IxChariot® User Guide

Corporate Headquarters	**Ixia Worldwide Headquarters** 26601 W. Agoura Rd. Calabasas, CA 91302 USA +1 877 FOR IXIA (877 367 4942) +1 818 871 1800 (International) (FAX) +1 818 871 1805 sales@ixiacom.com	**Web site:** www.ixiacom.com **General:** info@ixiacom.com **Investor Relations:** ir@ixiacom.com **Training:** training@ixiacom.com Support: support@ixiacom.com +1 818 595 2599 For the online support form, go to: http://www.ixiacom.com/support/inquiry/
EMEA	**Ixia Technologies Europe Limited** Part 2nd floor, Clarion House, Norreys Drive Maidenhead, UK SL6 4FL +44 (1628) 408750 FAX +44 (1628) 639916 salesemea@ixiacom.com	**Support:** eurosupport@ixiacom.com +40 21 3015699 For the online support form, go to: http://www.ixiacom.com/support/inquiry/?location=emea
Asia Pacific	**Ixia Pte Ltd** 210 Middle Road #08-01 IOI Plaza Singapore 188994	**Support:** Support-AsiaPac@ixiacom.com +65 6332125 For the online support form, go to: http://www.ixiacom.com/support/inquiry/
Japan	**Ixia Communications KK** Nishi-Shinjuku Mitsui Bldg 11F 6-24-1, Nishi-Shinjuku, Shinjuku-ku Tokyo 160-0023 Japan	**Support:** Support-Japan@ixiacom.com +81 3 5326 1948 For the online support form, go to: http://www.ixiacom.com/support/inquiry/
India	**Ixia Technologies Pvt Ltd** 2nd Floor, 19/1, Vithall Malya Road, Bangalore 560 001 India	**Support:** Support-India@ixiacom.com +91 80 22161000 For the online support form, go to: http://www.ixiacom.com/support/inquiry/?location=india

For viewing the FAQs related to the product, go to Ixia Technical Support Online:

https://ebsoprod.ixiacom.com/OA_HTML/jtflogin.jsp

Table of Contents

IxChariot® User Guide

Chapter 2   IxChariot Operational Overview

Chapter 3   Getting Started with the IxChariot Console User Interface

Chapter 4   Using Ixia Test Ports

Chapter 5   How To

Chapter 6   IxChariot User Settings

Chapter 7   Test Run Options

Chapter 8   Large-Scale Tests in IxChariot

Chapter 9   Quality of Service Testing

Chapter 10   Concepts

Chapter 11   Understanding Results

Chapter 12   Troubleshooting

Appendix A IxChariot File Types

1

About This Guide

The information in this section is provided to help you navigate this manual and make better use of its content. A list of related documentation is also included.

Topics:

- *Purpose* on page 1-1
- *Manual Content* on page 1-2
- *What's New in IxChariot 7.10 SP5* on page 1-3
- *Version 7.10 SP5 Compatibility Considerations* on page 1-4
- *Related Documentation* on page 1-5
- *Technical Support* on page 1-6

Purpose

This manual provides a complete reference for IxChariot users who need to:

- Set up an IxChariot test network.
- Define IxChariot tests.
- Schedule and execute IxChariot tests.
- Analyze the test results obtained from IxChariot tests.

Manual Content

This manual contains the following sections:

Name	Description
About This Guide	Provides information on this manual, including its purpose, content, and related documentation. Also explains how to contact technical support.
Chapter 2, *IxChariot Operational Overview*	Provides a high-level overview of the IxChariot test architecture and operations.
Chapter 3, *Getting Started with the IxChariot Console User Interface*	Describes the major features and functions of the IxChariot Console GUI.
Chapter 4, *Using Ixia Test Ports*	Provides instructions for setting up Ixia ports for use in IxChariot tests.
Chapter 5, *How To*	Provides instructions for using the IxChariot Console tools and features, including test definition and execution.
Chapter 6, *IxChariot User Settings*	Provides a description of the IxChariot global user settings. The user settings gives you a great deal of control over how tests are run and how results are reported.
Chapter 7, *Test Run Options*	Provides a description of the run options that you can set for individual IxChariot tests.
Chapter 8, *Large-Scale Tests in IxChariot*	This chapter focuses on the requirements for creating large-scale tests (tests with as many as 100,000 endpoint pairs).
Chapter 9, *Quality of Service Testing*	Provides a detailed description of the requirements, options, and procedures for incorporating QoS into your IxChariot testing.
Chapter 10, *Concepts*	Provides in-depth, conceptual support for specialized testing situations.
Chapter 11, *Understanding Results*	Describes the results generated by IxChariot tests, provides explanations intended to help you interpret your results, and includes technical information about how IxChariot generates timing measurements.
Chapter 12, *Troubleshooting*	Describes IxChariot Error messages, and provides guidelines to help you find the information necessary to solve problems you encounter.
Appendix A, *IxChariot File Types*	Provides a description of the types of files that IxChariot and the endpoints use and create.

What's New in IxChariot 7.10 SP5

Ixia is pleased to release a number of new features and feature enhancements in IxChariot 7.10 SP5. Principal new functionality is summarized as follows:

- **IxOS 6.10 GA is now supported**

- **Android version 3.2 is now supported by our Android endpoint**

Refer to the IxChariot 7.10 SP5 Release Notes for information about additional enhancements and product changes.

Version 7.10 SP5 Compatibility Considerations

Test files

The tests you used in previous versions of IxChariot are compatible with version 7.10 SP5. You can save test files in version 7.10 SP5/SP4, 7.10 SP3, 7.10 SP1/ 7.10 SP2, 7.10, 7.0, 6.70, 6.60/6.50 SP2, 6.50, 6.40, 6.30, 6.25/6.20, 6.10, 6.0, 5.40/ 5.20/5.10, 5.0, or 4.3 format.

Script files

Scripts you create or modify in IxChariot version 7.10 SP5 may be saved in script formats from versions 7.10 SP1/7.10 SP2/7.10 SP3/7.10 SP4/7.10 SP5, 7.10/7.0, 6.50/6.60/6.70, 6.40, 6.20/6.25/6.30, 6.0/6.10, 5.10/5.20/5.40, 4.1/4.2/4.3/5.0 of IxChariot.

If you load a script from an older version of IxChariot, it is automatically upgraded to the version 7.10 SP1/7.10 SP2/7.10 SP3/7.10 SP4/7.10 SP5 format.

Refer to *What's New in IxChariot 7.10 SP5* on page 1-3 for a summary of the new features in this release.

Socket buffers

As of IxChariot 7.10 SP1, the value standing for DEFAULT used in setting socket buffers has changed from 0 to 2147483647. Consequently, the file formats for IxChariot tests, scripts, application groups, IPTV channels and IPTV receivers have changed as well. As such, the following IxChariot tests upgrading/ downgrading rules apply starting with this release:

a) Upgrading

If the socket buffer was set to DEFAULT, its value must be updated from 0 to 2147483647;

If the socket buffer was set to 2147483647 (currently the value of DEFAULT), its value must be updated to 2147483646;

b) Downgrading

If the socket buffer was set to DEFAULT, its value must be updated from 2147483647 to 0;

If socket buffer was set to 0, the same value must be maintained (in this case, 0=DEFAULT).

Deconfigure Ports Feature Compatibility

As of IxChariot 7.10 SP3, you can release ownership on Ixia ports when the test ends using the **Deconfigure ports and release ownership after the test ends** option.

When saving IxChariot tests to an older format, this option is removed.

When loading an IxChariot test created using an older version, the option to deconfigure ports is set to the value configured in the User Settings menu.

Refer to *What's New in IxChariot 7.10 SP5* on page 1-3 for a summary of the new features in this release.

RUNTST and FMTTST Programs

These command-line programs can read test files produced by all versions beginning with IxChariot 4.3 (versions older than 4.3 are not supported). RUNTST writes test files in version 6.30 format. However, versions of RUNTST and FMTTST prior to version 6.40 cannot read test files in version 6.40 format.

Related Documentation

The IxChariot documentation suite includes the following manuals:

- *Getting Started with IxChariot*

  Contains detailed instructions for installing and registering IxChariot. This guide also provides an overview of the IxChariot user interfaces and includes simple examples of test setup and execution.

- *IxChariot User Guide* (this guide)

  Contains comprehensive guidance for using IxChariot, including test definition, test execution, and test results analysis.

- *IxChariot Runtime User Guide*

  Describes the IxChariot features that are included in the IxChariot Runtime product.

- *IxChariot Performance Endpoints*

  Provides step-by-step guidance for installing and configuring Performance Endpoints for different operating systems.

- *IxChariot Scripts and Streams Library Reference*

  Provides a detailed description of each of the IxChariot scripts and the Ixia streams that are provided with IxChariot.

- *IxChariot Scripts Development and Editing Guide*

Provides step-by-step guidance for creating and editing IxChariot scripts.

*   *IxChariot API Reference*

    Provides information about writing C programs or Tcl scripts to the IxChariot application programming interface that use and extend the capabilities of IxChariot.

The IxChariot manuals are provided in hardcopy and PDF format.

IxChariot also provides comprehensive context-sensitive online help.

The following additional manuals are provided for customers who are using Ixia Test Factory and Discovery Server in their testing:

*   *Getting Started with Ixia Test Factory*

*   *Ixia Test Factory API Reference*

*   *Getting Started with Ixia Discovery Server*

The following additional manuals are provided for customers who are using Ixia chassis in their testing:

*   *Stack Manager User Guide*

*   *Stack Manager API Reference*

Technical Support

You can obtain technical support for any Ixia product by contacting Ixia Technical Support by any of the methods mentioned on the inside cover of this manual. Technical support from Ixia's corporate headquarters is available Monday through Friday from 6 a.m. to 6 p.m., Pacific Standard Time (excluding American holidays). Technical support from Ixia's EMEA and India locations is available Monday through Friday, 8 a.m. to 5 p.m. local time (excluding local holidays).

2 *IxChariot Operational Overview*

This chapter provides an overview of the IxChariot test architecture. It is intended to provide test designers and test engineers with an understanding of all the key elements involved in setting up and executing tests.

Topics in this chapter:

IxChariot Test Network Overview

An IxChariot test network requires three essential elements:

- An *IxChariot Console*:

  This is a Windows machine on which you have installed the IxChariot software. The IxChariot software provides all the tools needed to define and execute your tests.

- An *Endpoint 1 Computer* running Performance Endpoint software:

  The Endpoint 1 computer communicates directly with the IxChariot Console computer and the Endpoint 2 computer. It receives test scripts and data from the IxChariot Console, coordinates the test actions with the Endpoint 2 computer, executes its test instructions, and returns all the test results to the IxChariot Console.

- An *Endpoint 2 Computer* running Performance Endpoint software:

  The Endpoint 2 computer typically communicates only with the Endpoint 1 computer. It receives test scripts and data from, and returns the test results to, the Endpoint 1 computer.

Figure 2-1 shows the basic IxChariot test network components. This figure shows a single pair of endpoints. You can, however, define tests with hundreds or thousands of endpoints.

Figure 2-1.   IxChariot Basic Test Process
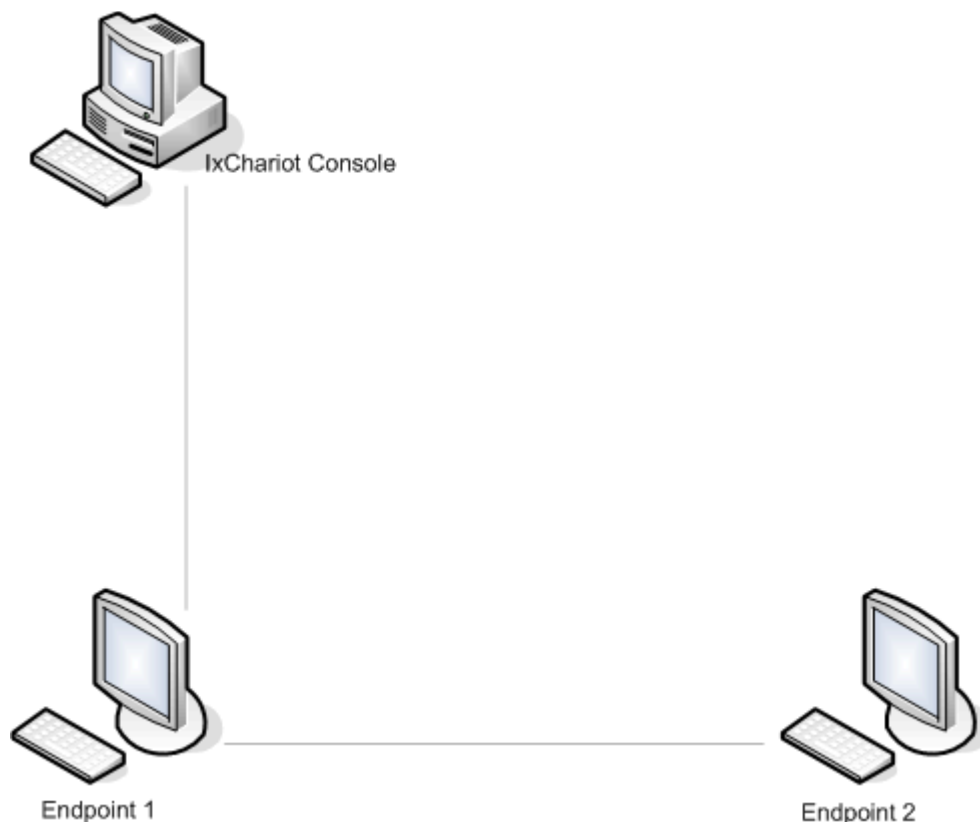
IxChariot Network Topology

An IxChariot test environment supports two types of traffic:

*   Test setup and results: This is the *management* traffic transmitted between the IxChariot Console and the Endpoint 1 computer.

*   Application data: This is the *test* traffic transmitted between the two endpoint computers during test execution.

If the IxChariot Console and the endpoint computers all reside in a single network, both management and test traffic will travel over the same network, which creates additional load on the network elements being tested. Therefore, it is more common for an IxChariot test network to employ two distinct IP networks:

*   Management network: The network over which the Console and Endpoint 1 send their traffic.

*   Test network: The network over which the Endpoint computers execute the tests.

A test network topology using two IP networks is illustrated in Figure 2-2.

Figure 2-2.   Two Network Test Environment
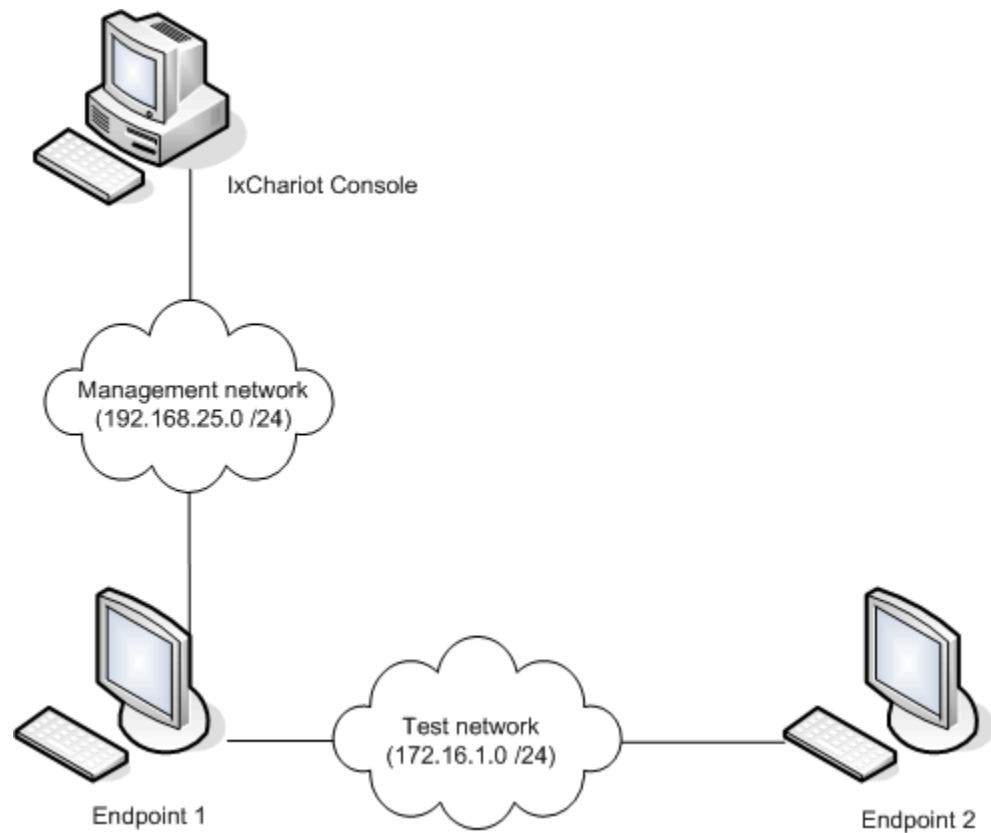
Using Ixia Ports as Endpoints

Rather than using workstations or servers as the endpoint computers in a test, you can use ports on an Ixia chassis as the endpoints. A single endpoint can emulate one or more computers. Figure 2-3 illustrates a two-network test environment in which an Ixia chassis provides the endpoints.

Figure 2-3.   Using Ixia Ports as Endpoints
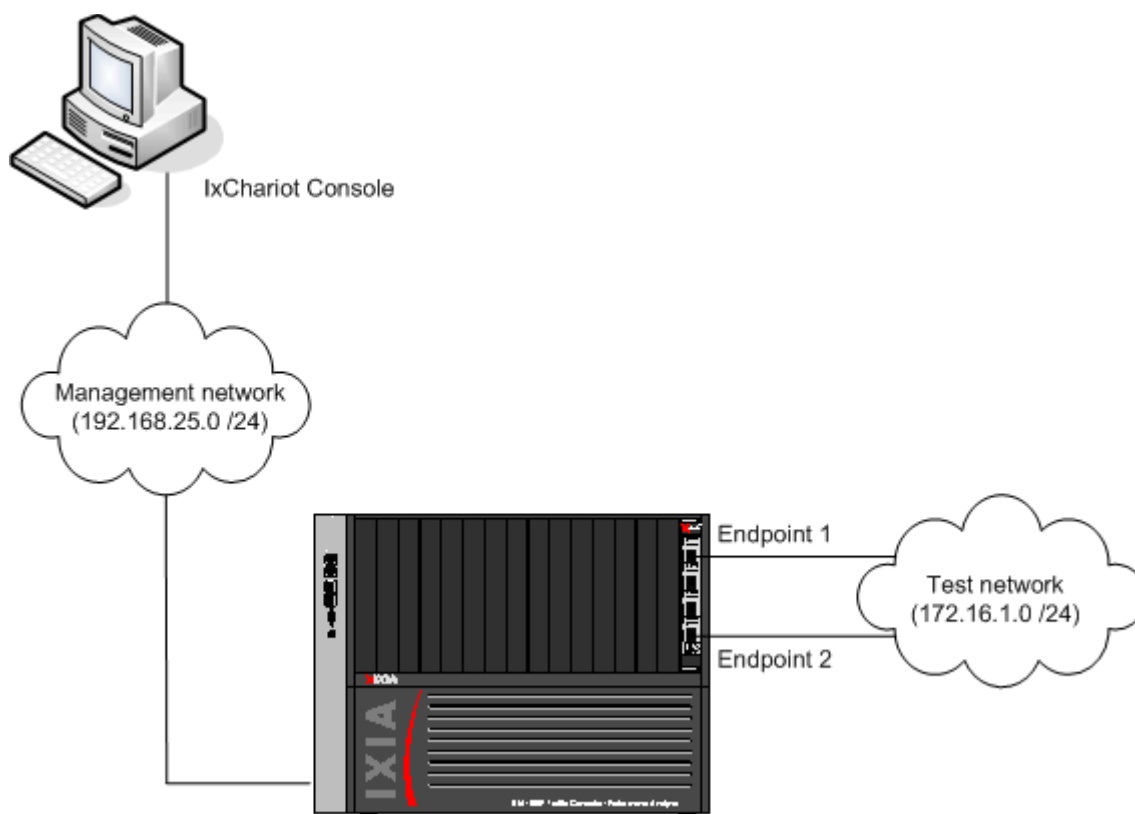
Transport Protocols Used in an IxChariot Test Network

The management network requires a stateful protocol: TCP or SPX. The test network, on the other hand, can use stateful or stateless protocols. This is summarized in Table 2-1.

Table 2-1.   Transport Protocols Used in IxChariot Tests

Network	Transport Protocols
Management	TCP, SPX
Test	TCP, TCP-IPv6, UDP, UDP-IPv6, RTP, RTP-IPv6, IPX/SPX.

IxChariot Test Process Overview

This section presents an overview of the major steps required to setup and execute an IxChariot test.

Major Steps in Running a Test

The basic process that IxChariot uses to run a test is shown in Figure 2-4 and described as follows:

1. You create a test on the IxChariot Console.

2. You initiate execution of the test.

    At this point, the Console establishes communications with, and sends the test setup information to, the Endpoint 1 computer.

3. The Endpoint 1 computer establishes communications with, and sends test setup information to, the Endpoint 2 computer.

    When Endpoint 2 has acknowledged it is ready, Endpoint 1 replies to the console. When all endpoint pairs are ready (in Figure 2-4, there is only one pair), the Console directs them all to start.

4. The two endpoint computers execute the test.

5. The Endpoint 1 computer collects the test results (timing records) and sends them to the IxChariot Console.

Figure 2-4.   IxChariot Test Process
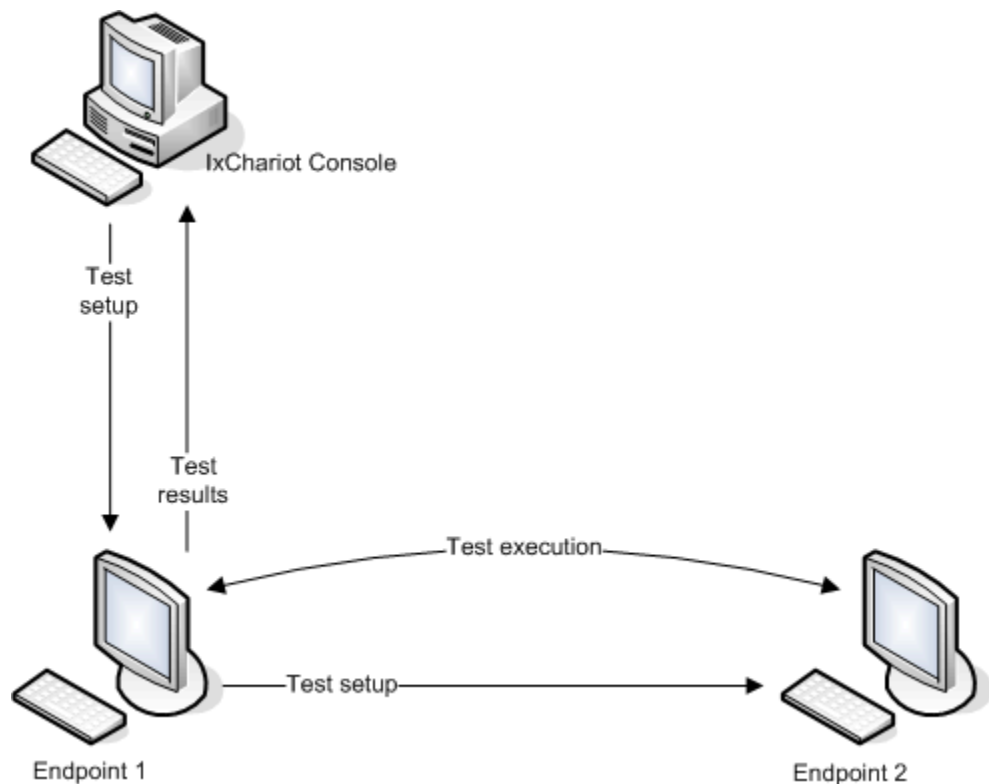
Stages of a Test Run

When you initiate execution of an test, IxChariot works through a staged series of actions to initialize and execute the test. Table 2-2 describes these stages.

