Creating a WebSphere MQ Virtual Service using CA Service Virtualization (DevTest) 8.0.X

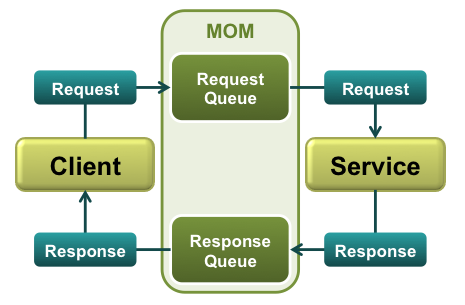
Overview:

The purpose of this document is to show how you can create a virtual image by using CA Service Virtualization (SV) MQ Recording and playback through WebSphere MQ proxy queues instead of WebSphere MQ live queues so you can develop and test in an unconstrained manner with live-like data and real application behavior.

In a virtualized environment, the CA SV virtual service stands in for the server application and provides the responses to the client application. Similarly, in order to record transactions, the SV recorder must be placed between the two applications. CA SV monitors requests and proxies them to the server application queues. CA SV also monitors responses and proxies them back to client application queues.

Agenda:

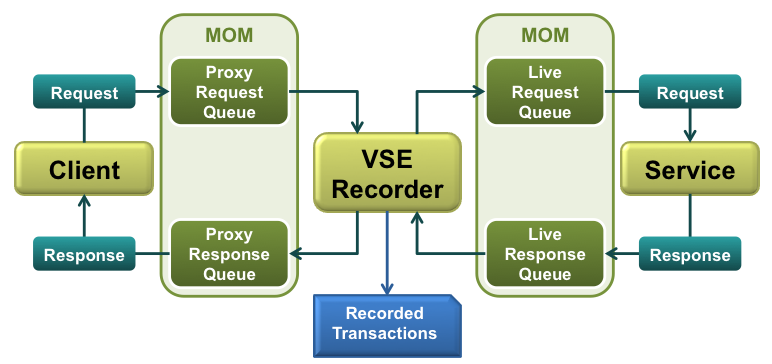
1. Messaging middleware basics and SV’s role in removing it’s constraint for development and testing
2. What you’ll need to demonstrate MQ service virtualization
   * IBM WebSphere MQ 7.0.1+ software download and installation
   * IBM WebSphere MQ software configuration (queue manager, queues, etc.)
   * IBM WebSphere MQ client jar’s for SV Workstation
   * The SV project for demo execution
3. Demonstrating MQ Service Virtualization using live system recording
4. Here is an architectural depiction of messaging middleware message flow, facilitated by a typical Message Oriented Middleware solution like IBM WebSphere MQ Series, SonicMQ, Oracle ESB, Mule, or any standard JMS-based implementations



The client sends request to the MQ live request queue in order to query data or invoke a service on the server.

The server reads the request from the live request queue, sending a response back to the Client containing the data that was queried or the result of the action. In messaging applications there can be more than one response for each request. The Client then reads the response from the live response queue.

Now let’s see how we can record using the proxy queue approach:



Proxy recording is the most reliable and widely-support method for recording a messaging application.  The VSE Recorder is inserted into the message flow between the Client and the Service.  Every request and response message has to flow through the VSE Recorder before it reaches the other side.

Proxy Request and Response Queues need to be created by the MQ Administrator on the Messaging Platform. Note: When preparing for a POC, it’s important to get the customer’s commitment to have these Proxy Queues ready a week before your SV installation

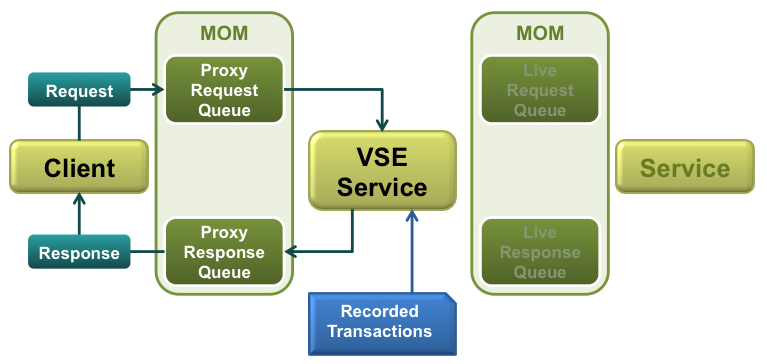
Proxy Request Queue: The request queue on which the Client sends the request. The client requires reconfiguring to point to the proxy queues.

