WebSphere MQ – Transaction Handler Test Tool Version 1.0

1 November 2007

Annette Green
IBM Global Services
6710 Rockledge Drive
Bethesda, Maryland 20817

greenac@us.ibm.com

Take Note!

Before using this report be sure to read the general information under "Notices".

First Edition, November 2007

This edition applies to Version 1.0 of *WebSphere MQ – Transaction Handler Test Tool* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Notices

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed “AS-IS”. The use of this information and the implementation of any of the techniques is the responsibility of the reader. Much depends on the ability of the reader to evaluate these data and project the results to their operational environment.

The performance data contained in this report was measured in a controlled environment and results obtained in other environments may vary significantly.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- WebSphere
- AIX
- Rational

The following terms are trademarks of other companies:

- Windows 2000, 2003, XP Microsoft Corporation
- Java Sun Microsystems, Inc.

Summary of Amendments

Date	Changes
1 November 2007	Initial release

Preface

This WebSphere MQ – Transaction Handler Test Tool is a Java utility program which simulates the processing of real-time business transactions between systems that interact using WebSphere MQ 6.0 or above. This highly flexible tool is geared towards developers and testers who need to verify application code against an interfacing system that may not be available or ready for integration. In place of the actual system, the user code interfaces with the Transaction Handler Test Tool via WebSphere MQ and, in turn, the test tool replicates the processing of the backend application (i.e. J2EE application, Web Service, legacy application). The Transaction Handler Test Tool can run as a standalone Java application; as a J2EE application client, connecting to WebSphere Application Server (WAS); or as a Web-based application running on WAS. Using a graphical user interface, the user can configure the Transaction Handler Test Tool to run either remotely as an MQ client; as a local program directly connected to an MQ server; or as a JMS client using Java Naming and Directory Interface (JNDI) to look up and connect to message queues.

This document describes the Transaction Handler Test Tool, which was developed to support testing efforts of the middleware transport layer. The document provides instructions on how to set up, run, and use the Transaction Handler Test Tool application. In addition, the document describes each of the screens used by the application.

Bibliography

- *WebSphere MQ Using Java,* IBM Corporation. SC34-5456.
- *WebSphere MQ Application Programming Reference*, IBM Corporation. SC33-1673
- *WebSphere MQ Application Programming Reference Summary*, IBM Corporation. SX33-6095
- *WebSphere MQ Application Programming Guide*, IBM Corporation. SC33-0807

Chapter 1. Overview

Using WebSphere MQ 6.0+, the Transaction Handler Test Tool provides users with the ability to test the processing of real-time business transactions without having to interface with a real backend system. Transaction Handler is a Java utility program which simulates the processing of real-time business transactions between systems that interact using WebSphere MQ; with WebSphere Application Server directly interoperating with WebSphere MQ; or with the default messaging provider incorporated within WebSphere Application Server (also known as the embedded WebSphere JMS provider for version 5.X or service integration bus for version 6.X). The Transaction Handler Test Tool is geared towards testers or developers who need to verify application code against an interfacing system that may not be available or ready for integration. In place of the actual system, the test tool can be used to replicate the processing of the backend application (such as a J2EE application, Web service, or legacy application).

Transaction Handler simply works by automatically retrieving messages placed on a user-specified request queue and then placing new application-specific messages onto a designated response queue. As a result, Transaction Handler frees testers and developers from the restrictions in accessing a separate system; allows users to validate round-trip business transactions prior to integration testing; and allows them to identify and minimize integration issues during development and early test phases.

Using a graphical user interface (GUI), the user configures the Transaction Handler Test Tool to specify 1) where to receive and send messages from/to the user code, 2) where to retrieve the content for reply messages, 3) whether to write request messages to a file, and 4) whether a time delay is expected between retrieving a message and sending a reply back to the user code. Transaction Handler can run as a standalone GUI application; as a J2EE application client, connecting to WebSphere Application Server (WAS); or as a Web-based application running on WAS. Transaction Handler uses both the WebSphere MQ base Java classes and the WebSphere MQ JMS classes to give the user the option of running the test tool either remotely as an MQ client; as a local program directly connected to an MQ server; or as a JMS client using Java Naming and Directory Interface (JNDI) to look up and connect to message queues.

Chapter 2. Functional Description

This chapter describes in more detail the three aspects of the WebSphere MQ - Transaction Handler Test Tool:

1) As a standalone application
2) As a J2EE client application, running within an application client container
3) As a Web-based application, running on an application server

Transaction Handler as a Standalone Application

As a standalone application, the user can start up the Transaction Handler Test Tool with either a WebSphere MQ-specific user display or a JMS-specific user display. In the case of the MQ-specific display screen, the user can configure the Transaction Handler Test Tool to run as either a WebSphere MQ client or directly connected to a WebSphere MQ server. In the case of the JMS-specific display screen, the Transaction Handler Test Tool will run as a JMS client and interact with message queues that are defined within a JNDI repository (either an LDAP or a file system based directory). In both cases, the Transaction Handler graphical user interface (GUI) allows the user to enter configuration information; initiate the processing of messages; and view the status of the message processing. Static configuration information can also be stored in properties files used by the test tool. When the Transaction Handler Test Tool starts up, the information stored in the configuration file will appear as default values within the graphical user interface.

The following subsections describe the differences in the two Transaction Handler user interfaces as well as describe how the test tool processes messages. For complete field-by-field descriptions on each of the screens, please see Chapter 4. ,"User Interface Description".

Describing the WebSphere MQ-specific Interface

When the Transaction Handler Test Tool displays the MQ-specific display screen, then the user has initiated the test tool using the WebSphere MQ-base Java classes. Within the MQ-specific display screen, the user can specify configuration information such as the server name, the queue manager name, the names of the message queues, the channel for connecting to the queue manager, and the port for listening to incoming connection requests. If the Transaction Handler Test Tool is running on the same machine as the queue manager, then the server and channel information are optional. These two fields are required though for running the application as an MQ client.

Once the user completes the configuration, the Transaction Handler Test Tool uses this user-supplied information to retrieve request messages from one queue and place application-specific response messages onto another queue. The same queue manager must manage both user-specified queues. The Transaction Handler Test Tool (referred to hereafter as “TransHandler”), processes messages according to the following rules:

- TransHandler looks for messages on the request queue in first-in first-out order. If no messages exist, then the status is displayed to the user and the application will continue to poll for messages until either the user manually stops the processing or TransHandler has been set to automatically disconnect after a specified number of iterations.
- When retrieving a message from the request queue, if the queue contains a logical message that has been segmented into multiple segments, then TransHandler will use WebSphere MQ to retrieve all segmented parts and reassemble them back into one single message.

- If specified by the user, the body of the retrieved response message gets written to a file in a directory designated by the user. The filename containing the message content will use the following naming convention: `WMQJ.<Current Date>.<Current Time>.txt`. Note that although the file extension indicates the content is text only, the format of the message body is preserved and can be binary.
- The body of the response message is taken from a file located in a directory specified by the user. The file format for the response message body can be either text or binary. If multiple files exist within this directory, then on subsequent iterations, a different file is used to create the message body.
- The message descriptor header of the response message is generated based on the following conventions:
    - The request message Message Id value is copied to the response message Correlation Id.
    - The response message Message Id value is automatically generated by WebSphere MQ.
    - The response message Message Type is set to MQMT_REPLY.
    - The request message Format is copied to the response message Format.
    - The response message Persistence is set to MQPER_PERSISTENT.
- If specified by the user, the application will wait a number of seconds before placing the response message onto the designated response queue.

This processing will continue unless manually interrupted by the user or unless the application is configured to automatically disconnect after a set number of iterations elapses. During the processing, the status of all application operations (MQ and non-MQ) is redirected to the user interface for the user to view.

Describing the JMS-specific Interface

When the Transaction Handler Test Tool displays the JMS-specific display screen, then the user has initiated the test tool using the WebSphere MQ JMS classes. This aspect of the Transaction Handler Test Tool allows the user to access JMS resources defined within a JNDI repository. Within the JMS-specific display screen, the user can specify configuration information such as the JNDI names for the message queues and the JNDI names for their corresponding queue connection factories. The queues looked up from the directory service must both be defined on the same queue manager.

Once the user completes the configuration, the Transaction Handler Test Tool uses this user-supplied information to retrieve request messages from one queue and place application-specific response messages onto another queue. The same queue manager must manage both user-specified queues. The JMS-version of Transaction Handler Test Tool (referred to hereafter as “JMSTransHandler”), processes messages according to the following rules:

- JMSTransHandler looks for messages on the request queue in first-in first-out order. If no messages exist, then the status is displayed to the user and the application will continue to poll for messages until either the user manually stops the processing or JMSTransHandler has been set to automatically disconnect after a specified number of iterations.
- If specified by the user, the body of the retrieved response message gets written to a file in a directory designated by the user. The filename containing the message content will use the following naming convention: `WMQJ.<Current Date>.<Current Time>.txt`. Unlike the MQ-specific version, JMSTransHandler processes the message content as text only.

- JMSTransHandler generates a JMS TextMessage to send back to the user code. The body of the response message is taken from a file located in a directory specified by the user. If multiple files exist within this directory, then on subsequent iterations, a different file is used to create the message body.
- The header fields of the response message are generated as follows:
  - The request message JMSMessageID value is copied to the response message JMSCorrelationID.
  - The response message JMSMessageID value is automatically generated by WebSphere MQ.
  - The JMSTimestamp field is set to the current time before sending the response message.
  - The response message JMSDeliveryMode field is set to DeliveryMode.PERSISTENT
- If specified by the user, the application will wait a number of seconds before placing the response message onto the designated queue.

