

ENSEMBLE TO RABBITMQ

JAVA CLIENT QUICK START GUIDE

SOFTWARE VERSIONS

SERVER

| | |
|----------------------|----------------------------|
| OS | Windows Server 2012 R2 x64 |
| Erlang | OTP 19.1 |
| RabbitMQ Server | 3.6.5 |
| RabbitMQ Java Client | 3.6.5 |

CLIENT

| | |
|----------------------|------------------------------|
| OS | Windows Server 2012 R2 x64 |
| Java | JDK SE 8u111 JRE SE 8u111 |
| RabbitMQ Java Client | 3.6.5 |
| Ensemble | 2016.2.0.736.0 |

RABBITMQ

INSTALL SERVER

Download and run the Erlang Windows Binary File.
Download and install the RabbitMQ Server.

<http://www.rabbitmq.com/install-windows.html>

OPEN PORT 5672 ON SERVER FIREWALL

5672 is the default non-SSL port the RabbitMQ server listens for AMQP connections.

If not already configured during the RabbitMQ Server installation then open the Windows Firewall with Advanced Security management console:

Control Panel > Administrative Tools > Windows Firewall with Advanced Security

Add a new Port Inbound Rule:

| | |
|----------|------|
| Protocol | TCP |
| Port | 5672 |

And any other ports required for this installation.

INSTALL CLIENT

Install client on both the server hosting the RabbitMQ Server and a machine acting as a remote client.

Download and install the Java JDK.

Set and check JAVA_HOME, example:

System > Advanced system settings > Environment Variables...

Variable name: **JAVA_HOME**

Variable value: **C:\Program Files\Java\jdk1.8.0_111**

C:\>echo %JAVA_HOME%

C:\Program Files\Java\jdk1.8.0_111

Add the JDK bin folder to the system PATH, example:

System > Advanced system settings > Environment Variables...

Variable name: Path

Variable value: %SystemRoot%;%SystemRoot%\system32;

%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell

\v1.0\;C:\Program Files\Java\jdk1.8.0_111\bin

Download, unzip and copy the RabbitMQ Java Client files to a folder:

C:\rabbitmq-java-client-bin-3.6.5\commons-cli-1.1.jar

C:\rabbitmq-java-client-bin-3.6.5\commons-io-1.2.jar

C:\rabbitmq-java-client-bin-3.6.5\junit.jar

...

C:\rabbitmq-java-client-bin-3.6.5\stresspersister.sh

RPC LOCAL QUICK TEST

Local to RabbitMQ Server to facilitate confidence testing.

Download RabbitMQ Performance Testing Tool jar file:

<http://central.maven.org/maven2/com/rabbitmq/perf-test/1.0.1/perf-test-1.0.1.jar>

Copy the jar file to the RabbitMQ Java Client folder:

C:\rabbitmq-java-client-bin-3.6.5\perf-test-1.0.1.jar

More details available at:

<https://www.rabbitmq.com/java-tools.html>

<https://github.com/rabbitmq/rabbitmq-perf-test>

NOTES:

The following are defaulted to:

| | |
|-------------|--------------------|
| host name | "amqp://localhost" |
| user name | "guest" |
| Password | "guest" |
| port number | 5672 |
| Vhost | "/" |

RUN HELLOSERVER EXAMPLE

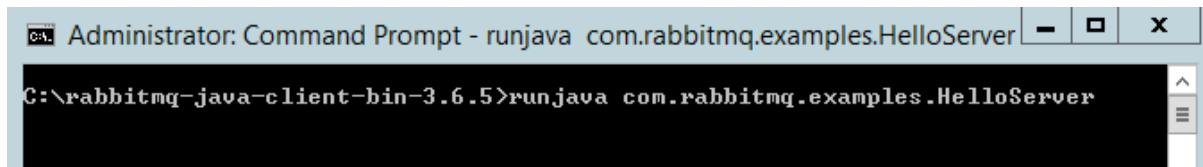
Open a command prompt, navigate to the RabbitMQ Java client folder and execute:

RabbitMQ Java client path>runjava com.rabbitmq.examples.HelloServer

Example:

Windows:

C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloServer



```
Administrator: Command Prompt - runjava com.rabbitmq.examples.HelloServer
C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloServer
```

Unix:
\$ sh runjava.sh com.rabbitmq.examples.HelloServer

RUN HELLOCLIENT EXAMPLE

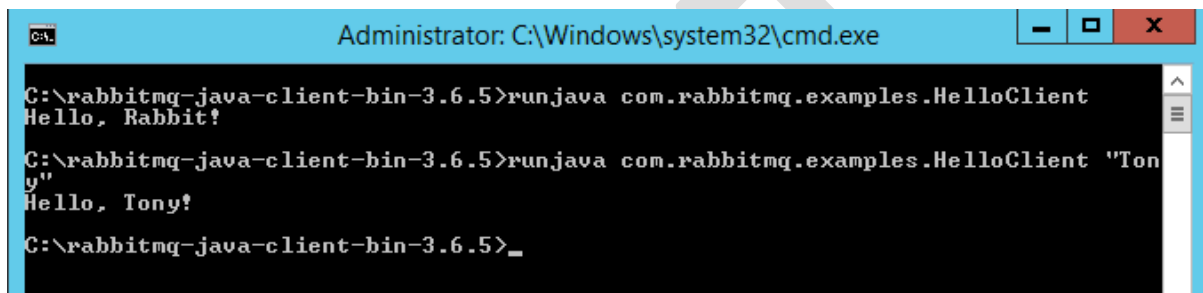
Open a separate command prompt, navigate to the Java client folder and execute:

Java client path>runjava com.rabbitmq.examples.HelloClient

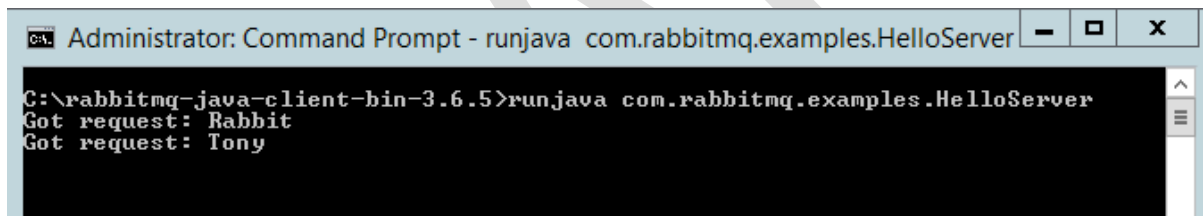
Example:

C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloClient

C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloClient "Tony"



```
Administrator: C:\Windows\system32\cmd.exe
C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloClient
Hello, Rabbit!
C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloClient "Tony"
Hello, Tony!
C:\rabbitmq-java-client-bin-3.6.5>_
```



```
Administrator: Command Prompt - runjava com.rabbitmq.examples.HelloServer
C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloServer
Got request: Rabbit
Got request: Tony
```

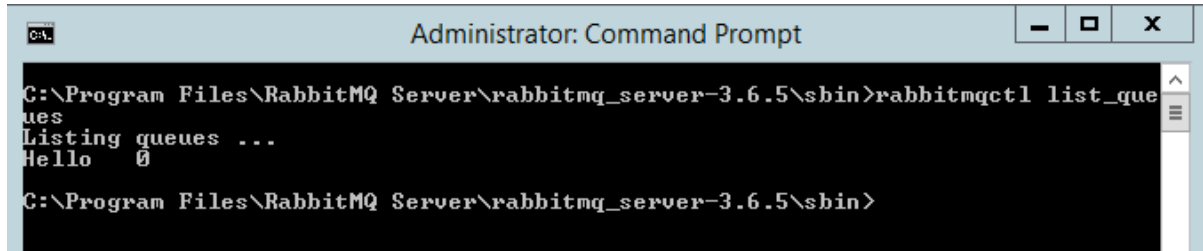
Unix:
\$ sh runjava.sh com.rabbitmq.examples.HelloClient
Hello, Rabbit!
\$ sh runjava.sh com.rabbitmq.examples.HelloClient "Tony"
Hello, Tony!