Table 2-2.   Stages of a Test Run

Stage	Description
Resolving names	The Console is determining the actual network addresses.
Initializing	The Console is contacting each Endpoint 1 and sending each of them the test script.
Initialized	An endpoint pair has finished Initializing and has reported back to the Console.
n/a	The test has either not started or has completed but does not have enough information to return data.
Running	The scripts are running between the endpoints.
Polling	The Console is polling the endpoints.
Requested stop	The test is over; the Console has sent a request to each pair to stop.
Stopping	Stopping can occur under three conditions: • An endpoint pair has completed its script, and the Console is stopping all the remaining pairs. • The Console user has issued a Stop command • An error occurred on one of the pairs.
Finished	The run has completed.

IxChariot Port Numbers

IxChariot uses a designated *management port* to transport management traffic between the console and the endpoints. The management port is one of the following:

• SPX transport: port 10117

• TCP transport: either port 10115 (the default) or a user-selected port. (For information about designating a management port, refer to *Management ports* on page 6-33 and the endpoint.ini information in the *IxChariot Performance Endpoints* guide.)

During test initialization, Endpoint 1 randomly picks an available port and transmits the test setup information to the management port on Endpoint 2. Endpoint 2 then sends an acknowledgment from the management port to the port from which Endpoint 1 transmitted the setup information.

IxChariot scripts designate the ports that the endpoints use to execute the test. You can edit the scripts to configure these port addresses. For each endpoint, you can either set the source and destination ports to specific port numbers or you can set the ports to the AUTO keyword (AUTO is the default). When set to AUTO, the endpoints assign the port number automatically. Setting the port numbers to AUTO gives the best performance and is the preferred choice when testing with

multiple pairs. (Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about the Script Editor.)

What Is an IxChariot Test?

You define an IxChariot test using the IxChariot Console. When you save your definition, IxChariot stores the test in a single file with a .tst file extension. An IxChariot *test* is a compound document that includes the following components:

- Layer 3 addresses for all the endpoint computers that will take part in a test.

- The test network protocol.

- QoS settings (if specified).

- The management network address and protocol.

- If you are using Ixia ports, the layer 2 addresses for all the endpoint computers that will take part in a test. This may include MAC addresses, VLAN tags, VCIs and VPIs.

- Identification of the endpoint pair types used in the test (see *IxChariot Endpoint Pair Types* on page 2-7).

- One or more IxChariot application scripts or stream files (see *IxChariot Scripts and Streams* on page 2-8).

- Optional payload files.

- Test run options, including: how to end a test, how to handle failures, how to report results, and so forth.

- Test results.

IxChariot Endpoint Pair Types

IxChariot provides several types of endpoint pairs, each of which allows you to model specific types of network traffic.

Table 2-3.   Endpoiint Pair Types

Endpoint Pair Type	Description
Regular Pair	A regular pair emulates a pair of endpoint computers that are transmitting application data (versus VoIP or video traffic).
Hardware Performance Pair	A hardware performance pair utilizes a pair of Ixia test ports as endpoints, and uses a stream file to generate background traffic that is sent from endpoint 1 to endpoint 2.
Multicast Group	Multicast group members are the Endpoint 2 computers designated as receivers of data from Endpoint 1 in IxChariot tests.
VoIP Pair	A VoIP pair emulates a pair of endpoint computers that are transmitting voice traffic using one of the available codec types (such as G7.11u).

Table 2-3.   Endpoiint Pair Types (Continued)

Endpoint Pair Type	Description
VoIP Hardware Performance Pair	A VoIP hardware performance pair utilizes a pair of Ixia test ports as endpoints, and uses a stream file to generate background VoIP traffic that is sent from endpoint 1 to endpoint 2.
Video Pair	A video pair emulates a pair of endpoint computers that are streaming video data.
Multicast Video Group	Multicast video group members are the Endpoint 2 computers designated as receivers of the video streams from Endpoint 1 in IxChariot tests.
IPTV Receiver Group	An IPTV receiver group represents an individual subscriber to one or more IPTV channels. In an IxChariot test, Endpoint 2 (the subscriber) receives channel video streams that are multicast from Endpoint 1. A receiver group includes one test pair for each channel to which that receiver has subscribed.

IxChariot Scripts and Streams

An IxChariot tests will use one of two methods to generate network traffic:

- Script-generated traffic, as described in *Scripts* on page 2-8.

- Stream-generated traffic, as described in *Streams* on page 2-9.

(Refer to the *IxChariot Scripts Development and Editing Guide* and the *IxChariot Scripts and Streams Library Reference* for detailed information about IxChariot scripts and Ixia streams.)

Scripts

IxChariot application scripts generate network traffic that emulates traffic patterns typical of a particular type of application. For example, IxChariot provides several scripts that emulate traffic generated by Lotus Notes. These scripts emulate the traffic patters of activities such as performing an indexed database lookup and retrieving email messages.

There are two categories of scripts:

- Streaming scripts: Streaming scripts emulate the traffic generated by network applications such as RealAudio and NetShow. In these scripts, Endpoint 1 sends a continuous stream of data to one or more receiving endpoints (Endpoint 2 computers).

- Non-streaming scripts: Non-streaming scripts emulate the traffic generated by applications that rely upon a two-way communication between the endpoint computers. For example, a database query application requires a series of requests and acknowledgements to complete a transaction.

Every IxChariot script contains a set of instructions for each of the endpoints (Endpoint 1 and Endpoint 2). Figure 2-5 shows an example of the Throughput

script. The statements on the left apply to Endpoint 1, while those on the right apply to Endpoint 2.

Figure 2-5.   Sample IxChariot Script
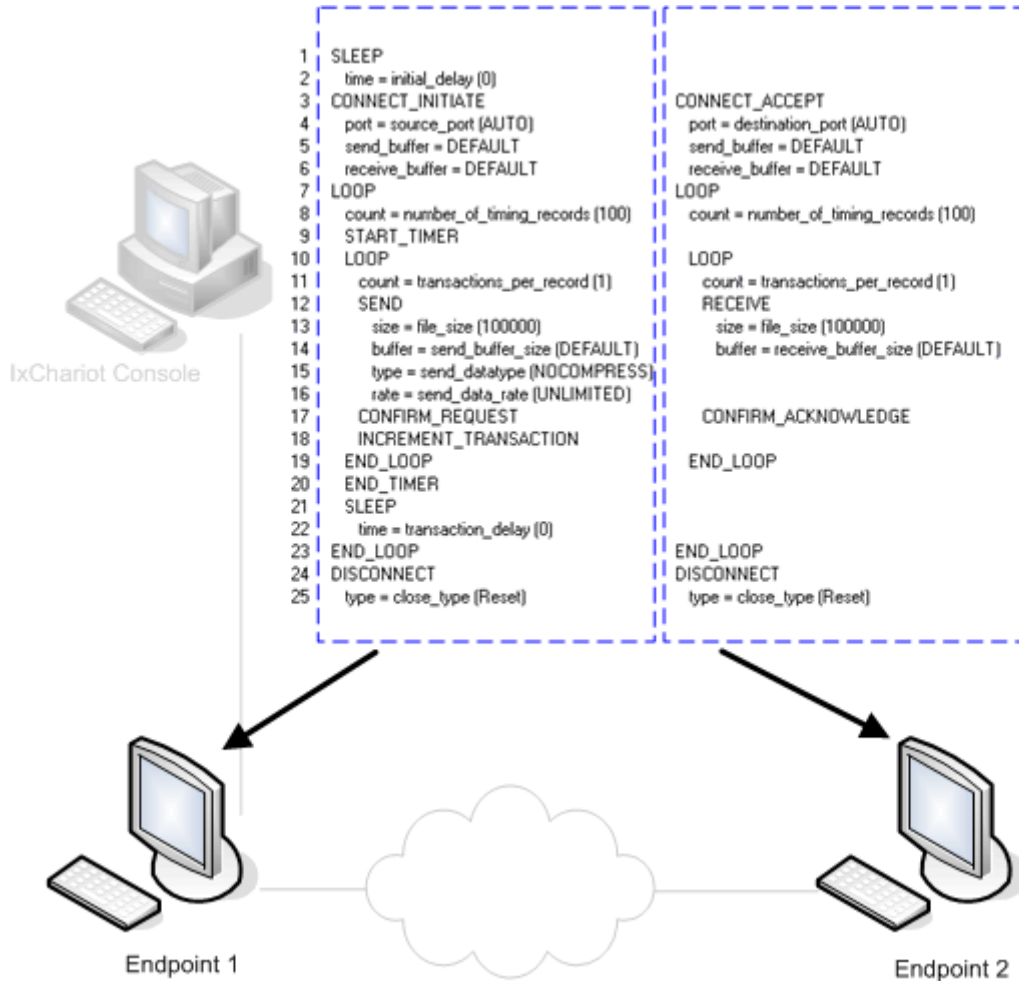
Streams

Ixia streams are used only when Ixia chassis ports serve as endpoints in a test (as shown in Figure 2-3 on page 2-4).

Ixia Streams are supported by most Ixia load modules. The load module uses one or more field programmable gate arrays (FPGAs) to create layer 2 and layer 3 network test traffic entirely in hardware, allowing tests to saturate network interfaces at up to wire speed.
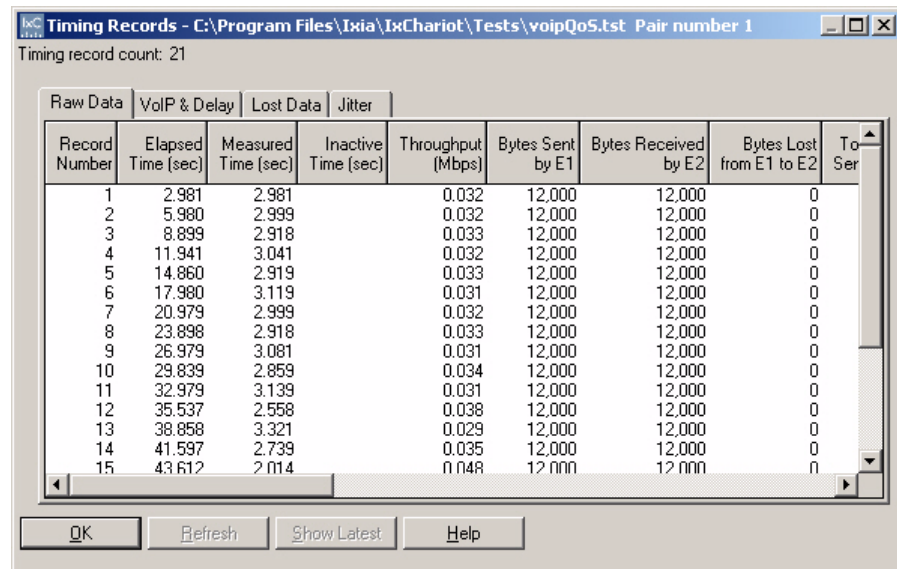
Ixia streams can generate various types of traffic. For example, the "Syn Flood" stream generates traffic for use in Denial of Service testing.

> **Note:** Both the port CPU and the FPGAs can generate traffic on an Ixia port. When both generate traffic at the same time, the Ixia ports do not use round robin for traffic arbitration. Instead, the Ixia ports use an arbitration method that always guarantees that the port CPU traffic has higher priority than stream engine traffic. Based on the burst traffic from the port CPU, the worst-case scenario for the stream engine is when the port CPU sends 1518-byte frames at the maximum throughput without receiving any packets, while the stream engine is trying to send 64 bytes frames at line rate. The bandwidth ratio for port CPU traffic and stream engine traffic is approximately 80/20.

Timing Records

While a test executes on one or more pair of endpoints, IxChariot collects test results in the form of timing records. Figure 2-6 shows an example of some of the timing records collected during a VoIP test.

Figure 2-6.   Timing Records Example

Data Contained in Timing Records

Table 2-4 provides a brief summary of the types of data contained in the timing records.

Table 2-4.   Summary of Test Data

Script Type	Timing Records Contain:
Streaming	• Elapsed, Measured, and Inactive Time values in seconds. • Response time. • Number of bytes and datagrams sent, received, and lost. • Number of datagrams received out-of-order. • Jitter statistics, delay, and other data specific to VoIP or video traffic.
Non-Streaming	• Number of timing records completed. • Transaction count. • Throughput (average, minimum, maximum). (IxChariot measures the throughput associated with packet payload, ignoring headers). • Transaction rate (average, minimum, maximum). • Response time (average, minimum, maximum). • Number of bytes sent and received. • Elapsed Time, Measured Time, and Inactive Time values in seconds. • 95% Confidence Interval. • Measured time. • Relative Precision.

Hardware Performance Endpoints and VoIP Hardware Performance Endpoints provide additional real-time statistics, including bytes received and sequence errors.

Refer to the *IxChariot User Guide* for a detailed description of the data collected in timing records.

Timing Record Collection

Every non-streaming script contains two loops, as shown in Figure 2-7 on page 2-12:

• The outer loop controls the number of timing records generated. For example, the first statement in the outer loop in the script shown in Figure 2-7 contains this statement:

```
LOOP
 count = number_of_timing_records (100)
```

For this script, the outer loop will execute 100 times, generating one record during each pass through the loop.

Specifically, a timing record is written each time the END_TIMER statement is executed. (The END_TIMER statement is on row 20 in Figure 2-7.)

- The inner loop controls the number of iterations for a transaction in a script. For example, the first statement in the inner loop in the script shown in Figure 2-7 contains this statement:

  ```
  count = transactions_per_record (1)
  ```

  For this script, each timing record will contain data for a single transaction. If you edit this script to change the transaction count from 1 to 10, you will still collect 100 timing records, but the volume of data will increase tenfold.

Figure 2-7.   Loops in an IxChariot Script.
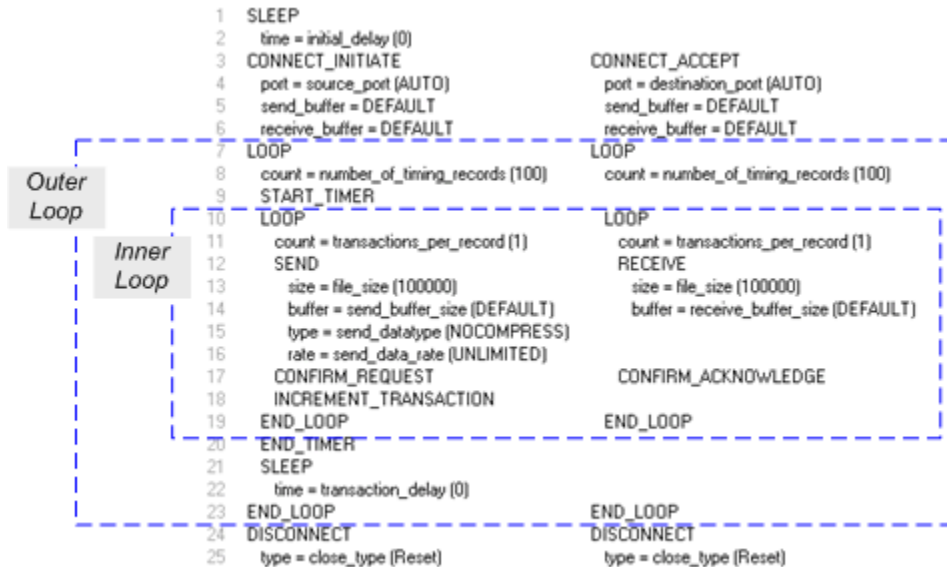
Running a Test for a Fixed Duration

When you configure a test, you determine when the test run will end. Your choices are:

- The test will stop when any one pair completes its execution.

- The test will stop only when every pair has completed its execution.

- The test will run for a fixed duration.

In the first two cases, a pair completes its execution after *number_of_timing_records* iterations of the outer loop (see row 8 in Figure 2-7).

In contrast, when you configure a test to run for a fixed duration, every endpoint pair in the test runs for the amount of time that you specify. In this case, the scripts ignore the `number_of_timing_records` value in their outer loop. Instead, IxChariot runs as many transactions during that time as it can. At the end of the test period, the endpoints stop and each Endpoint 1 returns its results to the IxChariot Console.

When a test is set to run for a fixed duration, the run time duration is checked every time an END_LOOP or END_TIMER command executes. This may cause the actual run time to slightly exceed the run time that you specified.

How Long Can a Test Run?

You can set a test to run for any period of time within the minimum and maximum durations:

- Minimum test duration: 1 second.

- Maximum test duration: 99 hours, 59 minutes, and 59 seconds.

Although the default setting is 1 minute, Ixia recommends 2 to 5 minutes for most performance testing. It is important to test for a sufficient length of time to generate at least 10 timing records. The records represent data samples, and you will need a sufficient number of samples to ensure test reliability.

For More Information

Refer to the *IxChariot Scripts Development and Editing Guide* for more information about configuring tests to run for a fixed duration.

3

Getting Started with the IxChariot Console User Interface

This chapter provides an description of the major features and functions of the IxChariot Console GUI.

Topics in this chapter:

- *Starting the IxChariot Console* on page 3-1
- *Test Window Overview* on page 3-2
- *Test Window Menus* on page 3-4
- *Test Window Status Bar* on page 3-14
- *Test Window Shortcut Keys* on page 3-15
- *Toolbar Icons* on page 3-16

Starting the IxChariot Console

Related Topics
Using Test Scheduler on page 5-37
Command-Line Programs on page 5-67

You can start the IxChariot Console using the Windows Start menu (**Start** > **Programs** > **IxChariot** > **IxChariot**), or by clicking the icon on the desktop:

When IxChariot Console starts running, it briefly displays a splash screen, then opens the Test window.

Test Window Overview

The IxChariot Console Test window lets you build, run, and view the results of a test. You can add new endpoint pairs, copy or delete existing endpoint pairs, edit endpoint pairs, or view the statistics and graphs of a test you've run. Figure 3-1 shows an example of the Test window.

Figure 3-1.   IxChariot Console Test Window
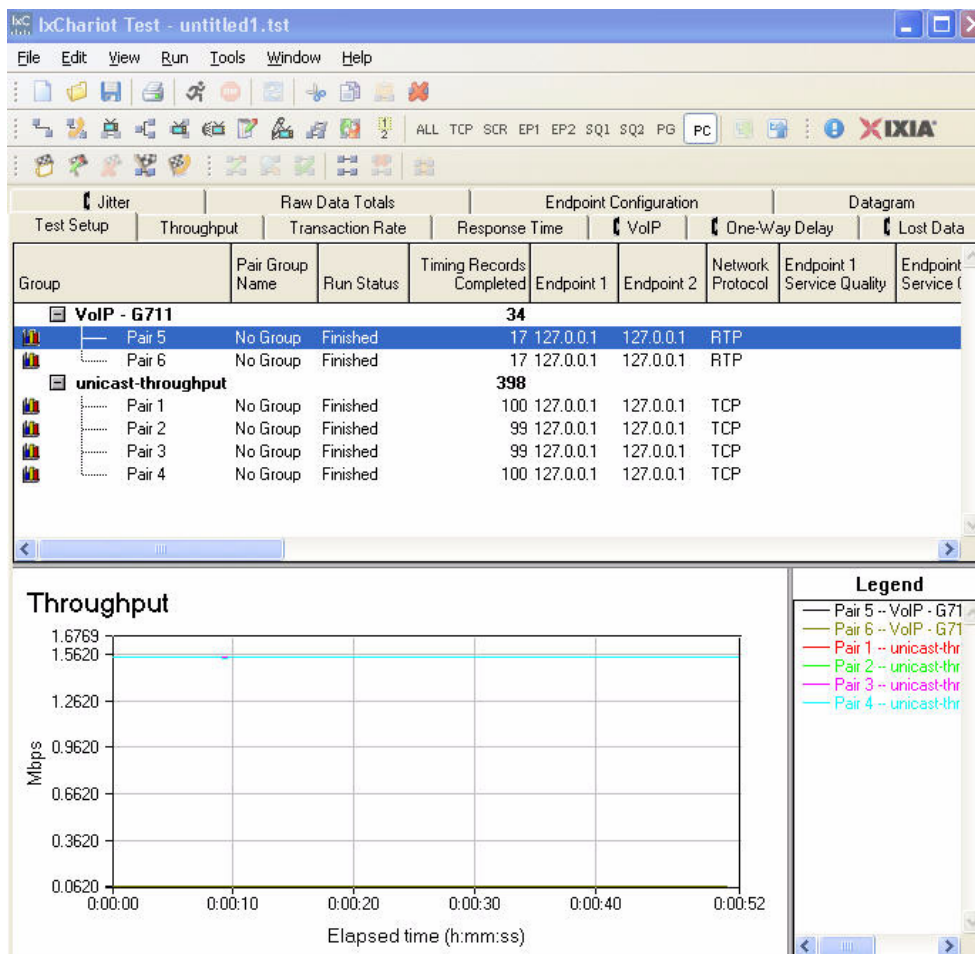
Each endpoint pair in a test is shown as a row in the Test window. The pairs are identified by a pair number, shown on the left-hand side of each row. You can edit a pair by double-clicking it, or by selecting it and using the menus or toolbar icons. The graph icon indicates that a pair is selected for graphing.

The Test window is partitioned into areas accessible by tabs. The first tab is for test setup; other tabs let you view the results of a test. You can save a test to a file, or export it in a variety of file formats.

You can have as many as nine Test windows open at one time. To open a new Test window, select **New** from the **File** menu. To open a test file that was previously saved to disk, select **Open** from the **File** menu, then select the test.

Floating Menus

Right-click to bring up a floating Edit menu when you are pointing to any end-point pair in a test window. Figure 3-2 shows an example.