This processing will continue unless manually interrupted by the user or unless the application is configured to automatically disconnect after a set number of iterations elapses. During the processing, the status of all application operations (messaging-related and non-messaging related) is redirected to the user interface for the user to view.

Transaction Handler as a J2EE Application Client

As a J2EE application client, the Transaction Handler Test Tool operates with the JMS-specific interface, using the WebSphere MQ JMS classes to retrieve request messages from one queue and place application-specific messages onto another queue. This aspect of the Transaction Handler Test Tool allows the user to access JMS resources defined on an instance of WebSphere Application Server. The user interface and the message processing of the application client version of the Transaction Handler Test Tool are identical to the JMS standalone version. For the description, please see section “Describing the JMS-specific Interface” within this chapter. For details on installing Transaction Handler as a J2EE application client, please see section “Installing the J2EE Application Client”.

Transaction Handler as a Web-based Application

As a Web-based application, the Transaction Handler Test Tool operates with the JMS-specific interface, using the WebSphere MQ JMS classes to retrieve request messages from one queue and place application-specific response messages onto another queue. This aspect of the Transaction Handler Test Tool allows the user to access JMS resources defined on an instance of WebSphere Application Server. In this case, the user enters the JMS configuration information within the test tool’s main Web page, and then submits this information for processing.

As with both the standalone and J2EE application client version, the Web-based version allows the user to specify the number of iterations for processing messages from the MQ server. However, unlike the other two versions, this value is required. It is required because the Web-based version of the Transaction Handler Test Tool does not provide a mechanism for the end user to manually interrupt message processing from the Web interface. With this exception, the rules for processing each of the messages is the same as provided in section “Describing the JMS-specific Interface” within this chapter.

During the processing, the status of all application operations (messaging and non-messaging related) is redirected to the Web browser for the user to view. Once the processing is complete, the application allows the user to restart processing, using the previously-entered configuration values as defaults.

Chapter 3. Setup and Configuration

This chapter describes how to install and configure the WebSphere MQ – Transaction Handler Test Tool.

Prerequisites

This section lists the required hardware and software components needed to run the Transaction Handler Test Tool.

Software Requirements

The following third party software components must be installed in order to properly run the Transaction Handler Test Tool:

- IBM WebSphere MQ Java Version 6.0, or above
- Java Runtime Environment 1.3.1 SR10, or 1.4.2 SR5, or above

The following additional third party software components may need to be installed depending upon the runtime configuration of the Transaction Handler application.

For the Standalone Application version:

- WebSphere MQ server software, version 6.0 or above
- JNDI service provider (LDAP or file system-based directory service)

For J2EE Application Client and Web-based Application version:

- WebSphere Application Server version 5.1 or above, or WebSphere Business Integration product (including WebSphere Business Integration Software Foundation, WebSphere Enterprise Service Bus, WebSphere Process Server) version 5.1 or above
- WebSphere MQ server software, version 6.0 or above

For running within an integration development environment (IDE):

- WebSphere Studio Application Developer/Rational Application Developer/WebSphere Integration Developer version 5.1 or above
  - WebSphere Test Environment
  - WebSphere MQ Embedded Messaging (which must be manually installed from the IDE installation media)
- WebSphere MQ server software (version 6.0 or above)

The table below displays the possible software configurations for the Transaction Handler Test Tool:

Figure 1 - Software Configuration

	WebSphere MQ Java	JRE	WebSphere MQ Server	JNDI Service Provider	WAS/WBI Server	WebSphere Application Client	WebSphere Test Environment	WebSphere MQ Embedded Messaging/ Default JMS Provider
Standalone version								
WebSphere MQ Bindings Mode	X	X	X (local)					
WebSphere MQ Client Mode	X	X	X (remote)					
WebSphere JMS Client	X	X	X (local or remote)	X				
J2EE Application Client version	X	X	+		X	X		+
Web-Based Application version	X	X	+		X			+
WebSphere Studio/Rational Application Developer								
Standalone - MQ Bindings	X	X	X (local)					
Standalone - MQ Client	X	X	X (remote)					
Standalone - JMS Client	X	X	X (local or remote)	X				
J2EE Application Client	X	X	+				X	+
Web-based Application	X	X	+				X	+

Note:

- o "X" represents a required component
- o "+" represents an optional component, however, at least one of these components must be used.

Hardware Requirements

The Transaction Handler Test Tool *should* be able to run on the following hardware platforms that support the WebSphere MQ Version 6.0+ product:

- Windows (2000, Server 2003, XP)
- UNIX (AIX, HP-UX, Linux, Solaris)

In addition, the Transaction Handler Test Tool requires some type of graphical display environment, such as Windows or X Windows.

Note: The Transaction Handler application has only been set up and tested on the both the Windows XP and AIX platforms, with a file-based JNDI service, WebSphere Application Server Express 6.0, WebSphere Studio Application Developer, Rational Application Developer.

Installation Instructions

The following subsections provide instructions on installing and starting up the Transaction Handler Test Tool as a standalone application, as a J2EE application client, and as a Web-based application. Instructions are provided for connecting directly to a WebSphere MQ queue manager, for connecting to a queue manager as an MQ client, for connecting to a messaging server via JNDI lookups, for connecting to or running within WebSphere

Application Server, and for running within WebSphere Studio or Rational Application Developer.

Installing the Standalone Application

This section provides instructions on installing and running the Transaction Handler Test Tool as a standalone application.

1. Install the required software components mentioned in the "Prerequisites" section. This version of the Transaction Handler application assumes WebSphere MQ has been installed either on the local machine or on a machine that is accessible by the local machine.

    a. If the user wants to connect the Transaction Handler application directly to a WebSphere MQ queue manager (i.e. bindings mode), then the WebSphere MQ server software must be installed on the local machine.

    b. If the user wants to run the JMS version of the application, JMSTransHandler, then a JNDI provider must be installed (either an LDAP or file system-based implementation).

    For developers, this version of the application can also be run within WebSphere Studio or Rational Application Developer. See the following subsection, Installation of Standalone Application for a Development Environment for details.

2. After the software components in step 1 have been installed, take note of the directory where the WebSphere MQ Java libraries have been installed as well as the directory where the JRE (Java Runtime Environment) executable is located.
3. Copy the `TransHandler.jar` file into a directory on your machine (i.e. c:\mqprog).
4. Copy the `trans_handler.properties` and `jms_trans_handler.properties` files into the same directory as the `TransHandler.jar` file. If running the MQ-specific version of Transaction Handler (TransHandler), optionally edit the `trans_handler.properties` file with application-specific configuration information. If running the JMS-specific version of Transaction Handler (JMSTransHandler), then make sure the values for both the initial context factory property and the provider URL property within the `jms_trans_handler.properties` file have been specified. See the "Static Configuration" section for more details on the configuration files.
5. Create a directory structure on the local machine which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

    Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

6. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

    Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

7. For the JMS version of Transaction Handler, define the JMS queues and queue connection factories given the installed LDAP or file-system based directory service. For information on defining JMS objects within a JNDI namespace, please see sections within the "WebSphere MQ Using Java" guide on using the JMSAdmin tool.
8. To run the application, enter the following commands into a batch file, shell script, or from the operating system command prompt.
   a. For WebSphere MQ base Java version

```
cd <TransHan_Directory>

<JRE_Directory>\java –Djava.library.path="%MQ_JAVA_LIB_PATH%"
-cp "%CLASSPATH%";<TransHan_Directory>\TransHandler.jar
mqjava.programs.TransHandler [-iterations <n>]
```

   b. For WebSphere MQ JMS version

```
cd <TransHan_Directory>

<JRE_Directory>\java –Djava.library.path="%MQ_JAVA_LIB_PATH%"
-cp "%CLASSPATH%";<TransHan_Directory>\TransHandler.jar
mqjava.programs.JMSTransHandler [-iterations <n>]
```

   where

   <JRE_Directory> = the directory where the JRE (Java Runtime Environment) executable is located.

   MQ_JAVA_LIB_PATH = an environment variable that specifies the WebSphere MQ Java Libraries (as specified in the WebSphere MQ Java installation instructions)

   CLASSPATH = the class path environment variable which contains the settings for running WebSphere MQ Java applications (as specified in the WebSphere MQ Java installation instructions)

   Note: The CLASSPATH environment variable must include the list of JAR files distributed with WebSphere MQ Java: `providerutil.jar, com.ibm.mqjms.jar, ldap.jar, jta.jar, jndi.jar, jms.jar, connector.jar, fscontext.jar, com.ibm.mq.jar`

   <TransHan_Directory> = the directory where TransHandler.jar is located (i.e. c:\mqprog)

   -iterations <n> = the application argument which specifies the number of iterations executed by the TransHandler/JMSTransHandler application when it connects to the messaging engine, where "n" is a positive integer value. This argument is optional. If no value or a negative value is specified then the application client will continually poll for and retrieve messages until manually interrupted by the user.

NOTE:

Attempting to handle large messages using this application may result in a ***java.lang.OutOfMemoryError***. In order to avoid this problem, the user will need to

increase the virtual memory of the Java runtime environment (the default value is 16M). To increase memory, the user should include the **–mx** option of the JRE executable, which specifies the maximum size of the Java virtual memory.