MONITOR QUEUES

Open a separate command prompt, navigate to the *sbin* folder in the RabbitMQ server folder and execute:
RabbitMQ Server path>\sbin\rabbitmqctl list_queues

Example:

C:\Program Files\RabbitMQ Server\rabbitmq_server-3.6.5\sbin>rabbitmqctl list_queues



```
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.6.5\sbin>rabbitmqctl list_queues
Listing queues ...
Hello 0
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.6.5\sbin>
```

Unix:

\$./rabbitmqctl list_queues

Listing queues ...

Hello 0

RPC REMOTE QUICK TEST

Remote to RabbitMQ server to facilitate confidence testing across network.

NOTES:

The following are defaulted to:

| | |
|-------------|---------|
| user name | "guest" |
| Password | "guest" |
| port number | 5672 |
| Vhost | "/" |

"guest" user can only connect via localhost:

<https://www.rabbitmq.com/access-control.html>

RUN HELLOSERVER EXAMPLE

On the remote machine; open a command prompt, navigate to the RabbitMQ Java client folder and execute:

RabbitMQ Java client path>runjava com.rabbitmq.examples.HelloServer "host"

Example:

Windows:

C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloServer "amqp://WIN-UK0KESAK7LB"

Unix:

\$ sh runjava.sh com.rabbitmq.examples.HelloServer

RUN HELLOCLIENT EXAMPLE

On the remote machine; open a separate command prompt, navigate to the Java client folder and execute:
Java client path>runjava com.rabbitmq.examples.HelloClient "message" "host"

Example:

Windows:

```
C:\rabbitmq-java-client-bin-3.6.5>runjava com.rabbitmq.examples.HelloClient "Tony" "amqp://WIN-UKOKESAK7LB"
```

Unix:

```
$ sh runjava.sh com.rabbitmq.examples.HelloClient "Tony" "amqp://WIN-UKOKESAK7LB"  
Hello, Tony!
```

MONITOR QUEUES

On the RabbitMQ Server server; open a separate command prompt, navigate to the *sbin* folder in the RabbitMQ server folder and execute:

```
RabbitMQ Server path>\sbin\rabbitmqctl list_queues
```

Example:

Windows:

```
C:\Program Files\RabbitMQ Server\rabbitmq_server-3.6.5\sbin>rabbitmqctl list_queues
```

Unix:

```
$ ./rabbitmqctl list_queues  
Listing queues ...  
Hello    0
```

ENSEMBLE OBJECT GATEWAY

CREATE JAVA OBJECT GATEWAY DEFINITION

Navigate to the Object Gateways page in the Management Portal:

System > Configuration > Connectivity > Object Gateways
and select Create New Gateway.

Menu

Home | About | Help | Logout

System > Configuration > Object Gateways > Edit Object Gateway

Edit: RabbitMQJava

Server: WIN-ESKQ6N0HF7F
User: UnknownUser
Namespace: %SYS
Licensed to: ISC Sales Engineering
Instance: ENSEMBLE

Save

Cancel

Use the form below to edit an existing Object Gateway Server definition:

Object Gateway For Java

Gateway Name

RabbitMQJava

Required.

Server Name / IP Address

127.0.0.1

Required.

Port

55555

Required.

Log File

rabbitmqjava.log

Browse...

Heartbeat Interval

10

Heartbeat Failure Timeout

30

Heartbeat Failure Action

Restart

Heartbeat Failure Retry

300

Initialization Timeout

5

Connection Timeout

5

Java Home

JDK Version

Java 1.8

Class Path

C:\rabbitmq-java-client-bin-3.6.5\rabbitmq-client.jar

Browse...