Proxy Response Queue: The response queue from which the Client receives responses. This will be different from the application's normal response queue.

So:

* Client communicates with the Proxy Queues
* Server continues to communicate with the Live Queues
* SV VSE Recorder (on User’s SV Workstation) is in the middle recording and routing traffic between Client and Service

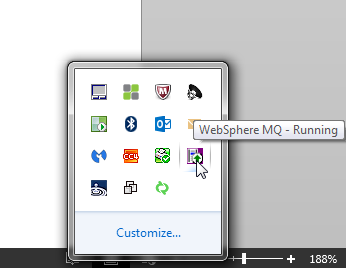
Virtual Service Playback:

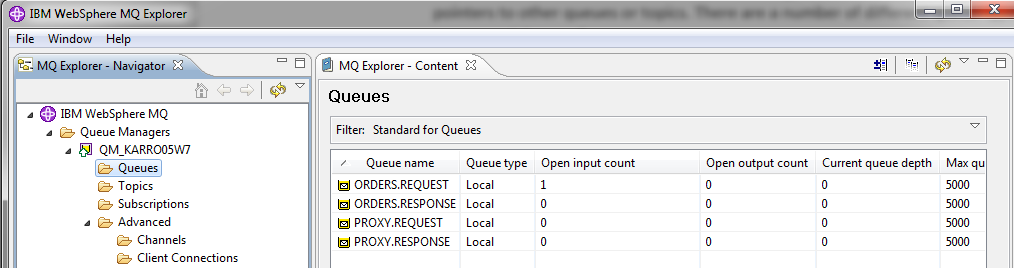


During playback the VSE Service takes the place of the VSE recorder. Instead of forwarding requests and responses between the proxy and live queues, the VSE recorder short-circuits the flow and cuts the live queues and the service out of the picture.

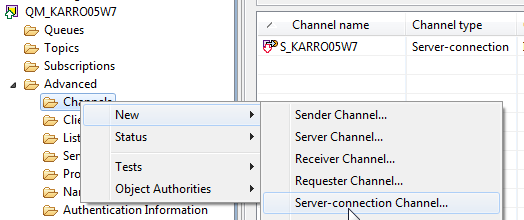
Currently the VSE service is generated to use the Proxy Queues, which means the client does not require reconfiguring between recording and playback. However, the user can easily change the queues which the VSE service operates by selecting different queues in the VSM's Listen and Respond steps.

1. What you’ll need to setup to demonstrate MQ service virtualization
   * IBM WebSphere MQ 7.0.1+ software download and installation
   * IBM WebSphere MQ software configuration (queue manager, queues, etc.)
   * IBM WebSphere MQ client jar’s for SV Workstation
   * The SV project for demo execution

* IBM WebSphere MQ software download and installation
  + Download WebSphere for MQ release 7 or 8 (I think you can get a temporary license from IBM if you don’t already own WAS for MQ)
  + Unzip the binaries and invoke the installer from Setup.exe in the IBM root directory
  + Under Software Requirements, the Eclipse 3.3 version is included in /IBM/prereqs/IES. Invoke the Setup.exe. This is used to bring up the WebSphere MQ Explorer client to add your queue manager and live/proxy queues
  + Once the software requirements are confirmed, select WebSphere MQ Installation and select “Launch IBM WebSphere MQ Installer”
  + Perform a typical installation
  + When the “Welcome to the Prepare WebSphere MQ Wizard” dialog appears, select next
  + When asked “Are any of the domain controllers in your network running Windows 2000 or greater” select No.
  + At “Completing the Prepare WebSphere MQ Wizard” dialog, ensure the “Launch WebSphere MQ Explorer” is checked and select Finish.
  + Note: If there are any issues with permissions and if the “Prepare” process prompts for your Administration password, you’ll need to go to Control Panel, User Accounts, Manage User Accounts and reset your Administrator’s password that will be used in the MQ setup.
* IBM WebSphere MQ software configuration (queue manager, queues, etc.)
  + Invoke the IBM WebSphere MQ Explorer client (you can right click on the tray icon) to invoke it: 
  + Queue manager: The first activity for a WebSphere MQ administrator is to create a queue manager. The queue manager coordinates all the applications updating messages on queues, and deals with the isolation and locking that are required to maintain consistency.
  + Queues: Queues are used either to store messages that are sent by programs or are pointers to other queues or topics. There are a number of different queue types and some special uses of queues.
  + Queue types: For simplicity and demonstration purposes, we going to use local queues. A local queue is the only place where messages are physically stored. All other queue types must be defined to point at a local queue. We support all WAS MQ queue types
  + Create a new Queue Manager, accepting the defaults. It should have the name of your host and look like QM\_ KARRO05W7.
  + Create four local queues, accepting the default settings:
    - ORDERS.REQUEST
    - ORDERS.RESPONSE
    - PROXY.REQUEST
    - PROXY.RESPONSE



