

Rational Integration Tester



Reference Guide for TCP/UDP Sockets

Version 8.0.0

Note

Before using this information and the product it supports, read the information in “Notices” on page 23.

This edition applies to version 8.0.0 of Rational Integration Tester 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
Sockets Overview	1
Application Layer	2
Transport Layer	3
Client or Server Designation	4
Connections	4
TCP/UDP Transport	5
Creating a TCP/UDP Transport	6
Configuring a TCP/UDP Transport	7
Using Schemas with a TCP/UDP Transport	17
Troubleshooting	20
No Messages are Received	21
No Watch Messages are Received	21
The Last Message is not Received	21
Bind Error	21
Error After Changing Transport Interaction Type	21
Standalone Windows	21

Glossary	22
Notices	23
Trademarks and service marks	26

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 Sockets plugin, which provides support for TCP and UDP socket-based communications.

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 IBM Rational Integration Tester, its configuration, and use alongside socket-based technologies.

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/

Sockets Overview

Contents

Application Layer

Transport Layer

Client or Server Designation

Rational Integration Tester provides the ability to create transports that facilitate the communication between clients and servers using both TCP and UDP based sockets.

This chapter provides a brief overview of the OSI Application and Transport layers and an explanation of how Rational Integration Tester provides access to them.

1.1 Application Layer

The application layer is the highest layer in the OSI model containing the application-layer protocols which define the rules that software applications use to exchange data. For example, web browsers communicate with web servers using HTTP (Hypertext Transfer Protocol), while email applications communicate using the SMTP (Simple Mail Transfer Protocol) and POP3 (Post Office Protocol) protocols.

Rational Integration Tester provides support for the application layer by packetising the data that it receives before making it available as individual messages to subscription-based actions within a test. See [Configuring a TCP/UDP Transport](#) for more information.

1.2 Transport Layer

The transport layer oversees the delivery of data from a process on one computer to a process on another computer. Transport layer protocols act as liaisons between the application layer protocols and the services provided by the network, there are two supported by Rational Integration Tester.

1.2.1 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is a connectionless (and therefore unreliable) transport protocol. It performs very little error checking and does not add anything to IP services except to provide process-to-process communication instead of host-to-host communication.

UDP is a simple protocol with minimum overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a message using UDP takes much less time than using TCP.

NOTE: UDP is a convenient protocol for multimedia and multicasting applications.

1.2.2 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is a reliable but complex transport-layer protocol. TCP adds connection-oriented features and reliability to IP.

TCP is a reliable, stream delivery service that guarantees delivery of a data stream sent from one host to another without duplication or lost data. Since packet transfer is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends, and waits for acknowledgment before sending the next packet. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires. The timer is needed in case a packet gets lost or corrupted.

1.3 Client or Server Designation

When working with sockets, it is necessary to denote which end of the two processes will initiate the conversation (the client) and which will wait for its peer to start (the server). This designation is described in more detail in the following chapter.

1.4 Connections

The connection from one peer to another will only be established (that is, initiated by a client or accepted by a server) when a test-based publisher or subscriber is actually executed in Rational Integration Tester.

The connection should exist for the duration of the test and will be shared by all subsequent messaging operations within the same test sequence. If the test sequence executes a child test, then the connection will be passed to the child so that it can communicate with the same peer (if necessary).

TCP/UDP Transport

Contents

[Creating a TCP/UDP Transport](#)

[Configuring a TCP/UDP Transport](#)

[Using Schemas with a TCP/UDP Transport](#)

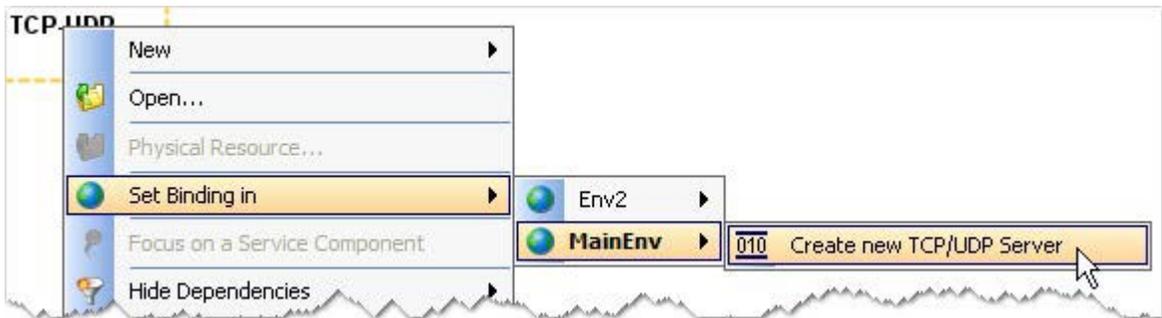
This chapter provides an overview of how to create and configure the TCP/UDP transport.

