

Rational IntegrationTester



Reference Guide for SonicMQ

Version 8.0.0



Note

Before using this information and the product it supports, read the information in “Notices” on page 30.

This edition applies to version 8.0.0 of Rational IntegrationTester and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2001, 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this Publication	v
Intended Audience	vi
Scope	vi
Typographical Conventions	vi
Contacting IBM Support	vi
Requirements	1
SonicMQ 6.1 Libraries	2
SonicMQ 7.0 Libraries	3
SonicMQ 7.6 Libraries	4
Configuring SonicMQ	5
Configure JMS Administered Objects	6
Viewing Sonic Queues	10
SonicMQ Transport	11
Creating the SonicMQ Transport	12
Configuring the SonicMQ Transport	13
SonicMQ Messages in Rational Integration Tester	19
Message Types	20
Publishing Messages	24
Consuming Messages	26
Troubleshooting	27
No Formatter Available	28
Inauthentic Client Error	28

Corrupted Multipart Messages with UTF-n Charset.	28
Glossary	29
Notices	30
Trademarks and service marks	33

About this Publication

Contents

Intended Audience

Scope

Typographical Conventions

Contacting IBM Support

This guide describes how to configure and run IBM® Rational® Integration Tester with the SonicMQ plugin, focusing specifically on the SonicMQ JMS Provider.

The JMS transport provides support for both Topic and Queue messaging-types, and Rational Integration Tester's access to, and use of, these are identical.

Intended Audience

This document intended to be read by those with a fair understanding and exposure to the concepts involved in both testing and development and in enterprise integration.

Scope

This document is concerned only with the configuration and use of IBM Rational Integration Tester alongside SonicMQ technologies. If you wish to familiarize yourself with these technologies please refer to documents provided by the relevant companies or individuals.

Typographical Conventions

The following typographical conventions are observed throughout this document:

Type	Usage
Constant Width	Program output, listings of code examples, file names, commands, options, configuration file parameters, and literal programming elements in running text.
<i>Italic</i>	Document title names in statements that refer you to other documents. Also used to highlight concepts when first introduced.
Bold	Menu items in graphical user interface windows (such as Microsoft Windows-based or UNIX X Window applications) from which you select options or execute macros and functions. Submenus and options of a menu item are indicated with a “greater than” sign, such as Menu > Submenu or Menu > Option .

Contacting IBM Support

To contact IBM Support, see: www.ibm.com/contact/us/en/

Requirements

Contents

SonicMQ 6.1 Libraries

SonicMQ 7.0 Libraries

SonicMQ 7.6 Libraries

This chapter describes the requirements for using the SonicMQ plugin for Rational Integration Tester.

Depending on the version of SonicMQ in use, specific product libraries are required. The following sections list the required libraries along with their default locations.

Default locations for specified libraries can be modified using Library Manager. For more information, refer to *IBM Rational Integration Tester Installation Guide*.

1.1 SonicMQ 6.1 Libraries

The following files can be found by default in C:\Sonic\MQ6.1\lib:

- broker.jar
- gnu-regexp-1.0.6.jar
- mfcontext.jar
- sonic_Client.jar
- xmlParserAPIs.jar
- sonic_SF.jar
- sonic_Crypto.jar
- mgmt_client.jar
- sonic_SSL.jar

1.2 SonicMQ 7.0 Libraries

The following files can be found by default in `C:\Sonic\MQ7.0\lib`:

- broker.jar
- gnu-regexp-1.0.6.jar
- mfcontext.jar
- sonic_Client.jar
- xmlParserAPIs.jar
- sonic_SF.jar
- sonic_Crypto.jar
- mgmt_client.jar
- asn1.jar
- jsafe.jar
- jsafeJCE.jar
- ras_ssl.jar
- sslj.jar
- sonic_SSL.jar

1.3 SonicMQ 7.6 Libraries

The following files can be found by default in `C:\Sonic\MQ7.6\lib`:

- `asn1.jar`
- `broker.jar`
- `jsafe.jar`
- `jsafeJCE.jar`
- `mfcontext.jar`
- `mgmt_client.jar`
- `ras_ssl.jar`
- `sonic_Client.jar`
- `sonic_Crypto.jar`
- `sonic_SF.jar`
- `sonic_SSL.jar`
- `sslj.jar`
- `xmlParserAPIs.jar`

Configuring SonicMQ

Contents

**Configure JMS Administered
Objects**

Viewing Sonic Queues

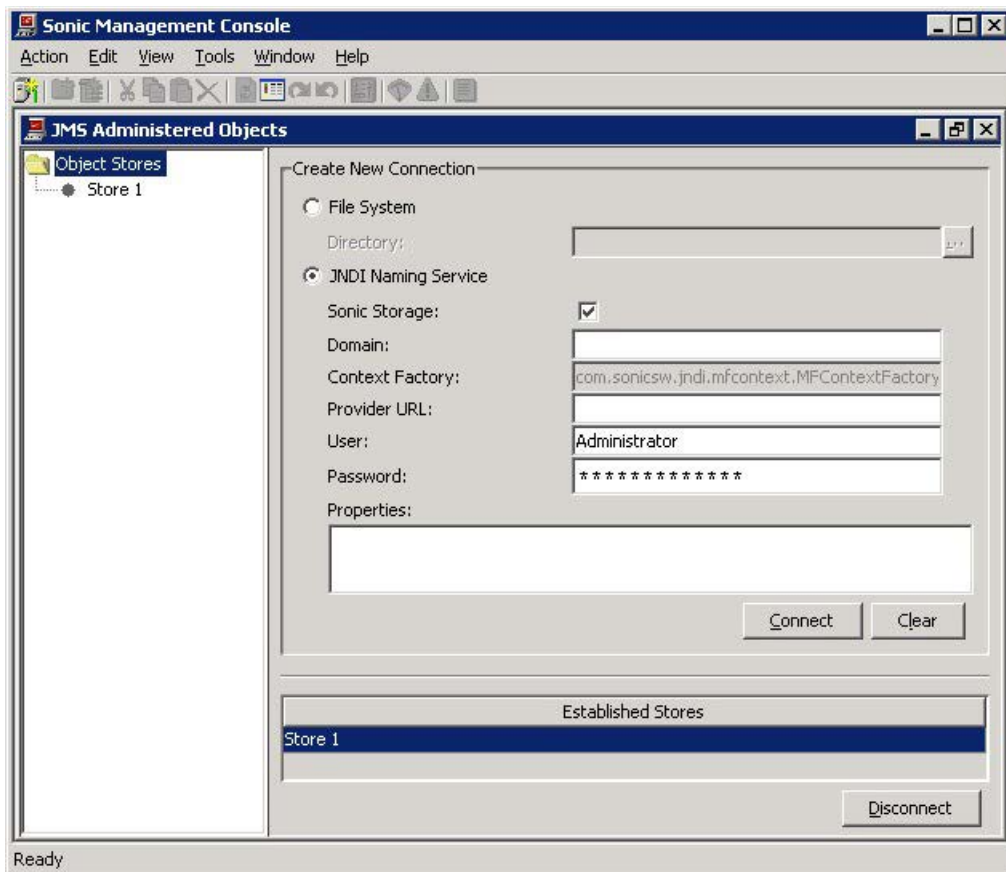
This chapter describes how to configure the Sonic Broker and JMS Object Store prior to use with Rational Integration Tester.