Figure 3-2.   Floating Menu for Test Pairs
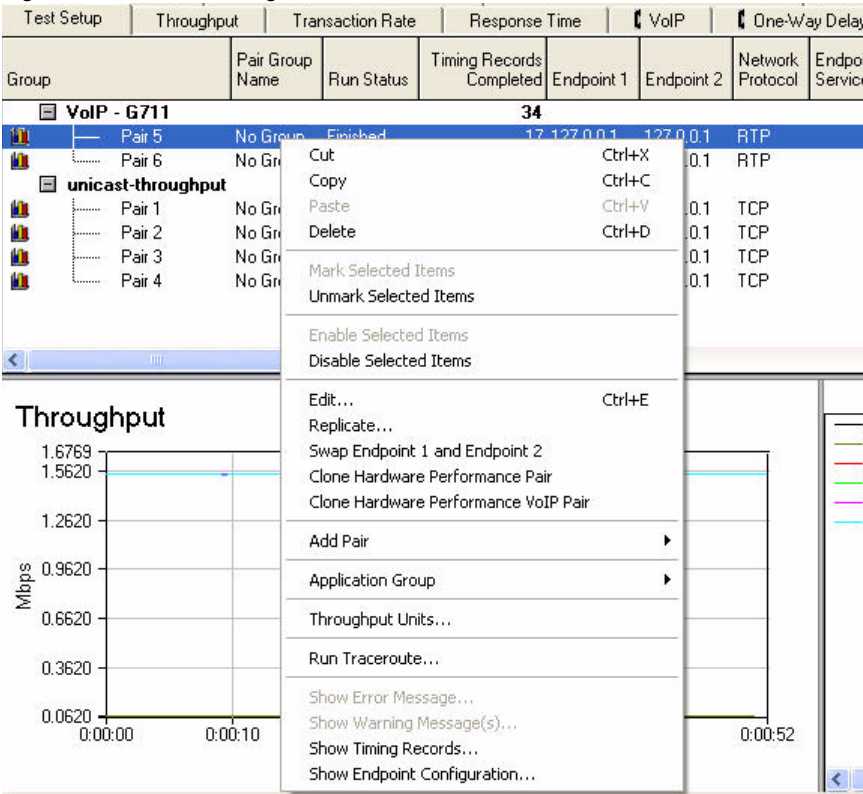
While in the graph region, right-click to bring up a menu that lets you change Graph Configuration or Throughput Units. Figure 3-3 shows an example.

Figure 3-3.   Floating Menu for Test Graph
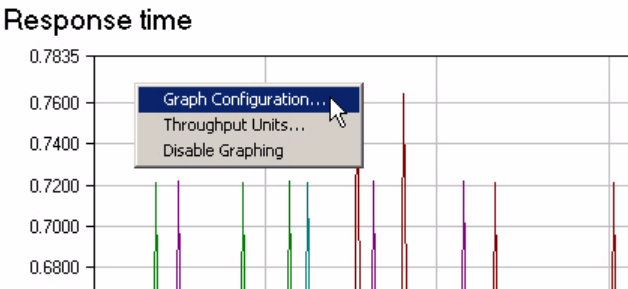
For More Information

Each of the menus in the IxChariot Test window is described in *Test Window Menus* on page 3-4. The status bar is described in *Test Window Status Bar* on page 3-14. The toolbar icons offer shortcuts to commonly-performed operations. Refer to *Toolbar Icons* on page 3-16 and *Test Window Shortcut Keys* on page 3-15 for more information. The button featuring the Ixia logo is a hyperlink to the Ixia Web site.

Each Test Window tab is discussed in *Understanding Results* on page 11-1. The **Test Setup** tab is always available; the other tabs are only available when a test has results. The **Throughput**, **Transaction Rate**, **Response Time**, **Lost Data**, and **Jitter** tabs display a graph of the appropriate results. Some tabs and sub-tabs may not be shown unless certain protocols are used in testing.

Test Window Menus

All IxChariot Console Test window functions are available through the following menus:

- *The File Menu* on page 3-4
- *The Edit Menu* on page 3-5
- *The View Menu* on page 3-9
- *The Run Menu* on page 3-10
- *The Tools Menu* on page 3-10
- *The Window Menu* on page 3-12
- *The Help Menu* on page 3-13

The File Menu

When you save a test, the test setup and results, if any, are stored together in a binary file. By default, IxChariot uses the file extension `.tst` for this file. For each endpoint pair in the test, IxChariot stores the names of the endpoints, their protocol and service quality, and the full script and script variables used by that pair. If results are available, IxChariot saves in the binary file all the timing records associated with the most recent test run.

- **New**

  Opens a new IxChariot Test window.

- **Open**

  Opens another IxChariot Test window.

- **Save**

  Saves a new test or saves a test (`.tst`) file using its existing filename.

  For additional information about saving a test that uses large payload files, Refer to the *IxChariot Scripts Development and Editing Guide*.

- **Save as**

  Saves a test (`.tst`) file with a new name, for the current or an earlier version of IxChariot. If you specify an earlier version, any features not supported in that version will be removed or modified for compatibility. The Save As Type drop-down menu lists the IxChariot versions that you can select. If the test

only contains pairs whose functions are not supported in the IxChariot ver-
sion in which you are saving the test, you cannot save the test in the specified
level.

- **Export**

  Exports test setup details and results to `HTML`, `.TXT`, or `.CSV` file format. If
  you've entered preferences on the Output tab of the Change User Settings
  notebook, these are used.

- **Print**

  Prints test setup details and results. If you've entered preferences on the Out-
  put tab of the Change User Settings notebook, these are used.

- **Clear Results**

  Erases the results from a test without affecting the test setup. Only the Test
  Setup tab remains in the Test window after you clear results. You might use
  this command to save only the setup for a test, without a large set of accom-
  panying results that you don't plan to keep.

- **Application Group**

  Imports saved application group files (.iag) into the Test window, and exports
  application groups from the Test window to disk. Refer to *Creating
  Application Group Tests* on page 5-35 for information about application
  groups.

- **Exit**

  Exits the current Test window, leaving any other IxChariot Test windows
  open. If a test is running in the window, you are asked if you want to stop it. If
  the test hasn't been saved to a file, you are asked if you want to save it.

The Edit Menu

The items in the Edit menu let you change the setup for a test. In addition to edit-
ing, deleting, and copying individual endpoint pairs, you can select multiple pairs
or multicast groups and perform these same operations across all of them. You
can also cut and paste pairs among different test windows.

Right-click to bring up a floating Edit menu when you are pointing to any end-
point pair in a test window. Following are brief descriptions of each item in the
Edit menu.

- **Cut**

  Removes a selected pair or group of pairs from the Test window and places
  them on the Windows clipboard. When selecting multiple pairs, hold down
  the Shift key and click the first and last pairs to be cut. The two pairs you
  clicked, as well as all pairs in between, are selected. If the Test window con-
  tains test results, you are prompted to confirm the cut because a cut operation
  clears all test results. You are given the option to cancel or proceed.

- **Copy**

  Copies a selected pair or group of pairs from the Test window to the Windows
  clipboard. When selecting multiple pairs, hold down the Shift key and click
  the first and last pairs to be copied. The two pairs you clicked, as well as all
  pairs in between, are selected. Copying does not clear test results.

- **Paste**

  Takes a pair or group of pairs from the Windows clipboard and pastes them into the Test window. You can only paste IxChariot data into the Test window. But you can paste data cut or copied from IxChariot into other applications that accept tab-delimited data, such as spreadsheets and editors.

  Before pasting the data into the selected Test window, IxChariot ensures that the Paste operation does not exceed the licensed number of endpoint pairs. If this number will be exceeded by the Paste operation, a dialog box alerts you and the paste operation is aborted.

  If the Test window contains test results, you are prompted to confirm the paste operation because it will clear all test results. You are given the option to cancel the paste operation or proceed.

- **Delete**

  Removes a selected pair or group of pairs from the Test window. To delete multiple pairs, hold down the Shift key and click the first and last rows to be removed.

  When you attempt to delete a pair, a warning asks, "Are you sure you want to delete the selected endpoint pair(s)?" You can disable this warning in the Change User Settings notebook.

- **Select All**

  Selects all of the pairs in the Test window. You must select a pair before you can cut, copy, or mark it.

- **Deselect All**

  Deselects all of the highlighted pairs in the Test window.

- **Mark Selected Items**

  Specifies a pair or pairs for inclusion in a graph or in a printed report. New pairs are initially marked when they're created. This command marks all pairs and groups that are currently selected in the Test window. A column to the left of the pairs displays a graph icon to indicate that pairs and groups are marked.

- **Unmark Selected Items**

  Excludes a pair or group from graphs or printed reports. Unmarks all pairs and groups currently selected in the Test window.

- **Enable Selected Items**

  Lets you enable pairs that you had previously disabled via the Disable Selected Items menu selection. Pairs are enabled by default.

- **Disable Selected Items**

  Lets you temporarily disable one or more pairs in a test. IxChariot ignores any disabled pairs when running a test. Use Enable Selected Items to re-enable the pairs.

- **Edit**

  Lets you modify a highlighted pair, set of pairs, or a multicast group in the Test window. If you highlighted multiple pairs, the Edit Multiple Endpoint Pairs dialog box opens so that you can modify the definitions of many pairs

simultaneously. You cannot edit multiple multicast groups simultaneously, nor can you edit multiple multicast groups or a combination of multicast pairs and single pairs.

- **IPTV Channels Editor**

  Launches the IxChariot IPTV Channels Editor.

- **Replicate**

  Lets you duplicate an existing pair, multicast group, or group of pairs. First highlight the pair(s) to be replicated. You must replicate pairs and multicast groups separately. In the Replicate Selected Pairs or Replicate a Multicast Group dialog box, enter the number of replications desired.

- **Swap Endpoint 1 and Endpoint 2**

  Changes the roles of the endpoints in the pair(s). First highlight the pair(s) of endpoints to be swapped. Swapping is not allowed if any of the selected pairs has a different pair setup address defined, or if a multicast pair is selected. See *Cloning Hardware Performance Pairs* on page 5-25 for more information.

- **Clone Hardware Performance Pair**

  Lets you create a hardware performance pair from a non hardware performance pair. This uses the addresses from the non hardware performance pair in the creation of a new hardware performance pair. See *Cloning Hardware Performance Pairs* on page 5-25 for more information.

- **Clone Hardware Performance VoIP Pair**

  Lets you create a hardware performance VoIP pair from a non hardware performance pair. This uses the addresses of a non hardware performance pair in order to create a hardware performance VoIP pair. See *Cloning VoIP Hardware Performance Pairs* on page 10-43 for more information.

- **Select Ixia Network Configuration**

  Lets you select a previously-defined Ixia Network Configuration for use in a test.

- **New/Edit Ixia Network Configuration**

  Invokes the Ixia Stack Manager application with which you define a new—or edit an existing—Ixia Network Configuration. An Ixia Network Configurations encompasses the configuration of Ixia ports, the selection of a protocol stack, and the possible selection of additional global services (such as DNS or Impairment). See *Using Stack Manager to Configure and Assign Ixia Ports* on page 4-8 for detailed information.

- **Clear Ixia Network Configuration**

  Lets you clear an Ixia Network Configuration that you had previously selected for use in a test.

- **Add Pair**

  Presents a submenu with the following options:

  - Add Pair

    Lets you add a pair of endpoints to an IxChariot test. See *Adding or Editing a Multicast Group* on page 5-27 for more information.

- • Add Multicast Group

  Lets you create a group of receivers for an IP Multicast test. See *Adding or Editing a Multicast Group* on page 5-27 for more information.

- • Add Video Multicast Group

  Lets you create a group of receivers for a Video Multicast test. See *Adding or Editing a Video Multicast Group* on page 10-58.

- • Add IPTV Receiver Group

  Lets you create receivers for an IPTV multicast test. See *IPTV Testing* on page 10-66.

- • Add VoIP Pair

  Lets you add a VoIP endpoint pair to an IxChariot test. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information.

- • Add Video Pair

  Lets you add a Video endpoint pair to an IxChariot test. See *Adding or Editing a Video Endpoint Pair* on page 10-54.

- • Add Hardware Performance Pair

  Lets you add a hardware performance endpoint pair to an IxChariot test. See *Adding or Editing a Hardware Performance Pair* on page 5-23 for more information.

- • Add VoIP Hardware Performance Pair

  Lets you add a VoIP hardware performance endpoint pair to an IxChariot test. See *Adding or Editing a VoIP Hardware Performance Pair* on page 10-42.

- • **Renumber All Pairs**

  Renumbers all the pairs in the Test window sequentially, beginning with 1. After you have edited or deleted pairs within a test window, pairs aren't automatically renumbered.

- • **Application Group**

  Presents a submenu with the following options:

  - • Paste Application Group

  - • Delete Application Group

  - • Edit Application Group

  - • Search and Replace Addresses

  - • Add to Application Group

  - • Remove from Application Group

  - • Validate Application Group

  Each of the Application Group submenu options is described in detail in the *IxChariot Scripts Development and Editing Guide*.

The View Menu

The View menu lets you alter the way information is displayed in the Test and Comparison windows, set graph display options, display error messages, display timing records, and display the configuration of an endpoint. Following are brief descriptions of each item in the Edit menu.

- **Sort...**

  Opens the Sort dialog box, where you specify the first, second, and third criteria by which the pairs will be sorted. For each of the three criteria, you can select ascending or descending order.

- **Group Sort Order**

  When you have endpoint pairs organized in groups, this option lets you sort the groups of pairs in either ascending or descending order.

- **Group By**

  The Group By feature lets you organize the tests into groups, based on a selected criteria (such as grouping by Pair Comment). Once pairs are grouped, their results appear in these groups when you print or export. By default, the no-grouping category "All Pairs" is used.

- **Information**

  Lets you choose which aspect of a test's results are graphed and shown in the Test window. If a test is still being configured, only the Test Setup tab is available. While a test is running, or after a run, all tabs that contain results are shown. If the test does not contain RTP pairs, for example, the Jitter tab is not shown.

- **Expand All Groups**

  Lets you view details about all pairs in a test. Test setup and results are displayed for all of the pairs.

- **Collapse All Groups**

  Decreases the level of detail shown in the Test window; details for individual endpoint pairs are not shown. Instead, endpoint pairs are subsumed within groups. With large tests, it is a good idea to collapse groups before you try to graph their results.

- **Disable Graphing**

  Shows test results without graphing them. This option may greatly improve GUI performance if you are running large tests, which can be tricky to graph. Automatically selected when you run a test of more than 5,000 endpoint pairs.

- **Graph Configuration**

  Lets you control how test data is graphed. See *Graph Configuration* on page 11-9 for more information.

- **Throughput Units**

  Lets you change the units in which data is displayed. See *Throughput Units Tab* on page 6-30 for information about default units.

- **Show Error Message**

  Displays any error message associated with the selected pair. See *Show Error Message* on page 12-6 for more information.

- **Show Warning Message(s)**

  Displays any warning message associated with the selected pair. See *Show Warning Messages* on page 12-8 for more information.

- **Show Timing Records**

  For pairs that have results, provides a breakdown of results by timing record. See *Examining Timing Records* on page 11-3 for more information.

- **Show Endpoint Configuration**

  Provides information about the endpoint computers. See *Endpoint Configuration Details Dialog Box* on page 11-17 for more information.

The Run Menu

The Run menu lets you run tests and traceroutes and modify the way IxChariot tests are performed.

- **Run**

  Starts a test that you've created.

- **Stop**

  Stops a running test before it completes.

  When you stop a running test, a warning box appears asking, "A test is currently running. Do you want to stop the test?" Click **Yes** to stop the test or click **No** to resume the running of the test. See *Stopping a Running Test* on page 5-31 for more information on stopping a test.

- **Set Run Options**

  Sets parameters determining how one test is run. A two-page notebook is shown, with a Run Options page and a Datagram page. Set run options for all tests by clicking the Run Options tab in the Change User Settings notebook. See *Test Run Options* on page 7-1.

- **Poll Endpoints Now**

  Causes the Console to contact each of the Endpoint 1 computers in a test while a test is running. The endpoints reply, returning the number of timing records they've created so far in this test. Poll Endpoints Now corresponds to the Poll icon on the toolbar. See *Polling the Endpoints* on page 6-6 for information about why you'd choose to poll during a running test.

The Tools Menu

The Tools menu provides a set of tools for managing and customizing your IxChariot test environment. The following items are available in the Tools menu:

- **Compare Tests**

  Opens the Comparison window, which lets you compare the results of multiple IxChariot tests. Refer to *Comparing Test Results* on page 5-43 for detailed information.

- **Edit Scripts**

  Opens the Script Editor, which lets you modify existing scripts or create a new script. The Script Editor can also be invoked from the Test window, but if you access the Script Editor this way, your modifications are saved at the

file level and are available to all new pairs associated with the script. See *Editing Script Variables* on page 5-57 for more information.

- **Edit Output Templates**

  Lets you specify printer output for reports ahead of time. See *Output Templates* on page 5-65 for more information.

- **Edit IPX/SPX Entries**

  Lets you enter a series of aliases that are stored and used in IxChariot tests instead of those long IPX network addresses. See *IPX/SPX Aliases* on page 5-3 for more information.

- **Edit QoS Templates**

  Lets you run tests with Quality of Service (QoS) settings. IxChariot offers four options for testing with QoS or prioritized traffic. Refer to Chapter 9, *Quality of Service Testing* for detailed information.

- **Run Traceroute**

  Runs a traceroute between a pair of endpoints in a test. See *Running a Traceroute* on page 5-32 for a full discussion.

- **Run Test Scheduler**

  Launches the IxChariot Test Scheduler, which lets you schedule tests for execution and set a recurrence pattern for the scheduled tests. Refer to *Using Test Scheduler* on page 5-37 for detailed information.

- **Run Test Designer**

  Opens the IxChariot Visual Test Designer, a tool for creating tests. For more information, refer to *Visual Test Designer* on page 5-75.

- **Run Test Factory**

  Launches the Ixia Test Factory application. Test Factory works with the Ixia Discovery Server and IxChariot to discover IxChariot endpoints installed within physical and virtualized datacenters, create sophisticated network performance tests of the infrastructure, and analyze the results of those tests. Refer to *Getting Started with Ixia Test Factory* for detailed information.

- **View Error Logs**

  Shows you more information on errors you receive while running tests. Provides an organized list of errors and information about them. See *The Error Log Viewer* on page 12-8 for more information.

- **Options**

  Selecting **Tools > Options** provides two selections: User Settings and Display Fonts. These are described in *Changing Global User Settings and Display Fonts* on page 3-12.

Changing Global User Settings and Display Fonts

Related Topics
IxChariot User Settings on page 6-1
Datagram Tab on page 6-16

Select **Tools > Options > User Settings...** to open a tabbed notebook of global IxChariot settings you can change. IxChariot's user settings affect the following:

- Default parameters used in all new tests you create or run:
    – Endpoint Pair Defaults tab
    – VoIP Pair Defaults tab
    – HPP Defaults tab
    – VoIP HPP Defaults tab
    – Video Pair Defaults tab
    – IPTV Defaults tab
    – Run Options tab
    – Datagram tab
    – Traceroute tab
- Default units in which test data is recorded and displayed (Throughput Units tab)
- Warnings you might see while setting up tests or viewing results (Warnings tab)
- Directories where IxChariot stores your files (Directories tab)
- Options for printing or exporting test results (Output tab)
- Ranges into which data is placed when results are reported (Result Ranges tab)
- Parameters affecting how IxChariot runs tests through firewalls in your network (Firewall Options tab).
- Options for using Ixia ports as endpoints (Ixia Port Configuration tab)
- Options for working with IxChariot application groups (Applications Groups tab)

Click **Help** for context-sensitive help about each tabbed notebook page in the Change User Settings notebook.

Select **Tools > Options > Display Fonts...** to change the fonts used in the Test and Comparison windows. The settings you choose here do not affect the Main window, nor do they change the fonts used in exported or printed test results.

The Window Menu

The Window menu presents a list of open IxChariot Test windows, allowing you to easily navigate among the open windows. It also lists the IxChariot Comparison window, if you have opened a comparison.

You can open as many as nine Test windows simultaneously. Each open Test window is shown on the Window menu. Click the name of a test to bring the Test window to the foreground.

The Help Menu

The Help menu gives you instant access to the IxChariot online help, provides access to the Ixia Registration Utility and to the Ixia web site, while also including information about your installer version of the product. It includes the following menu selections:

- Select **Contents and Index** to access the IxChariot online help for the IxChariot Console, Performance Endpoints, IxChariot API, application scripts, and messages.

- Select **Current Window** to get descriptive information about the IxChariot window you are currently viewing.

- Select **Shortcut Keys** for a list of all shortcut keys and key combinations available for the current window.

- Select **Contact Us** to open the Ixia web site in your web browser.

- Select **Registration** to view the IxChariot release number, the license expiration date, and a list of licensed features.

  Click **Manage License...** to access the Ixia Registration Utility (IRU). A dialog similar to Figure 3-4 on page 3-13 is displayed.

  Figure 3-4.   Ixia Registration Utility Dialog

  For detailed information about the IRU, refer to the Ixia *License Management User Guide*.

- Click **Check for updates at startup** to have IxChariot automatically search for more recent versions of the application as soon as IxChariot starts.

- Click **Check for updates...** to have IxChariot search for more recent versions of the application at the precise moment when this option is selected.

If either of the above two options are selected, the update check will yield one of the following results:

- a warning, notifying that the update check could not be performed.

  In addition, if the version check is performed at startup, the Stop checking for updates at startup option is available.

- an information window opens, notifying that the installed IxChariot version is the latest available

- a dialog window opens, notifying that there is a more recent IxChariot version available and directing you to the download page

- Click **About IxChariot** for details on the IxChariot version and build level, and for information about service and support. The About IxChariot dialog box contains copyright and release information.

  Click **Support Info** in the About IxChariot dialog box for IxChariot technical support information.

Test Window Status Bar

Related Topics
Understanding the Run Status on page 11-7

The Status Bar, at the bottom of the Test window, shows summary information about the progress of a test as it moves from initialization to completion. The status bar can include up to five fields, depending on the test's current state.

Status Bar Fields include the following:

- The number of pairs used in the current test.

- Start status (shown when the test is initializing and running)—gives the overall progress of the test. When the test ends, this field displays the start date and time of the current test;

- End status (only shown while the test is running)—indicates the elapsed time (hr:min:sec). When the test ends, this field shows the ending date and time.

- Duration (shown while the test is running)—estimates the time remaining until the test will complete (hr:min:sec). This field shows no value until enough timing records are received to calculate an estimate. When the test ends, this field shows the actual total run time. You might see the estimated remaining time increase in large increments, if one or more pairs are using random `SLEEP` durations.

- Completion status (displayed when the test ends)—informs you whether the test ran to completion, was stopped by the user, or stopped because an error was detected.

Test Window Shortcut Keys

Related Topics

Shortcut Keys for the Test Designer on page 5-89
Shortcut Keys for the Comparison Window on page 5-47

You can use the following keys and key combinations in any IxChariot test window instead of using the mouse. In addition to these keys, the **Alt** key can be used in combination with any underscored letter to invoke a menu function. The menu function must be visible and not shown in gray. For example, clicking **Alt**+**F** shows the File menu.

Table 3-1.   Shortcut Keys for the Test Window

Key or Key Combination	Command Invoked
F1	Get help for the IxChariot Test window.
F2	See an index of all the available IxChariot help topics.
F5	Poll the endpoints during a running test.
F9	Show the keys and key combinations available in a window.
F11	Open the About IxChariot dialog box, which shows your version and build level, and gives you product support information.
Ctrl+A	Select all the pairs in a test.
Ctrl+C	Copy the test setup for one or more pairs to the clipboard.
Ctrl+D	Delete the selected pairs. .
Ctrl+E	Edit the highlighted endpoint pair(s).
Ctrl+G	Add a new multicast group.
Ctrl+I	Add a Video pair.
Ctrl+J	Add a new VoIP Hardware Performance Pair.
Ctrl+N	Open a new test window.
Ctrl+O	Open a test previously saved to disk.
Ctrl+P	Add a new endpoint pair to a test. If you are working with a test that already has results, IxChariot asks whether you want to discard your existing results.
Ctrl+R	Run this test. Only one test can be run at a time, to avoid conflicting performance data.
Ctrl+S	Save this test setup and its results to an IxChariot test file.
Ctrl+T	Stop a running test.