2.1 Creating a TCP/UDP Transport

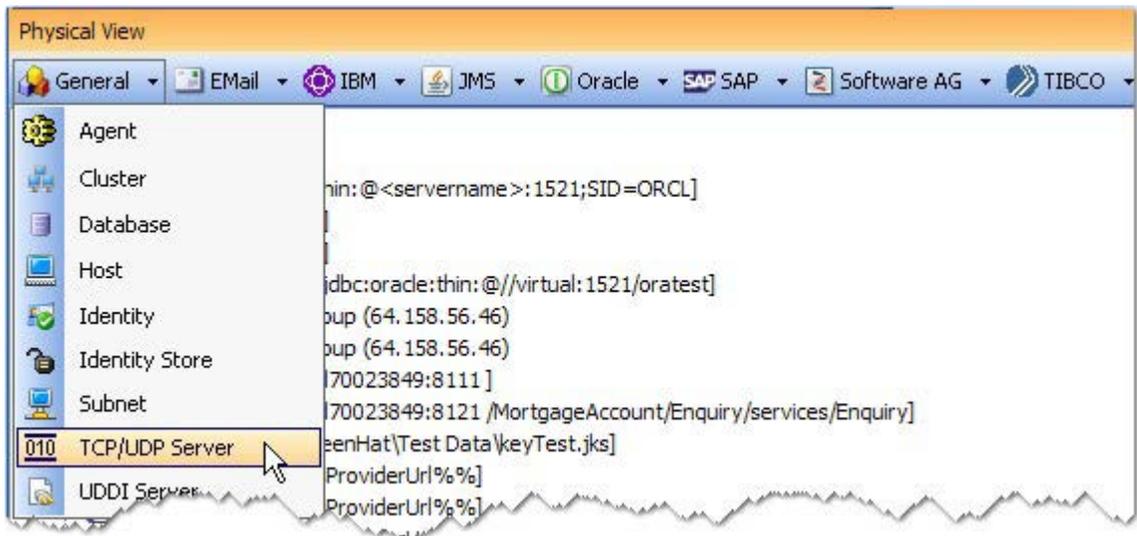
A TCP/UDP transport is created when you create a physical TCP/UDP Server resource in Rational Integration Tester's Architecture School.

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

- In the Logical View, right-click on a TCP/UDP Connection and select the **Set Binding in > [environment] > Create new TCP/UDP Server** option.



- In the Physical View, select the **General > TCP/UDP Server** option.