2.1 Configure JMS Administered Objects

This section provides details about configuring JMS administered objects using the Sonic Management Console.

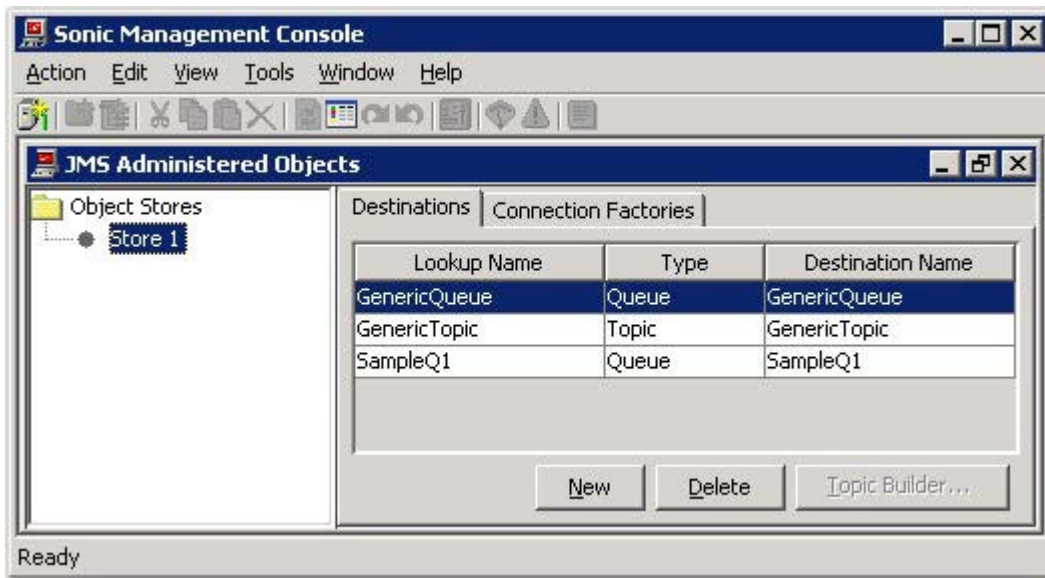
1. Open the Sonic Management Console and select **JMS Administered Objects** from the **Tools** menu.
2. Create a new connection with the following details:

Sonic Storage	Enabled
Context Factory	com.sonicsw.jndi.mfcontext.MFContextFactory
User/Password	Use the user name and password that were created when SonicMQ was installed.



3. Click **Connect**.
4. Select the newly created store under **Object Stores**.

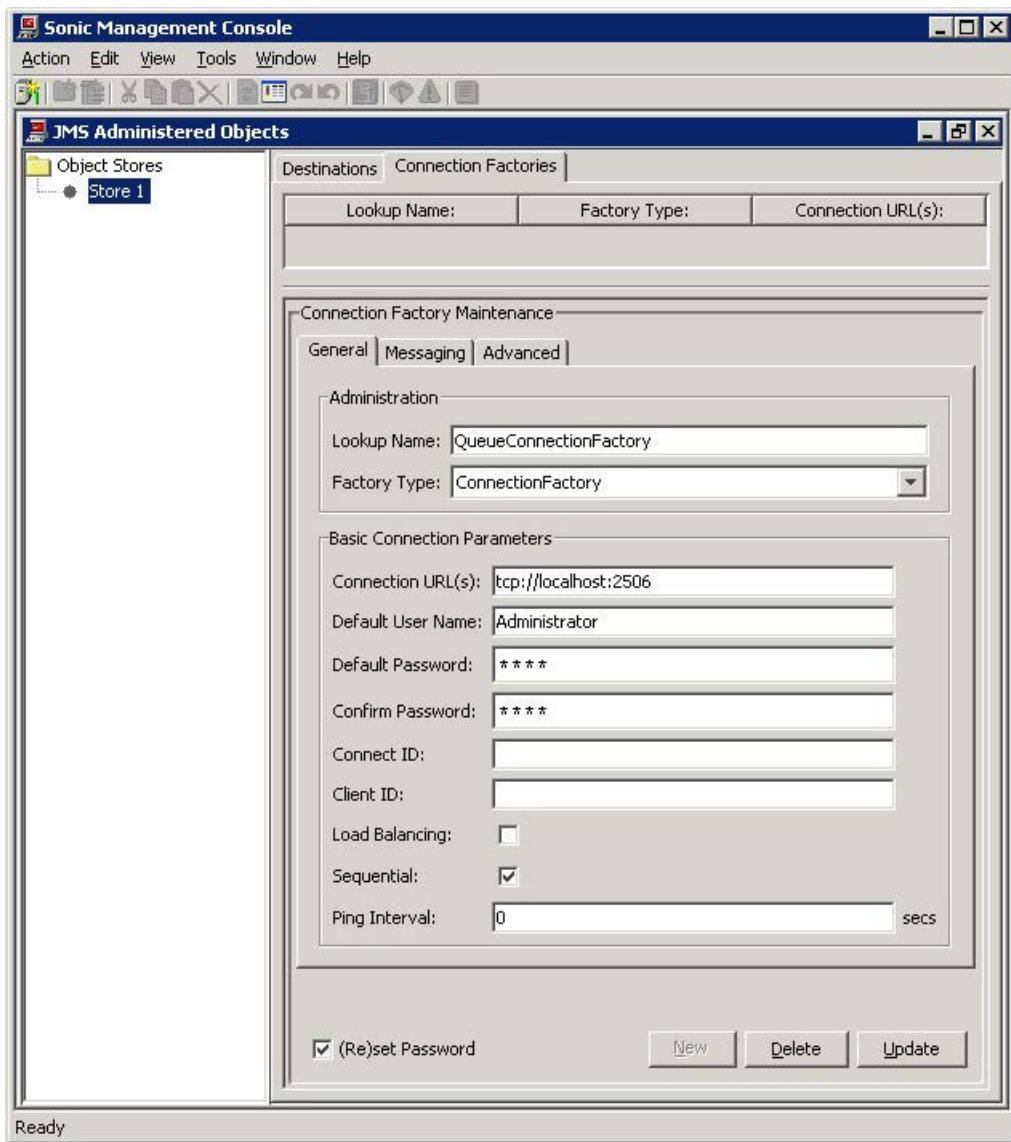
Under the **Destinations** tab, all of the specified queues should point to queues defined within the Broker.