For example,

```
java -mx32m –cp <…> mqjava.programs.TransHandler –iterations 1
```

The above command doubles the default memory from 16 megabytes to 32 megabytes. The JRE tool reference documentation can provide more information on this option.

9. See sections entitled "Using the MQ-specific Interface (Standalone version only) and "Using the JMS-specific Interface (J2EE Application Client/Standalone versions)" for a detailed description of the user interface

Installation of Standalone Application for a Development Environment

This section provides details on running the Transaction Handler standalone application within WebSphere Studio or Rational Application Developer.

1. Import either EAR file, `trans_handler_app.ear` or `trans_handler_appclient.ear`, into the Application Developer workspace. Within the Import wizard, under the option for "Select the utility JARs from the list to be imported as utility projects", make sure the `TransHandler.jar` file is checked. Also, if available, select the option to make a binary project for the utility JAR. If importing both EAR files into the workspace, then when importing the second EAR file, uncheck the `TransHandler.jar` file from the above mentioned list.
2. Copy the `trans_handler.properties` and `jms_trans_handler.properties` files into the TransHandler project. If running the MQ-specific version of Transaction Handler (TransHandler), optionally edit the `trans_handler.properties` file with application-specific configuration information. If running the JMS-specific version of Transaction Handler test tool (JMSTransHandler), then make sure the values for both the initial context factory property and the provider URL property within the `jms_trans_handler.properties` file have been specified. These two values are required for the standalone version of the JMS-specific Transaction Handler application. See the "Static Configuration" section for more details on the configuration files.
3. Create a directory structure on the local machine which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

   Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

4. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

   Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

5. To run the standalone application, go to the main toolbar of the Application Developer IDE and select Run→Run… From the Configuration pane, select a new Java Application configuration.
6. Under the Main tab, select TransHandler for the project. For the Main class, type or search for either **`mqjava.programs.TransHandler`** (for the WebSphere MQ base Java version of the application) or **`mqjava.programs.JMSTransHandler`** (for the WebSphere MQ JMS version of the application). If typing in the class name, make sure that there are no extra spaces before or after the class name.
7. Under the Arguments tab, optionally add the program argument **-iterations <n>**, which specifies the number of iterations executed by the TransHandler/JMSTransHandler application client when it connects to the messaging engine. The parameter "n" should be a positive integer value. If no value or a negative value is specified then the application client will continually poll for and retrieve messages until manually interrupted by the user.
8. Under the Classpath tab, add the following external JAR files which are located in the WebSphere MQ Java lib directory: `providerutil.jar, com.ibm.mqjms.jar, ldap.jar, jta.jar, jndi.jar, jms.jar, connector.jar, fscontext.jar, com.ibm.mq.jar`. If applicable, you may need to uncheck the option "Use default class path". In addition, add the TransHandler folder, where the configuration property files are located. This entry can be added by pressing the Advanced button.
9. Save the above configuration, and then press the Run button to launch the application.
10. See sections entitled "Using the MQ-specific Interface (Standalone version only) and "Using the JMS-specific Interface (J2EE Application Client/Standalone versions)" for a detailed description of the user interface.

Installing the J2EE Application Client

This section describes how to install and start up the Transaction Handler J2EE application client against a WebSphere Application Server or from a development environment using WebSphere Studio or Rational Application Developer. For installation within an IDE, please see "Installation of J2EE Application Client for a Development Environment" for details.

Installation of J2EE Application Client for WebSphere Application Server

1. Install the required software components mentioned in the "Prerequisites" section. The WebSphere Application Client software must be installed on the machine running this version of the application and should be configured to access an existing WebSphere Application Server. Note that if the application client software has not yet been installed, then the installation software can normally be found with the installation media used for installing the application server.
2. Within the administrative console of the WebSphere Application Server, under the Resources heading, define the messaging provider (and corresponding JMS objects) using either the WebSphere MQ JMS provider or the default messaging provider (WebSphere JMS provider).
3. Copy the `trans_handler_appclient.ear` file into a directory on your machine (i.e. c:\mqprog)
4. Copy the `jms_trans_handler.properties` file into the same directory as the `trans_handler_appclient.ear` file. Optionally, edit this file with application-specific configuration information. See the "Static Configuration" section for more details about the configuration files.

Please note that the values for both the initial context factory property and the provider URL property are not required for this version of the Transaction Handler application. Both of these values are provided to the test tool by the application server.

5. Create a directory structure on the local which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

   Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

6. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

   Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

7. Use the `launchClient` tool, located in the bin directory of the application client, to run the Transaction Handler J2EE application client (JMSTransHandler). The following command can be used to launch the application client:

```
<AppClient_Install_Dir>\bin\launchClient
<TransHan_Directory>\trans_handler_appclient.ear –CCcclasspath=
"%CLASSPATH%";"%MQ_JAVA_LIB_PATH%";<TransHan_Directory> -CCproviderURL=
<ProviderURL> -CCverbose=true [-iterations <n>]
```

   where

   <AppClient_Install_Dir> = the root directory where the WebSphere Application Server Application Client software has been installed.

   <TransHan_Directory> = the directory where `trans_handler_appclient.ear` is located (i.e. c:\mqprog)

   CLASSPATH = the class path environment variable which contains the settings for running WebSphere MQ Java applications (as specified in the WebSphere MQ Java installation instructions)

   Note: The CLASSPATH environment variable must include the list of JAR files distributed with WebSphere MQ Java: `providerutil.jar, com.ibm.mqjms.jar, ldap.jar, jta.jar, jndi.jar, jms.jar, connector.jar, fscontext.jar, com.ibm.mq.jar`

   MQ_JAVA_LIB_PATH = an environment variable that specifies the WebSphere MQ Java Libraries (as specified in the WebSphere MQ Java installation instructions)

   ProviderURL = the bootstrap server information used by the initial context factory to obtain an initial context. Please note that the “-CCproviderURL” parameter is specific to the `launchClient` tool (ex. iiop://localhost:2810)

-iterations <n> = the application argument which specifies the number of iterations executed by the Transaction Handler application when it connects to the messaging server, where "n" is a positive integer value. This argument is optional. If no value or a negative value is specified then the application client will continually poll for and retrieve messages until manually interrupted by the user.

NOTE: For more details on the `launchClient` tool, please refer to the specific version of the WebSphere Application Center Information Center.

8. See the section entitled "Using the JMS-specific Interface (J2EE Application Client/Standalone versions)" for a detailed description of the user interface.

Installation of J2EE Application Client for a Development Environment

This section provides details on running the Transaction Handler J2EE application client within WebSphere Studio or Rational Application Developer.

1. Ensure that the required software components have been installed as mentioned in the "Prerequisites" section. Also ensure that an appropriate version of the WebSphere Test Environment has been installed within the development environment.
2. Configure the test server to use the embedded messaging provider or the WebSphere MQ JMS provider. The MQ Simulator for Java Developers cannot be used with any JMS application client. A processing error will occur if the Transaction Handler application client attempts to access a JMS object which has been defined using the MQ Simulator for Java Developers.
3. Import the EAR file, `trans_handler_appclient.ear`, into the Application Developer workspace. Within the Import wizard, under the option for "Select the utility JARs from the list to be imported as utility projects", make sure the `TransHandler.jar` file is checked. Also, if available, select the option to make a binary project for the utility JAR. If the EAR file for the Web-based application (`trans_handler_app.ear`) has already been imported into the workspace, then uncheck the `TransHandler.jar` file from the above mentioned list.
4. Copy the `jms_trans_handler.properties` file into the TransHandler project. Optionally, edit this file with application-specific configuration information. See the "Static Configuration" section for more details about the configuration file.

   Please note that the values for both the initial context factory property and the provider URL property are not required for this version of the Transaction Handler application. Both of these values are provided to the test tool by the application server test environment.

5. Create a directory structure on the local machine which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

   Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

6. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

   Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

7. Start up the test server. The test server must be running before starting up the Transaction Handler application client.
8. To run the application client, go to the main toolbar of the Application Developer IDE and select Run→Run… From the Configuration pane, select a new Application Client configuration for the specific server type.
9. Under the Application tab, select `trans_handler_appclient` for the Enterprise Application and `trans_handler_client` for the Application Client.
10. Under the Classpath tab, add the following external JAR files under user classes. These files should be located in the test server `lib` directory: `bootstrap.jar, j2ee.jar, lmproxy.jar`. Also add the external `properties` folder from the test server `lib` directory.
11. Under the Classpath tab, add the following external JAR files which are located in the WebSphere MQ Java lib directory: `providerutil.jar, com.ibm.mqjms.jar, ldap.jar, jta.jar, jndi.jar, jms.jar, connector.jar, fscontext.jar, com.ibm.mq.jar.` In addition, add the TransHandler project folder, where the `jms_trans_handler` configuration property file is located. This entry can be added by pressing the Advanced button.
12. Under the Arguments tab, optionally add the program argument **-iterations <n>**, which specifies the number of iterations executed by the Transaction Handler application client when it connects to the messaging server. The parameter “n” should be a positive integer value. If no value or a negative value is specified then the application client will continually poll for and retrieve messages until manually interrupted by the user.
13. Save the above configuration, and then press the Run button to launch the application client.
14. See the section entitled “Using the JMS-specific Interface (J2EE Application Client/Standalone versions)” for a detailed description of the user interface.

Installing the Web-based Application

This section describes how to install and start up the Web-based version of the Transaction Handler application within WebSphere Application Server or within the WebSphere Test Environment for WebSphere Studio or Rational Application Developer. For installation within an IDE, please see “Installation of Web Application for a Development Environment” for details.