JVM arguments

JAVA_HOME Environment Variable

C:\Program Files\Java\jdk1.8.0_111

START INSTANCE OF GATEWAY

You can start the Java Gateway server in one of the following ways:

- Manually, by selecting the Start link of a previously configured gateway (see below)
- Manually, by calling the business service StartGateway method
- Manually, by entering a command at the Terminal command prompt
- Automatically, by adding a Java Gateway business service to the production
 - o The Java Gateway server starts when the production starts

localhost:5772/csp/sys/mgr/%25CSP.UI.Portal.ObjectGatewayStart?PID=RabbitMQJava

Menu

Home | About | Help | Logout

System > Configuration > Object Gateways > Start Object Gateway

Start Object Gateway

Server: WIN-ESKQ6N0HF7F
User: UnknownUser
Namespace: %SYS
Licensed to: ISC Sales Engineering
Instance: ENSEMBLE

Start Object Gateway

Start Object Gateway Server RabbitMQJava.

Please wait...result will show below:

2016-10-28 17:42:51 Starting Java Gateway Server 'RabbitMQJava'

2016-10-28 17:42:51 Executing O.S. command: java.exe -Xrs -classpath C:\rabbitmq-java-client-bin-3.6.5.C:\InterSystems\Ensemble\dev\java\lib\UDK18\cachejdbc.jar;C:\InterSystems\Ensemble\dev\java\lib\UDK18\cachejdbc.jar com.intersys.gateway.JavaGateway 55555 rabbitmqjava.log 2-&1

2016-10-28 17:42:52 Execution returned."

2016-10-28 17:42:52 Gateway Server successfully started

2016-10-28 17:42:52 Starting background process to monitor the Gateway Server

2016-10-28 17:42:52 Job command successful, started monitor process :2684

Done

PROXY CLASS

The object gateway provides a proxy class mechanism to execute, in this example, Java code, from within Ensemble.

RABBITMQ JAVA WRAPPER CLASS

It is usually not practical to import a complete library, so the recommendation is to create a wrapper class that provides a simplified, subset of the required functionality.

CREATE A JAVA SOURCE FILE

Example: `<root folder>\com\myorgname\rabbitmq\Wrapper.java`

```
package com.myorgname.rabbitmq;

import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.QueueingConsumer;

public class Wrapper {

    public void sendMsg(String hostName, String queueName, byte[] msg) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost(hostName);
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
        channel.queueDeclare(queueName, false, false, false, null);

        channel.basicPublish("", queueName, null, msg);

        channel.close();
        connection.close();
    }

    public int readMsg(String hostName, String queueName, byte[] msg) throws Exception {
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost(hostName);
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
        channel.queueDeclare(queueName, false, false, false, null);
        QueueingConsumer consumer = new QueueingConsumer(channel);
        channel.basicConsume(queueName, true, consumer);

        QueueingConsumer.Delivery delivery = consumer.nextDelivery();
        int len = delivery.getBody().length;
        System.arraycopy(delivery.getBody(), 0, msg, 0, len);

        channel.close();
        connection.close();

        return len;
    }
}
```

COMPILE THE WRAPPER CLASS

Compile the class using for example:

```
javac -verbose -cp C:\rabbitmq-java-client-bin-3.6.5\rabbitmq-client.jar com\myorgname\rabbitmq\Wrapper.java
```

Example output:

```
C:\rabbitmq-java-proxy-3.6.5>javac -verbose -cp C:\rabbitmq-java-client-bin-3.6.5\rabbitmq-client.jar com\myorgname\rabbitmq\Wrapper.java
[parsing started RegularFileObject[com\myorgname\rabbitmq\Wrapper.java]]
[parsing completed 30ms]
[search path for source files: C:\rabbitmq-java-client-bin-3.6.5\rabbitmq-client.jar]
...
[loading ZipFileIndexFileObject[C:\rabbitmq-java-client-bin-3.6.5\rabbitmq-client.jar(com\rabbitmq\client\AMQP$BasicProperties.class)]]
[wrote RegularFileObject[com\myorgname\rabbitmq\Wrapper.class]]
[total 631ms]
```