NOTE: It is possible to specify a lookup name that doesn't resolve to an available queue within the Broker. To see how to set up the actual queues in the Broker, refer to [Viewing Sonic Queues](#)

5. Under the **Type** column, designate each queue as a JMS queue or a topic.
6. Set the destination name (that is, the name that the queue/topic will expose by means of JNDI) – this does not have to be the same as the actual lookup name within the Broker.

7. Select the **Connection Factories** tab to provide clients with a route to the provider.



-
8. Under the General tab, click **New** to create a connection factory with the following settings:

Lookup Name/Factory Type	QueueConnectionFactory
Connection URL(s)	tcp://localhost:2506
Default User Name	Administrator
Default Password/Confirm Password	Enter the desired password for the selected user name.
Sequential	Enabled
Ping Interval	0

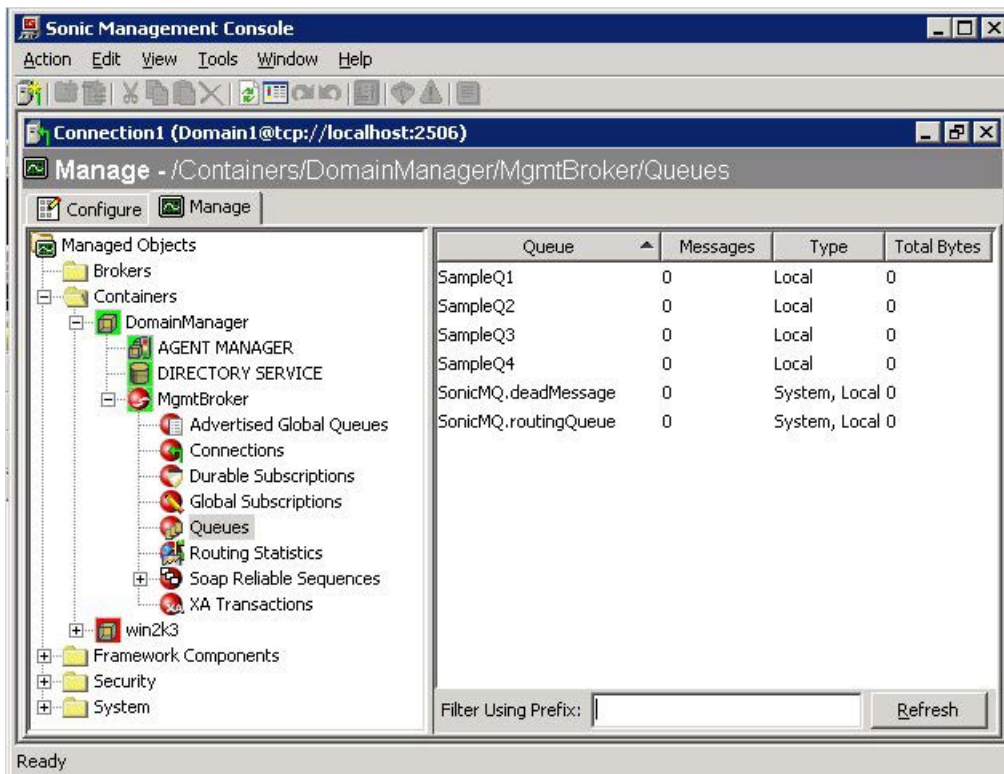
NOTE: If you don't supply the a password or if you specify a different password in the two fields, Rational Integration Tester will not be able to authenticate itself.

NOTE: The settings under the Messaging and Advanced tabs can be left unchanged.

9. When finished, click **Update** to save the new connection.

2.2 Viewing Sonic Queues

If you need to inspect any queues, open the main connection window from the Sonic Management Console and select the **Manage** tab. Navigate through the Containers and, if necessary, clear existing messages by using the right-click menu.



SonicMQ Transport

Contents

Creating the SonicMQ Transport

Configuring the SonicMQ Transport

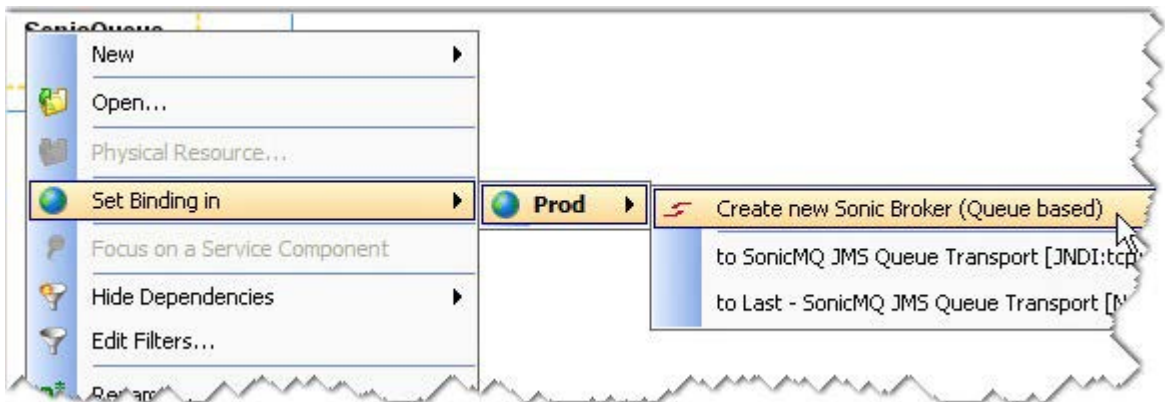
This chapter provides an overview of how to create and configure the SonicMQ transport.