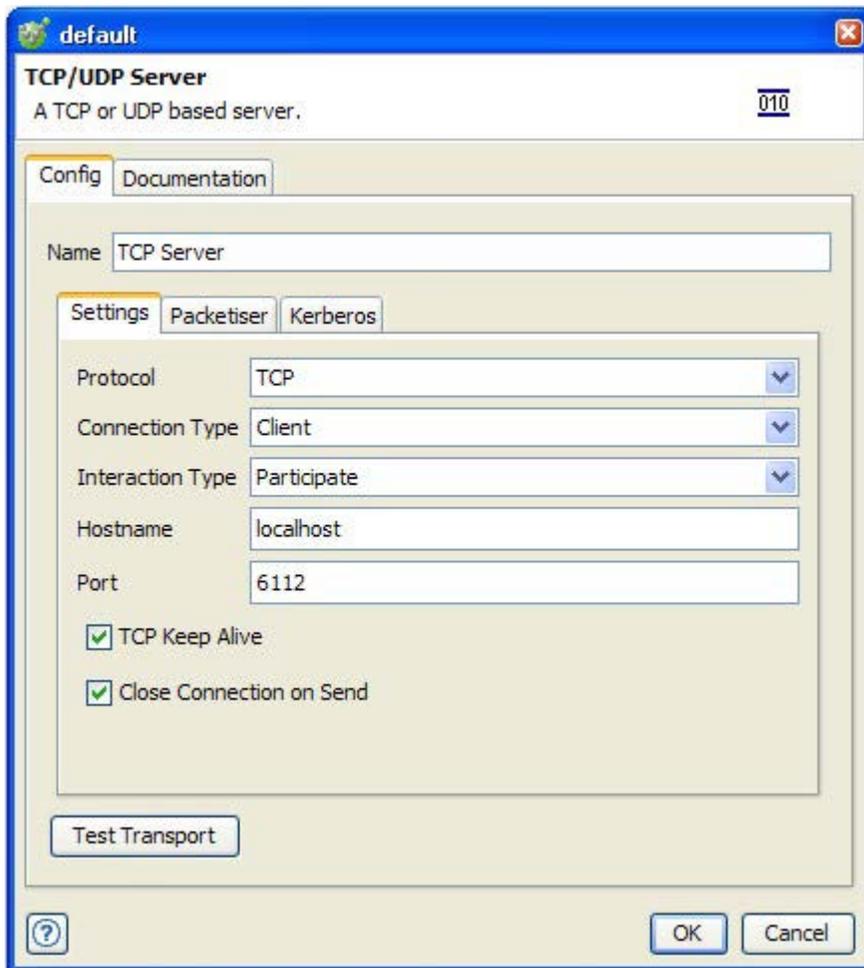


Each physical resource will represent a TCP/UDP transport that can be selected and configured later on.

NOTE: Before it can be utilized, the physical TCP/UDP Server must be bound to a (logical) TCP/UDP Connection.

2.2 Configuring a TCP/UDP Transport

To configure a transport, double-click the appropriate TCP/UDP Server resource in Architecture School's Physical View. If desired, enter a name for the transport in the **Name** field (for example, to help identify it when multiple TCP/UDP servers are available).



The transport settings are broken into the [Transport Layer Settings](#) (**Settings** tab), the [Application Layer Settings](#) (**Packetiser** tab), and the [Kerberos Authentication Settings](#) (**Kerberos** tab).

NOTE: You can test the connection to the indicated host name and port by clicking the **Test Transport** button.

NOTE: If you change any of the transport settings, you need to close and re-open any message editors that are using the transport.

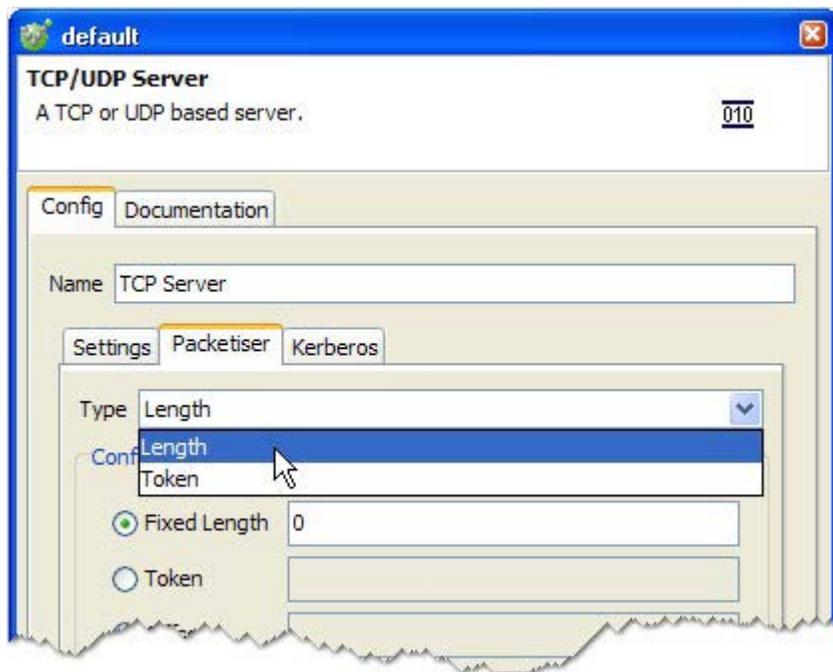
2.2.1 Transport Layer Settings

Transport layer settings are configured under the **Settings** tab (shown in the previous section). The available settings are described in the following table:

Protocol	Select the protocol for the transport – TCP or UDP – from the dropdown menu.
Connection Type	Select whether the transport will act as a client or a server. When running as a client, Rational Integration Tester will attempt to initiate a connection to the specified host (using the configured port and protocol) before sending or receiving its data. When running as a server, Rational Integration Tester will await an incoming connection before exchanging information.
Interaction Type	When using TCP, indicate whether the transport should only watch (snoop) messages or actively participate in them. Note: You must ensure that the correct network device is selected in Library Manager (for more information, refer to <i>IBM Rational Integration Tester Installation Guide</i>). Note: It is not possible to watch your own messages as they are sent internally (that is, not by means of the network interface).
Hostname	The IP address or host name of the peer with which you want to communicate. DNS must be configured properly on the local machine if a host name is provided.
Port	The TCP or UDP port upon which the connection should be established.
TCP Keep Alive	For TCP connections, enable this option to request that the socket implementation enable the use of “keep-alive” packets, which are used to maintain the stream connection where possible.
Close Connection on Send	Enable this option to close the connection as soon as data has been sent.

2.2.2 Application Layer Settings

Application layer settings, configured under the **Packetiser** tab, control how Rational Integration Tester will provide information to subscriber-based actions.



For all packetiser types, you have the option to update outbound messages with relevant packet information by enabling the **Update outbound message with packet details below** option.

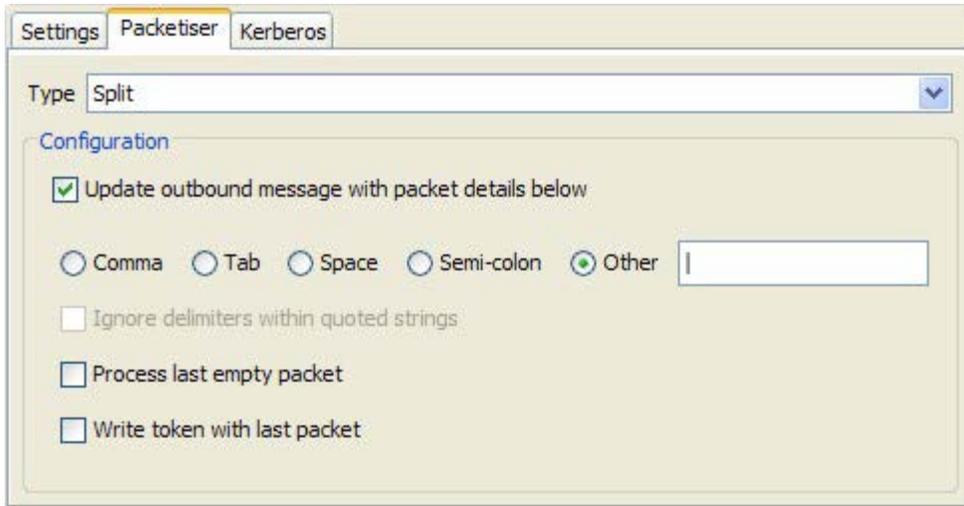
If the configured packetiser settings support it, outbound messages will be updated to include relevant packet information (for example, for a length prefix packetiser, the first *n* bytes are used to denote how many more bytes should be read to constitute a message). If the box is ticked then when publishing, the outbound packet will have *n* bytes prepended to indicate the length of the remaining data.

This option is disabled if the configuration of the packetiser does not support updating.

The following packetizers are supported: [Split](#), [Length](#), [Token](#), [Delimited](#), and [Swift](#).

Split

The **Split** option can packetize (split) the data stream based on a user-defined delimiter.



You can select one of the existing delimiter types (comma, tab, space, semi-colon), or select **Other** and enter the delimiter character(s) in the field provided.

When reading data, use the **Process last empty packet** option depending how the stream ends. Enable this option if the data ends with your delimiter and you want to process one more packet as an empty string (that is, ""). Disable this option if the delimiter indicates that there are no more packets.