Installation of Web Application for WebSphere Application Server

1. Install the required software components mentioned in the Prerequisites section.
2. Copy the `trans_handler_app.ear` file into a directory on your machine (i.e. c:\mqprog).
3. Within the administrative console of the WebSphere Application Server, go to Applications→Install New Application. Follow the installation wizard and accept the default values. As instructed by the wizard, map the TransHandlerWeb module to the given server(s) and map the virtual host to the TransHandlerWeb module. When complete, save changes to the master configuration file.

4. Within the administrative console, go to Applications→Enterprise Applications and start the JMSTransHandlerEAR application.
5. Within the administrative console of the WebSphere Application Server, under the Resources heading, define the messaging provider (and corresponding JMS objects) under Resources for either the WebSphere MQ JMS provider or the default messaging provider (WebSphere JMS provider).
6. Create a directory structure on the local machine which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

   Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

7. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

   Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

8. If the application server is up and running, enter the following URL into a Web browser to go to the Transaction Handler main Web page, http://<servername>:<serverport>/TransHandlerWeb/JMSTransHandler/Main
9. See section entitled “Using the Web-based Interface” for a detailed description of the user interface.

Installation of Web Application for a Development Environment

This section provides details on running the Web-based version of the Transaction Handler application within WebSphere Studio or Rational Application Developer.

1. Ensure that the required software components have been installed as mentioned in the Prerequisites section. Also ensure that an appropriate version of the WebSphere Test Environment has been installed within the development environment.
2. Make sure the test server has been configured with the appropriate JMS queue and queue connection factories used by the user code. Unlike the application client, the MQ Simulator for Java Developers can be used as well as either the embedded messaging provider or the WebSphere MQ JMS provider.
3. Import the EAR file, `trans_handler_app.ear`, into the Application Developer workspace. Keep default value for project name. Within the Import wizard, under the option for “Select the utility JARs from the list to be imported as utility projects”, make sure the `TransHandler.jar` file is checked. Also, if available, select the option to make a binary project for the utility JAR. If the EAR file for the application client (`trans_handler_appclient.ear`) has already been imported into the workspace, then uncheck the `TransHandler.jar` file from the above mentioned list.
4. Create a directory structure on the local machine which will contain the content for the response messages. At least one file must exist within the directory. Make sure the directory and file(s) listed within the directory are read-accessible by the Transaction Handler application.

Please note that response messages are sent by the Transaction Handler application to the user code. As a result, when placing files into this directory, make sure that the user code can understand the content of these files, whether plain text, XML, or some other format.

5. Optionally, create a directory structure on the local machine which will contain the content of the request messages. The directory must be write-accessible by the Transaction Handler application.

   Please note that request messages are sent by the user code to the Transaction Handler application. As a result, creating and using this directory can help developers in debugging and troubleshooting their own code.

6. Start up and add the `trans_handler_app` project to the test server.
7. Once the application has started on the test server, enter the following URL into a Web browser to go to the JMSTransHandler main page, http://<servername>:<serverport>/TransHandlerWeb/JMSTransHandler/Main
8. See the section entitled "Using the Web-based Interface" for a detailed description of the user interface.

Static Configuration

This section describes the property files used by the Transaction Handler Test Tool using the MQ-specific interface (TransHandler) and using the JMS-specific interface (JMSTransHandler). The use of these files is optional and is only provided as a convenience for storing static configuration information. When either the Transaction Handler Test Tool starts up, the information stored in these files will appear as default values within its graphical user interface. When the Transaction Handler Test Tool is initiated with the MQ-specific interface, then the test tool will only retrieve values from the file named `trans_handler.properties`. Likewise, when the JMS-version of the Transaction Handler Test Tool is initiated, it will only retrieve values from the file named `jms_trans_handler.properties`. (Please note that this does not apply to the Web-based version of the test tool). The configuration information in each of these files is in the form of <`Property Key`>=<`Configuration Value`>. A pound sign (#) at the beginning of a line indicates a comment. The following subsections describe each of the property keys.

Configuring the MQ-specific Interface

The following table provides a short description of the property keys found in the file `trans_handler.properties`. All properties in this file map to a field in the user interface. For a more detailed description, please see the section entitled, "Using the MQ-specific Interface (Standalone version only)".

Table 1 - TransHandler Configuration File Property Description

GUI Field	Property Key	Description
Queue Manager	trans.handler.queue.manager.name	Specifies the name of the Queue Manager to which to connect.
Request Queue Name	trans.handler.request.queue.name	Specifies the name of the queue within the specified Queue Manger to

GUI Field	Property Key	Description
		open for input.
Response Queue Name	trans.handler.response.queue.name	Specifies the name of the queue within the specified Queue Manger to open for output.
Channel	trans.handler.channel	Specifies the name of the channel to connect to on the target Queue Manager.
Hostname	trans.handler.hostname	Specifies the TCP/IP hostname of the machine on which the WebSphere MQ server resides.
Port	trans.handler.port	Specifies the port on which the WebSphere MQ server is listening for incoming connection requests.
Response File Path	trans.handler.response.file.path	Specifies the location on the local machine where the application can retrieve the content of the response message body.
Response Delay Time	trans.handler.response.delay.time	Specifies the number of milliseconds the application should wait between retrieving a message from the request queue and placing a message on the response queue.
Request File Path	trans.handler.request.file.path	Specifies the location on the local machine where the application can create files containing the message body of request messages.

Configuring the JMS-specific Interface

The following table provides a short description of the property keys found in the file `jms_trans_handler.properties`. Except for two properties, all properties in this file map to a field in the user interface. For a more detailed description, please see section “Using the JMS-specific Interface (J2EE Application Client/Standalone versions)”. The other two properties for the initial context factory and the provider URL are required only by the JMS standalone client and should contain values that are used by the JNDI service provider.

Table 2 - JMSTransHandler Configuration File Property Description

GUI Field	Property Key	Description
Request Queue	trans.handler.request.queue.name	Specifies the JNDI name of the JMS queue where the application will

GUI Field	Property Key	Description
Name		retrieve request messages.
Response Queue Name	trans.handler.response.queue.name	Specifies the JNDI name of the JMS queue where the application will retrieve response messages.
Request Connection Factory	trans.handler.request.connection.factory	Specifies the JNDI name of the JMS connection factory for the corresponding request queue.
Response Connection Factory	trans.handler.response.connection.factory	Specifies the JNDI name of the JMS connection factory for the corresponding response queue.
Response File Path	trans.handler.response.file.path	Specifies the location on the local machine where the application can retrieve the message text for the body of the response messages.
Request File Path	trans.handler.request.file.path	Specifies the location on the local machine where the application can create files containing the message body of request messages.
Response Delay Time	trans.handler.response.delay.time	Specifies the number of milliseconds the application should wait between retrieving a message from the request queue and placing a message on the response queue.
N/A	trans.handler.initial.context.factory	Specifies the fully-qualified class name of the factory responsible for creating an InitialContext object for a JNDI service. This property field is required for the JMS standalone client only. The value for this property should be obtained from the JNDI service provider.
N/A	trans.handler.provider.url	Specifies URL used by JNDI service provider for designating configuration information. This property field is required for the JMS standalone client only. The value for this property should be obtained from the JNDI service provider.

Chapter 4. User Interface Description

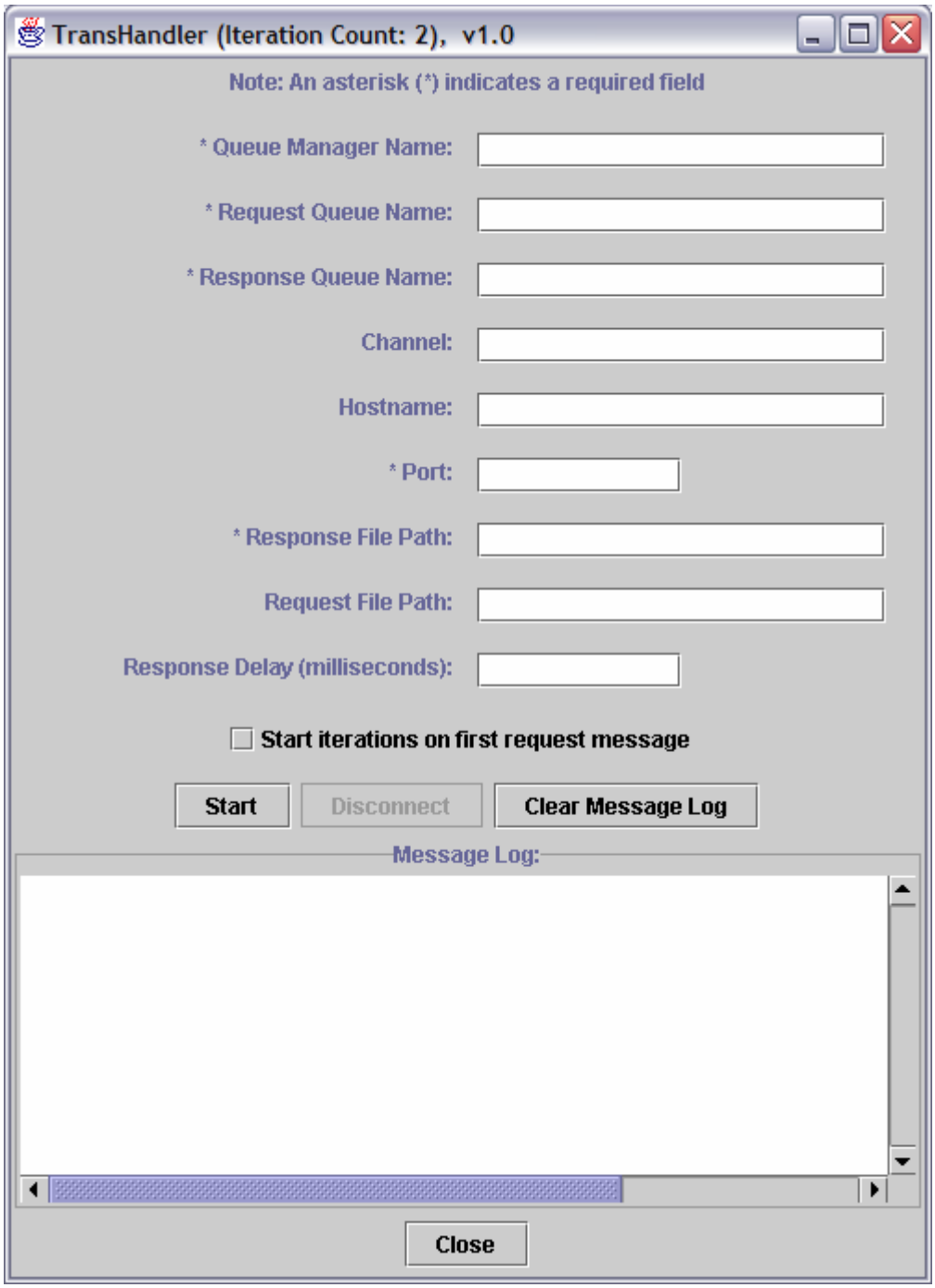
The WebSphere MQ – Transaction Handler Test Tool consists of a graphical user interface (GUI) which allows the user to connect to a WebSphere MQ server, put messages to and get messages from WebSphere MQ queues. This chapter provides detailed descriptions on the graphical user interface provided for the MQ-specific, JMS-specific, and Web-based versions of the Transaction Handler Test Tool. Additionally, each section describes how the test tool interacts with the user input.