3.1 Creating the SonicMQ Transport

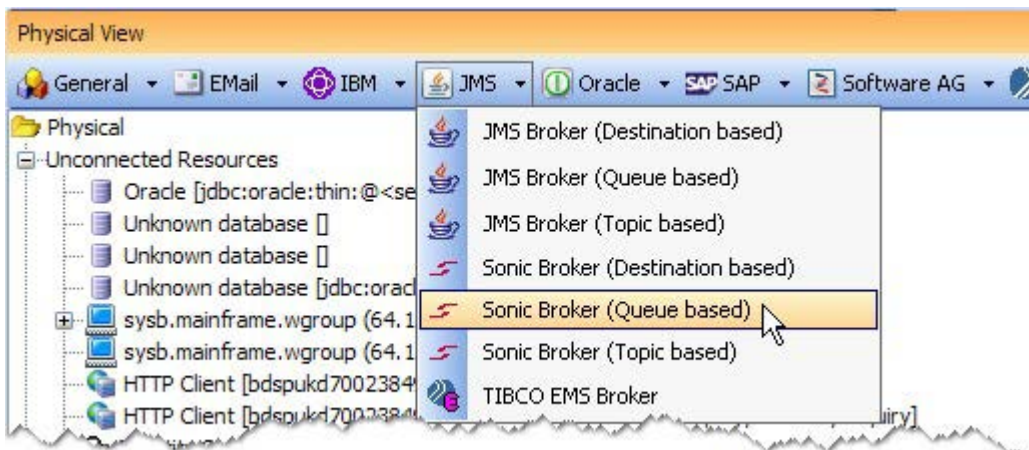
The SonicMQ transport is created when you create a physical Sonic Broker resource in Rational Integration Tester's Architecture School.

In Architecture School, you can create a new resource in two ways:

- In the Logical View, right-click on a webMethods Broker Domain and select the **Set Binding in > [environment] > Create new Sonic Broker** option.



- In the Physical View, select one of the **JMS > Sonic Broker** options.



Each physical Sonic Broker resource will represent a SonicMQ Broker transport that can be selected and configured later on.

3.2 Configuring the SonicMQ Transport

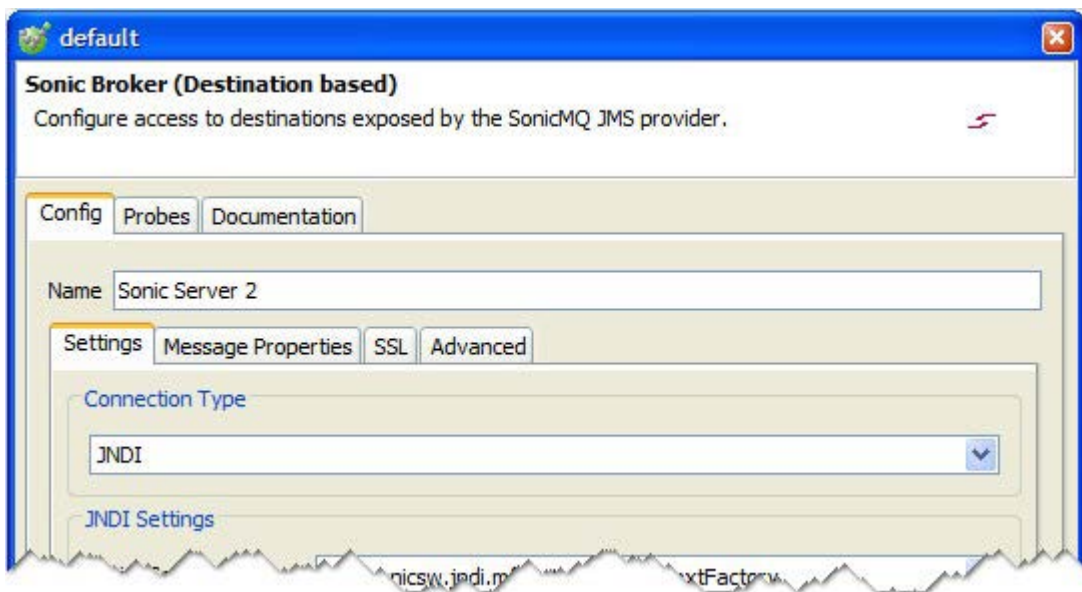
To configure a Sonic Broker transport, double-click the appropriate Sonic Broker resource in Architecture School's Physical View.

The transport settings are broken into [JNDI Connections](#) (**Settings** tab), [Non JNDI Connections](#) (**Direct** tab), [Message Properties](#), [SSL Settings](#), and [Advanced Settings](#).

NOTE: All of the configuration fields support the use of tags. Tags can be entered manually or from the context menu, except in the **Password** field, where tag names must be entered directly (for example, %%pswd%%). Since this field is encrypted, any characters entered will be hidden.

NOTE: You can test the connection parameters at any time by clicking the **Test Transport** button.

If desired, enter a name for the transport in the **Name** field. This can help identify the transport if more than one of the same type is available.



From the combo box under **Connection Type**, select the type of connection to the SonicMQ server that you want to configure. The details of each connection type are described in [JNDI Connections](#) and [Non JNDI Connections](#).

3.2.1 JNDI Connections

For JNDI connections, you must configure the JNDI and Connection settings.

The screenshot shows a configuration window with tabs: Settings, Message Properties, SSL, and Advanced. The 'Settings' tab is active. It contains two main sections: 'Connection Type' and 'JNDI Settings'.