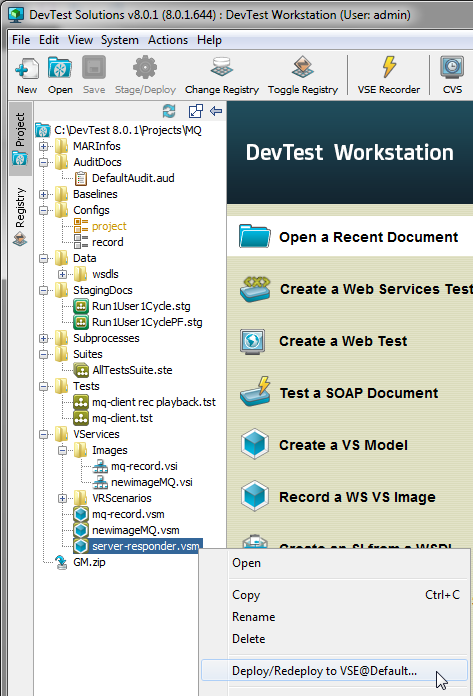
* + Create the server connection channel (e.g., S\_KARRO05W7)



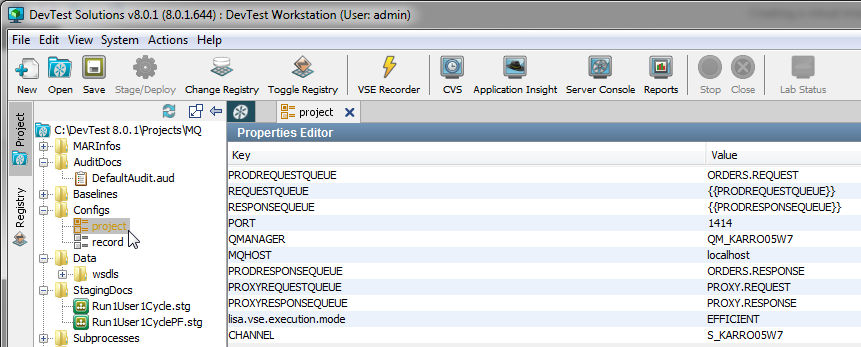
* IBM WebSphere MQ client jar’s for SV Workstation
  + Download the MQ Client Jar’s from the IBM site or you can use my 7.0.1 client jars
  + Unzip and place the binaries in DevTestRoot/lib/core
  + There should be 13 jar files
* The SV project for demo execution
  + Download the SV project for creating and executing the MQ demo from the dropbox:
  + Unzip the file and include in DevTestRoot/projects