Please note that the screen images may vary slightly depending upon the platform where the application is running.

Using the MQ-specific Interface (Standalone version only)

The MQ-specific version of the Transaction Handler Test Tool (called “TransHandler”) consists of a graphical user interface (GUI) which allows the user to connect to a WebSphere MQ server and process messages between MQ queues. When the user initiates the TransHandler application, the following screen is displayed:

Figure 2 - TransHandler Screen



The table below describes the GUI components of the TransHandler application:

Table 3 - TransHandler GUI Description

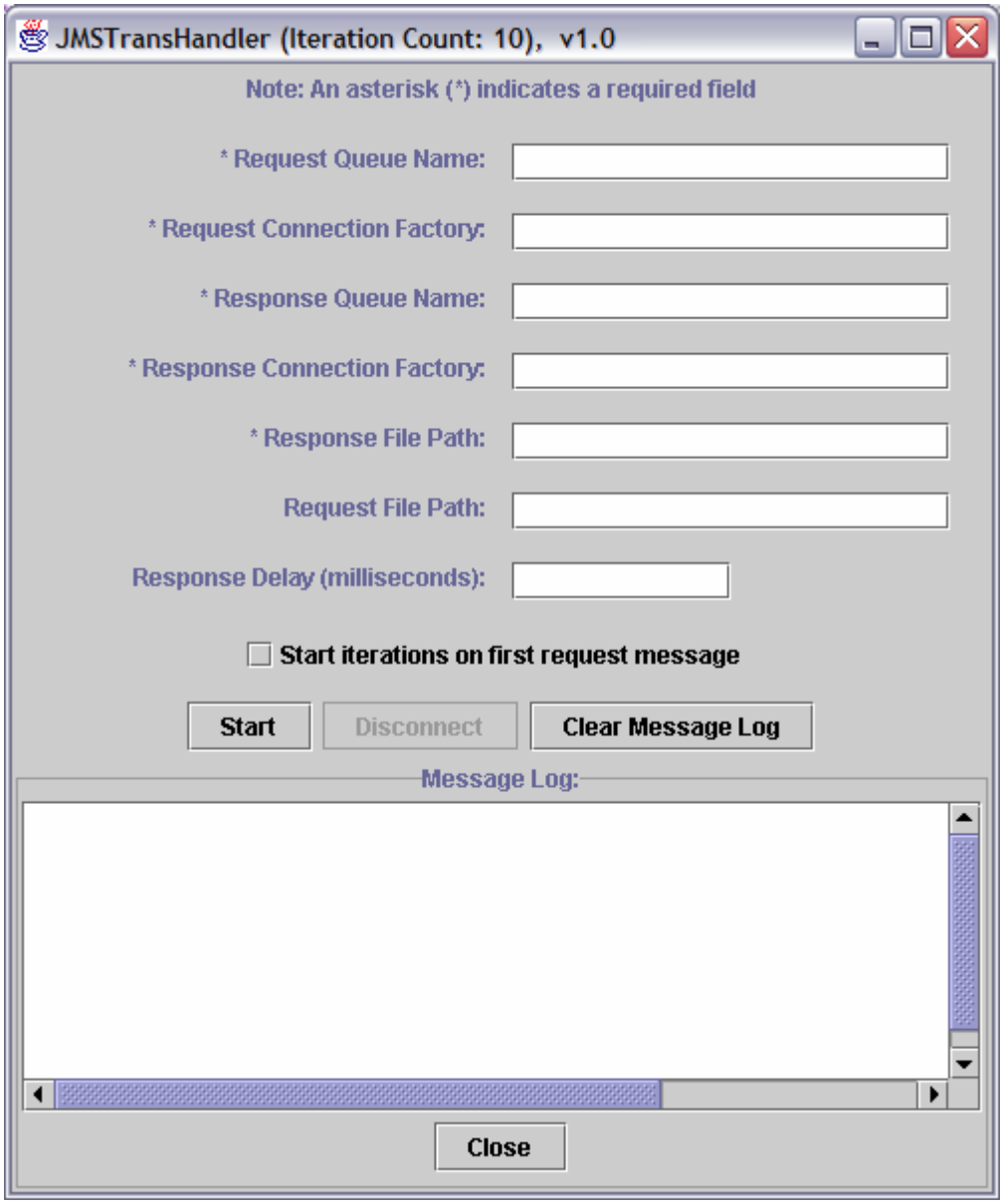
Name	Type	Description
TransHandler (Iteration Count: NN) , v1.0	Title	Displays the name of the application followed by the iteration count specified by the user at start-up and the version number. If the iterations argument was not specified by the user at start-up, then the text "(Iteration Count: NN)" is not displayed.
Queue Manager	Text Field	Enter the name of the Queue Manager to which to connect. This field is **required** by the application in order to make a connection to the WebSphere MQ server.
Request Queue	Text Field	Enter the name of the queue within the specified Queue Manger to open for input. This field is **required** by the application in order to make a connection to the WebSphere MQ server.
Response Queue	Text Field	Enter the name of the queue within the specified Queue Manger to open for output. This field is **required** by the application in order to make a connection to the WebSphere MQ server.
Channel	Text Field	Enter the name of the channel to connect to on the target Queue Manager. This field is **optional**. If the Channel is not set, then the Hostname must not be set as well.
Hostname	Text Field	Enter the TCP/IP hostname of the machine on which the WebSphere MQ server resides. This field is **optional**. If the Hostname is not specified, then a direct connection will be made to the WebSphere MQ server via Bindings Mode. Otherwise a Client connection is made via the network.
Port	Text Field	Enter the port to connect to. This should be the port on which the WebSphere MQ server is listening for incoming connection requests. This field is **required** by the application in order to make a connection to the WebSphere MQ server.
Response File Path	Text Field	Enter the location on the local machine where the application can retrieve the content of the response message body. This field is **required** by the application and at least one file must be present in the directory. If more then one file is present, then the application will use the round robin approach to selecting a file for creating the response message body.
Request File Path	Text Field	Enter the location on the local machine where the application can create files containing the body of the request messages. This field is **optional**, however if a value is set, it cannot be the same value as the Response File Path field. The files that are created in this directory will having the following naming convention: `WMQJ.<Current Date>.<Current Time>.txt`

Name	Type	Description
Response Delay (milliseconds)	Text Field	Enter the number of milliseconds the application should wait between retrieving a message from the request queue and placing a message on the response queue. This field is **optional**. If no value is specified then no delay will occur and a message will immediately be placed onto the response queue during processing.
Start iterations on first request message	Check Box	Select whether to start counting down the number of iterations immediately (default) or wait until the first request message is retrieved by the application. If the default is used, then the application will automatically disconnect after the specified number of iterations elapses despite whether or not a request message was retrieved from the queue. Otherwise, if the checkbox is checked, then the application will wait until at least one message is in the request queue before counting down the number of iterations and eventually disconnecting. This checkbox only displays if the iterations argument was specified by the user at start-up.
Start	Button	When pressed, the application will attempt to connect to the specified Queue Manager and open the specified Queues. The application will then start the process of getting messages from the request queue and putting messages onto the response queue. If the user specified a value for the number of get/put iterations per connection, then the application will automatically disconnect from the Queue Manager once the number of iterations has been completed. If a problem occurs, then a corresponding error message will appear within the message log. If the connection is successful then the "Start" button will become disabled and the “Disconnect” button will become enabled. In addition, the Queue Manager, Request Queue, Response Queue, Hostname, Channel, Port, Response File Path, Request File Path, Response Delay fields and Clear Message Log button will be disabled.
Disconnect	Button	When pressed, the application will close the specified Queues and disconnect from the specified Queue Manager. The "Disconnect" button will become disabled and the “Start” button will become enabled again. In addition, the Queue Manager, Request Queue, Response Queue, Hostname, Channel, Port, Response File Path, Request File Path, Response Delay fields and Clear Message Log button will be enabled as well.
Clear Message Log	Button	When pressed, the application will remove all existing text from the Message Log area.
Message Log	Text Area	This area logs the status of the TransHandler application as it executes the actions specified by the user.
Close	Button	When pressed, the application will disconnect from the specified Queue Manager (if connected) and then exit the application.