Connection Type

Connection Type: JNDI

JNDI Settings

Initial Context Factory: com.sonicsw.jndi.mfcontext.MFContextFactory

Provider URL: tcp://192.168.10.114:2506

Domain: Domain1

Idle Timeout: 6000

Username: Administrator

Password:

Name	Type	Value
------	------	-------

Connection Settings

Connection Factory: MyConnFactory

☐ Connection authentication same as JNDI ☐ Use JNDI to lookup destination

Username: admin




Password:

Client ID: MyClientID

The details of each set of configuration options can be found in [JNDI Settings](#) and [Connection Settings](#), below.

JNDI Settings

JNDI settings are described in the following table:

Initial Context Factory	The Java class used to obtain context information to perform naming and directory service functions through JNDI. Default values are provided for the JMS implementations supported by Rational Integration Tester.
Provider URL	URL of the JMS server's JNDI tree, specific to the selected JMS implementation. The required format of the URL is provided when one of the supplied context factory entries is selected. This field can be populated with a comma separated list of URLs as per the Sonic format.
Domain	The name of the SonicMQ domain (configured on the Sonic Domain Manager) to which you want to connect.
Idle Timeout	The amount of time (in milliseconds) to attempt the connection before timing out.
Username/Password	The user name and password to send when connecting to JNDI, which should be the same as what was configured for the connection on the Sonic Domain Manager.
Context Properties	<p>In the table at the bottom of the JNDI Settings you can enter additional JNDI properties (name-value pairs) that should be set. These properties are specific to the server you are connecting to, and you should refer to the server's documentation if you are unsure about any of these settings.</p> <ul style="list-style-type: none">• Click  to create a new property, then enter the property name, type, and value in the New Message Property dialog.• Select an existing property and click  to edit it.• Select an existing property and click  to delete it.

NOTE: SSL connections can be configured using the additional context properties.

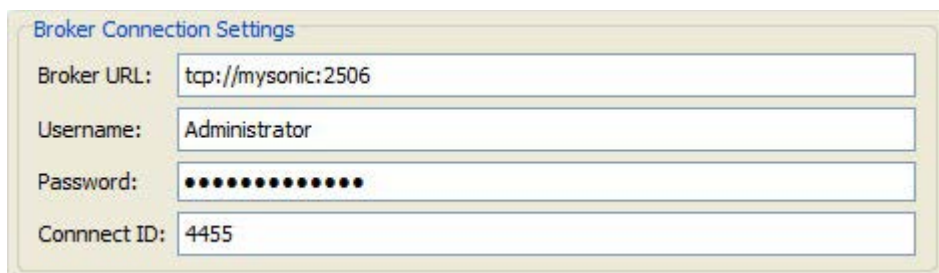
Connection Settings

Connection settings are described in the following table:

Connection Factory	The default JNDI lookup name of the Sonic connection factory, which indicates the location in the JNDI tree to find a Connection Factory object.
Connection authentication same as JNDI	Enable this option to send the user name/password specified under JNDI settings when obtaining a connection from the connection factory. Enable this option only if the authentication for the connection factory is the same as for the JNDI connection (configured in the Sonic Domain Manager).
Use JNDI to lookup destination	Enable this option to use the specified JNDI settings to look up destinations.
Username/Password	The user name and password to use when obtaining a connection from the connection factory.
Client ID	The Sonic client identifier needed for durable topic subscriptions on all connections created using this connection factory.

3.2.2 Non JNDI Connections

It is possible to connect Rational Integration Tester directly to the Sonic destination without connecting to a JNDI by using the **Non JNDI** connection type.



Broker Connection Settings

Broker URL: tcp://mysonic:2506

Username: Administrator

Password:

Connect ID: 4455

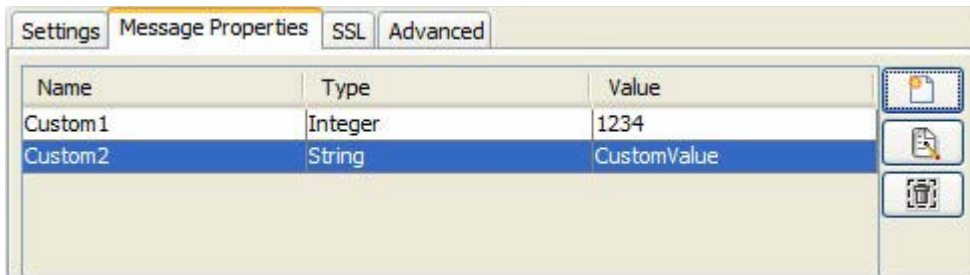
Non JNDI connection settings are described in the following table:




Broker URL	URL of the Sonic Broker to which you want to connect, in the following format: tcp://<server>:<port>
Username/Password	The user name and password to use when obtaining a connection to the Sonic Broker.
Connect ID	The Connect ID (configured on the Broker) to use when obtaining a connection to the Sonic Broker.

3.2.3 Message Properties

It is possible to send additional properties in the header of Sonic messages. These properties can be used, for example, to let recipients make decisions about which messages should be presented to the receiving application.

The **Message Properties** tab allows you to specify additional properties that can be set on each message sent using the specified transport.



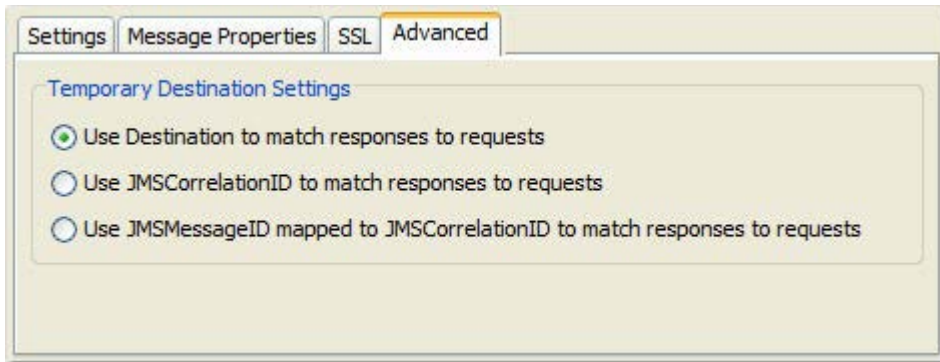
- Click  to create a new property, then enter the property name, type, and value in the **New Message Property** dialog.
- Select an existing property and click  to edit it.
- Select an existing property and click  to delete it.