1. Creating a WebSphere MQ virtual service by recording and playback:

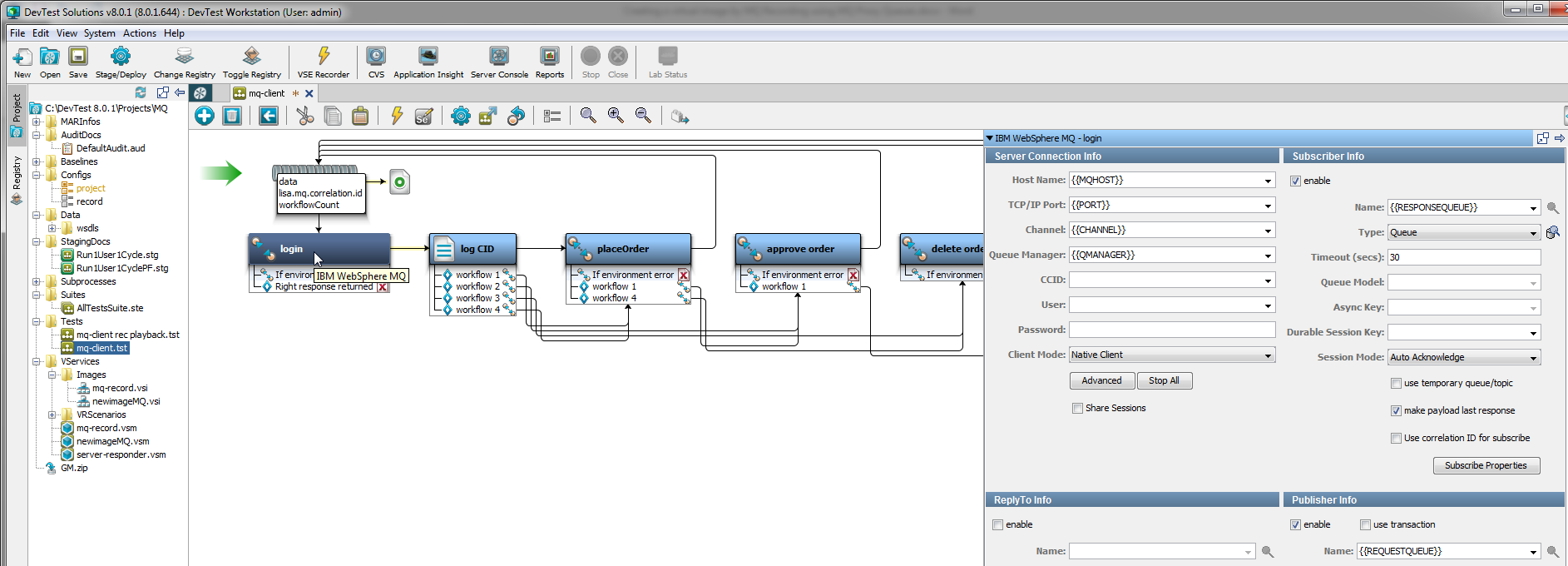
* Invoke the SV DevTest Console in Firefox: <http://localhost:1505>
* Deploy server-responder.vsm to the VSE environment. This is a pre-configured virtual service is in lieu of an actual application server (service) to read the request from the live request queue, sending a response back to the Client containing the data that was queried.



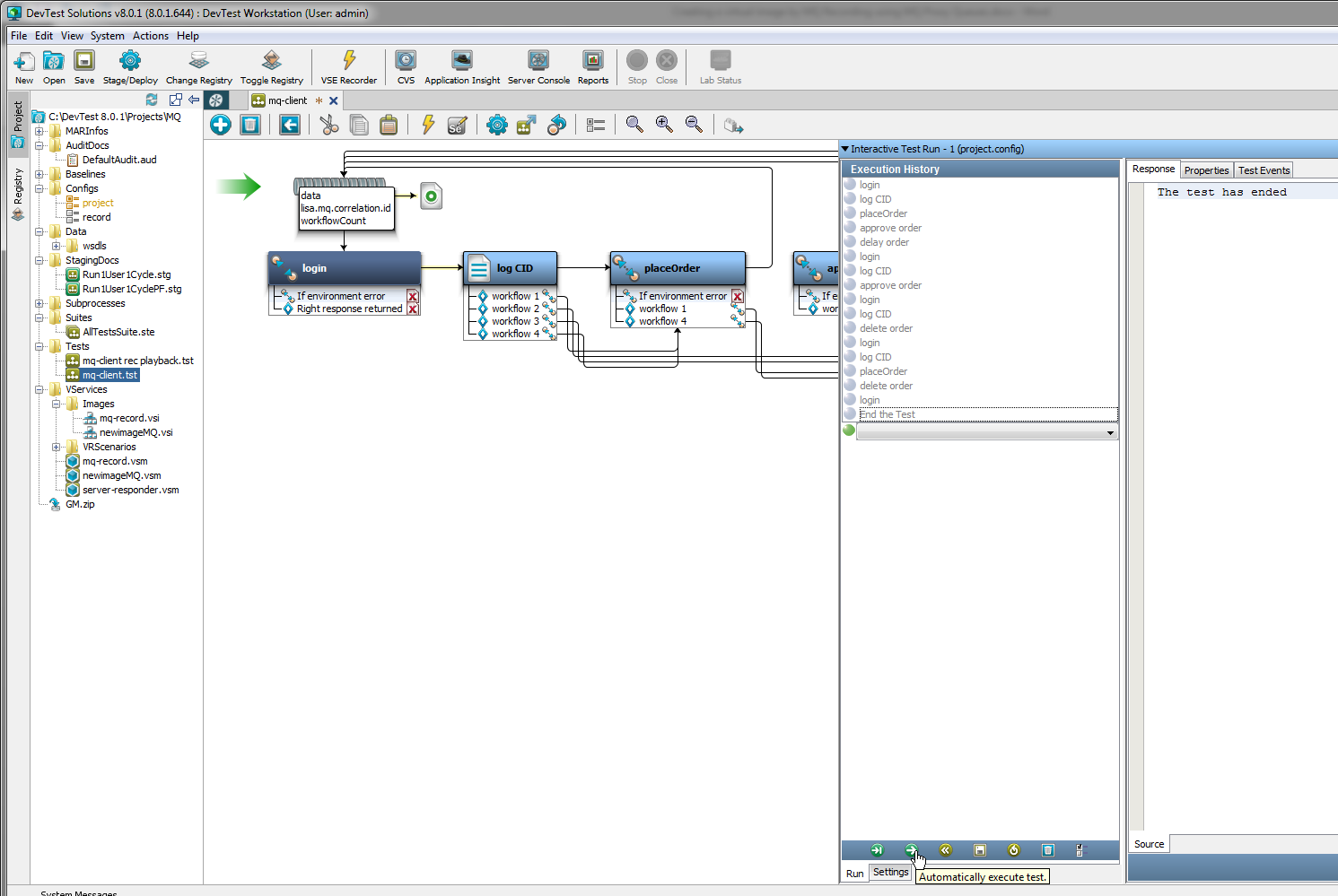
* Open the project configuration file to observe the property editor key and value entries you’ll use to demo. The {{REQUESTQUEUE}} and {{RESPONSEQUEUE}} are property designations for the MQ live queues and their associative values of {{PRODREQUESTQUEUE}} and {{PRODRESPONSEQUEUE}} will be used to create the proxy queues. We’ll hit the live queues first to validate the environment is working.