Using the JMS-specific Interface (J2EE Application Client/Standalone versions)

The JMS version of the Transaction Handler Test Tool (called “JMSTransHandler”) consists of a graphical user interface (GUI) which allows the user to specify the JNDI names of the JMS objects, connect to a MQ server, and process messages between the designated queues. When the user initiates the JMSTransHandler application, the following screen is displayed:

Figure 3 – JMSTransHandler Screen



The table below describes the GUI components of the JMSTransHandler application:

Table 4 - JMSTransHandler GUI Description

Name	Type	Description
JMSTransHandler (Iteration Count: XX), v1.0	Title	Displays the name of the application followed by the iteration count specified by the user at start-up and the version number. If the iterations argument was not specified by the user at start-up, then the text "(Iteration Count: NN)" is not displayed.
Request Queue	Text Field	Enter the JNDI name of the JMS queue where the application will retrieve request messages. This field is **required** by the application.
Request Connection Factory	Text Field	Enter the JNDI name of the JMS queue connection factory for the corresponding request queue. This field is **required** by the application and is used for creating a connection to the MQ server.
Response Queue	Text Field	Enter the JNDI name of the JMS queue where the application will send response messages. This field is **required** by the application.
Response Connection Factory	Text Field	Enter the JNDI name of the JMS connection factory for the corresponding response queue. This field is **required** by the application and is used for creating a connection to the MQ server.
Response File Path	Text Field	Enter the location on the local machine where the application can retrieve the message text for the body of the response messages. This field is **required** by the application and at least one file must be present in the directory. If more then one file is present, then the application will use the round robin approach to selecting a file for creating the response message.
Request File Path	Text Field	Enter the location on the local machine where the application can create files containing the message body of request messages. This field is **optional**, however if a value is set, it cannot be the same value as the Response File Path field. The files that are created in this directory will having the following naming convention: `WMQJ.<Current Date>.<Current Time>.txt`
Response Delay (milliseconds)	Text Field	Enter the number of milliseconds the application should wait between retrieving a message from the request queue and placing a message on the response queue. This field is **optional**. If no value is specified then no delay will occur and a message will immediately be placed onto the response queue during processing.
Start iterations on first request message	Check Box	Select whether to start counting down the number of iterations immediately (default) or wait until the first request message is retrieved by the application. If the default is used, then the application will automatically disconnect after the specified number of iterations elapses despite whether or not a request message was retrieved from the queue. Otherwise, if the checkbox is checked, then the application will wait until at least one message is in the request queue

Name	Type	Description
		before counting down the number of iterations and eventually disconnecting. This checkbox only displays if the iterations argument was specified by the user at start-up.
Start	Button	When pressed, the application will attempt to connect to the MQ server and open the specified Queues. The application will then start the process of getting messages from the request queue and putting messages onto the response queue. If the user specified a value for the number of get/put iterations per connection, then the application will automatically disconnect from the MQ server once the number of iterations has been completed. If a problem occurs, then a corresponding error message will appear within the message log. If the connection is successful then the "Start" button will become disabled and the “Disconnect” button will become enabled. In addition, the Request Queue Name, Request Connection Factory, Response Queue Name, Response Connection Factory, Response File Path, Request File Path, Response Delay fields and Clear Message Log button will be disabled.
Disconnect	Button	When pressed, the application will close the specified Queues and disconnect from the specified Queue Manager. The "Disconnect" button will become disabled and the “Start” button will become enabled. In addition, the Request Queue Name, Request Connection Factory, Response Queue Name, Response Connection Factory, Response File Path, Request File Path, Response Delay fields and Clear Message Log button will be enabled as well.
Clear Message Log	Button	When pressed, the application will remove all existing text from the Message Log area.
Message Log	Text Area	This area logs the status of the JMSTransHandler application as it executes the actions specified by the user.
Close	Button	When pressed, the application will disconnect from the MQ server (if connected) and then exit the application.

(**Note**: For information on defining JMS objects within a JNDI namespace, please see sections on using the JMSAdmin tool in the “WebSphere MQ Using Java” guide)

Using the Web-based Interface

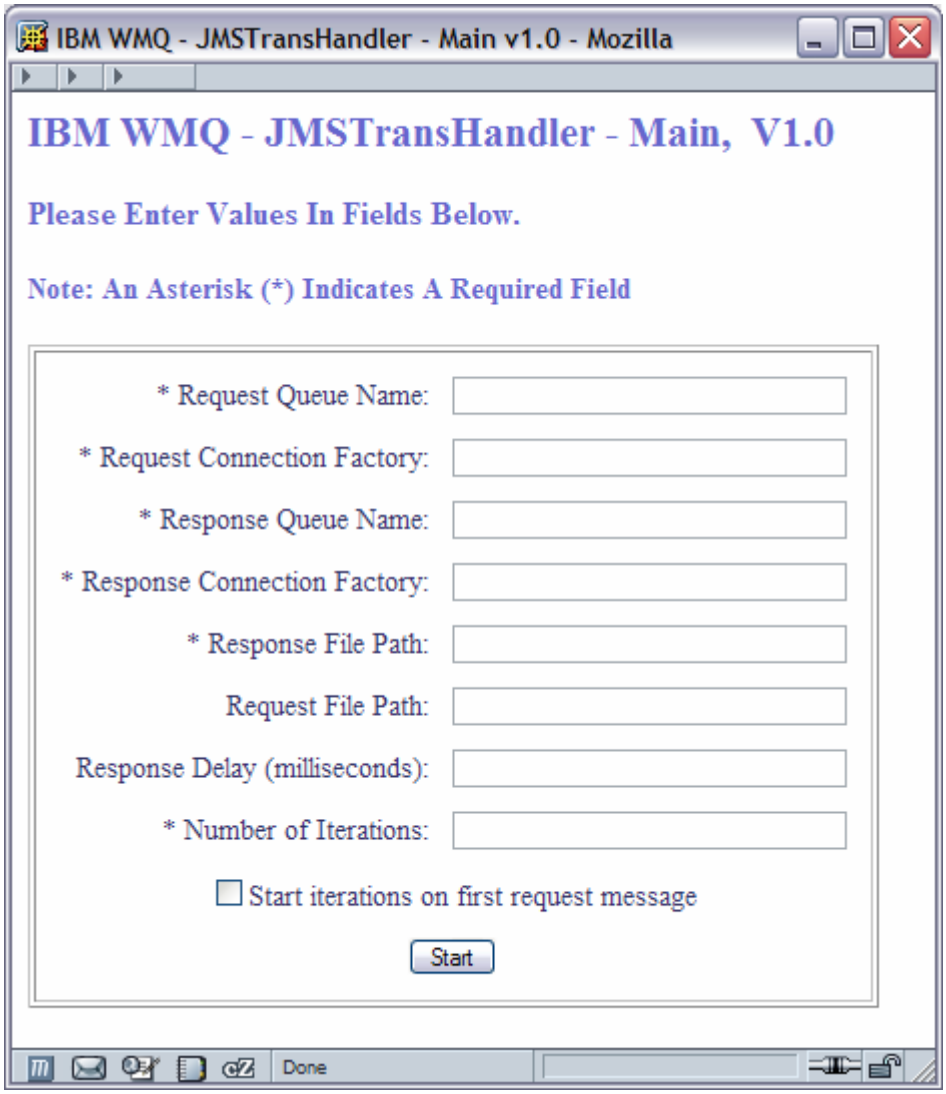
The Web-based version of the Transaction Handler Test Tool (called “JMSTransHandler”) consists of a browser interface which allows the user to specify the JNDI names of the JMS objects, connect to a MQ server, and process messages between the designated queues.