PACKAGE THE CLASS/ES IN A JAR FILE

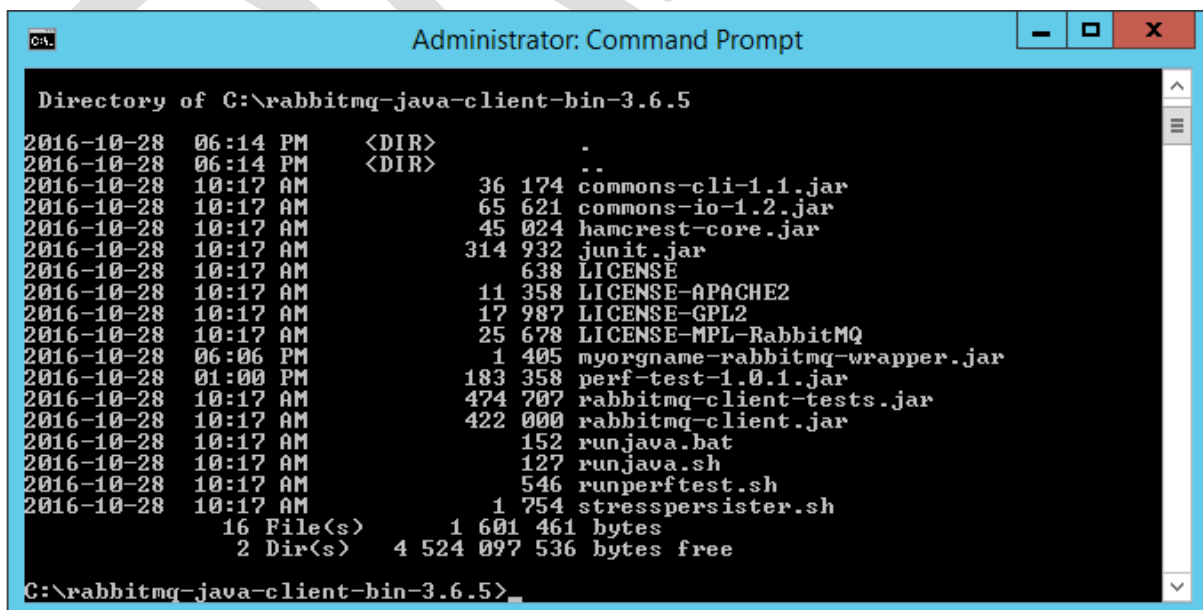
Create a jar using for example:

```
jar cvf myorgname-rabbitmq-wrapper.jar com\myorgname\rabbitmq\Wrapper.class
```

Example output:

```
C:\rabbitmq-java-proxy-3.6.5>jar cvf myorgname-rabbitmq-wrapper.jar com\myorgname\rabbitmq\Wrapper.class
added manifest
adding: com/myorgname/rabbitmq/Wrapper.class(in = 1938) (out= 899)(deflated 53%)
```