* Open the mq-client.tst test case and change the following blocks to the live {{REQUESTQUEUE}} and {{RESPONSEQUEUE}}:
  + Login
  + placeOrder
  + approve order
  + delete order
  + delay order



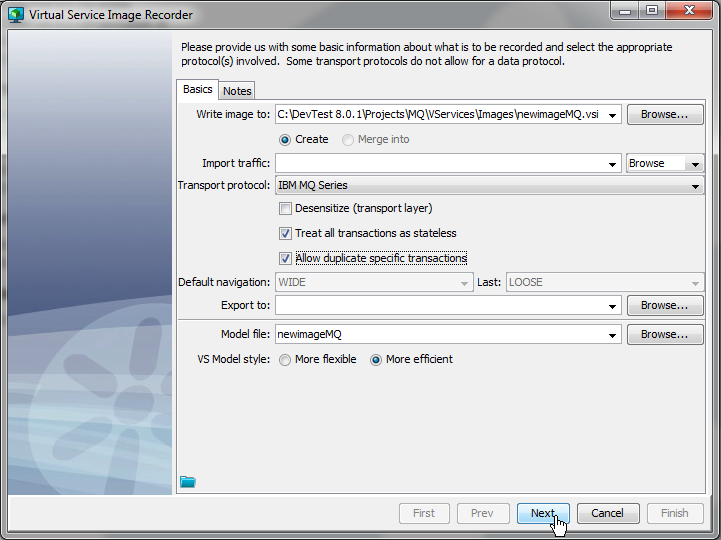
* Invoke the ITR and execute the test



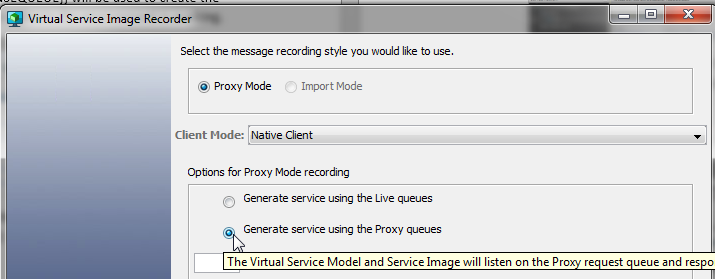
* Confirm 11 transaction against the server-responder model in the DevTest Console



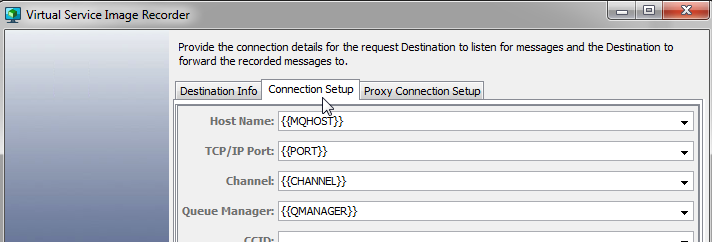
* Reset the server-responder transaction count
* Create a new virtual image by recording
  + newimageMQ.vsi
  + newimageMQ.vsm
  + transport protocol: IBM MQSeries