3.2.4 SSL Settings

The configuration of SSL over JMS will vary according to the specified provider. For this reason, no specific configuration options are available in JMS transports. SSL can be enabled, however, using the additional context properties under [JNDI Settings](#).

3.2.5 Advanced Settings

The use of temporary destinations for receiving replies can be configured under the **Advanced** tab.



The details of each option are described below:

Use Destination to match responses to requests

The default option creates a unique temporary destination for receiving responses. Using this configuration, a separate reply destination is created for each published request.

Use JMSCorrelationID to match responses to requests

This option uses a single temporary reply destination, using the JMSCorrelationID (set as an additional message property) to correlate responses with requests.

Use JMSMessageID mapped to JMSCorrelationID to match responses to requests

This option uses a single temporary reply destination, mapping the JMSMessageID (set as an additional message property) to the JMSCorrelationID to correlate responses with requests.

SonicMQ Messages in Rational Integration Tester

Contents

Message Types

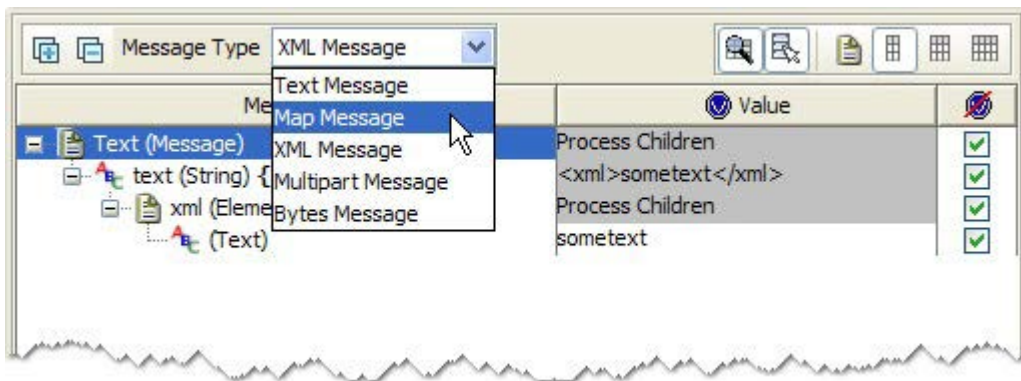
Publishing Messages

Consuming Messages

This chapter provides an overview of how to format messages in Rational Integration Tester using the SonicMQ transport.

4.1 Message Types

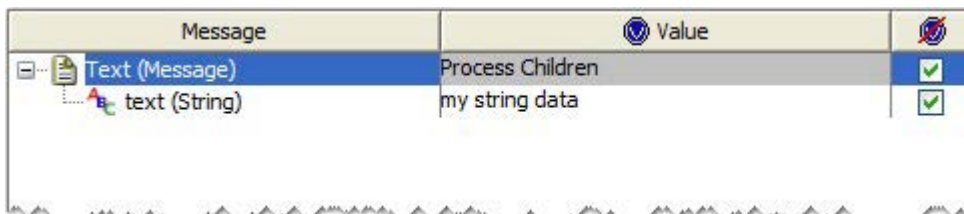
When publishing and consuming SonicMQ messages, the body of the message data type can be specified from a number of types defined by the JMS API specification.



Rational Integration Tester provides generic support for two of these types (Text and Map) and also the Multipart message type that Sonic implement as an extension to the standard API.

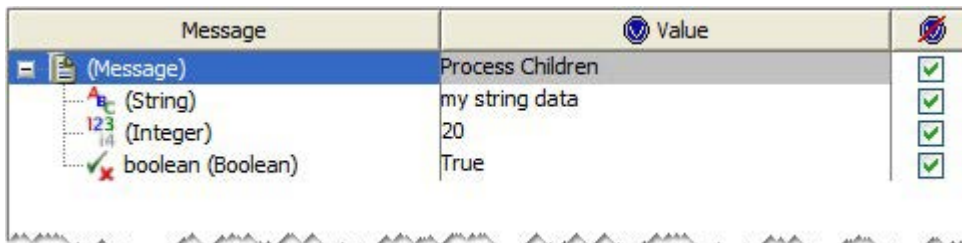
4.1.1 Text Messages

The Text message formatter is constrained by a schema that allows a single field of type String to be set on an outbound message. On receiving a JMS message, a single field message is decompiled with the String data.



4.1.2 Map Messages

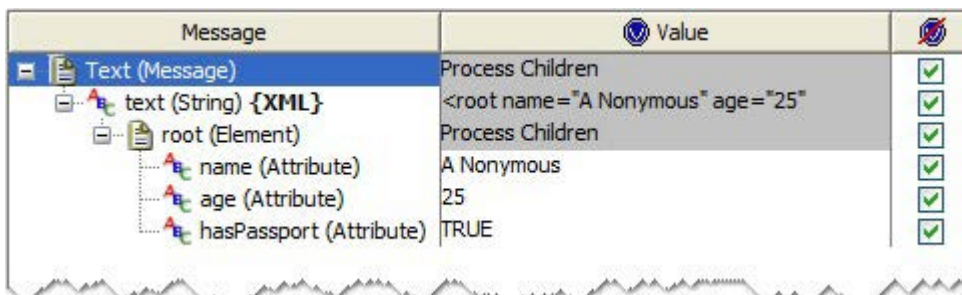
The Map message formatter allows multiple key value pairs to be defined as immediate child fields of the root of the message. These fields can be most primitive scalar types (no embedded messages).



Message	Value	
(Message)	Process Children	<input checked="" type="checkbox"/>
(String)	my string data	<input checked="" type="checkbox"/>
(Integer)	20	<input checked="" type="checkbox"/>
boolean (Boolean)	True	<input checked="" type="checkbox"/>

4.1.3 XML Messages

The XML message formatter sits on top of the Text message formatter and provides guided creation and modification of XML documents within Rational Integration Tester.