COPY THE JAR FILE TO THE RABBITMQ JAVA CLIENT INSTALL FOLDER



```
Administrator: Command Prompt

Directory of C:\rabbitmq-java-client-bin-3.6.5
2016-10-28 06:14 PM <DIR> .
2016-10-28 06:14 PM <DIR> ..
2016-10-28 10:17 AM 36 174 commons-cli-1.1.jar
2016-10-28 10:17 AM 65 621 commons-io-1.2.jar
2016-10-28 10:17 AM 45 024 hamcrest-core.jar
2016-10-28 10:17 AM 314 932 junit.jar
2016-10-28 10:17 AM 638 LICENSE
2016-10-28 10:17 AM 11 358 LICENSE-APACHE2
2016-10-28 10:17 AM 17 987 LICENSE-GPL2
2016-10-28 10:17 AM 25 678 LICENSE-MPL-RabbitMQ
2016-10-28 06:06 PM 1 405 myorgname-rabbitmq-wrapper.jar
2016-10-28 01:00 PM 183 358 perf-test-1.0.1.jar
2016-10-28 10:17 AM 474 707 rabbitmq-client-tests.jar
2016-10-28 10:17 AM 422 000 rabbitmq-client.jar
2016-10-28 10:17 AM 152 runjava.bat
2016-10-28 10:17 AM 127 runjava.sh
2016-10-28 10:17 AM 546 runperfctest.sh
2016-10-28 10:17 AM 1 754 stresspersister.sh
16 File(s) 1 601 461 bytes
2 Dir(s) 4 524 097 536 bytes free

C:\rabbitmq-java-client-bin-3.6.5>
```


ENSEMBLE PROXY CLASS

IMPORT THE WRAPPER INTO ENSEMBLE

- Open the Java Gateway Wizard in Studio:
Tools > Add-Ins > Java Gateway Wizard
- Select Jar File and enter the path and name of the wrapper jar file
- Specify the Java Gateway server name or IP address and its port number
- Select Next

The screenshot shows the 'Java Gateway Wizard' dialog box. The title bar says 'Java Gateway Wizard'. The main area has a blue header with the Studio logo and 'Java Gateway Wizard'. Below the header, it says 'This wizard will help you import a class file or a jar file from Java and create a set of corresponding classes.' There are two radio buttons: 'Jar File' (selected) and 'Class Name'. Below that, a text field contains 'C:\rabbitmq-java-client-bin-3.6.5\myorgname-rabbitmq-wrapper.jar' with a 'Browse' button. Below that, a text field contains '127.0.0.1' for 'Java Gateway server name / IP address: *'. Below that, a text field contains '55555' for 'Java Gateway server port: *'. There are two empty text fields for 'Additional classpaths to be used in finding dependent classes:' and 'Exclude dependent classes matching the following prefixes:'. At the bottom, there are buttons: 'Back', 'Next', 'Finish', 'Cancel', and 'Help'.

Java Gateway Wizard

Studio Template
Java Gateway Wizard

User: UnknownUser
Namespace: ENSEMBLE

This wizard will help you import a class file or a jar file from Java and create a set of corresponding classes.

Select a .jar file or fully qualified class name: *

☒ Jar File ☐ Class Name

Enter the path and name of a Jar file: *

C:\rabbitmq-java-client-bin-3.6.5\myorgname-rabbitmq-wrapper.jar Browse

Required.

Java Gateway server name / IP address: *

127.0.0.1

Required.

Java Gateway server port: *

55555

Required.

Additional classpaths to be used in finding dependent classes:

Specify a list of jar files or directories, separated by semi-colons.

Exclude dependent classes matching the following prefixes:

Specify a list of package and class name prefixes, separated by semi-colons.

Back Next Finish Cancel Help

- Select the Wrapper class
- Select Finish

The screenshot shows the 'Java Gateway Wizard' dialog box, Step 2. The title bar says 'Java Gateway Wizard'. The main area has a blue header with the Studio logo and 'Java Gateway Wizard'. Below the header, it says 'The Java Gateway Wizard has determined that your jar file contains the following Java classes:'. Below that, there is a table with two columns: 'Classname' and a checkbox. The first row has an unchecked checkbox and 'com.myorgname.rabbitmq.Wrapper'. The second row has a checked checkbox and 'com.myorgname.rabbitmq.Wrapper'. Below the table, there are navigation arrows '<' and '>'. At the bottom, there are buttons: 'Back', 'Next', 'Finish', 'Cancel', and 'Help'.