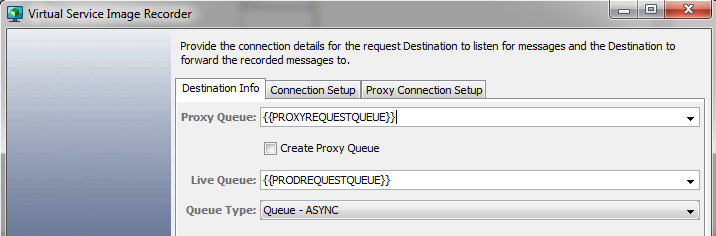
* + next: Generate the service using the proxy queues



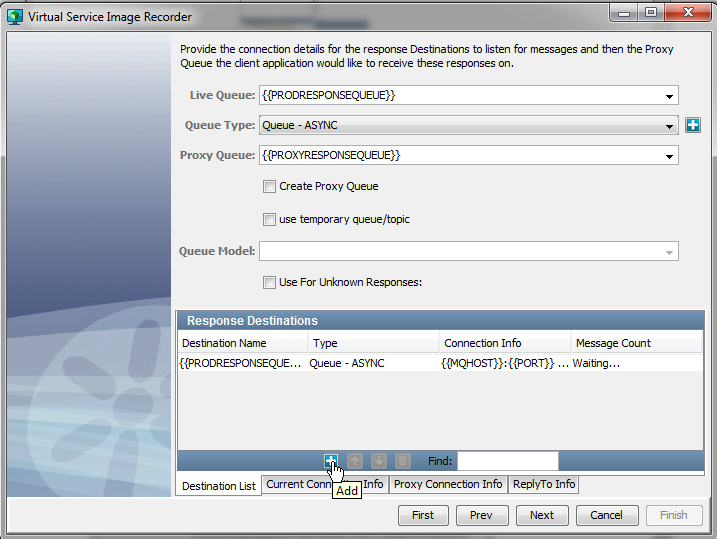
* + Select the Connection Setup tab and use the parameterized keys for the following:
    - Host Name: {{MQHOST}}
    - TCP/IP port: {{PORT}}
    - Channel: {{CHANNEL}}
    - Queue Manager: {{QMANAGER}}



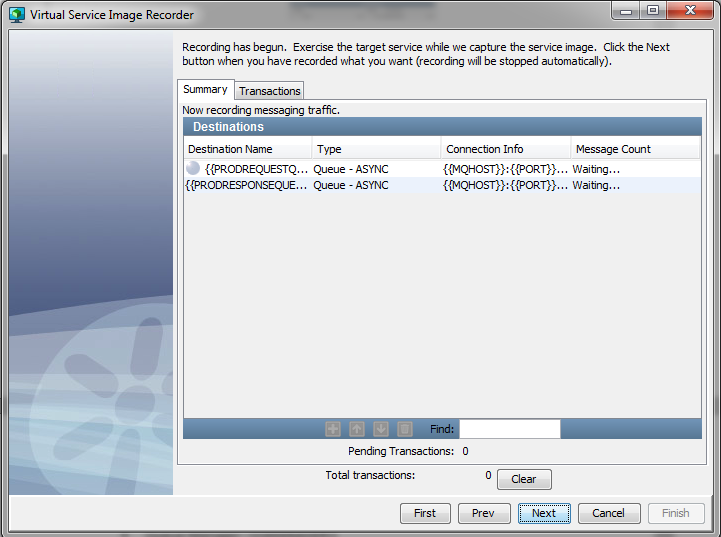
* + Selection the Destination Info tab
    - Request destination live queue: {{PRODREQUESTQUEUE}}
    - Request destination proxy queue: {{PROXYREQUESTQUEUE}}