Table 3-1.   Shortcut Keys for the Test Window (Continued)

Key or Key Combination	Command Invoked (Continued)
Ctrl+V	Paste the test setup for one or more pairs from the Windows clipboard.
Ctrl+W	Add a VoIP pair.
Ctrl+X	Cut the test setup for one or more pairs and place it on the Windows clipboard.
Alt+F4	Close any window or dialog box. Has the same effect as clicking the **Esc** key or clicking **Cancel** with the mouse.

Toolbar Icons

The icons on the toolbar in IxChariot's Test window, where you'll perform most test configuration, correspond to the most commonly performed tasks. Each task is also represented in one of the Test window's menus.

Table 3-2 lists the buttons that you use when creating, managing, and running tests.

Table 3-2.   Toolbar Buttons for Creating and Running Tests

Icon	Tasks for Creating and Running Tests
	Create a new test.
	Open a test.
	Run a test.
	Stop the test that is currently running.
	Poll the endpoints during a running test.
	Delete a pair from a test.
	Add an endpoint pair to a test.
	Add a VoIP pair to a test.
	Add a Video pair to a test.

Table 3-2.   Toolbar Buttons for Creating and Running Tests (Continued)

Icon	Tasks for Creating and Running Tests
	Add a multicast group to a test.
	Add a Video multicast group to a test.
	Add IPTV receiver group to a test.
	Edit a pair.
	Start the IPTV Channels Editor.
	Replicate a pair.
	Renumber pairs.
	Select an Ixia Network Configuration.
	Create a new Ixia Network Configuration.
	Clear an Ixia Network Configuration.
	Add a Hardware Performance pair to a test.
	Add a VoIP Hardware Performance pair to a test.

Table 3-3 lists the toolbar buttons that you use to adjust the display of the test results in the Test window.

Table 3-3.   Toolbar Buttons for Viewing Endpoint Pairs

Icon	Viewing Endpoint Pairs
	Swap Endpoint 1 and Endpoint 2.
ALL	No grouping (default).
TCP	Group by network protocol.

Table 3-3.   Toolbar Buttons for Viewing Endpoint Pairs (Continued)

Icon	Viewing Endpoint Pairs
SCR	Group by script filename.
EP1	Group by Endpoint 1 address.
EP2	Group by Endpoint 2 address.
SQ1	Group by Service Quality for Endpoint 1.
SQ2	Group by Service Quality for Endpoint 2.
PG	Group by pair group name.
PC	Group by pair comment.
	Expand all groups.
	Collapse all groups.

A separate set of icons is provided for working with application groups; these are listed in Table 3-4. (Refer to *Creating Application Group Tests* on page 5-35 and the *IxChariot Scripts Development and Editing Guide* for detailed information about application groups.)

Table 3-4.   Application group icons

Icon	Application Group Task
	Edit application group.
	Search and replace addresses in an application group.
	Validate an application group.
	Add a pair to an application group.
	Remove a pair from an application group.
	Paste application group.

The toolbar buttons listed in Table 3-5 represent standard Windows tasks:

Table 3-5.   Standard Windows task icons

Icon	Windows Tasks
	Save a test file.
	Print the results of a test run.
	Cut a pair or group of pairs from the Test window.
	Copy a pair or group of pairs to the Windows clipboard.
	Paste a pair from the Windows clipboard into another IxChariot Test window.
	Access online help for the Test window.
	Access the Ixia web site.

Using Ixia Test Ports

4

Ixia provides Performance Endpoint software for several operating systems, including the Linux operating system that runs on Ixia load module ports. The Ixia load module Performance Endpoint allows you to use Ixia ports in much the same manner as other Performance Endpoints. This chapter provides instructions for setting up Ixia ports for use in IxChariot tests. It is organized into the following topics:

- *Requirements for Using Ixia Ports in IxChariot Tests* on page 4-1

- *Theory of Operation* on page 4-5

- *Using Stack Manager to Configure and Assign Ixia Ports* on page 4-8

- *Using Stack Manager to Configure VLANs* on page 4-20

Requirements for Using Ixia Ports in IxChariot Tests

The following topics describe the requirements for using Ixia ports in IxChariot tests:

- *Ixia Chassis Software Requirements* on page 4-2

- *Ixia Chassis Chain Requirements* on page 4-2

- *Ixia Load Modules Supported* on page 4-2

- *Ixia Port Access Restriction* on page 4-4

- *Stack Manager* on page 4-4

Ixia Chassis
Software
Requirements

IxChariot requires an Ixia 400T, 1600T, 250, or Optixia chassis configured with the following software components:

* IxOS software version 4.10 SP8 or later. Refer to the IxChariot Release Notes to determine which IxOS versions are supported by your version of IxChariot.

    Refer to the "IxOS Support" topic in the *Aptixia Stack Manager User Guide* for a list of the Stack Manager features supported in each version of IxOS.

* The Tcl Server component of IxOS. IxChariot 6.10 and higher requires TclServer version 8.4 or higher.

Ixia Chassis Chain
Requirements

Each port on an Ixia chassis has a base management address defined as 10.0.x.x, where 10.0.0.0 /16 is the chassis base IP address and x.x designates the module and port on the chassis. For example, 10.0.1.1 is the management address for the first port on the first module on a chassis. This is the case for every chassis.

If you are using multiple chassis in a test, you need to ensure that each chassis has a unique base IP address. Otherwise, your tests will fail because you will have duplicate IP addresses within the set of chassis that you are using.

There are two ways that you can ensure that each port that you are using in a test has a unique management address:

* Use the Endpoint 2 address as the management address, or

* Assign a unique base IP address to each chassis that you are using. (If you change the base address for one or more chassis, you will also need to set up routes such that the chassis can communicate with one another.)

Instructions for changing the base IP address on a chassis are provided in the *Stack Manager User Guide*. (You can also use IxExplorer to change the base IP address. For instructions, refer to the IxRemoteIp topic in the *IxExplorer User Guide*.)

Ixia Load Modules
Supported

To use Ixia load module ports in your IxChariot tests, you need one or more of the following Ixia load modules:[1]

Table 4-1.   Ixia Load Modules Supported by IxChariot

Load Module	Description
ALM1000T8	8-port 10/100/1000 Mbps Ethernet.
ASM1000XMV12X-01	10 Gigabit aggregation module, Non-Aggregate Mode.
CPM1000T8	8-port 10/100/1000 Mbps Ethernet content processing module.

---

1. Note the following measured limits on the number of pairs supported by these load modules: 500 for the LM1000TXS4 and ALM1000T-8 and 200 for the LM1000TXS8.

Table 4-1.   Ixia Load Modules Supported by IxChariot (Continued)

Load Module	Description
ELM1000ST2	2-port 10/100/1000 Mbps Copper/Fiber Ethernet IPSec Encryption Load Module.
LM1000SFPS4 and LM1000SFPS4-256	4-port 1000 Mbps Ethernet.
LM1000STX4 and LM1000STX4-256	4-port 10/100/1000 Mbps Ethernet with dual copper/fiber interfaces.
LM1000STXS2	2-port 10/100/1000 Mbps Ethernet with dual copper/fiber interfaces.
LM1000STXS4 and LM1000STXS4-256	4-port 10/100/1000 Mbps Ethernet with dual copper/fiber interfaces.
LM1000TXS1	Built-in test port in the Ixia 250 chassis.
LM1000TXS4	4-port 10/100/1000 Mbps Ethernet.
LM100TXS2	2-port 10/100 Mbps Ethernet.
LM100TXS8	8-port 10/100 Mbps Ethernet.
LM10GE700F1B-P	10 Gigabit Ethernet port with Xenpak interface and advanced processor.
LM622MR and LM622MR-512	2-port ATM load module.
LSM1000XMS12	12-Port Dual-PHY (RJ45 and SFP) 10/100/1000 Mbps Ethernet Load Module for Optixia XM12.
LSM1000XMV12	12-port 10/100/1000 dual-phy load module.
LSM1000XMV16	16-port 10/100/1000 dual-phy load module.
LSM1000XMV4	4-port 10/100/1000 dual-phy for XM form factor.
LSM1000XMV8	8-Port Dual-PHY (RJ45 and SFP) 10/100/1000 Mbps load module.
LSM10G1-01	10 Gigabit Ethernet LAN Service Module.
LSM10GXL6	10 Gigabit 6-port load module for XL10 chassis.
LSM10GXM2XP	10 Gigabit 2-port IxYukon, XFP interface, Extra Performance (XP), load module.
LSM10GXM3	3-port 10 Gigabit Ethernet XFP LAN Services Module for Optixia XM12.
LSM10GXM4	10 GE Load Module, 4-Port LAN/WAN module.
LSM10GXM4XP	4-port IxYukon, XFP interface, Extra Performance (XP), load module.
LSM10GXM8XP	8-port IxYukon, XFP interface, Extra Performance (XP), load module.
MSM10G1	10 Gigabit Universal Multi-Services Module.

Table 4-1.   Ixia Load Modules Supported by IxChariot (Continued)

Load Module	Description
MSM2.5G1-01	2.5 Gigabit Universal Multi-Services Module.
OLM1000STX24	24-port 10/100/1000 Mbps Ethernet with dual copper/fiber interfaces for use with Optixia chassis.
OLM1000STXS24	24-port 10/100/1000 Mbps Ethernet with dual copper/fiber interfaces for use with Optixia chassis.

For all load modules, refer to the *Ixia Hardware Guide* for full load module specifications.

Ixia Port Access Restriction

When an IxChariot test is using an Ixia port, it is recommended that you do not access that same port through any other Ixia application, such as IxExplorer.

If you use IxExplorer or a Tcl script to reset the hardware timestamps or change the mode on a port, IxOS resets the hardware clock on that port. Any IxChariot test using the clock or hardware timestamps may experience errors and will either fail or will yield invalid results. Refer to *Run Options Tab* on page 7-2 for information about clock synchronization and Ixia hardware timestamp run options.

Stack Manager

IxChariot 6.10 (and higher) includes Stack Manager as an integrated tool for configuring Ixia ports for use in IxChariot tests. The examples in this chapter show the basic Stack Manager operations.

Stack Manager is supported by IxOS 4.0 and higher. However, not all versions of IxOS support every feature in Stack Manager. Refer to the "IxOS Support" topic in the *Aptixia Stack Manager User Guide* for a list of the features supported in each version of IxOS.

Note that Stack Manager is backward compatible with IxApplifier files (`.its` files), version 1.30 and up.

TCP Ports 6809 and 2809

When using Ixia ports in IxChariot tests, you need to bidirectionally open TCP ports 6809 and 2809. Stack Manager uses port 6809 for console-to-chassis communications and port 2809 for chassis-to-console communications.

Limitations on NAT Usage

When using Ixia ports in IxChariot tests, the following limitations on the use of Network Address Translation (NAT) are in effect:

• You cannot use NAT between the console and the chassis.

• You cannot use a dual-PAT network for IxChariot tests. This is a network in which PAT is used on each side.

Theory of Operation

Ixia load module ports may be used in three modes, either individually or simultaneously:[1]

• **Performance Endpoint**–In the same manner as any other endpoint.

• **Hardware Performance Pair**–As a background traffic generator. In addition to normal endpoint parameters, a Hardware Performance Pair uses an additional file to specify the content of background traffic to be sent from endpoint 1 to endpoint 2. Throughput and latency may be measured for the pair.

• **VoIP Hardware Performance Pair**–As a background VoIP traffic generator. Throughput and latency may be measured for the pair.

Regardless of their use as Performance Endpoints and/or Hardware Performance Pairs, Ixia ports are connected to the test network in the same manner - as discussed here.

---

1. When using LM100TXS8 cards in both modes, it is necessary to take some precautions. Refer to "Configuring Filters" in the *Stack Manager User Guide*.

Single Network Operation

One possible test model with Ixia Performance Endpoints is illustrated in Figure 4-1.

Figure 4-1.   IxChariot Test Model - Using One Network
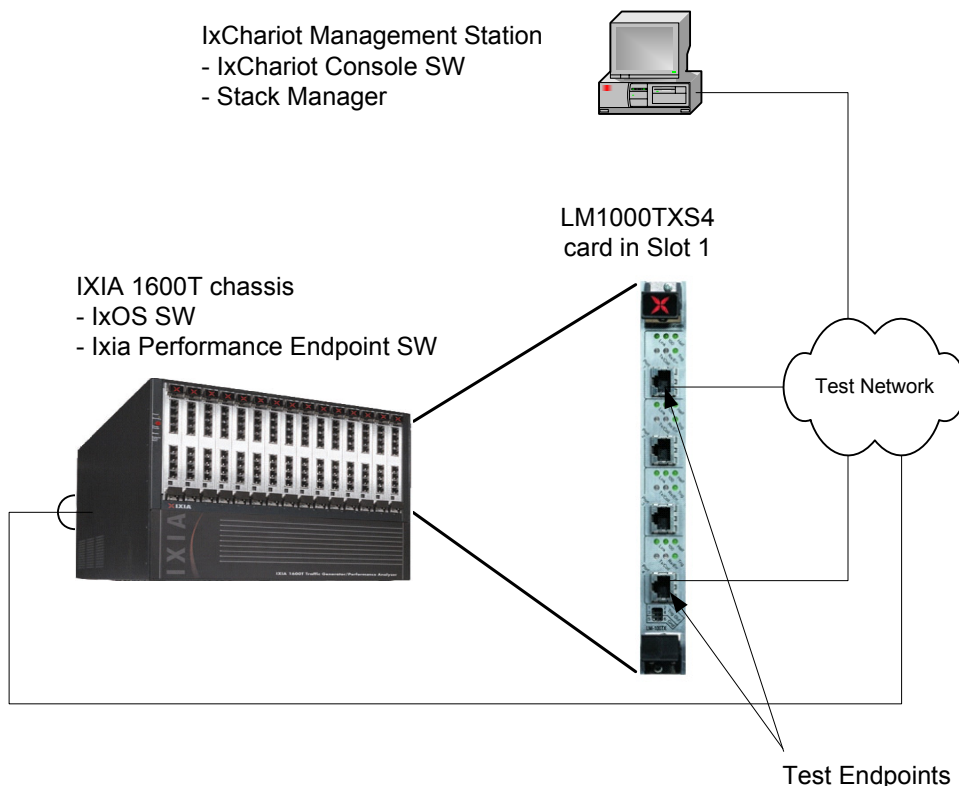
In this model, Ixia ports in an Ixia chassis substitute for the test systems. The components in this model are:

*   **IXIA chassis**–holds the Ixia load modules and must be loaded with appropriate software. Although only one chassis is shown, multiple Ixia chassis may be used; their sync out/sync in cables must be daisy chained.

    *   IxOS software.

    *   IxChariot Ixia Performance Endpoint Software–this is a set of IxChariot endpoint software specifically compiled and packaged to execute on the processors embedded on the Ixia load modules.

*   **Ixia Load Module**–one of the set of applicable Ixia load modules installed in the Ixia chassis. The list of applicable load modules is listed at the beginning of this chapter.

*   **IxChariot Management Station**–a Windows-based PC which hosts:

    *   IxChariot Console Software.

    *   Stack Manager – An integrated tool that sets up IP addresses and other configuration options on the Ixia ports and downloads the IxChariot Ixia Performance Endpoint Software to the Ixia ports.

- **Test Network**–a single network that is used to connect the management station, chassis and test ports. Note that the chassis has a network interface of its own, separate from the test interfaces. The test network, of course, contains the network equipment that is the object of the test.

Two Network Configuration

Although the one network model is simple, it is not the recommended approach for IxChariot testing. When a single network is used, both management and test traffic travel over the same network, providing additional load to the network elements being tested. The preferred method uses two networks and serves to isolate test traffic from all other traffic. Figure 4-2 illustrates a two network configuration.

Figure 4-2.   IxChariot test Model - Using Two Networks
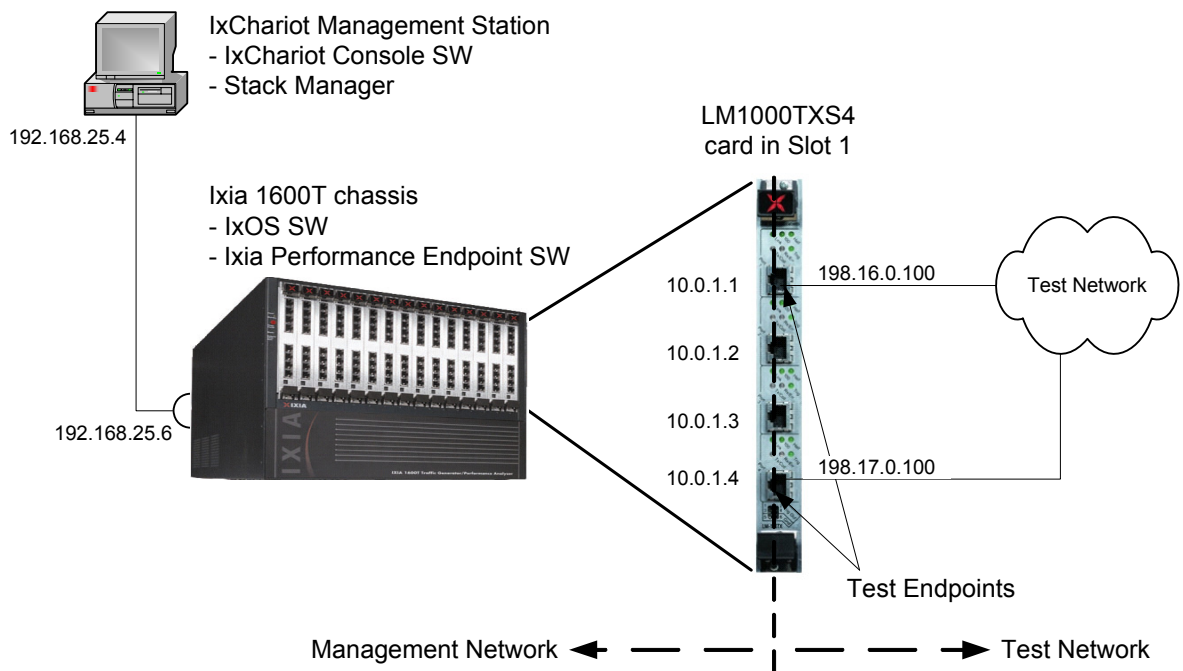
This configuration logically divides the management network from the test network. The management network includes the IxChariot Management Station, Ixia chassis and the management addresses associated with the ports (10.0.1.1-10.0.1.4 in the figure. The test network contains the network components which are part of the test network (198.*.*.* in the figure).

> **Note:** Best test results will be obtained when the test network is devoid of all other network traffic.

Setting up Ixia Ports

Ixia ports are general purpose Linux computers, with additional capabilities related to efficient network testing.

You will use Stack Manager to set up one or more interfaces on the ports with a number of standard networking characteristics, including:

- IP Address.

- MAC Address (stand-alone PCs have a hardware configured MAC address, whereas Ixia ports have a programmed value).

- Routing table, including a default gateway.

- VLAN.

- DNS.

In addition, Ixia ports allow filters to be applied to incoming network traffic to ignore irrelevant traffic that would affect test-specific operation.

You configure these parameters using Stack Manager. The normal flow of control is shown in Figure 4-3.

Figure 4-3.   Flow of Control
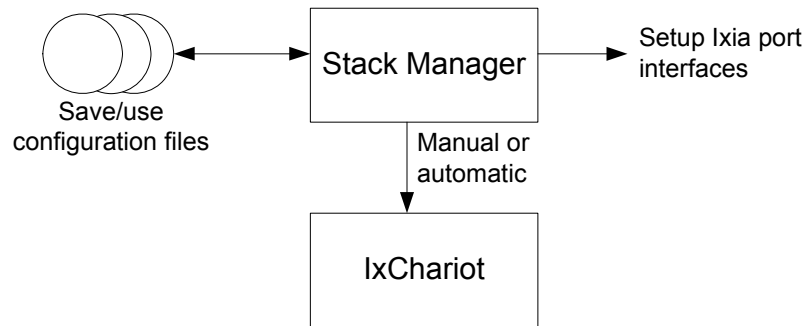
Stack Manager configurations may be saved for later re-use.

Using Stack Manager to Configure and Assign Ixia Ports

Planning

IP network addresses must be assigned to the various components in the system. Refer to the figures in *Theory of Operation* for a description of alternate networking topologies. In some cases, existing addresses may need to be changed to conform to IxChariot rules.

The following networks must not overlap:

- The test network, including the test endpoint addresses.

- The networks between the Ixia chassis and management station.

- The networks corresponding to all the chassis base addresses. Remember that these are /16 networks.

If you are using Ixia ports in your testing, you can use the procedures presented in this section to set up and execute a basic test.

For More Information

For detailed information about Stack Manager, refer to the following manuals:

* *Stack Manager User Guide*
* *Stack Manager API Reference*

Step by Step Operations

This section describes the basic steps required to configure and assign Ixia ports for an IxChariot test:

* *Start Stack Manager* on page 4-9
* *Configure Layer 1 and 2 Parameters* on page 4-10
* *Configure IP Addresses* on page 4-11
* *Continue to Build the Stack* on page 4-14
* *Assign Ixia Ports to Port Groups* on page 4-15
* *Completing the Stack Manager Session* on page 4-16
* *Selecting the Ixia Ports for a Test* on page 4-17
* *Setting the Management Addresses* on page 4-18

Step 1: Start Stack Manager

Stack Manager is fully integrated with the IxChariot Console. IxChariot starts Stack Manager when you make one of the *Ixia Network Configuration* selections from the menu or toolbar. An *Ixia Network Configuration* comprises all the configuration parameters for one or more Ixia port groups.

To start Stack Manager from the IxChariot Console:

1. Start IxChariot if it is not already running.

2. Select **New** from the IxChariot File menu.

   IxChariot opens a new Test window.

3. Click the **New Ixia Configuration** button (or select New Ixia Network Configuration from the Edit menu):

   IxChariot opens the Stack Manager window (Figure 4-4).