Java Gateway Wizard

Studio Template
Java Gateway Wizard

User: UnknownUser
Namespace: ENSEMBLE

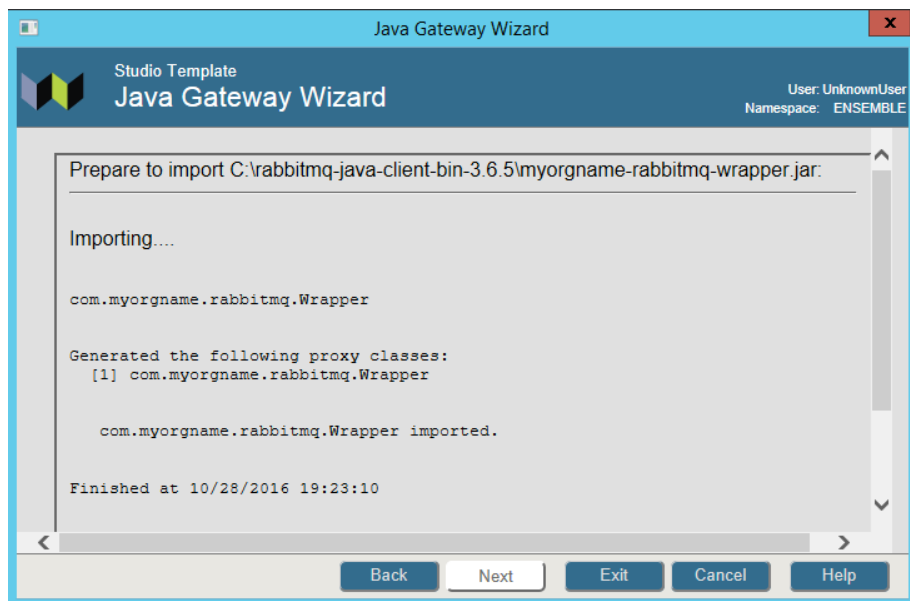
The Java Gateway Wizard has determined that your jar file contains the following Java classes:
(Check the top checkbox to select/unselect all classes)

| | Classname |
|-------------------------------------|--------------------------------|
| <input type="checkbox"/> | com.myorgname.rabbitmq.Wrapper |
| <input checked="" type="checkbox"/> | com.myorgname.rabbitmq.Wrapper |

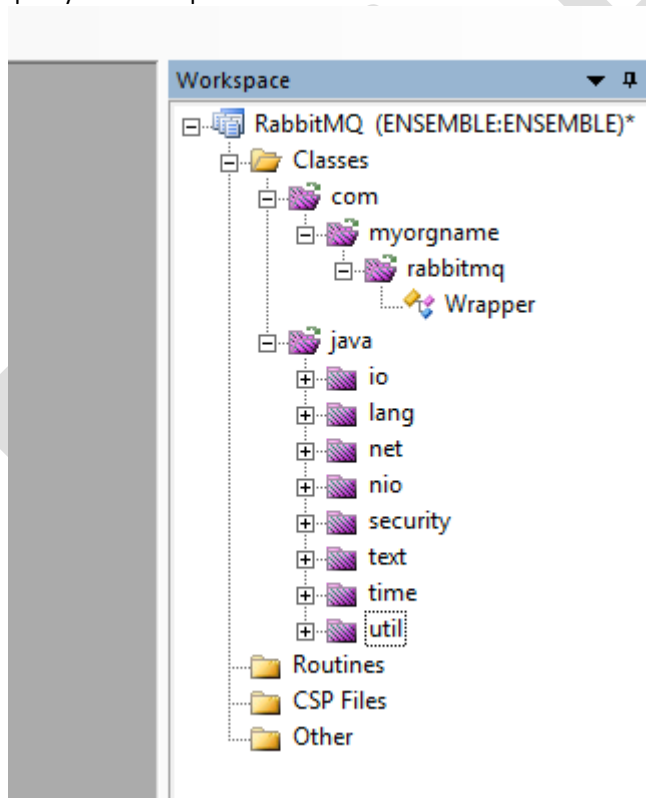
< >

Back Next Finish Cancel Help

Example result of an import:



Note the proxy classes imported:



USE THE PROXY CLASSES IN YOUR APPLICATION

CREATE A UTILITY CLASS TO TEST CONNECTIVITY TO THE RABBITMQ SERVER

Class RabbitMQ.Java.HelloWorld Extends %RegisteredObject
{

Parameter CLASSPATH = "C:\rabbitmq-java-client-bin-3.6.5\myorgname-rabbitmq-wrapper.jar";

Parameter HOST = "localhost";

Parameter QUEUE = "hello";

```
/// s sc=##class(RabbitMQ.Java.HelloWorld).SendMsg()
ClassMethod SendMsg(pMsg = "Hello from Ensemble!") As %Status
{
    #dim tGateway as %Net.Remote.Gateway
    #dim tException as %Exception.AbstractException

    Set tSC=$$$OK
    Try    {
        Set tGateway=..Connect()
        Set tRabbitMQWrapper=##class(com.myorgname.rabbitmq.Wrapper).%New(tGateway)
        /*** Do not use this syntax. It does not work! ***/
        //      Set tByteStream=##class(%Library.GlobalBinaryStream).%New("Hello
World!")

        // *****
        Set tByteStream=##class(%Library.GlobalBinaryStream).%New()
        Set tSC = tByteStream.Write(pMsg)
        Do tRabbitMQWrapper.sendMsg(..#HOST,..#QUEUE, tByteStream)
        Write !,"Sent message via "_tByteStream.Read()

        Set tSC=tGateway.%Disconnect()
    } Catch tException {
        Set tSC = tException.AsStatus()
    }

    Quit tSC
}
```

```
/// s sc=##class(RabbitMQ.Java.HelloWorld).ReadMsg()
ClassMethod ReadMsg(pMsgLen = 32000) As %Status
{
    #dim tGateway as %Net.Remote.Gateway
    #dim tException as %Exception.AbstractException

    Set tSC=$$$OK
    Try    {
        Set tGateway=..Connect()
        Set tRabbitMQWrapper=##class(com.myorgname.rabbitmq.Wrapper).%New(tGateway)

        Set tReadStream=##class(%GlobalBinaryStream).%New()
        // we need to 'reserve' a number of bytes since we are passing the stream
        // by reference (Java's equivalent is byte[] ba = new byte[max];)
        For i=1:1:pMsgLen Do tReadStream.Write("0")
    }
```

```

        Set tBytesRead=tRabbitMQWrapper.readMsg(..#HOST,..#QUEUE, .tReadStream)
        Write tReadStream.Read(tBytesRead),!
        Write "Bytes Read: ",tBytesRead,!

        Set tSC=tGateway.%Disconnect()

    } Catch tException {
        Set tSC = tException.AsStatus()
    }

    Quit tSC
}

ClassMethod Connect(pPort As %Integer = 55555, pHost As %String = "127.0.0.1") As %Net.Remote.Gateway
{
    // connect to current namespace, use 2 second timeout
    Set tSC=$$$OK,tNamespace=$zu(5),tTimeout=2
    Set tClassPath=##class(%ListOfDataTypes).%New()
    Do tClassPath.Insert(..#CLASSPATH)

    // get a connection handle and connect
    Set tGateway=##class(%Net.Remote.Gateway).%New()
    Set tSC=tGateway.%Connect(pHost,pPort,tNamespace,tTimeout,tClassPath)
    If tSC'=$$$OK {
        Write $system.OBJ.DisplayError(tSC)
        Set tGateway=""
    }
    Quit tGateway
}

```

TEST THE UTILITY CLASS



```

Cache TRM:6648 (ENSEMBLE)
File Edit Help

ENSEMBLE>s sc=##class(RabbitMQ.Java.HelloWorld).SendMsg()

Sent message via Hello from Ensemble!
ENSEMBLE>s sc=##class(RabbitMQ.Java.HelloWorld).ReadMsg()
Hello from Ensemble!
Bytes Read: 20

ENSEMBLE>

```