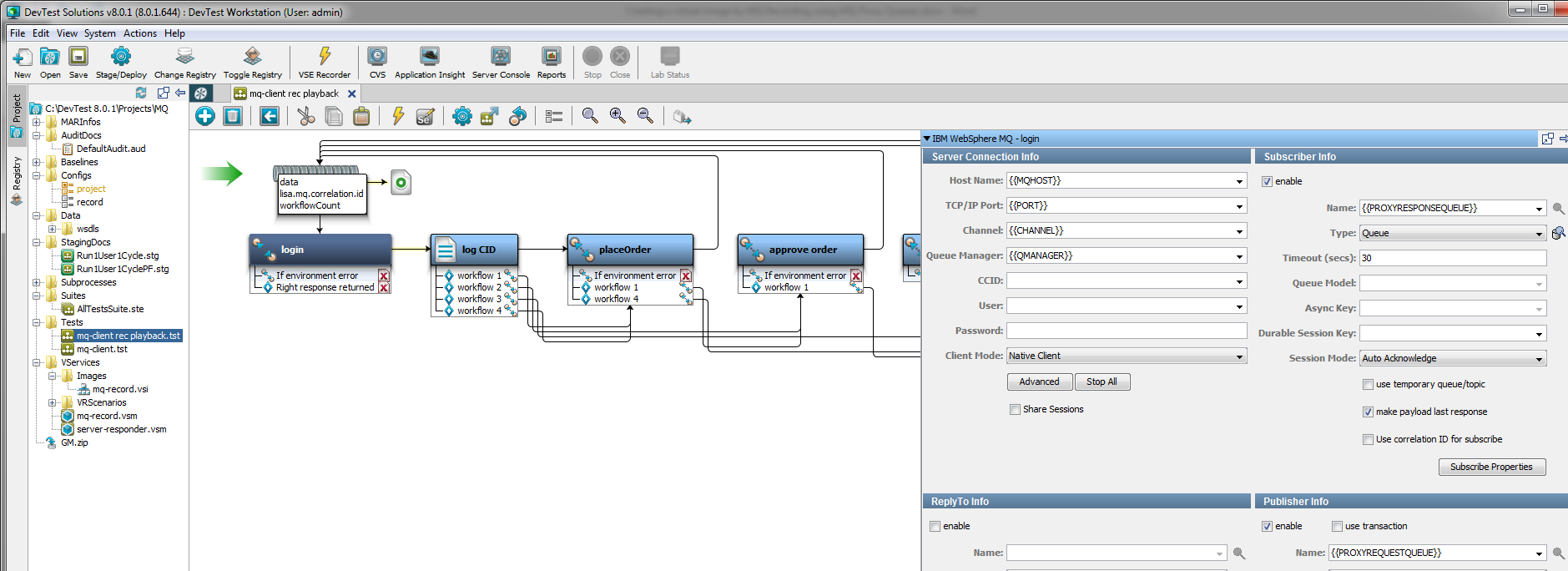
* + - Next : response destinations
    - Response destination proxy queue: {{PROXYRESPONSEQUEUE}}
    - Response destination live queue: {{PRODRESPONSEQUEUE}}
    - Press the ‘+’ button on the Response Destination list to add the queue designations (ensure the Current Connection Info tab is set with the Connection Setup properties in the previous step)



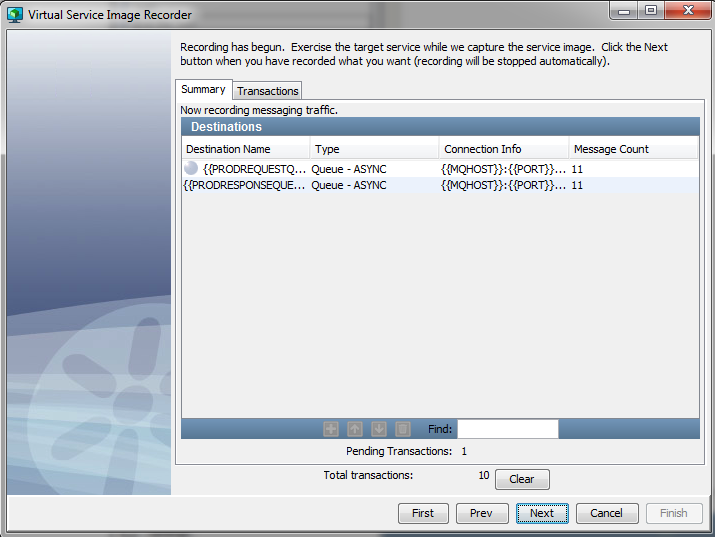
* + - Next: Recording has begun. Open up your mq-client.txt and change the 5 blocks to the proxy queue values {{PROXYREQUESTQUEUE}} and {{PROXYRESPONSEQUEUE}}