Figure 4-4.   Stack Manager Window
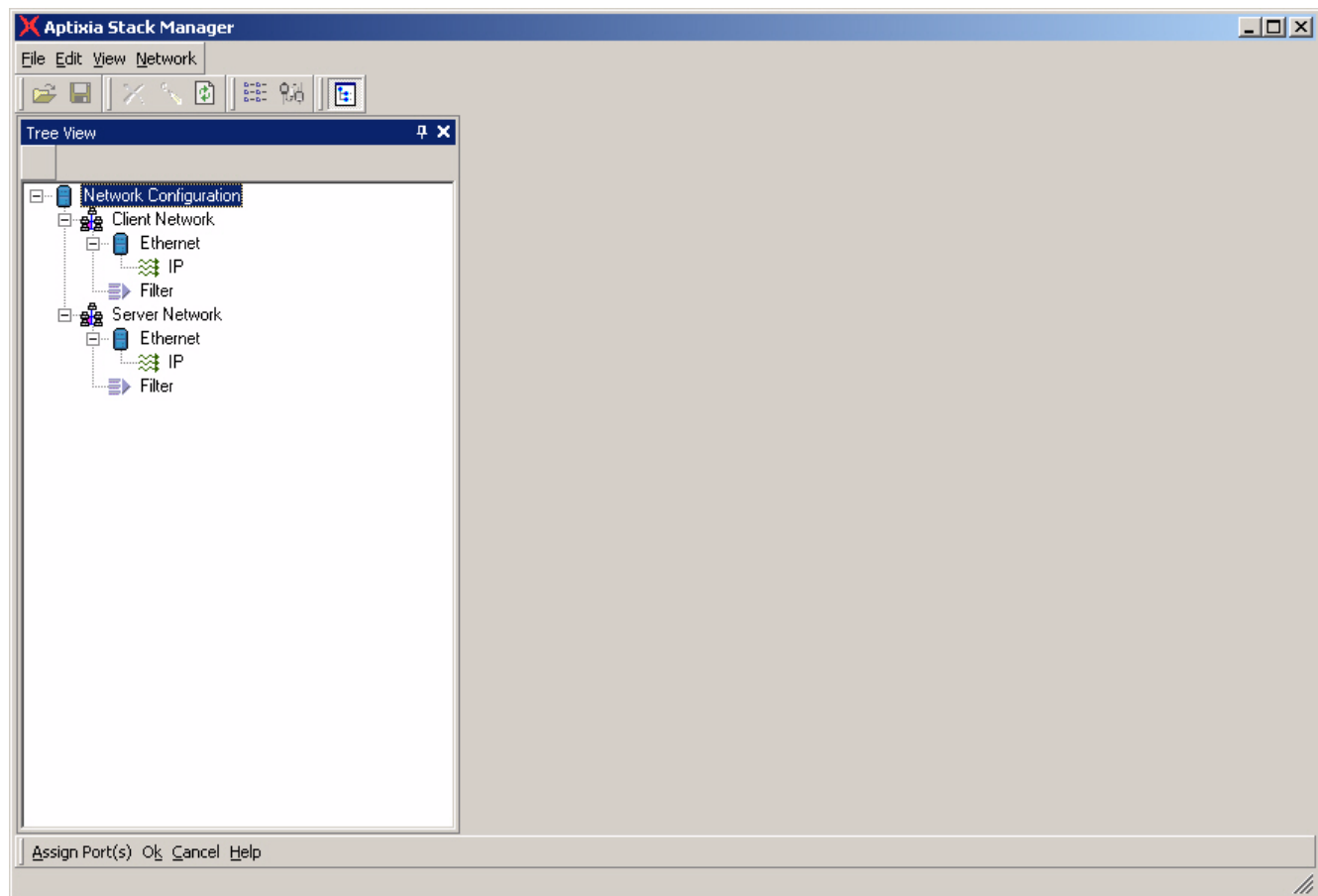
IxChariot automatically creates two port groups: Client Network and Server Network. You can use these two port groups, rename them, or delete them. You can also create additional port groups. The following steps assume that you are using the two default port groups. The procedures for configuring each are the same.

Step 2: Configure Layer 1 and 2 Parameters

For each port group, you will select the desired Hardware Type and enter the desired parameters for that type. The examples described in this section assume that Ethernet is used at layers 1 and 2.

1. To access the Ethernet properties:

   a: First select the port group (Client Network or Server Network).

   b: Then select the Ethernet module in the stack diagram (or in the navigation tree.

Stack Manager displays the available tabs and parameters for the selected hardware type, as shown in Figure 4-5.

Figure 4-5. Stack Manager - Ethernet Basic Parameters
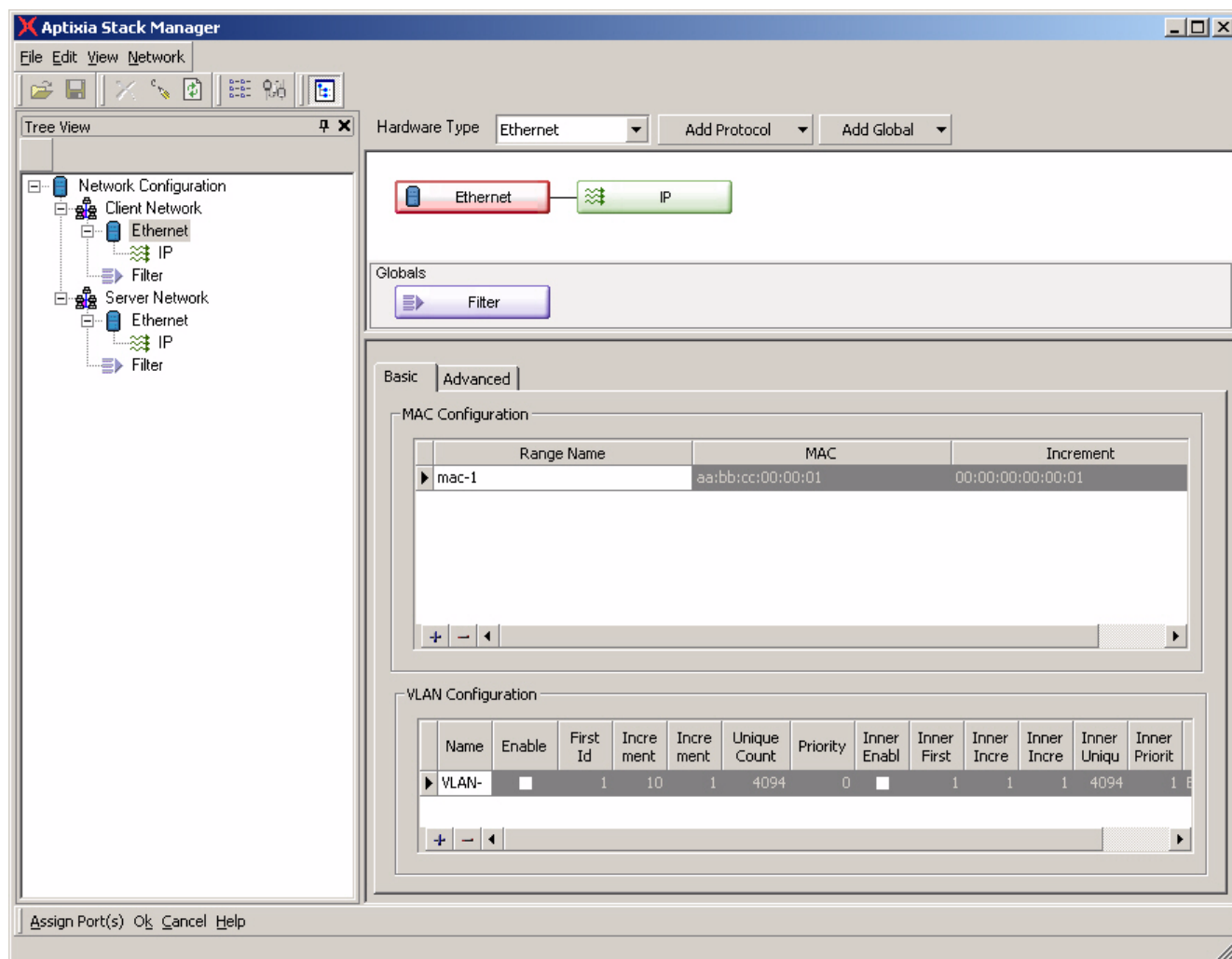
2. Either accept the default Range Name, MAC address, and Increment value, or modify them.

This is the specification for the first MAC address that will be associated with the first interface in the port group. Additional MAC addresses will be formed by adding the Increment By field to this value for each new interface.

Refer to *Stack Manager User Guide* for detailed information about the Ethernet Advanced parameters.

3. If you need to configure VLANs on your port groups, refer to *Using Stack Manager to Configure VLANs* on page 4-20 for detailed instructions.

Step 3: Configure
IP Addresses

IP addresses must be set for each port group. Therefore, you must include either an IP component or a PPPoX component in your protocol stack. You can assign multiple IP address ranges for each port group. Stack Manager supports both static and dynamic IP address allocation, as described in *Configuring Static IP Addresses* on page 4-12 and *Configuring Dynamic IP Addresses* on page 4-13.

Configuring Static IP Addresses

To configure static IP addresses for the port groups:

1. First select the port group and then select the IP module in the figure (or select it in the navigation tree).

   Stack Manager displays the window for configuring IP addresses for this port group, as shown in Figure 4-6. IxChariot provides a default IP address range, using address 172.16.1.1 /16.

Figure 4-6.   Default IP Values
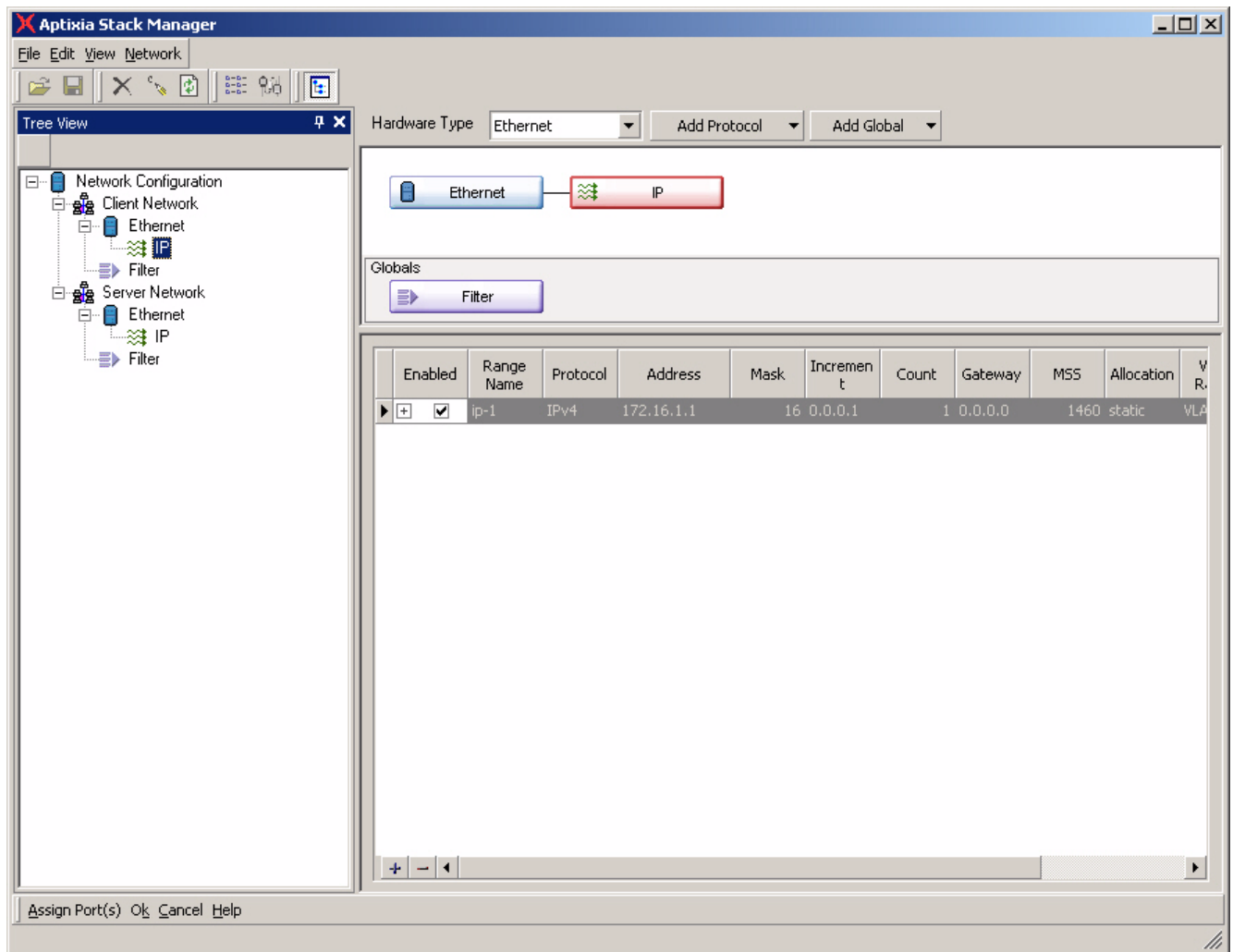
2. Either accept the values in the default IP range, or modify them to suite your needs.

3. To add additional IP interfaces for the selected port group:

   a: Click the **Append** button:

Stack Manager creates a new IP interface item, using default values. The default values are shown in Figure 4-6.

Figure 4-7.   Adding IP Interfaces

b: Either accept the default values or modify them as desired.

c: Repeat steps a and b for each for each additional IP interface that you will use in your test.

Configuring Dynamic IP Addresses

When using dynamic IP address allocation, you have the option of specifying the DHCP server to use. If you do not specify a DHCP server IP address, Stack Manager uses the services of the first DHCP server that responds to the DHCP Discover message.

1. Select the IP address range for which you require dynamic address allocation.

2. Set the Allocation parameter to *dhcp*.

3. Click the Show Nested Table button. (This is small button located on the left side of the Range Name column.)

Stack Manager opens the nested table, as shown in Figure 4-8.

4. Select the DHCP tab on the nested table.

Figure 4-8.   Setting the DHCP Server IP Address

5. Either accept the default DHCP parameters, or modify them to suit your test requirements.

Refer to *Stack Manager User Guide* for detailed information about the DHCP parameters.

Removing IP Interfaces from a Port Group

To remove an IP interface from a port group:

1. First select the port group and then select the IP module.

2. Select the IP interface from the table.

3. Click the **Delete** button:

Stack Manager removes the entry from the table.

Disabling IP Interfaces in a Port Group

To disable an IP interface for a port group:

1. First select the port group and then select the IP module.

2. Select the IP interface from the table.

3. De-select (uncheck) the **Enabled** checkbox.

A disabled IP interface is not available for use.

Step 4: Continue to Build the Stack

To use Ixia ports for an IxChariot test, you need to establish—as a minimum—a basic protocol stack that includes layer 2 and 3 addresses (as described in *Configure Layer 1 and 2 Parameters* on page 4-10 and *Configure IP Addresses* on page 4-11). You can also include additional protocols and services in your protocol stack. Stack Manager supports the following protocols and services:

Encapsulations:

- Impairment
- Emulated Router
- PPPoX
- IPSec

Services:

- Routes
- Filters
- DNS
- TCP

For detailed information, refer to the *Stack Manager User Guide*.

Step 5: Assign Ixia Ports to Port Groups

Before you begin your test, you must associate one or more Ixia ports with each of the port groups. You need not perform this step when you first define your port groups, but you will need to do so before actually running your test.

As part of the assignment of port groups, you will select the Ixia chassis in which the ports are located.

To assign Ixia ports to port groups:

1. Click **Assign Port(s)** (in the lower left corner of the Stack Manager window).

   Stack Manager opens the Port Assignment window.

2. Click the **Add Chassis** button.

   Stack Manager opens the Add New Chassis dialog (Figure 4-9).

Figure 4-9.   Add New Chassis Dialog
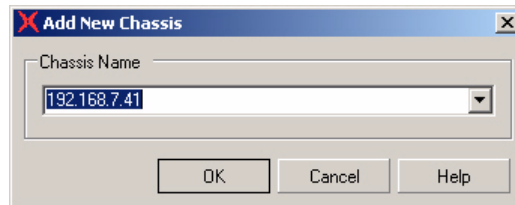
3. Enter the name or IP address of the Ixia chassis that you are using for this test, then click **OK**.

   This example uses a chassis with an IP address of 192.168.41.7.

   Stack Manager establishes contact with the chassis, then displays the chassis ports in the Port Assignment window.

4. To assign port groups to ports on the chassis:

   a: Select the desired port group from the right pane (Client Network in this example), and also select one of the ports that you are using for this test. (That is, click each of them once, as show in Figure 4-10.)

Figure 4-10. Assigning a Port in Stack Manager
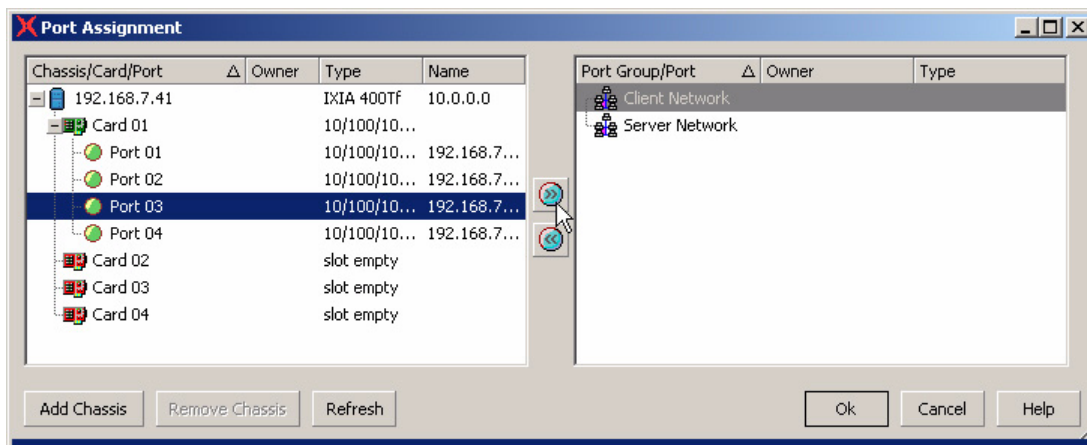
b: Click the Assign Port button to make the assignment:

Stack Manager adds the port to the Client Network port group (in the right pane), and also removes it from the list of available ports (in the left pane).

c: Repeat the process for each port group.

Figure 4-11 shows the Port Assignment window with two ports assigned.

Figure 4-11. Ports Assigned in Stack Manager
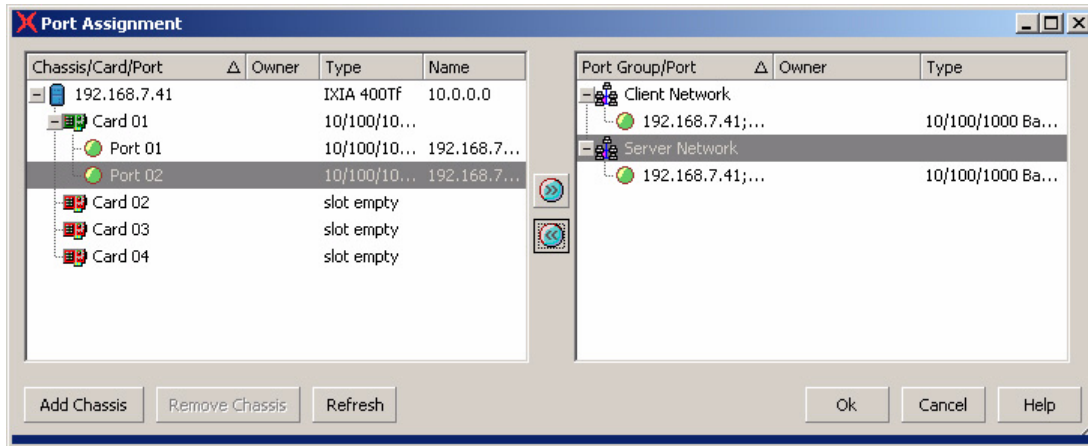
d: Click **OK** to complete the port assignment.

Stack Manager again displays the main window.

Step 6: Completing the Stack Manager Session

Once you have completed the port assignments, Stack Manager redisplays its main window. To complete your Stack Manager session:

1. Optionally, save the stack configuration to a file for later re-use:

   a: Select **Save** from the File menu.

   Stack Manager opens the Save As dialog.

   b: Enter a name for your Ixia network configuration file.

   c: Click **Save**.

   Stack Manager saves your file to the IxChariot Tests folder, giving it a file extension of .ixn.

2. To complete your Stack Manager session and return to the IxChariot Test window, Click **OK** in the main Stack Manager window.

IxChariot closes the Stack Manager window and redisplays the Test window. The Ixia ports that you configured are now ready for use in your test.

Step 7: Selecting the Ixia Ports for a Test

Once you complete your Stack Manager session, the Ixia ports that you configured are available for use in your test:

• If you have configured static IP addresses (as described in *Configure IP Addresses* on page 4-11), those addresses will be available for selection from the drop down menu, as is shown in Figure 4-12. You can select a specific IP address from the drop-down menu, or you can select the name of the port group (notice that *Client Network* and *Server Network* are available from the drop-down menu in Figure 4-12).

• If you have configured dynamic IP addresses (as described in the *Stack Manager User Guide*), you must select the port group names rather than IP addresses from the drop down menu.

Figure 4-12. Selecting an IP Address Configured in Stack Manager
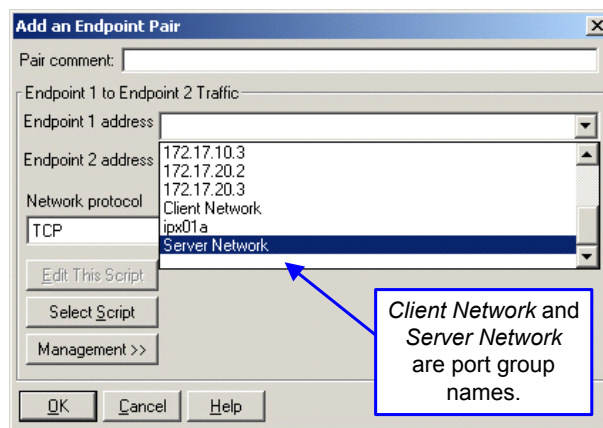
When you select a port group name (rather than a specific IP address), Stack Manager provides IxChariot with the actual IP addresses each time the test is run. For each subsequent run of the test, Stack Manager may provide different IP addresses for the test pairs. Notice in Figure 4-13 on page 4-18 that the assigned IP addresses differ for the two test runs. For example, 172.17.10.1 is the *E1 actual address* for pair 2 in the first test run, whereas in the second test run, 172.17.10.1 is assigned to pair 3.

If your test requires consistent IP addresses across multiple test run iterations, you can assign specific IP addresses rather than assigning port group names—assuming that you are using static IP addressing. If you are using dynamic IP addressing, however, the address assignments from one run to another may differ.

Figure 4-13. IP Addresses Assigned by Stack Manager

Step 8: Setting the Management Addresses

Once you select the endpoint addresses, IxChariot automatically sets the management addresses using an address format that designates the chassis, the card, and the port. For example:

```
192.168.7.41 /01 /03
```

In this example, 192.168.7.41 is the IP address of the chassis, /01 designates card 1 in the chassis, and /03 designates port 3 on that card. As shown in Figure 4-14 on page 4-19, IxChariot assigns both of the management addresses (Console to Endpoint 1 Management address, and Endpoint 1 to Endpoint 2 Management address).

Figure 4-14. Management Address Assignment

If you use these default address assignments, IxChariot displays an information message (as shown in Figure 4-15) when you click OK. This dialog asks you to confirm these management address assignments.

Figure 4-15. Management Address Assignment Confirmation

If you prefer to use the test network for endpoint management traffic, select the **Use Endpoint 2 address as management address** checkbox in the Add an Endpoint Pair dialog, as shown in Figure 4-16.

Figure 4-16. Changing the Management Address Assignment

In the example shown in this section, selecting the **Use Endpoint 2 address as management address** checkbox instructs IxChariot to use 172.16.2.1 for the management traffic passed between the two endpoints.

Using Stack Manager to Configure VLANs

You can use Stack Manager to configure Virtual LANs (VLANs) for regular IxChariot pairs, hardware performance pairs, VoIP pairs, and VoIP hardware performance pairs. When you define and enable VLAN tags, the frames generated by the IxChariot scripts or VoIP codecs are IEEE 802.1Q tagged frames.

Using VLANs with Hardware Performance Pairs

You can create tests using hardware performance pairs that are associated with VLANs. In this case:

*   Your test environment requires appropriate DUTs to support the test objectives. Typically, you will use VLAN-aware switches and routers to implement a test network such as this.