When writing to a record, enable the **Write token with last packet** if you want the packetiser to write out the delimiter as the last character, so the record ends with a delimiter. For example: "... |myfieldvalue|" when enabled and "... |myfieldvalue" when disabled.

Length

Length-based data is broken up according to a fixed length or a dynamic length that is encoded in the packet. For dynamic lengths, the position can be specified in relation to a known identifier (a token) or as a fixed offset from the start of the message.

The screenshot shows a configuration window for the 'Length' type. The 'Type' dropdown is set to 'Length'. Under the 'Configuration' section, the checkbox 'Update outbound message with packet details below' is checked. Four radio buttons are present: 'Fixed Length' (selected with '0' in the input field), 'Token' (empty input), 'Offset' (selected), and 'Prefix' (empty input). Below these, there are two more input fields: 'Size' with '4' and 'Format' with 'Bytes'. At the bottom, there are two unchecked checkboxes: 'Swap Bytes' and 'Length includes prefix'.

The available settings are described in the following table:

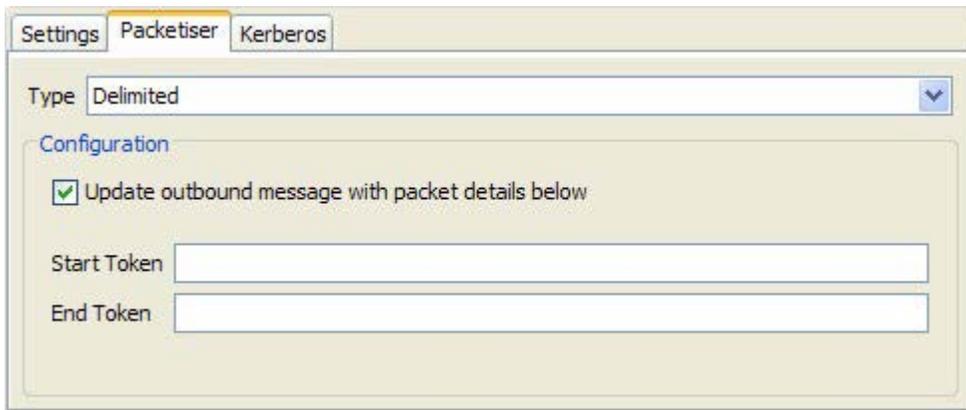
Fixed Length	Each packet will be the same designated size. Once this many bytes are received from the socket they will be passed to the subscriber. To test an input stream, set a fixed length of 0 to indicate that data will be packetised in-line with the physical packet size that has been received from the underlying network.
Token	Indicates that the string of entered characters will mark the start of where the length information is contained within the packet.
Offset	Length information can be found following the entered number of bytes. In the above example, 16 bytes of data precede the length information.
Prefix	A number of bytes / characters at the start of the packet that denote the length of the remainder of the packet.

When using the Token, Offset, or Prefix modes, the following options control how the actual packet length will be read from the stream of information.

Size	The number of bytes that contain the length information.
Format	Indicates whether the values should be treated as raw values (Bytes) or translated from their ASCII equivalent (ASCII).
Swap Bytes	Indicates whether the packetiser should swap the order of the bytes before treating the data as the length of the packet.
Length includes prefix	Indicates whether the length information found within the packet includes the size of the prefix.

Delimited

The **Delimited** option can be used to extract data that is designated by a start and end token.



The available settings are described in the following table:

Start Token	A series of characters that denotes the start of a record.
End Token	A series of characters that denotes the end of a record.

Token

Token-based data is separated by known characters (tokens) in the incoming data stream. In this case, the packetiser looks for one token at the start of the data stream and another token at the end. A number of additional bytes of data can be included following the end token.