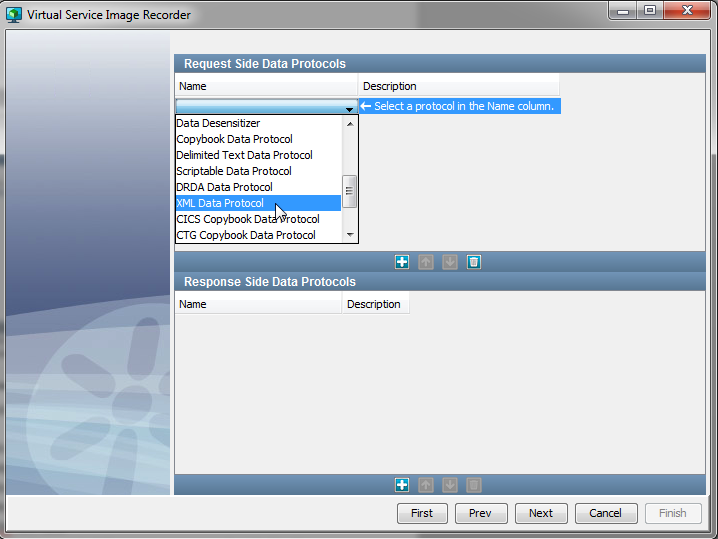
* Open the record and replay test client, invoke the ITR and execute the test



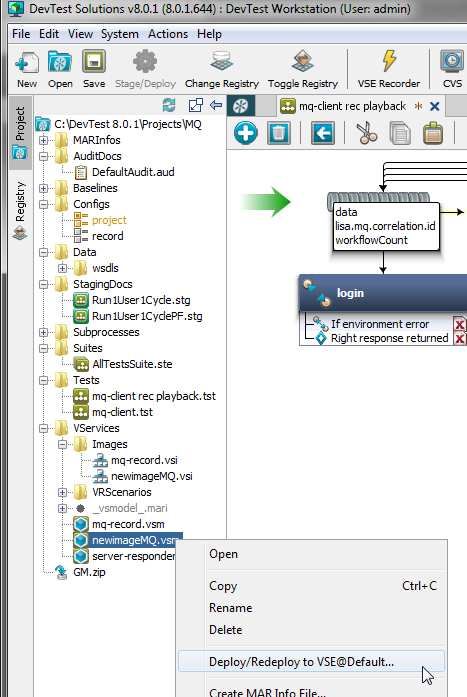
* Confirm 11 messages and 10 transaction recorded



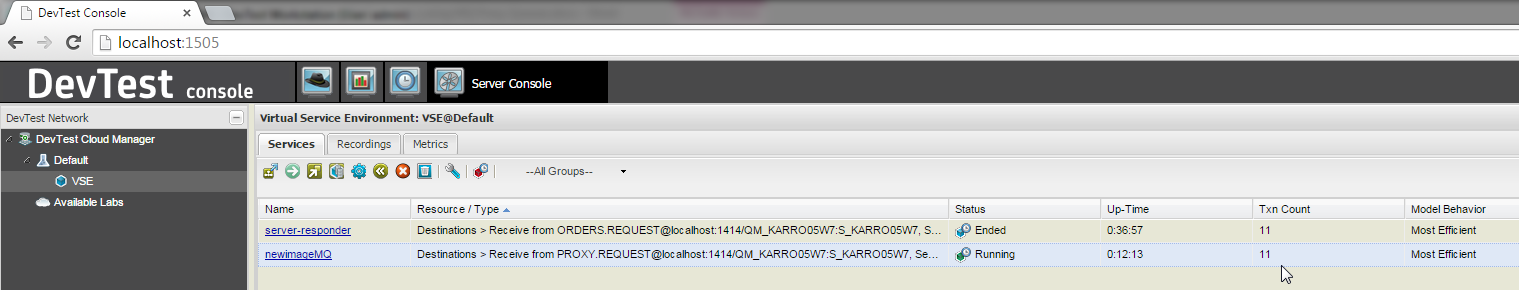
* Next: Add the XML Data Protocol as the request side data protocol. No response side data protocol needed



* Deploy the newimageMQ.vsm to the DevTest Console



* Stop the server-responder in the DevTest Console
* Reset the ITR for the record and replay test case
* Run the mq-client.txt in the ITR, this time against the proxy queues as depicted in the Virtual Service Playback illustration
* Go to the DevTest Console and confirm the 11 transactions using the proxy queues



* Select the newimageMQ services, click on the Inspection View icon in the toolbar to observe the matched transactions.