*   The Ixia ports will emulate end-user nodes generating traffic streams.

For example, Figure 4-17 shows an IxChariot test and its associated Stack Manager configuration.

Figure 4-17. VLAN Example Using Hardware Performance Pairs

This test uses two hardware performance pairs. Figure 4-17 shows the two IP ranges in the Client Network port group (the Server Network port group is defined in similar fashion). In this example, all four endpoints are in VLAN 23, but two of the endpoints are in the 172.16.0.0 /16 network while the other two are in the 172.17.0.0 /16 network.

Using VLANs with Regular Pairs

You can also create VLAN tests using regular pairs, VoIP pairs, and video pairs. In this case, you can configure the Ixia ports to emulate the entire test network: the end-user nodes, the layer 2 switches, and the layer 3 routers.

For example, Figure 4-18 shows a test network with four workstations, two switches, and one router. Two workstations are in VLAN 101 and the other two are in VLAN 103. The router provides connectivity between the VLANs.

Figure 4-18. Two LANs Connected Through a Router
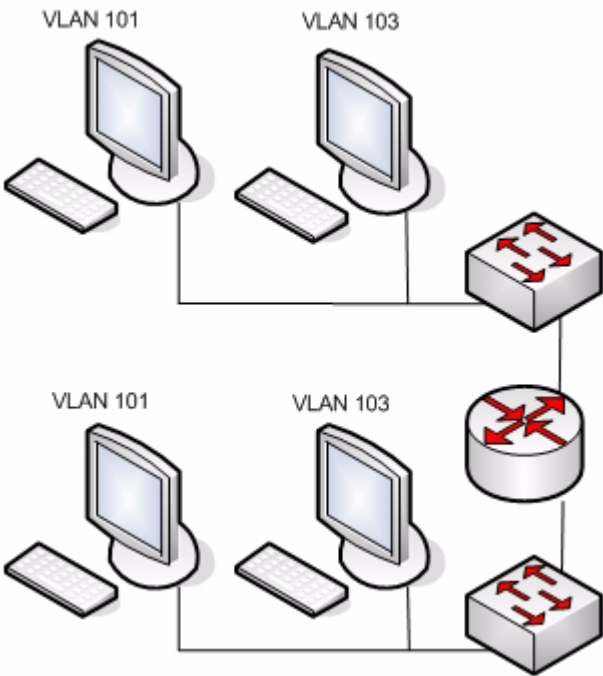
Figure 4-19 shows an IxChariot test with two pairs, each running the Throughput script. This test emulates the topology shown in Figure 4-18.

Figure 4-19. VLAN Test Using Regular Pairs

To emulate a router in this test, we include an Emulated Router component in the Stack Manager configuration. Figure 4-20 shows the Emulated Routers for both of the port groups. Notice that the Address fields and Gateway fields are exactly the opposite in the two Emulated Router definitions.

Figure 4-20. Emulated Router Definition for VLAN Test
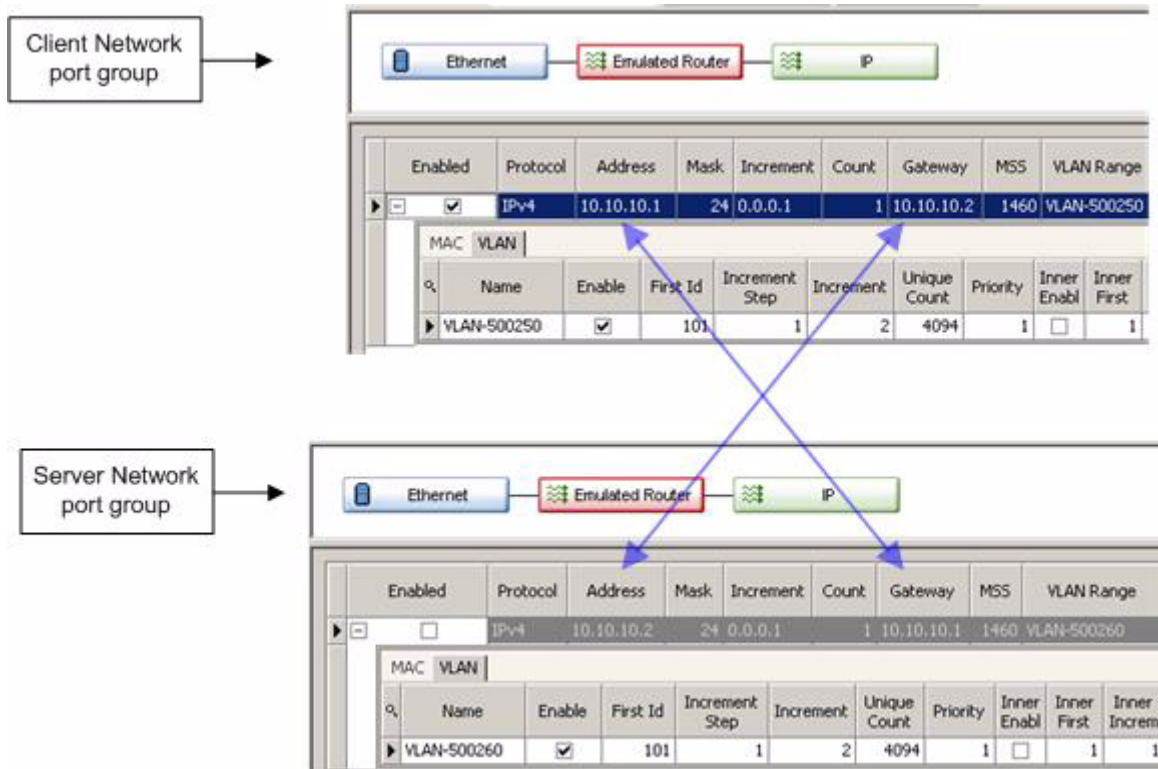
Figure 4-21 shows the Stack Manager definition of the IP components for the Client Network and Server Network port groups. Note that the Mask length is 24; therefore, the addresses defined in each port group are in separate subnets.

Figure 4-21. IP Component Definition for VLAN Test
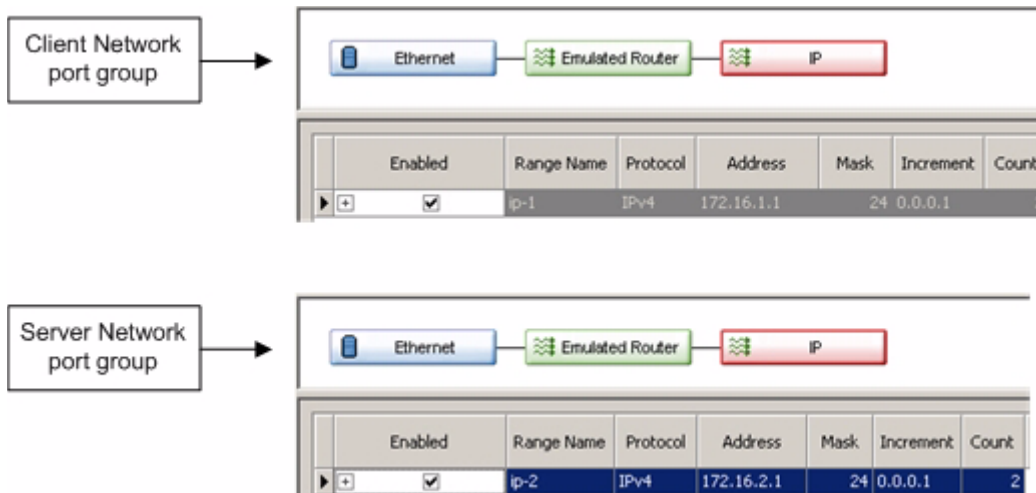
The effect of this example is as follows:

- Each Stack Manager Ethernet component defines and enables a VLAN. For each port group, the starting VLAN ID is 101 and the Increment value is two. Each time Stack Manager creates a new MAC address, it will create a new VLAN ID. In this example, the VLAN IDs in both port groups will be 101 and 103.

- Each Stack Manager IP component defines a starting IP address and specifies a count of 2, thereby creating two IP addresses per port group. For each IP address created, Stack Manager will create a new MAC address and a new VLAN ID.

- The emulated routers establish IP addresses that will be used to route traffic between the two subnets.

- With this configuration, the test (shown in Figure 4-19) uses the following VLAN IDs and IP addresses:

Pair	VLAN ID	Endpoint 1	Endpoint 2
1	101	172.16.1.1	172.16.2.1
2	103	172.16.1.2	172.16.2.2

The endpoints defined for each of the two pairs are in the same VLAN but are in different subnets. The Emulated Router provides the mechanism to pass traffic between the subnets.

Using an AFD1 in an IxChariot Test Network

This section describes how to set up a virtual chassis chain based on GPS clocking, using Ixia AFD1 units for the GPS functionality.

Physical Setup   The following topics provide a summary description of the physical setup required by a virtual chassis chain that uses AFD1 units for GPS clocking.

About the AFD1

The IXIA Auxiliary Function Device 1 (AFD1) provides the means for accurate worldwide timing using integrated Global Positioning System (GPS) technology. It is designed for distributed end-to-end measurements of key metrics, including point-to-point latency and jitter.

Refer to the Ixia Auxiliary Function Device (AFD1) chapter in the *Ixia Hardware and Reference Manual* for detailed information about the IXIA Auxiliary Function Device 1 (AFD1).

Chassis Connection

The Ixia AFD1 GPS receiver is controlled by an Ixia chassis through a USB port or a serial port. Chassis timing is provided by connecting the Sync Out of the AFD1 to the Sync In of the chassis, as shown in Figure 4-22. This configuration then enables the chassis to operate as a slave in a virtual chassis chain, with the Ixia AFD1 as the master.

Figure 4-22. AFD1 in a Chassis Chain
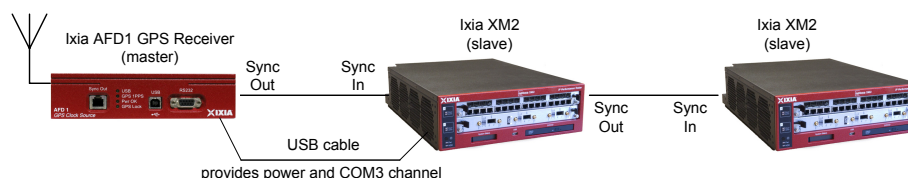
Worldwide Synchronization

Two or more Ixia chassis connected to a time reference may be distributed worldwide forming a virtual chassis chain based on GPS timing. One possible configuration is shown in Figure 4-23 on page 4-25.

Figure 4-23. Worldwide Deployment of Synchronized Chassis
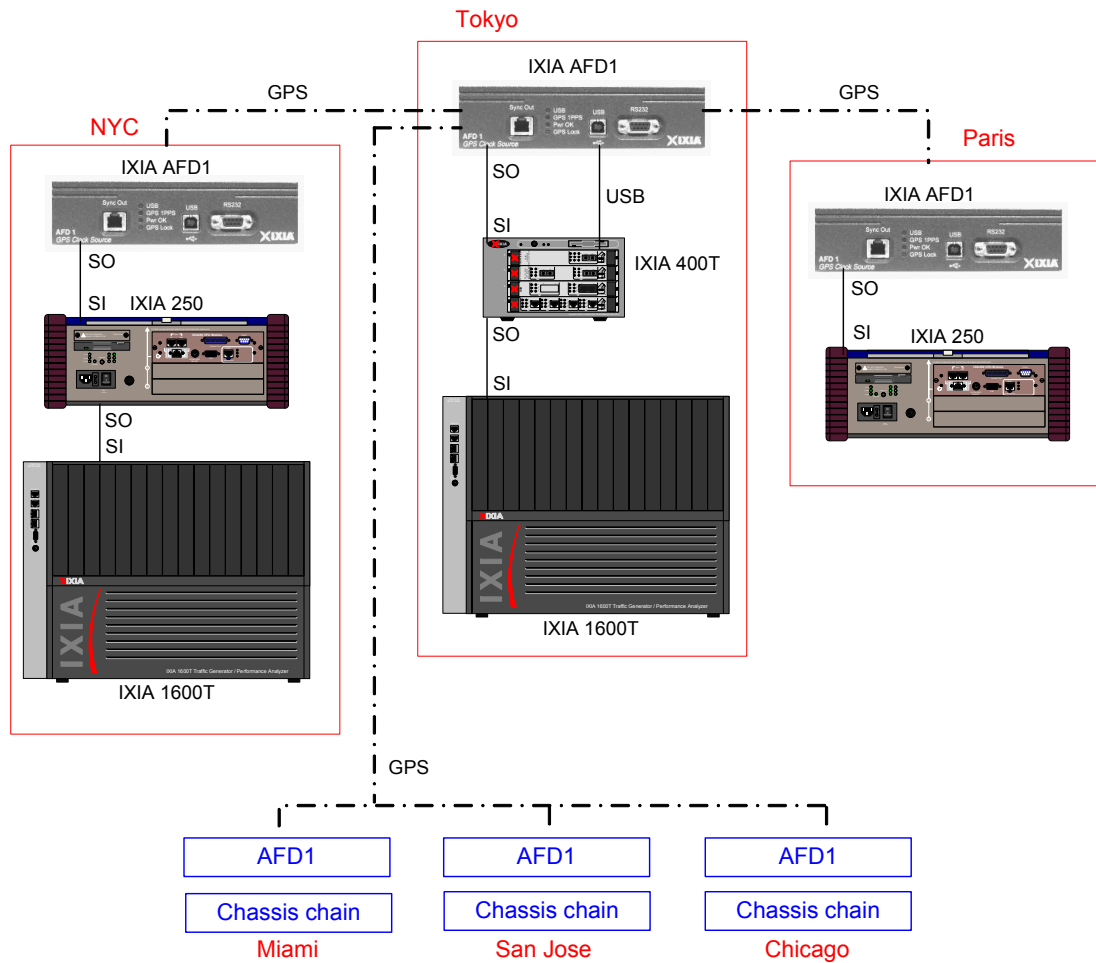
Once the timing features of the chassis are configured, operating a worldwide set of Ixia chassis is the same as local operation. The Ixia hardware and software program the clocks such that they all send a master trigger pulse to all Ixia chassis, within a tolerance of ±150 ns with GPS and ±100 us for CDMA. Ixia chassis timing operates by resetting at a fixed time-of-day on all chassis from one source, and then maintaining the time accuracy through various means.

Setting the Transmit Delay

When setting up an IxChariot test using a virtual chassis chain, you may need to change the transmit delay. The transmit delay is a chassis chain time delay that is added to Start Transmit and Clear Timestamp actions. This value is a global value for the entire chassis chain. The delay ensures that chassis that are geographically distant will be correctly synchronized and will issue commands at the same time.

To set the transmit delay:

1. Verify that you have already established the physical connections between the AFD1s and chassis, as described in *Physical Setup* on page 4-23.

2. Verfiy that each chassis in each chassis chain has a unique base IP address.

Unless each chassis has a unique base IP address, your tests will fail because you will have duplicate IP addresses within the chain.

Refer to "Changing a Chassis Chain Base IP Address" in the *Aptixia Stack Manager User Guide* for instructions.

3. Create an Ixia Network Configuration (or open an existing configuration).

Refer to *Using Stack Manager to Configure and Assign Ixia Ports* on page 4-8 for information about creating Ixia Network Configurations.

4. Display the properties of the first chassis in the chain:

a: Select the chassis name (or IP address) in the Chassis/Card/Port column of the Port Assignment dialog.

b: Right-click the mouse to display the pop-up menu.

c: Select **Properties** from the menu, as shown in Figure 4-24.

Figure 4-24.   Selecting the Stack Manager Properties Window
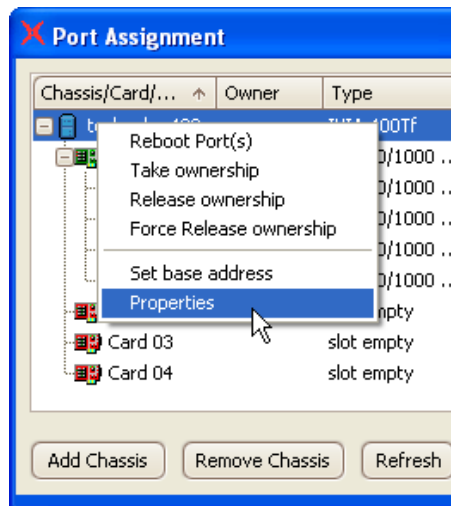
Stack Manager displays the Chassis Properties window, as shown in Figure 4-25 on page 4-27. Note that the Transmit Delay parameter is displayed only when the Chassis Mode is *Master (GPS)*.

5. Modify the Transmit Delay value as appropriate for your test network.

Figure 4-25. Chassis Properties Window
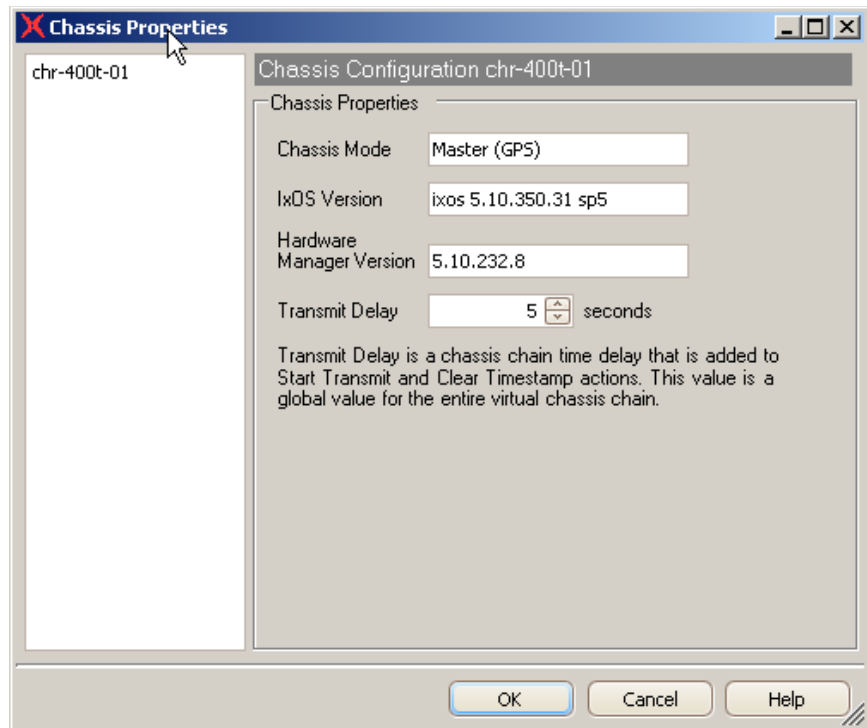
Management Address Configuration

In this type of test environment, the endpoints will typically be on different chassis. In this case, you need to ensure that the management addresses for your endpoint pairs identify the correct ports:

• Set the *Console to Endpoint 1 Management* address to the port used by endpoint 1.

• Set the *Endpoint 1 to Endpoint 2 Management* address to the port used by endpoint 2.

As illustrated in Figure 4-26 on page 4-28, Endpoint 1 uses port 3 on module 1 on the *te-10* chassis, while Endpoint 2 uses port 3 on module 1 on the *te-20* chassis. The management IP addresses for these two ports are 10.0.0.1 and 20.0.0.1, respectively.

Figure 4-26. Specifying Management Addresses
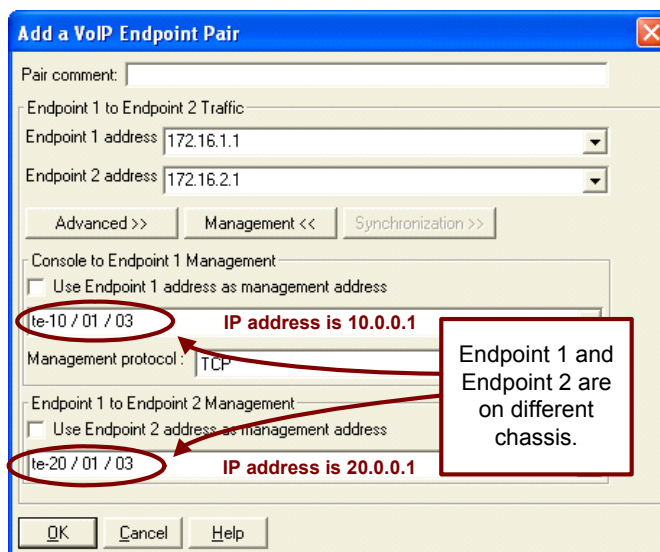
Clock
Synchronization

To use GPS clocking from the AFD1, you must select *Use Ixia hardware clock synchronization* as the clock synchronization setting for the test. This is shown in Figure 4-27 on page 4-29.

Figure 4-27. Clock Synchronization Setting for AFD1 Usage
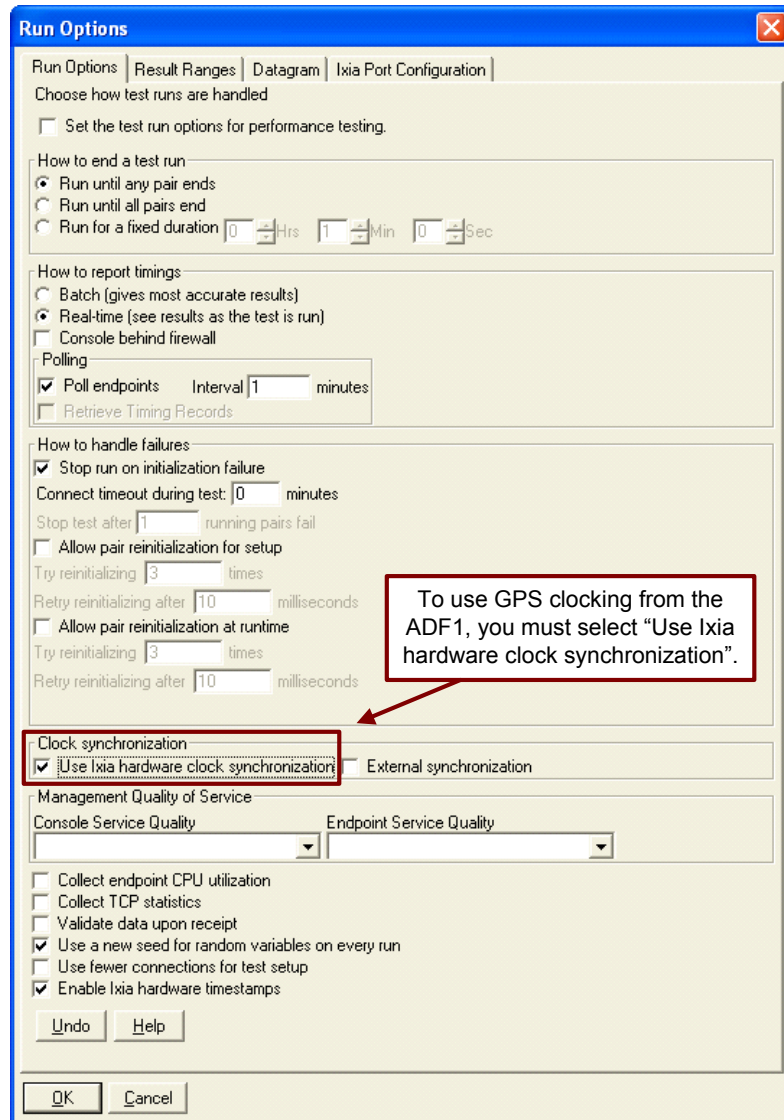
Configuring Routes

In a test environment in which the endpoints are on different chassis, you must also configure static routes for the test.