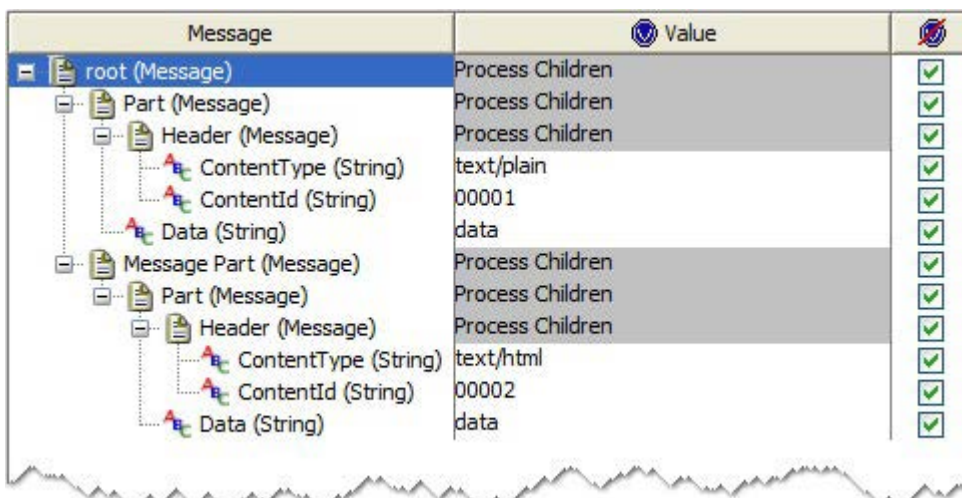


Message	Value	
Text (Message)	Process Children	<input checked="" type="checkbox"/>
text (String) {XML}	<root name="A Nonymous" age="25"	<input checked="" type="checkbox"/>
root (Element)	Process Children	<input checked="" type="checkbox"/>
name (Attribute)	A Nonymous	<input checked="" type="checkbox"/>
age (Attribute)	25	<input checked="" type="checkbox"/>
hasPassport (Attribute)	TRUE	<input checked="" type="checkbox"/>

When publishing or consuming, XML message data is still processed as a standard JMS text message.

4.1.4 Multipart Messages



The Multipart message formatter deals with a specific extension to the JMS API provided by SonicMQ. A Multipart message can contain multiple message parts and multiple parts. Each message part can contain multiple parts, and each part can be of a different type. Rational Integration Tester supports multiple parts where data can be either String, byte array types, or embedded message parts. The message formatter is schema constrained and allows simple adding and editing of parts based on a fixed structure. Each part has its own header properties that contain two fixed fields, ContentType and ContentID, and any number of optional user defined key value pairs.



Message	Value	
root (Message)	Process Children	<input checked="" type="checkbox"/>
Part (Message)	Process Children	<input checked="" type="checkbox"/>
Header (Message)	Process Children	<input checked="" type="checkbox"/>
ContentType (String)	text/plain	<input checked="" type="checkbox"/>
ContentId (String)	00001	<input checked="" type="checkbox"/>
Data (String)	data	<input checked="" type="checkbox"/>
Message Part (Message)	Process Children	<input checked="" type="checkbox"/>
Part (Message)	Process Children	<input checked="" type="checkbox"/>
Header (Message)	Process Children	<input checked="" type="checkbox"/>
ContentType (String)	text/html	<input checked="" type="checkbox"/>
ContentId (String)	00002	<input checked="" type="checkbox"/>
Data (String)	data	<input checked="" type="checkbox"/>

4.1.5 Bytes Messages

The Bytes message formatter is constrained by a schema that allows a single data field of type `ByteArray` to be set on an outbound message. The field must contain hexadecimal or byte values (that is, a whole number of bytes). The Bytes message type can be used to convey payloads such as images or serialized program data (for example, market price updates).

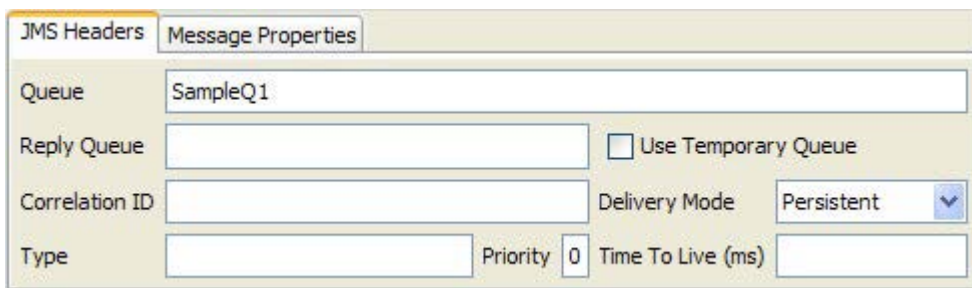
Message	Value	
(Message)	Process Children	
data (ByteArray)	64	

4.2 Publishing Messages

When a message is published within a Rational Integration Tester test, a number of properties can be set on the outbound message.

4.2.1 JMS Header Properties

A number of fixed properties can be defined in the header of a SonicMQ message. The configuration of the SonicMQ transport is the same for topics, queues, and destinations – only the names of the corresponding fields will differ.



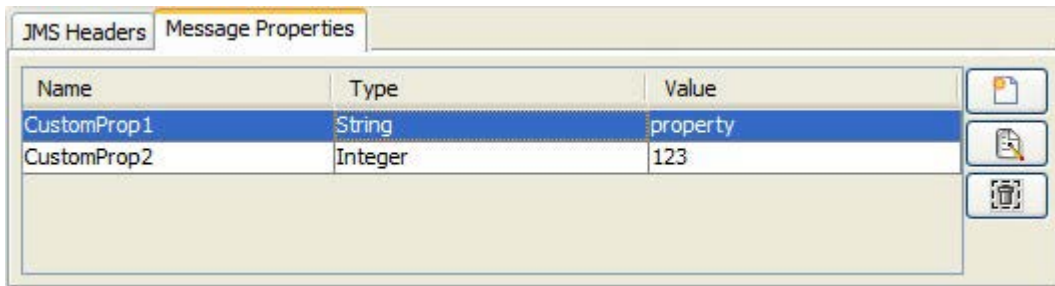
The JMS header properties are described in the following table:




Queue	The SonicMQ queue to which the message should be published.
Reply Queue	The destination to which a reply to the published message should be sent.
Use Temporary Queue	Specifies that a temporary queue should be used (a unique Queue object created for the duration of a QueueConnection).
Correlation ID	An identifier that can be used to link one message with another (for example, a response with a request). This field can be a provider-specific message ID, an application-specific string, or a provider-native byte[] value.
Delivery Mode	The delivery mode to use when publishing, either Persistent or Non-persistent.
Type	The message type, defined by the JMS provider.
Priority	The priority with which the message should be published, from 0 to 9. While this field may be ignored by the provider, messages with a higher priority should be delivered before those with a lower priority.
Time to Live	The amount of time (in milliseconds) after which an undelivered message should be destroyed. If this field is set to "0," the message will never expire.