The available settings are described in the following table:

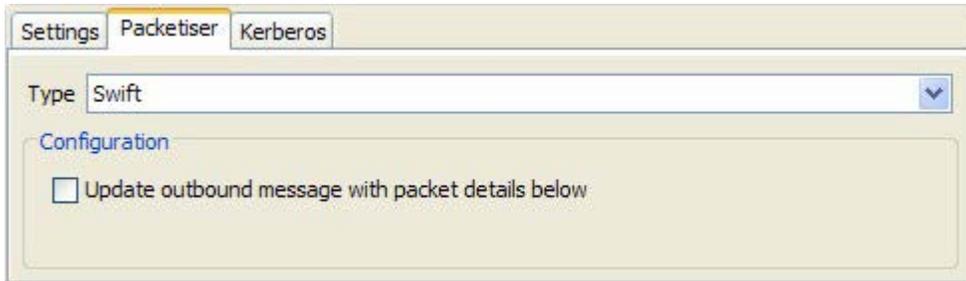
Start Token	A series of characters that denotes the start of a packet.
End Token	A series of characters that denotes the end of a packet.
End Token Data Length	An optional quantity of data that can be present following the end token (for example, this is often used to convey checksum information that is appended to a fully formed packet prior to transmission).

Start and end tokens can accept representations of certain non-printable characters, as follows:

Esc Character	Definition	Unicode Character
\b	Backspace character	\u008
\t	Tab character	\u009
\n	Newline character	\u00A
\f	Form feed	\u00C
\r	Carriage return	\u00D
\0	Null	\u000
\0xHH	2 Char Hex value, for example, \0x02 (STX)	\u002

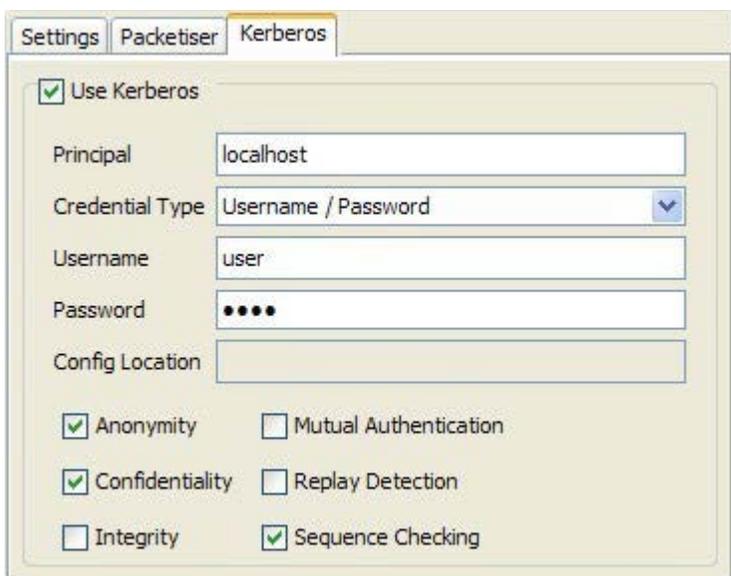
Swift

The Swift option can be used to break up the data stream into a Swift message based on simple Swift rules (that is, {n:...}).



2.2.3 Kerberos Authentication Settings

Kerberos authentication settings are configured under the **Kerberos** tab.



The screenshot shows a dialog box with three tabs: 'Settings', 'Packetiser', and 'Kerberos'. The 'Kerberos' tab is active. At the top, there is a checkbox labeled 'Use Kerberos' which is checked. Below this are several input fields: 'Principal' with the text 'localhost', 'Credential Type' with a dropdown menu showing 'Username / Password', 'Username' with the text 'user', and 'Password' with a masked field of dots. There is also an empty 'Config Location' field. At the bottom, there are six checkboxes for security options: 'Anonymity' (checked), 'Confidentiality' (checked), 'Integrity' (unchecked), 'Mutual Authentication' (unchecked), 'Replay Detection' (unchecked), and 'Sequence Checking' (checked).

To enable Kerberos authentication, tick the box next to **Use Kerberos**. The configuration options are described in the following table:

Principal	Specifies the alias of the principal.
Credential Type	Indicates whether the identity details are a user name/password pair or specified in a file.
Username/Password	The user name/password of the principal.
Config Location	Full path to a file containing principal configuration details.
Anonymity	Requests that the initiator's identity not be disclosed to the acceptor.
Confidentiality	Requests that data confidentiality be used when sending data.
Integrity	Requests that data integrity be enabled when sending data.
Mutual Authentication	Requests that mutual authentication be done during context establishment.
Replay Detection	Requests that replay detection be enabled for the per-message security services after context establishment.
Sequence Checking	Requests that sequence checking be enabled for the per-message security services after context establishment.

