client applications as well as a reference for an overview of the APARs in this area.

In conclusion though, the four APARs discussed in this blog post affect different product versions which I'll summarise below:

|  |  |
| --- | --- |
| **MQ Product Version** | **Applicable APARs** |
| WebSphere MQ V7.1 | IT06775, IV66840, IT09423 and IT10837 |
| WebSphere MQ V7.5 | IT06775, IV66840, IT09423 and IT10837 |
| IBM MQ V8 | IV66840 and IT10837 |

**Other References:**

The list of MQ CipherSpecs and details about each:

IBM MQ V8: [Enabling CipherSpecs](http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q014260_.htm?lang=en)

WebSphere MQ v7.5: [Specifying CipherSpecs](http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q014260_.htm?lang=en)