Configure Routes in Stack Manager

You need to configure static routes in Stack Manager. These static routes ensure that each endpoint will use the backplane gateway to reach the other endpoint:

• For the Client Network port group, add a route to enable Endpoint 1 to reach Endpoint 2 on the other chassis.

• For the Server Network port group, add a route to enable Endpoint to reach Endpoint 1 on the other chassis.

This is illustrated in Figure 4-28 on page 4-30.

Figure 4-28. Setting Static Routes in Stack Manager
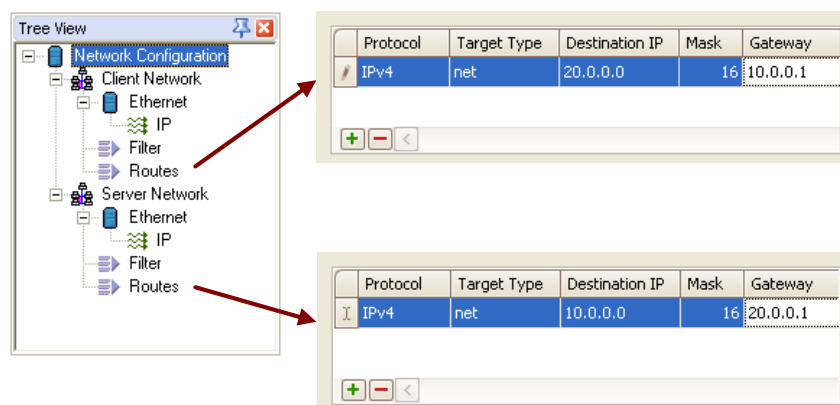
Configure Routes on Each Chassis

You also need to configure a static route for Windows on each chassis. This is the route one chassis will use to reach the other chassis:

1. On the first chassis (used for Endpoint 1), use the Windows command line to add a static route to Chassis 2. For example:

```
route add 20.0.0.0 mask 255.255.0.0 gateway 10.205.16.52
```

2. On the second chassis (used for Endpoint 2), use the Windows command line to add a static route to Chassis 1. For example:

```
route add 10.0.0.0 mask 255.255.0.0 gateway 10.205.16.51
```

In these examples, 10.205.16.52 and 10.205.16.51 are the addresses of the management ports for each of the two chassis.

5

How To

The following topics are intended to help you get started running some basic tests and using some of IxChariot's most popular features.

- *Network Protocol Configuration* on page 5-2 offers tips on running tests with the IPX/SPX, RTP, TCP, and UDP protocols.

- *IPv6 Configuration and Testing* on page 5-10 explains how to perform IPv6 tests.

- *Creating and Running Tests* on page 5-17 goes over the basics of setting up and running IxChariot tests.

- *Creating Application Group Tests* on page 5-35 describes the procedures for creating a synchronized pair test based on an application group.

- *Using Test Scheduler* on page 5-37 describes the use of the IxChariot Test Scheduler tool.

- *Comparing Test Results* on page 5-43 provides instructions for comparing the results of two or more tests.

- *Encrypting Setup Flows* on page 5-49 describes the options for encrypting test setup data.

- *Modifying the TCP Window Size* on page 5-52 provides instructions for tuning the TCP window buffers to support the TCP window scale option defined in RFC 1323.

- *Editing Script Variables* on page 5-57 provides instructions for modifying script variables for a test.

- *Exporting and Printing Results* on page 5-60 explains how you can configure your printing and exporting options to get reports containing the results you need.

- *Command-Line Programs* on page 5-67 is a comprehensive guide to IxChariot's command-line utilities, including tips for using them.

- *Visual Test Designer* on page 5-75 walks you through IxChariot's quick and easy test-building utility.

Network Protocol Configuration

The following topics describes the layer 3 and layer 4 protocols that you can use to execute tests in your IxChariot test network.

- *Choosing Protocols for Use in Tests* on page 5-2
- *IPX and SPX Configuration* on page 5-3
- *TCP Configuration* on page 5-5
- *UDP Configuration* on page 5-6
- *RTP Configuration* on page 5-7
- *Determining Packet Sizes* on page 5-10

Choosing Protocols for Use in Tests

The protocols supported by the IxChariot Console are shown in the **Network Protocol** list when you create new endpoint pairs. You can create tests using these protocols, but the tests will not run unless the protocols are also supported on the respective endpoints. Not all endpoint operating systems support testing with all of these protocols; refer to "Endpoint Requirements and Capabilities" in the *Performance Endpoints* guide for more information.

Communication Between the Console and Endpoint 1

For communications between the IxChariot Console and Endpoint 1, IxChariot uses only connection-oriented protocols:

- SPX
- TCP

Communication Between Endpoints

Performance Endpoints can communicate with each other using any of the following protocols (provided that the endpoint computers are configured to support them):

- IPX
- SPX
- RTP
- RTP-IPv6
- TCP
- TCP-IPv6
- UDP
- UDP-IPv6

Refer to *IPv6 Configuration and Testing* on page 5-10 for more information about testing in networks that support IPv6.

Test Setup Communications

The Edit Pair Setup dialog box lets you configure a different connection for test setup communications between the Console and Endpoint 1 and between the endpoints. By default, the Console will "Use Endpoint 1 address as management address" when sending test setup information to E1. To use a different address, protocol, or service quality, click **Edit Pair Setup** on the Edit menu in the Test window. If you are using a connectionless protocol between the endpoints and "Use Endpoint 1 address as management address" is checked, the Console automatically uses the corresponding connection-oriented protocol for its connection to Endpoint 1. So if you choose the IPX protocol for the test, by default the Console uses SPX between the Console and Endpoint 1. The default connection-oriented protocol for the UDP or RTP protocols is TCP. Although you can enter an alternate network address for test setup communications between the endpoints, you cannot choose a different protocol or service quality.

IPX and SPX Configuration

To use the IPX or SPX protocol in IxChariot tests, IPX addresses must be supplied as the network address at the Console when adding an endpoint pair. IPX addresses consist of a 4-byte network number (8 hexadecimal digits) followed by a 6-byte node ID (12 hex digits). The network number and node ID are separated by a colon. The 6-byte node ID (or the *device number*) is usually the same as the MAC address of the LAN adapter you are using.

For Windows, IxChariot makes WinSock version 1.1 Sockets-compatible calls when using the IPX or SPX network protocol. For NetWare, IxChariot makes calls to the TLI API when using IPX or SPX.

IPX/SPX Aliases

It can be time-consuming to enter IPX addresses; they consist of 8 hex digits, a colon, then 12 more hex digits. When using the IPX or SPX protocol in IxChariot tests, you can instead enter an easy-to-remember alias in the Edit Pair dialog box. IxChariot maintains these aliases for you in a file at the Console. You set up the mapping only once; the alias names are valid until you change them.

The IPX/SPX alias values are stored in the SPXDIR.dat file, located in the same directory as IxChariot. You can use this file to edit aliases or move the alias values to an installation of IxChariot on another computer.

To map IPX/SPX aliases, click **Edit IPX/SPX Entries** on the Tools menu. The IPX/SPX Alias List dialog box is shown. This dialog box provides the current list of aliases and lets you add new aliases and modify, delete, and copy existing aliases. The list of aliases is preserved when you upgrade to a new version of IxChariot.

Add IPX/SPX Aliases

To add a new IPX/SPX alias:

1. Select **Tools** > **Edit IPX/SPX Entries…**

IxChariot opens the IPX/SPX Alias List dialog.

2. Click **Add**.

3. Enter the name of the new alias in the **Alias Name** field.

4. In the **Address** field, enter an "8:12" or "12:8" hex address.

5. To save the new alias, click **OK**.

The new alias appears in the IPX/SPX Alias List.

Modify IPX/SPX Aliases

To modify an IPX/SPX alias:

1. Select **Tools** > **Edit IPX/SPX Entries…**

   IxChariot opens the IPX/SPX Alias List dialog.

2. Select the alias that you want to modify.

3. Click **Modify**.

4. Make the desired modification. You can modify the name, the address, or both the name and the address.

5. To save the modified alias, click **OK**.

The modified alias appears in the IPX/SPX Alias List.

Determining Your IPX Network Address in Windows

You must find the IPX address of an endpoint via a command prompt at the endpoint. These instructions assume that your endpoint software is version 3.5 or later.

Use the "ipxroute config" command to determine an IPX address on a Windows XP, Windows 2000, or Windows 2003 system.

If your IPX software support is configured correctly, your output looks similar to the following:

```
NWLink IPX Routing and Source Routing Control Program
v2.00
 net 1: network number 00000002, frame type 802.2, device
AMDPCN1 (0207011a3082)
```

The 8-digit network number is shown first; here, it is 00000002. The 12-digit node ID is shown in parentheses at the end; here it is 0207011a3082, which is our Ethernet MAC address. Thus, the IPX address to be used in tests is 00000002:0207011a3082.

TCP Configuration

When you select TCP or TCP-IPv6 as your network protocol, IxChariot uses the standard TCP header for the reliable transport of your test's application traffic. TCP supports the collection and reporting of test statistics without the need for any modifications to the header.

Note that IxChariot does not emulate TCP half-close behavior. (TCP *half-close* refers to the ability for one end of a connection to terminate its output while it continues to receive data from the other end.)

Determining Your IP Network Address

To find the local IP address on a Windows computer, enter the following command from the command line:

```
ipconfig
```

To find the local IP address on a UNIX or Linux computer, enter the following command from the command line:

```
ifconfig
```

Sockets Port Number

IP networks use *network addresses* to forward traffic across a network to a specific device, and they use *port numbers* to deliver traffic to a specific application running on the selected device.

IxChariot uses a designated *management port* to transport management traffic between the console and the endpoints. The management port is one of the following:

- SPX transport: port 10117

- TCP transport: either port 10115 (the default) or a user-selected port. (For information about designating a management port, refer to *Management ports* on page 6-33 and the endpoint.ini information in the *IxChariot Performance Endpoints* guide.)

During test initialization, Endpoint 1 randomly picks an available port and transmits the test setup information to the management port on Endpoint 2. Endpoint 2 then sends an acknowledgment from the management port to the port from which Endpoint 1 transmitted the setup information.

IxChariot scripts designate the ports that the endpoints use to execute the test. You edit the scripts to configure these port addresses. For each endpoint, you can either set the source and destination ports to specific port numbers or you can set the ports to the AUTO keyword (AUTO is the default). When set to AUTO, the endpoints assign the port number automatically. Setting the port numbers to AUTO gives the best performance and is the preferred choice when testing with multiple pairs. (Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about the Script Editor.)

UDP Configuration

IxChariot uses three distinct variations of the UDP protocol:

- **IxChariot Reliable UDP for Non-Streaming Scripts**

  When you select UDP as the network protocol for a non-streaming script, IxChariot automatically uses this proprietary UDP implementation. This version of UDP is a reliable transport protocol (uses acknowledgements and windowing) and includes a custom header within the Data field.

- **IxChariot UDP for Streaming Scripts**

  When you select UDP as the network protocol for a streaming script, IxChariot automatically uses this proprietary UDP implementation. It is the default for all streaming scripts (unicast and multicast, VoIP, and Video). This version of UDP includes a custom header within the Data field.

- **RFC 768 UDP**

  When you select UDP as the network protocol for a streaming script, you can modify that script (using the Script Editor) to use the RFC 768 compliant UDP protocol. You can select this UDP variant for unicast and multicast streaming scripts, but you cannot select it for use with VoIP or Video pairs. A reduced set of statistics can be measured when using this implementation of UDP. It can be of value, however, when you are using IxChariot as a traffic generator.

The first two implementations are specifically designed for IxChariot testing. They both modify standard UDP packets by embedding custom headers within the Data field. These modifications are needed to measure and report statistics, such as duplicate and out-of-order datagrams.

How to select RFC 768 UDP

To use the RFC 768 compliant version of UDP as the network protocol for a test:

1. Open the script in the Script Editor.

2. Select the first line in the script (line 1).

3. Select **Insert** > **UDP compliant with RFC 768 option…**

   The Script Editor insert the OPTION command to enable RFC 768 UDP.

4. Select **File** > **Save to Pair** to save your modified script with the selected pair.

UDP Protocol Compatibility

IxChariot introduced the RFC 768 compliant version of UDP in release 6.40. Therefore, the RFC 768 UDP option requires release 6.40 for the IxChariot Console and both endpoints.

If you save a test that uses the RFC 768 UDP option to an older test format (6.30 or earlier), IxChariot saves the script in any pair that uses the RFC 768 UDP option as a standard IxChariot streaming script. It removes the RFC 768 UDP option from the script and also clears the results for that pair.

Comparison of UDP Statistics Measures

The RFC 768 compliant version of UDP differs from the proprietary IxChariot version of UDP in the number of statistics that can be measured, and also the point at which the statistics are delivered and presented in the Test window. Following is a summary of the differences exhibited by the RFC 768 compliant version of UDP (versus UDP for streaming scripts):

Lost Data Tab

- Maximum Consecutive Lost Datagrams statistics are not collected.

- The following statistics are displayed only when the test is finished, rather than after every timing record: Bytes Sent by E1, Bytes Lost from E1 to E2, and Percent Bytes Lost E1 to E2.

Raw Data Totals tab

- The following statistics are displayed only when the test is finished, rather than after every timing record: Bytes Sent by E1.

Datagram Tab

- Duplicate DGs Received by E2, and Datagrams Out of Order statistics are not collected.

- The following statistics are displayed only when the test is finished, rather than after every timing record: Total DGs Sent by E1; DGs Lost, E1 to E2.

RTP Configuration

IxChariot tests support RTP, the real-time transport protocol defined in RFC 3550. RTP provides end-to-end network transport functions for applications transmitting real-time data—such as audio, video, or simulation data—over multicast or unicast network services. Many leading voice and video applications use RTP. IxChariot VoIP endpoint pairs always use RTP.

RTP Header

In IxChariot tests, RTP is carried on top of UDP. The RTP fixed header fields are shown in Figure 5-1.

Figure 5-1.   RTP Fixed Header Fields

```
0                             1                       2                           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

V=2|P|X|  CC   |M|     PT     |         sequence number

                         timestamp

         synchronization source (SSRC) identifier

       contributing source (CSRC) identifiers [optional]
```
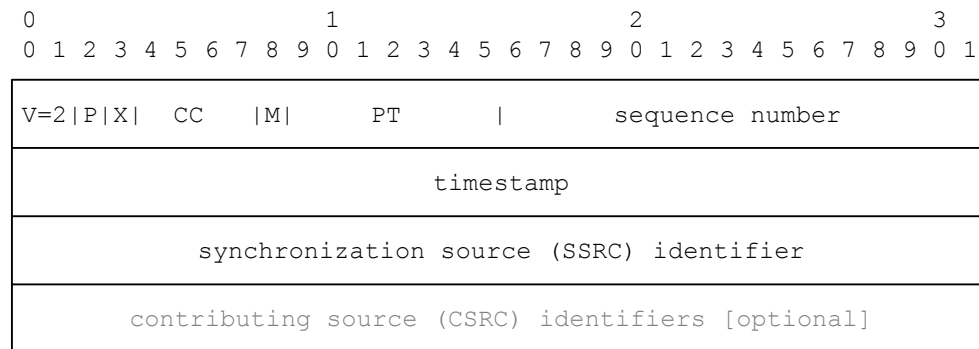
The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer. (An RTP mixer is an intermediate system that receives RTP packets from one or more sources, possibly changes the data format, combines the packets in some manner, and then forwards a new RTP packet.) Datagrams generated by IxChariot do not include the CSRC field.

RTP Header Timestamp

IxChariot tests support two timestamp options in RTP headers:

- IxChariot legacy timestamps: In IxChariot 6.40 and earlier, the *timestamp* field in the RTP header always contained a timestamp derived from the system clock. In this case, the RTP headers contain the actual SEND time for each datagram. This allows IxChariot to calculate the one-way delay for a test pair. When this timestamp option is used, the RTP header is 12 bytes in length.

- RFC 3550-compatible timestamps: In IxChariot 6.50 and higher, users can choose RTP headers that are compatible with RFC 3550. In this case, the values in the *timestamp* field are monotonic. (A sequence increases monotonically if, for every n, P $n+1$ is greater than or equal to P n.) RFC 3550 states that:

   The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations.

   When this timestamp option is used, the RTP header is 20 bytes in length.

IxChariot implements the RFC 3550-compatible timestamps by using the extension (X) bit, as shown in Figure 5-2. In this case, the *timestamp* field contains a monotonous timestamp value, while the header extension contains a time value derived from the system clock.