2.2.4 Recording Settings

TCP transport recording settings are configured under the **Recording** tab.

The **Recording Mode** list has the following options:

- **Packet Capture**

This requires WinPcap. For information about installing WinPcap, refer to *IBM Rational Integration Tester Installation Guide*.

- **External Proxy Server**

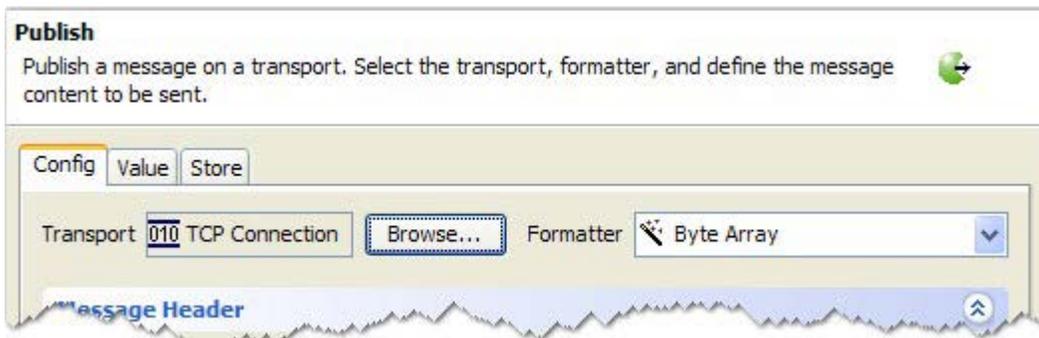
The proxies in the Rational Integration Tester Platform Pack also enable Rational Integration Tester and Rational Test Virtualization Server to record all TCP traffic routed through the proxy. For information about this option, refer to *IBM Rational Integration Tester Reference Guide* and *IBM Rational Test Virtualization Server Reference Guide*.

For more information about TCP recording, refer to *IBM Rational Integration Tester Reference Guide*.

2.3 Using Schemas with a TCP/UDP Transport

While it may not seem obvious, schemas – including text-based schemas – can be applied to messages that use the TCP/UDP transport. The following steps illustrate how to apply an XML-based schema to a publisher that uses the TCP/UDP transport.

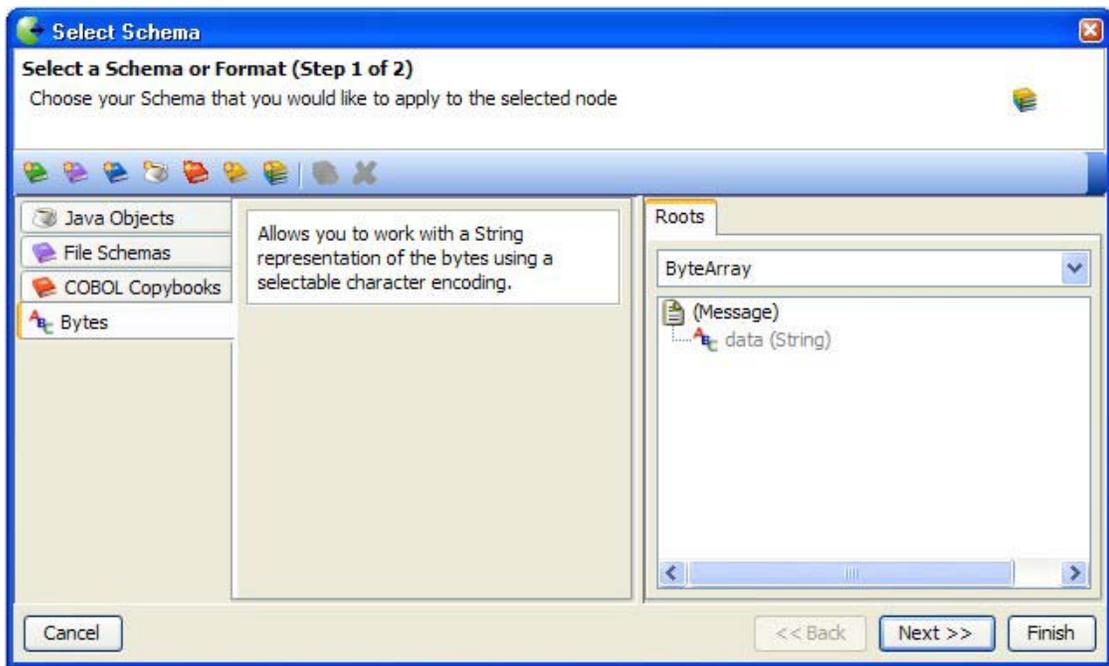
1. In a new or existing test, create or open a Publish test action.
2. Select an existing **TCP/UDP Connection** (logical resource) as the transport (the formatter will be **Byte Array**).