Main Page Description

When the user enters in the URL http://<servername>:<serverport>/TransHandlerWeb/JMSTransHandler/Main for the

JMSTransHandler Web application, the following page (Figure 4 - JMSTransHandler Web – Main) is displayed:

Figure 4 - JMSTransHandler Web – Main



The table below describes the initial page of the JMSTransHandler Web application:

Table 5 – JMSTransHandler Web-Based Interface Description

Name	Type	Description
Request Queue	Text Field	Enter the JNDI name of the JMS queue where the application will retrieve request messages. This field is **required** by the application.
Request Connection Factory	Text Field	Enter the JNDI name of the JMS connection factory for the corresponding request queue. This field is **required** by the application and is used for creating a connection to the MQ

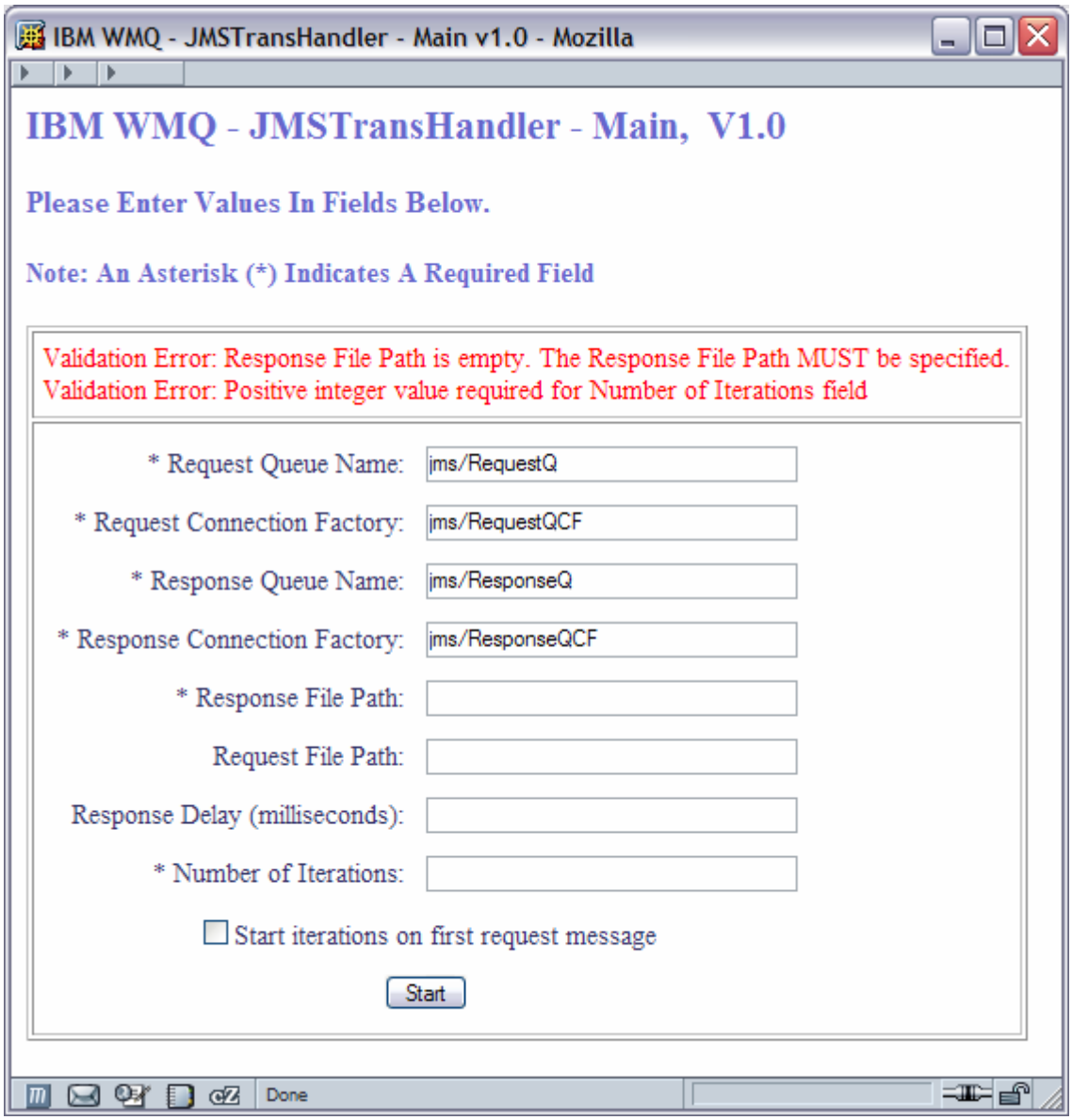
Name	Type	Description
		server.
Response Queue	Text Field	Enter the JNDI name of the JMS queue where the application will send response messages. This field is **required** by the application.
Response Connection Factory	Text Field	Enter the JNDI name of the JMS connection factory for the corresponding response queue. This field is **required** by the application and is used for creating a connection to the MQ server.
Response File Path	Text Field	Enter the location on the local machine where the application can retrieve the message text for the body of the response messages. This field is **required** by the application and at least one file must be present in the directory. If more then one file is present, then the application will use the round robin approach to selecting a file for creating the response message.
Request File Path	Text Field	Enter the location on the local machine where the application can create files containing the message body of request messages. This field is **optional**, however if a value is set, it cannot be the same value as the Response File Path field. The files that are created in this directory will having the following naming convention: `WMQJ.<Current Date>.<Current Time>.txt`
Response Delay (milliseconds)	Text Field	Enter the number of milliseconds the application should wait between retrieving a message from the request queue and placing a message on the response queue. This field is **optional**. If no value is specified then no delay will occur and a message will immediately be placed onto the response queue during processing.
Number of Iterations	Text Field	Enter the number of “get request”/“put response” iterations the application will make before automatically disconnecting from the MQ server. This field is **required**.
Start iterations on first request message	Check Box	Select whether to start counting down the number of iterations immediately (default) or wait until the first request message is retrieved by the application. If the default is used, then the application will automatically disconnect after the specified number of iterations elapses despite whether or not a request message was retrieved from the queue. Otherwise, if the checkbox is checked, then the application will wait until at least one message is in the request queue before counting down the number of iterations and eventually disconnecting. **NOTE:** Please take care when selecting this option. If checked, at least one request message must be received by the application before it is able to automatically disconnect from the messaging server. The Web interface does not

Name	Type	Description
		provide a way for the user to manually disconnect from the messaging server.
Start	Button	When pressed, the application will attempt to connect to the MQ server and open the specified Queues. The application will then start the process of getting messages from the request queue and putting messages onto the response queue. Once started, all fields will become disabled and the Web application will display a message log showing the current progress (see Figure 7). If a problem occurs, then a corresponding error message will also appear within the message log.

Main Page with Errors Description

When the user presses the Start button on the main page, the JMSTransHandler application will validate the values entered by the user. If any of these values fails the validation check, i.e. missing required field, then the Main page will be redisplayed along with a list of the errors (as shown in Figure 5 – JMSTransHandler Web - Main with Errors).

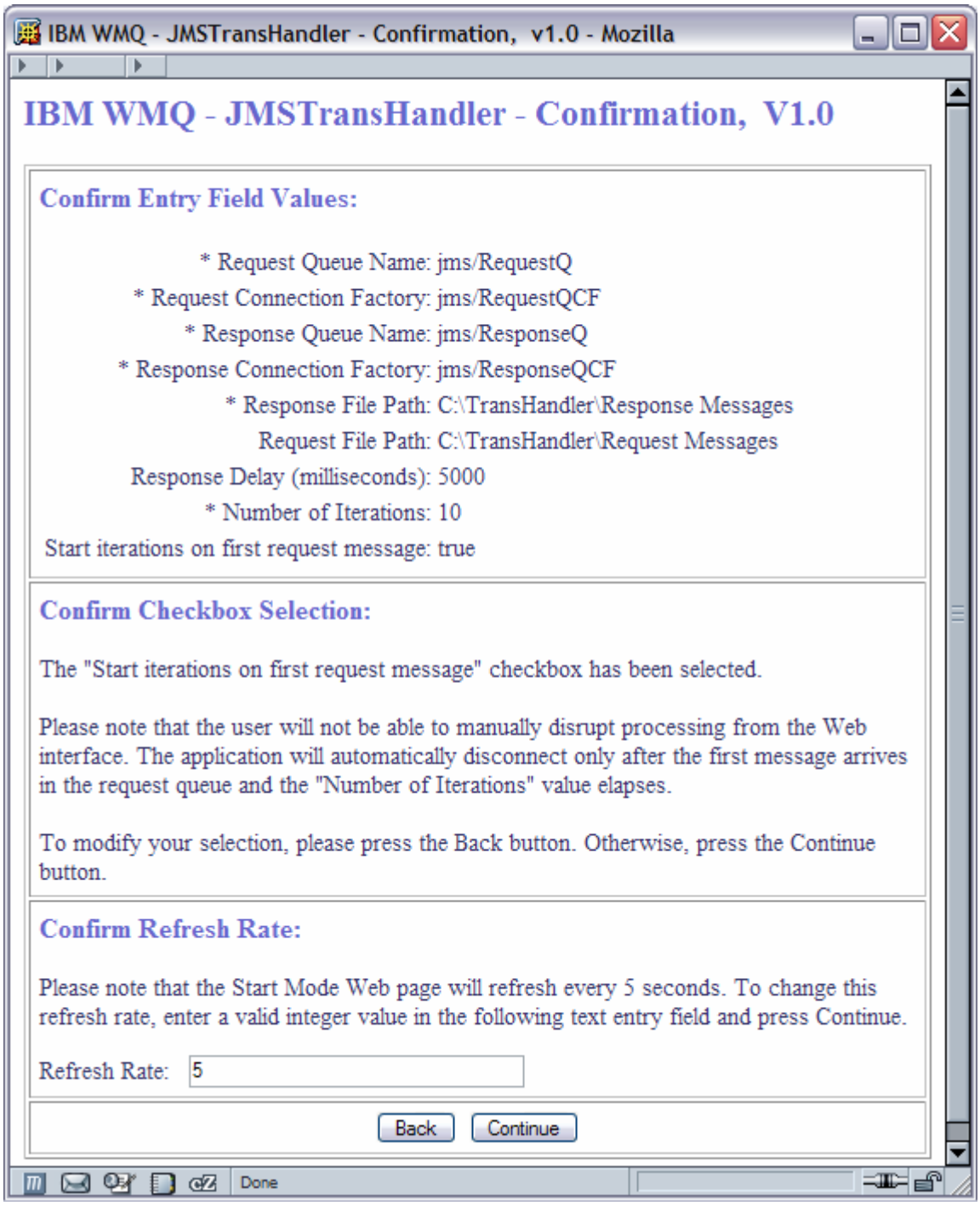
Figure 5 – JMSTransHandler Web - Main with Errors



Confirmation Page Description

Once the user presses Start button and all initial validation checks have passed successfully, the page (shown in Figure 6 - JMSTransHandler Web - Confirmation) is displayed to the user. The page allows the user to verify the values that were entered from the Main page (shown in Figure 4 - JMSTransHandler Web – Main) and also allows the user to specify a refresh interval for the subsequent Start Mode page (shown in Figure 7 - JMSTransHandler Web - Start Mode). This interval value allows the user to specify how often the text within the message log area will be updated and displayed by the application. The user can press the Back button to update the values entered on the Main page or can press the Continue button to go to the Start Mode page.