See the JMS specification for more information about these properties.

4.2.2 Message Properties

The JMS specification also allows any number of additional user defined header properties to be defined. Under the Message Properties tab, these key value pairs can be created and modified.



- Click  to create a new property, then enter the property name, type, and value in the **New Message Property** dialog.
- Select an existing property and click  to edit it.
- Select an existing property and click  to delete it.

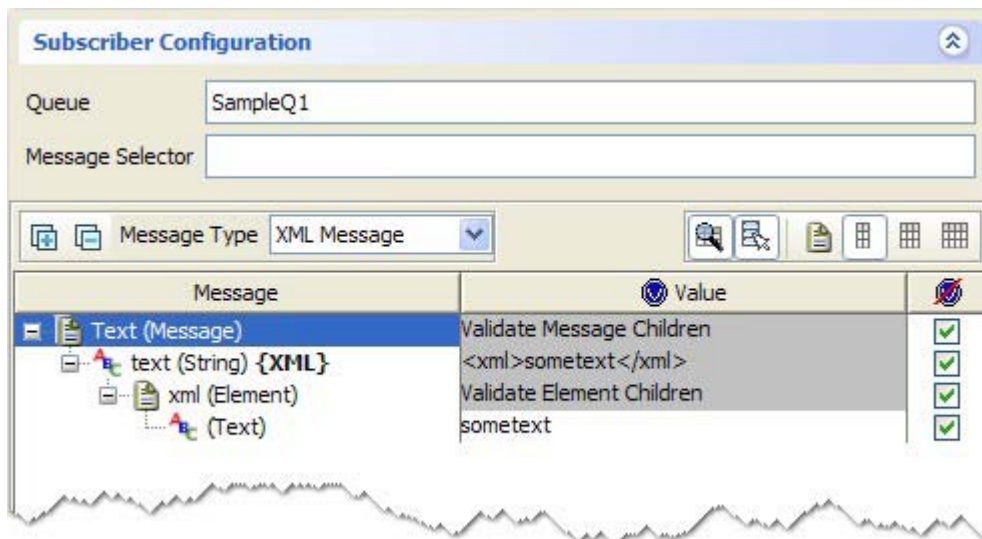
NOTE: These properties can also be used as additional criteria for filtering when consuming messages.

4.2.3 Message Body

The body of the message is built using the message tree. The way in which the message is built is constrained by the schema associated with the message format. See [Message Types](#) for information about message formats and how they affect the construction of messages.

4.3 Consuming Messages

When a message is consumed within a Rational Integration Tester test (for example, by a Subscribe action), you must define the destination from which the message should be received. Additionally, you can apply filters (message selectors) that will determine which messages to consume and which to ignore.



The **Queue/Topic/Destination** is the source from which messages should be consumed. The **Message Selector** allows the usage of a subset of SQL-92 to specify the messages to consume – please see the JMS specification for more information.

Messages will be consumed and decompiled using the formatter selected in the **Message Type** field.

NOTE: Errors will occur if the type of message consumed does not match the type that is specified.

Troubleshooting

Contents

No Formatter Available

Inauthentic Client Error

**Corrupted Multipart Messages
with UTF-n Charset**

This chapter provides answers to common questions and issues that may arise when using the SonicMQ transport.

5.1 No Formatter Available

If the **Formatter** combo box is blank in a messaging action editor, it is likely that the requisite JAR files have not been configured. For information about adding the required JAR files using the Library Manager application, refer to *IBM Rational Integration Tester Installation Guide*.

5.2 Inauthentic Client Error

When creating a Connection Factory in SonicMQ, you must supply and confirm a default password. If you don't supply the correct password on the Rational Integration Tester side, the inauthentic client error will occur.

5.3 Corrupted Multipart Messages with UTF-n Charset

String encoding and decoding of Multipart messages – specifically the Part objects – is performed through the Sonic APIs and is restricted to the system's current encoding type. If necessary, this can be changed by adding a "-Dfile.encoding=UTF-8" or similar line to the JVM settings area in the Library Manager (for more information, refer to *IBM Rational Integration Tester Installation Guide*).

Glossary

The following table below lists some of the key terms used in this document, and provides a description of each.

Term	Description
Field	A bit of data constituent to a message. Most fields are scalar and therefore unitary, equivalent to data attributes. Vector fields are an aggregation of fields both scalar and vector, and are usually referred to as Messages. See also Message.
Message	A unit of information made up of a header consisting of meta-information and a body consisting of the message data.
Host	The computer on which a software process runs.
Publisher-Subscriber	A messaging paradigm whereby a messaging network consists of Publishers and Subscribers.
Transport	Informally, the messaging software in use. For instance, TIBCO Rendezvous, TIBCO ActiveEnterprise, IBM WebSphere® MQ (JMS).
Publishing	Making a message (data) available on a message channel.
Subscribing	Receiving a stream of messages (data) on a given message channel.
Server	A host computer on a network shared by more than one user.
JMS	Java Message Service, a J2EE technology. Several implementations of JMS exist, for instance SonicMQ and TIBCO EMS.
JMS Topic	The JMS equivalent of a subject, used typically for one-to-many messaging, where a message is broadcast rather than sent to a specific recipient.
JMS Queue	A JMS Queue is normally used for one-to-one messaging.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT,

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited
Intellectual Property Law
Hursley Park
Winchester
SO21 2JN
Hampshire
United Kingdom

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the

capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corporation 2001, 2012.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