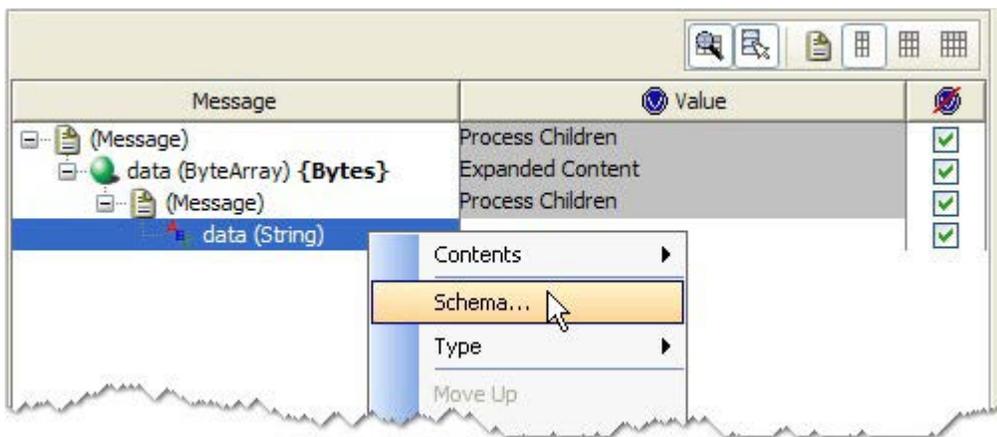
3. In the message body, right-click the **data (ByteArray)** node and select **Schema** from the context menu.



-
4. In the **Select Schema** wizard, select the **Bytes** schema type tab, then click **Finish**.



-
5. Apply a schema to the **data (String)** element in the same way as before.



Now when the **Select Schema** wizard is displayed, you can select a text-based schema, as desired.

See the “Schemas” chapter in the *IBM Rational Integration Tester Reference Guide* for more information.

Troubleshooting

Contents

No Messages are Received

No Watch Messages are Received

The Last Message is not Received

Bind Error

**Error After Changing Transport
Interaction Type**

Standalone Windows

This chapter provides answers to common questions and issues that may arise when using TCP/UDP transports.

3.1 No Messages are Received

Verify the packetiser settings. If these are wrong then the incoming stream of data can not be broken up properly and it can not be presented to the subscriber.

3.2 No Watch Messages are Received

The most likely cause is that you are trying to watch messages sent from the same machine. These messages are sent internally without using the network card so cannot be captured.

3.3 The Last Message is not Received

If no end token is specified when using the token-based packetiser, packets will be determined by awaiting the next start token. This means that each message will only be presented to the subscriber when a subsequent message is received.

3.4 Bind Error

When running as a server, the transport will open a listener on the specified port. If another process or another transport has already started listening on this port, then a collision will occur. This will result in a bind error being reported to the user.

You need to stop the other process that is using the port or configure the other transport to use a different value.

3.5 Error After Changing Transport Interaction Type

If you change the transport's Interaction Type (that is, Watch or Participate) while any editors that use the transport are opened, you may see the following when attempting to publish: "Unable to send information whilst configured in watch mode".

In this case, close any editors after saving the transport and re-open them.

3.6 Standalone Windows

If there are issues accepting subsequent connections when running as a server, the server component can be restarted by using the Disconnect option in the context menu of the underlying transport (in Architecture School's Physical View).

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.

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.

IBM[®]