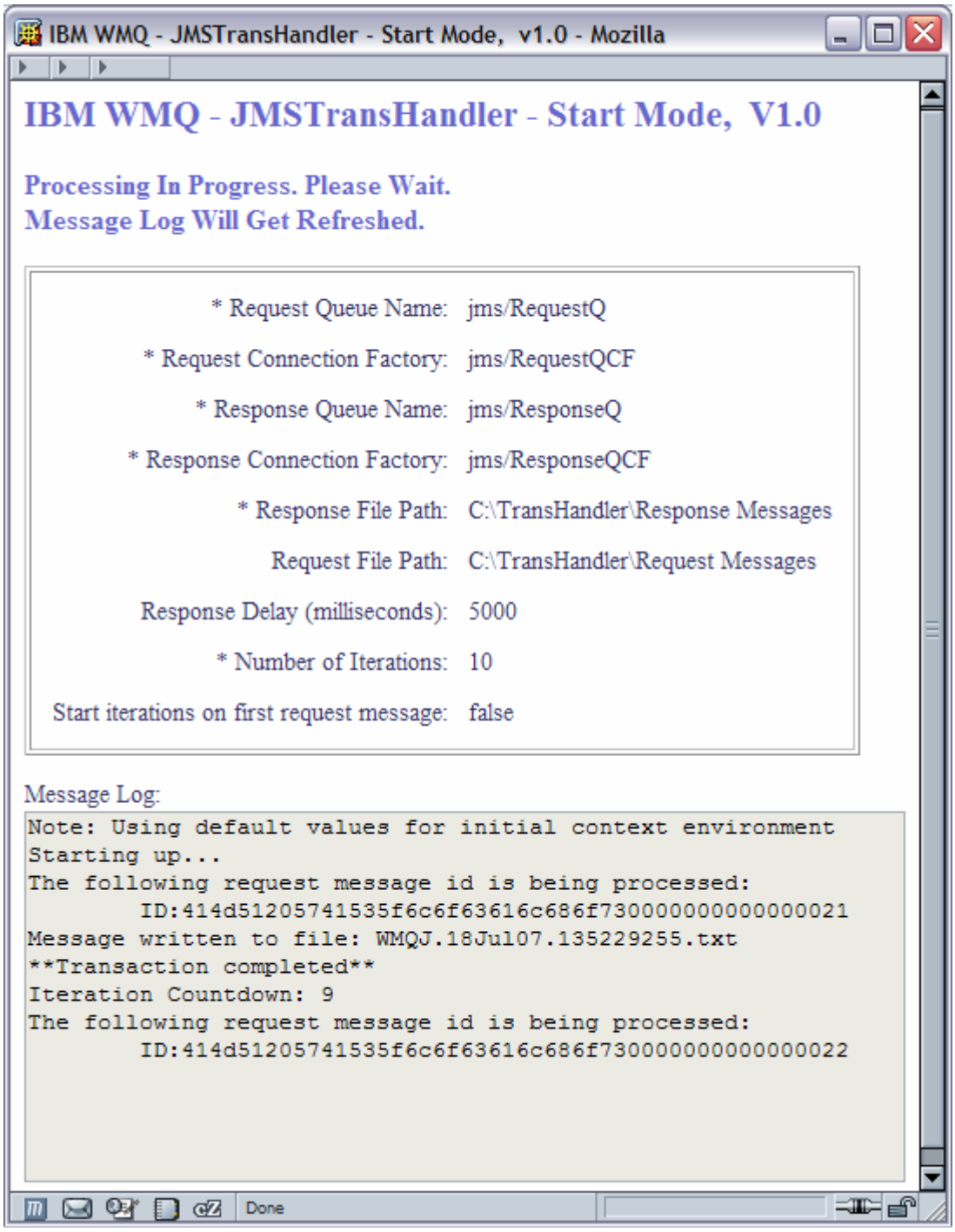
Figure 6 - JMSTransHandler Web - Confirmation



Start Mode Page Description

Once the user presses the Continue button from the Confirmation page, the following page (Figure 7 - JMSTransHandler Web - Start Mode) is displayed to the user. Note that all the user-entered fields are now read-only and a message log (at the bottom of the page) shows the status of the application. The text within the message log will get updated based on the refresh rate specified by the user on the Confirmation page.

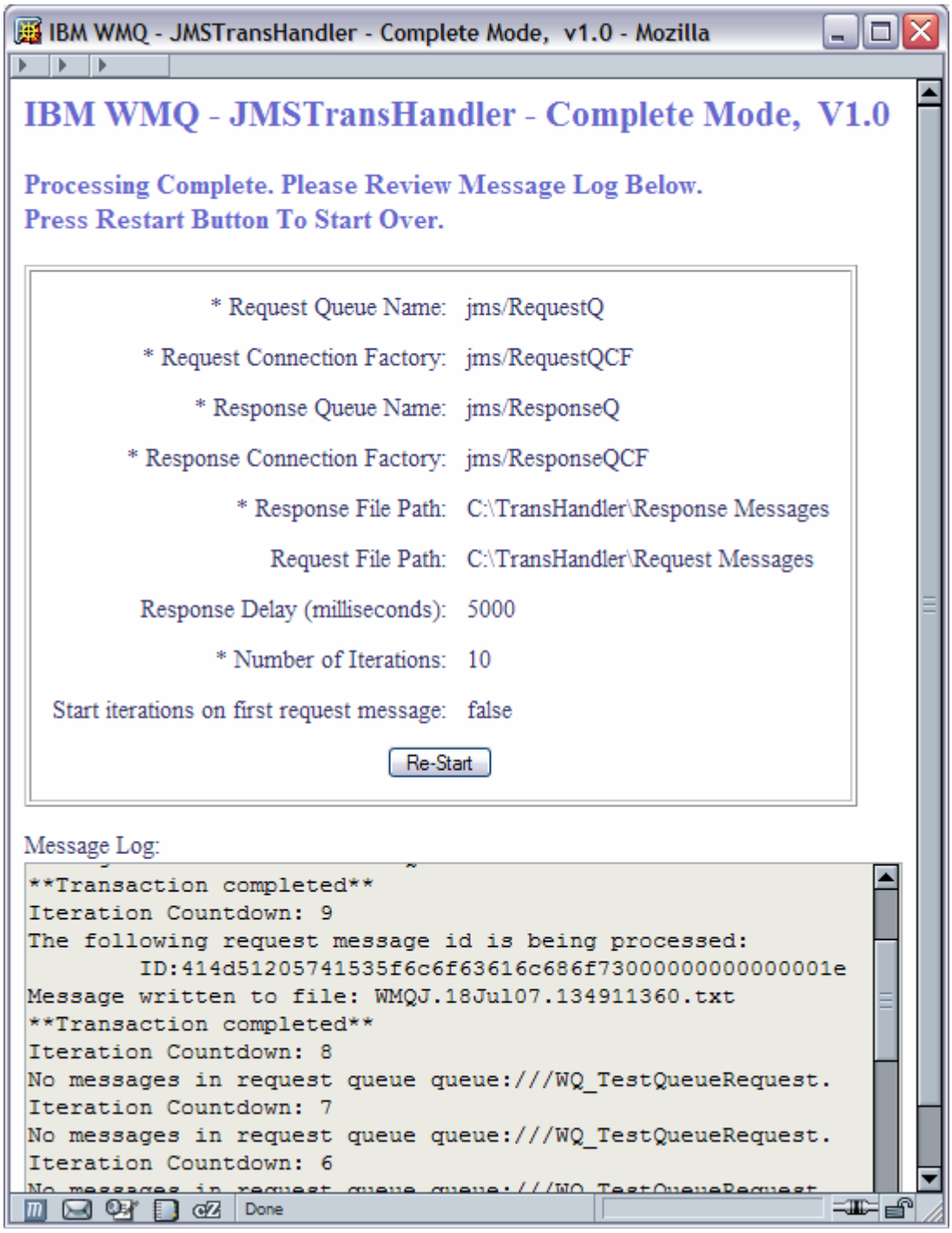
Figure 7 - JMSTransHandler Web - Start Mode



Complete Mode Page Description

Once the application has completed its processing, the page (shown in Figure 8 - JMSTransHandler Web - Complete Mode) is displayed to the user. The user can then press the “Re-Start” button to begin processing again. If the Re-Start button is pressed, then the main entry page is redisplayed and the previously-entered values are automatically entered in each of the entry fields as defaults.

Figure 8 - JMSTransHandler Web - Complete Mode



End of Document