Figure 5-2.   RTP Header Extension Field

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
V=2|P|X|  CC   |M|    PT     |         sequence number
                        timestamp
        synchronization source (SSRC) identifier
      defined by profile      |           length
    header extension - timestamp derived from system clock
```
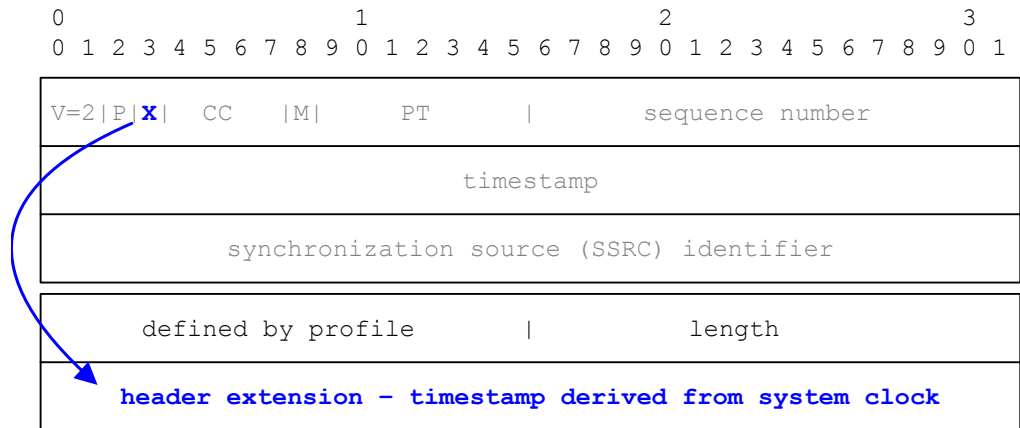
When the X bit in the RTP header has a value of 1, a variable-length header extension is appended to the RTP header, following the RTP fixed header fields. The header extension contains a 16-bit *defined by profile* field (which is not used by IxChariot) and a 16-bit *length* field that counts the number of 32-bit words in the extension (which IxChariot sets to 1).

Setting the RTP Timestamp Option

To use the extension header for RTP timestamps in an IxChariot test:

1. Open the test file (or create a new test).

2. Select **Run** >**Set Run Options…**

   IxChariot opens the Run Options dialog.

3. Select the Datagram tab.

4. Select or de-select the *Use extended headers for RTP timestamps* option:

   When you select this option, IxChariot will generate RTP packets with the header extension described in *RTP Header Timestamp* on page 5-8.

   When you de-select this option, IxChariot will generate RTP packets with IxChariot legacy timestamps.

5. Click **OK**.

When a release 6.50 (or higher) Performance Endpoint is running RTP traffic with a pre-6.50 Performance Endpoint, the RTP extension header will not be used. In this case, the RTP packets will contain IxChariot legacy timestamps.

RTP Port Numbers

RTP flows use even port numbers. The recommended value for the port number is between 16384 and 65535. If you select AUTO for the source_port or destination_port variable, IxChariot uses an even port number in this range.

Determining Packet
Sizes

Typical multimedia applications use various packet sizes. When emulating them, you should account for any header that may be included in the size of the packet. Endpoint multimedia support uses a 9-byte header for UDP and either a 12-byte or a 20-byte header for RTP. In addition, the protocol stack adds on 8 bytes for UDP and 20 bytes for IP.

For example, if you are using UDP and the desired packet size is 512 bytes, the `send_buffer_size` should be 475 (512 – 9 – 8 – 20=475). If you are using RTP (without the header extension option) and the desired packet size is 512 bytes, the `send_buffer_size` should be 472 (512 – 12 – 8 – 20=472).

IPv6 Configuration and Testing

Related Topics

You can create and run tests that use IPv6 to route test traffic between endpoints. In addition, you can also use IPv6 for your management network (for sending test setup information from the console to Endpoint 1 and sending test report flows from Endpoint 1 back to the console).

IPv6 Test Module
Features

The IPv6 Test Module includes these features:

Table 5-1.   IPv6 Test Module Features

Feature Category	IPv6 Test Module Features
IxChariot endpoint and hardware performance pair options	The following network protocol choices: TCP—IPv6 UDP—IPv6 RTP—IPv6
VoIP endpoint and VoIP hardware performance pair options	The following network protocol choice: RTP–IPv6

Endpoint Support
for IPv6

Ixia Performance Endpoints provide IPv6 support for the following platforms:

• Ixia ports

• Windows XP and newer (32- and 64-bit)

• RedHat Linux versions 8.0 and higher, with Linux kernel 2.4.20.

• Sun Solaris SPARC and x86

General Information
about IPv6

IPv6 was designed to coexist with—and eventually to replace—IPv4, while enlarging some of its capabilities. The IETF explains that IPv6 adds the following to IP:

• **Extended Addressing**

IPv6 reserves 128 bits for the IP address space. (IPv4 reserves only 32 bits.) The larger address space allows for more levels in the addressing hierarchy and an exponentially increased number of addresses.

The longer IPv6 address now identifies a network *interface*, not a network node. In effect, an address identifying any of a node's interfaces thus identifies the node as well.

*   **New Address Type**

    A new type of address—called an "anycast address"—in the IPv6 spec identifies sets of interfaces; a packet sent to an anycast address is delivered to one of these interfaces. IPv6 multicast addresses work in a similar manner to route packets to *all* interfaces in a set. By contrast, IPv4 multicast addresses route packets to *all nodes* in a set. Testing with anycast addresses isn't supported in this version of the IPV6 Test Module.

*   **Header Minimization**

    To minimize bandwidth consumption and processing cost, some IPv4 header fields have been dropped or made optional. Therefore, the IPv6 header is only twice the size of the IPv4 header despite the vastly increased address space.

*   **Quality of Service Support**

    With the 24-bit Flow Label field, packets within particular traffic "flows"— sequences of packets bound for the same receiver—may be designated for special handling, including quality of service and "real-time" service. The 4-bit Priority field allows for distinctions in handling to be made between low- and high-fidelity audio or video packets, for example. IPv6 test flows are not supported.

*   **Authentication and Security**

    IPv6 extensions support authentication, data integrity checking, and privacy. Testing with IPv6 extensions for authentication and security isn't supported in this version of the IPV6 Test Module.

The IPv6 protocol comprises the basic IPv6 header plus extensions. See www.ipv6.org/ for more information.

At present, many IPv6 deployments employ a tunneling mechanism, such as 6to4 or 6over4, to send IPv6 traffic over the IPv4 portion of the Internet. Such mechanisms may use an interim IPv6 address prefix and encapsulate IPv6 packets within IPv4 headers. In our testing, we used manually configured tunnels to send IPv6 traffic over an IPv4 network.

The IPv6 Header

An IPv6 packet consists of three parts: the IPv6 header, any (optional) extension headers, and the data payload:

Figure 5-3.   IPv6 Packet

IPv6 Header 40 bytes	Extension Headers	Data

The final byte in the extension headers, which are included for future extensions to IPv6, indicates the upper-layer protocol of the data payload bytes.

Within the IPv6 header are eight fields. In most cases, they correspond to an equivalent field in the IPv4 header, with some exceptions:

Table 5-2.   IPv6 Header Fields

IPv4 Header Field	IPv6 Header Field
Version—4 bits. IPv4.	Version—4 bits. IPv6.
Internet Header Length—4 bits	None. IPv6 header is always 40 bytes.
Type of Service—8 bits	Traffic Class—8 bits
None.	Flow Label—20 bits. Used for IPv6 QoS.
Total Length—16 bits. Payload + header.	Payload Length—16 bits. Payload size only.
Identification—16 bits. Used to reassemble datagram after fragmentation.	None. Fragmentation information included in an extension header.
Fragmentation flags—4 bits. Indicates whether datagram can be fragmented.	None. Fragmentation information included in an extension header.
Fragment Offset—13 bits. Used to reassemble datagram after fragmentation.	None. Fragmentation information included in an extension header.
Time to Live—8 bits. Indicates how many hops the datagram can pass over.	Hop Limit—8 bits. Indicates how many hops the datagram can pass over.
Protocol—8 bits. Protocol of the data payload.	Next Header—8 bits. Protocol of the data payload.
Header Checksum—16 bits. Indicates whether errors have been introduced into packet by a router.	None. Error detection performed at Link layer.
Source Address—32 bits	Source Address—128 bits
Destination Address—32 bits	Destination Address—128 bits
Options—Variable length.	None. Replaced by extension headers.

Tips for Running Tests with the IPv6 Test Module

The following topics provide general guidelines and tips for using IPv6 network addressing in IxChariot tests.

Mixing IPv4 and IPv6 Addresses

When using Ixia ports as endpoints, do not try to use IPv4 and IPv6 addresses within a single port group. If your test requires both, create separate port groups for each type. For example, use one port group named ClientNetv4 for your IPv4 addresses and another port group named ClientNetv6 for your IPv6 addresses.

IPv6 addressing

IPv6 addresses are extremely lengthy and difficult to type correctly. We recommend that you stick to host names and don't try to type the full set of eight hex character combinations. However, if you do choose to type the full hex address, keep the following in mind:

- IxChariot's `endpoint.dat` file, found in the `IxChariot` root directory, keeps track of any endpoint addresses you enter and displays them in the Endpoint lists when you add new pairs. If you make a mistake, the `endpoint.dat` file is a text file you can edit.

- All eight blocks of the hex IPv6 address are separated from each other by a single colon (:).

- Double colons indicate zero compression, in which a contiguous series of 16-bit binary address blocks containing only zeroes have been reduced. Zero compression can only be used once when expressing an IPv6 network address.

- It Is also possible to leave out any zeroes that begin a new block in the hex address. For example, the blocks `34BD:00FE` can be reduced to `34BD:FE`. Just make sure each remaining block contains at least a single digit. Thus the following are two legal expressions of the same address:

      34BD:00FE:0000:3D2A:02BB:00FF:DA33:0C4C
      34BD:FE:0:3D2A:2BB:FF:DA33:C4C

Zone IDs

Adding a zone ID to a link-local or site-local address makes the zone, or area of the network, more specific. Microsoft uses the following example of a situation that requires the addition of zone IDs: "a computer with multiple Ethernet adapters that are connected to separate links," where each adapter has been assigned a link-local address. They explain, "Destination link-local addresses in this configuration are ambiguous because a specific link-local address can be assigned to multiple nodes located on the links" that can be reached by all the adapters. In a case like this, zone IDs are used "to indicate the Ethernet adapter over which traffic is sent and received." With a link-local address, the zone ID is equivalent to the interface index.

If you're using site-local addresses and are connected to multiple sites, each site is assigned a site identifier. In a case like this, use a zone ID to indicate the site identifier, further defining the area of the network intended as the destination of the traffic.

- Find the interface index using the following command:

      netsh interface ipv6 show interface

- Find the "Zone ID for Site" using the following command:

      netsh interface ipv6 show interface level=verbose

This command returns the default site identifier, 1, if you're only connected to a single site. No zone ID is needed in such a case.

When you enter the IPv6 address at a command prompt, affix the zone ID to the IPv6 address after a percent sign, as follows:

FE80::949:83FF:FE05:8EA5%3

In this case, the zone ID for the interface is 3.

Loopback Testing

The IPv6 standard as implemented on Windows protocol stacks specifies "`::1`" as a loopback address (it is the IPv6 equivalent of the IPv4 address `127.0.0.1` or `localhost`). Loopback testing with IPv6 only succeeds if you enter "`::1`" as the Endpoint 2 address. To run loopback tests with IPv6, enter a valid network address for E1 (the domain name, or the address in hex) and "`::1`" (loopback) as the E2 address.

Configuring Your Equipment for IPv6 Testing

Testing with the IPv6 protocols is supported on the endpoints listed in *Endpoint Support for IPv6* on page 5-10. Some advance configuration at the endpoints and on your routers may be needed for IPv6 testing. You will at least need to enable IPv6 on Windows or Linux, if you haven't already done so. See the following sections for more information:

- *Windows Configuration* below
- *Linux Configuration* on page 5-15

> **Note:** Make sure your endpoints are running at least version 4.5 of the Performance Endpoint software. The latest version supports the newest IPv6 testing capabilities.

Windows Configuration

Refer to your Windows product documentation for full instructions for installing and configuring IPv6. The following basic instructions should be sufficient to ensure that the IPv6 service is installed and operational.

Windows Server 2003

To install IPv6 in Windows Server 2003:

1. Open the Control Panel, then double-click Network Connections.
2. Right-click the network adapter on which you want to enable IPv6, then select Properties.
3. Click Install.
4. Select Protocol from the list of installation choices, then click Add.
5. Select Microsoft TCP/IP Version 6
6. Click OK.

Windows XP Professional

To install IPv6 on Windows XP Professional:

1. Open the Control Panel, click Network and Internet Connections, and then click Network Connections.

2. Right-click any local area connection, and then click Properties.

3. Click Install.

4. In the Select Network Component Type dialog box, click Protocol, and then click Add.

5. In the Select Network Protocol dialog box, click Microsoft TCP/IP version 6, and then click OK.

6. Click Close to save changes to your network connection.

Windows 2000

For Windows 2000, you must install a patch before you can enable IPv6. The patch requires that you have Service Pack 1 for Windows 2000 previously installed. Take these steps to install the patch and enable IPv6:

1. Navigate to the following Web site: http://msdn.microsoft.com/Downloads/ sdks/platform/tpipv6/readme.asp

2. After you've read the README for the "Microsoft IPv6 Technology Preview for Windows 2000 Network Protocol Stack," click **Download** in the left pane. When you click "**I agree**" to the license agreement, the self-extracting file is downloaded to your computer. Unzip the downloaded files.

3. In your Windows Explorer, navigate to the folder where you unzipped the files (the default location is C:\IPv6Kit). Double-click Setup.exe to launch the setup program.

4. Click **Start>Settings>Network and Dial-up Connections**.

5. Right-click the Ethernet-based connection to which you want to add the IPv6 protocol, and then click **Properties**. (You'll probably want to use **Local Area Connection**.) Click **Install**.

6. In the **Select Network Component Type** dialog box, click **Protocol**, and then click **Add**.

7. In the **Select Network Protocol** dialog box, click **Microsoft IPv6 Protocol.** Click **OK**.

8. Click **Close** to close the **Local Area Connection Properties** dialog box. The IPv6 protocol is now enabled on all your Ethernet interfaces.

Linux Configuration

Ixia officially supports IxChariot IPv6 testing on Red Hat Linux versions 8.0 and higher. IPv6 addresses can be used for test setup between Console and Endpoint 1 and between the endpoints.

To enable IPv6 support on Linux, you need to make a configuration change. Navigate to the /etc/sysconfig folder on each Linux endpoint computer. You'll need to modify the network file to add the following line:

```
NETWORKING_IPV6=yes
```

After this file is changed, restart the computer. The computer will then receive an IPv6 address from the router. To see the IPv6 address, run the following command:

```
/sbin/ifconfig
```

The output is as follows:

```
eth0    Link encap:Ethernet  HWaddr 00:60:97:7F:82:85
        inet addr:10.42.1.102  Bcast:10.42.1.255 Mask:255.255.255.0
        inet6 addr: fec0:a2a:100:0:260:97ff:fe7f:8285/64 Scope:Site
        inet6 addr: fe80::260:97ff:fe7f:8285/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:998768 errors:4 dropped:0 overruns:0 frame:4
        TX packets:270190 errors:0 dropped:0 overruns:0 carrier:0
        collisions:227 txqueuelen:100
        RX bytes:207268334 (197.6 Mb)  TX bytes:56687548 (54.0 Mb)
        Interrupt:9 Base address:0xff00
```

You can see the IPv4, IPv6 site, and IPv6 link addresses for each interface.

Router Configuration

If you are using Cisco routers, you'll need to ensure that the following software requirements are met:

• IOS 12.2(4)T1 or later
• PLUS software feature

You'll also need to enable IPv6 routing on your routers. Check your router documentation for information.

Creating and Running Tests

The topics in this section describe the procedures for using the IxChariot Console to interactively set up and run IxChariot tests:

Refer to *Getting Started with IxChariot* for step-by-step examples of some small IxChariot tests.

Test Execution Methods

You use IxChariot Console to create all tests. Once a test has been created, you can use any of the following methods to execute the test:

- Execute the test interactively in the IxChariot Console, as described in *Initiating Test Execution* on page 5-19.

- Execute the test using command line programs, as described in *Command-Line Programs* on page 5-67.

- Execute the test using the IxChariot Test Scheduler, as described in *Using Test Scheduler* on page 5-37.

- Schedule and execute the test using the Test Conductor application.

  Test Conductor is an Ixia application that supports the scheduling and execution of tests for a number of compatible Ixia applications, including IxChariot, IxLoad, and IxVoice. In Test Conductor, an *application configuration file* is the file used as the basis for running tests. In the case of IxChariot, the .tst files serve as the application configuration files. For detailed information, refer to the *Test Conductor User Guide*.

• Execute tests using TestComposer.

IxChariot provides a *TestComposer* plug-in that enables automation of complex, multi-step tests. The IxChariot plug-in exposes the basic IxChariot functionality, in support of the following activities:

• Load, save, and run tests of any kind.

• Create and edit tests that use regular, VoIP, and video pairs.

The TestComposer module is a software component integrated into the Ixia Test Conductor application. Refer to the *TestComposer User Guide* for detailed instructions for using the IxChariot plug-in for TestComposer.

Before Running a Test

Related Topics

Network Protocol Configuration on page 5-2

Before you run a test, the endpoint programs for the pairs in your test must be active, and likewise, so must their underlying network software. The following preparatory steps are recommended, prior to initiating a test run:

• Be sure the network software for the protocols you are using is configured and active at the Console and each endpoint in the test. It is probably best to start the network software when you power up the computer.

• Be sure the endpoint program is active on every endpoint participating in the test. When the endpoint program is running (and its output is visible), it shows whether it is successfully accessing the underlying network protocol.

• Save your test configuration. Test runs involve complex, extended interaction with multiple computers and network programs. Unexpected errors can cause problems and a run may not complete. Saving the test before the run lets you avoid repeating the selection and configuration of endpoint pairs and scripts.

• Set your desired run options. These are described in detail in *Run Options Tab* on page 7-2. (See also *Setting Options for Initialization Failure* on page 5-18.)

For more information about performance endpoints, refer to *Getting Started with IxChariot* and *IxChariot Performance Endpoints*.

Setting Options for Initialization Failure

Related Topics

Run Options Tab on page 7-2

When you start a test, IxChariot performs the test execution in two main phases (described in more detail in *Understanding the Run Status* on page 11-7):

1. Test initialization phase: IxChariot verifies network connectivity with all the endpoints participating in the test.

2. Test execution phase: IxChariot begins running the test and collecting the test data from endpoint 1.

You can set various run options for each test that you create, including the options that determine how IxChariot will respond to failures during the initialization phase and the test execution phase. These options allow you to:

- Decide whether or not IxChariot should stop the run if one or more endpoints cannot be contacted during the initialization phase.

- Specify that IxChariot should stop the test only if a specified number of pairs fail during test execution.

- Set reinitialization options for pairs that fail during the initialization phase. You can specify the number of times that IxChariot should attempt to reinitialize a pair, as well as the number of seconds to wait between reinitialization attempts.

- Set reinitialization options for pairs that fail during test execution. You can specify the number of times that IxChariot should attempt to reinitialize a pair, as well as the number of seconds to wait between reinitialization attempts.

The reinitialization feature provides flexibility in test planning and execution. When running large-scale tests (up to 100,000 pairs), it is often desirable—if not necessary—for a test to proceed even if some of the pairs fail during test initialization or test execution, and to allow IxChariot to reinitialize the pairs that fail. As another example, in many WLAN tests it is not uncommon for a pair to fail when the performance endpoint moves out of the coverage area of the access point (AP). By using appropriate reinitialization options, you can allow IxChariot to reinitialize the pairs that fail.

Initiating Test Execution

Related Topics

Stopping a Running Test on page 5-31
Using Test Scheduler on page 5-37

To run an IxChariot test, you can use the graphical user interface of the IxChariot Console program, use the command-line program named RUNTST, or use the IxChariot Test Scheduler. The underlying software is the same, so the test results from each are the same.

You will typically run smaller-sized tests using the IxChariot Console program. In a Test window, select **Run** from the Run menu to start a test, or click the Run button:

See *RUNTST: Running Tests* on page 5-67 for details on running a test from the command line. If you want to run tests of at least 5000 endpoint pairs, you should plan to use RUNTST.

Adding or Editing an Endpoint Pair

Related Topics

Creating and Running Tests on page 5-17
Adding or Editing a Multicast Group on page 5-27
Adding or Editing a VoIP Endpoint Pair on page 10-38
Adding or Editing a Video Endpoint Pair on page 10-54
Network Protocol on page 5-22

An endpoint pair includes the network addresses of the two computers, the protocol to use between them, the script they should run, and a service quality, if desired.

By default, Endpoint 1 executes a test script, collects results, and reports them to the Console over the same network segment. Depending on the size of your test or the reliability of your test network, you may want to isolate test setup and results traffic from actual test traffic; doing so can increase the accuracy of results or decrease the likelihood that setup data will be lost, causing the test to fail.

If you want to use an alternate network for setup/results reporting, or a different network protocol or service quality between the Console and E1 and between the endpoints, fill in the values for *Use Endpoint 1 values from pair* and *Use Endpoint 2 address as management address*. For example, you might use TCP to connect from the Console to E1, yet run a UDP test between E1 and E2. But keep in mind that IxChariot only uses connection-oriented protocols for setup communications between the Console and E1, and that TCP is required for setup communications between E1 and E2.

To use alternate networks, enter a different network address at which the Console will "know" E1 so that you can use separate networks for setup communications between the Console and E1 and for actual test traffic.

- **Pair Comment (optional)**

    A descriptive word or phrase that lets you easily identify each pair in the Test window.

- **Endpoint 1 to Endpoint 2 Traffic**

    - **Endpoint 1 address**

        A computer playing the role of Endpoint 1 in an IxChariot test. The IxChariot Console contacts Endpoint 1 computers directly, sending them the application script and test setup information you entered. Endpoint 1 acts something like a client computer in a client-server network. Application traffic flows from Endpoint 1 to Endpoint 2 and back.

    - **Endpoint 2 address**

        A computer playing the role of Endpoint 2 in an IxChariot test. Endpoint 1 computers contact the Endpoint 2 computers with test setup information. Endpoint 2 acts something like a server in a client-server network.

        - **For IPX or SPX**, enter an IPX address in hexadecimal format or enter its alias. An example of an IPX address in hex format is `03F2E410:0A024F32ED02`. The first 8 digits (4 bytes) are the network number; the 12 digits (6 bytes) following the colon are the node ID (you may also hear these referred to as the *network address* and *node address*). But no one wants to enter hex numbers like that more than once. IxChariot lets you associate aliases with these addresses. See *IPX/SPX Aliases* on page 5-3.

        - **For RTP, TCP or UDP**, enter an a hostname or an IPv4 or IPv6 address in standard notation. An example of domain name format is

www.ixiacom.com, while an example of and IPv4 address is `199.72.46.202.`, and an example of an IPv6 address is `2001::2.`

- **Network Protocol**

  The protocol the endpoints will use as they execute the commands in the test script. When the test is run, the protocols you select for all pairs must be correctly configured and started on the endpoint computers. They also must be appropriate for the type of work being performed in the script. For example, if you selected a streaming script, you must select IPX, RTP, or UDP. See *Network Protocol* on page 5-22 for more information.

- **Service Quality (optional)**

  If a service quality is required by the network protocol, enter or select a value defined on the endpoint computers. Any quality of service templates you've configured are available in the list. The service quality values you enter are remembered in the file `servqual.dat.` Service quality may be selected for both TCP-IPv4 and TCP-IPv6 traffic. Refer to Chapter 9, *Quality of Service Testing* for detailed information.

- **Select Script**

  Lets you browse IxChariot's application script library. Six broad categories of scripts are available to emulate a wide range of applications. See the *Application Scripts* guide for information on scripts. You must select a script to complete the endpoint pair definition.

- **Edit this Script**

  Opens the Script Editor so that you can tailor the application script to emulate your unique environment and testing requirements. You must first select a script; see "Select Script," above.

The remaining fields in this dialog are visible by pressing the **Management >>** button.

- **Console to Endpoint 1 Management**

  - **Use Endpoint 1 address as management address**

    The Console will communicate with Endpoint 1 using the network address and protocol for E1 specified in the Endpoint 1 to Endpoint 2 Traffic box. If the Console needs to send test setup information, such as the address of E2, to E1 through an alternate network, clear the box and specify a setup address. Or clear the box to select a different protocol. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1. For example, if you choose the IPX protocol for the test, the default is to use SPX between the Console and Endpoint 1. For RTP and UDP, the default is TCP.

    The field below the checkbox holds the network address where the Console can contact E1. It lets you choose a different address for test setup and results flows between the Console and E1 than the endpoint address specified in the pair itself. Endpoint computers can have multiple network addresses. For example, a computer with multiple adapters may have multiple IP addresses.

> > > - If you are changing the network address at which E1 contacts E2 with setup information (see **Endpoint 1 to Endpoint 2 Management** below), you'll probably need to change the address at which the Console contacts E1 as well.
> > >
> > > - In tests running streaming scripts (and in VoIP tests), be aware that E2 sends results back to E1. If the network between E2 and E1 may become saturated due to the operation of the test and an alternate network and address for E1 is available, you should specify that here.
> >
> > - **Management Protocol**
> >
> >   The protocol that the Console will use to contact Endpoint 1 for the purpose of setup.
>
> - **Endpoint 1 to Endpoint 2 Management**
>
>   - **Use Endpoint 1 address as management address**
>
>   The address and protocol used for test setup and results flows will be the same as those you specified when you created the endpoint pair. If E1 needs to send setup information, such as the application script, to E2 through an alternate network, clear the box and specify a setup address. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to E1. If you're using IPv6 and this box is checked, E1 uses TCP for IPv6 for this connection. If you clear this box and supply an alternate address, TCP for IPv4 is used.
>
>   The field below the checkbox holds the network address where E1 can contact E2. It lets you enter a different address for setup and results flows between the endpoints than is specified for test traffic in the pair itself. Endpoint computers can have multiple network addresses. Often, a more reliable network connection can be made between the endpoints at different addresses to ensure that setup and reporting information is received. In tests running streaming scripts (and in VoIP tests), E2 sends results back to E1. If the network is unreliable, you should specify a more reliable address for E2 here so that the endpoints use a reliable network for test setup and reporting.

As you enter the network addresses of endpoints, IxChariot stores them in the lists. The names are saved in a file named `endpoint.dat`, which you can edit with an ASCII text editor.

Do not enter an IP Multicast address in the **Endpoint 1** or **Endpoint 2 address** fields. If you enter an IP Multicast address in either of these fields, the test fails with error **CHR0209**. See *Adding or Editing a Multicast Group* on page 5-27 for instructions on setting up these groups properly.

Network Protocol

When selecting the protocol the endpoints will use as they execute the commands in the test script, be aware of the following:

- Once the test is run, the protocols you select for all pairs must be correctly configured and started on the endpoint computers.

- Protocols also must be appropriate for the type of work being performed in the script. For example, if you selected a streaming script, you must select IPX, RTP, or UDP.

Consult the *Application Scripts* guide for more information about how the scripts work, and which types of scripts are available.

You have the option to select TCP, UDP, or RTP for IPv6 for your tests. Refer to *IPv6 Configuration and Testing* on page 5-10 for more information.

Confirmation Messages for Endpoint Pair Edits

When you edit an endpoint pair, IxChariot may present one or more confirmation messages before saving the pair. IxChariot displays a confirmation for the following conditions:

- The Endpoint 1 address differs from the Console-to-Endpoint 1 management address. This is simply an informational message. If you are using an address for test setup and results flows between the Console and E1 that is different from the endpoint address specified in the pair itself, simply select Yes and continue with your test preparations.

- You have selected an endpoint IP address defined in the Ixia network configuration. This is simply an informational message to inform you that IxChariot has assigned management network settings based on the settings found in the Ixia network configuration. Refer to *Selecting the Ixia Ports for a Test* on page 4-17 for more information.

- You have selected alternate management address settings that do not match the addresses found in the Ixia network configuration. This may indicate an incorrect configuration setting.

In each case, you can select Yes to proceed with your test setup and preparation, or select No to remain in the Edit an Endpoint Pair dialog.

Use of IPv6 addresses for Endpoint 1

An IPv6 address may be used for Endpoint 1's management address.

Adding or Editing a Hardware Performance Pair

Related Topics
Creating and Running Tests on page 5-17
HPP Defaults Tab on page 6-23
Examining Timing Records on page 11-3

A hardware endpoint pair utilizes a pair of Ixia test ports as endpoints. It includes the network and management addresses of the two computers, and an Ixia stream to used to generate background traffic.

> **Restriction:** Hardware Performance Pairs and VoIP Hardware Performance Pairs support neither PPP nor IPSec. Although IxChariot allows you to select port groups that have PPP and IPSec addresses, the streams that IxChariot sends out will not have PPP or IPSec headers. Only the IP addresses that you specify in the PPP and IPSec Stack Manager plug-ins will be used.

The *Add a Hardware Performance Pair* and *Edit a Hardware Performance Pair* dialogs contains the following parameters:

• **Port 1 and Port 2 management addresses**

Ixia ports each have an IPv4 management address by which the IxChariot Console controls the hardware. This management address is of the form:

    <base octet 1>.<base octet 2>.<card #>.<port #>

    • The first two octets are associated with the *base address* of the Ixia chassis, which is normally 10.0.0.0. When two or more chassis are used in an IxChariot test, all of the chassis need to have different base addresses. These base addresses may be observed and modified using Stack Manager.

    • The third octet is the card number for the port.

    • The fourth octet is the port number on the card.

The management addresses for both ports must be entered.

• **Port 1 network address**

Port 1 will be used to generate unidirectional network traffic destined for Port 2. The network address is used in the establishment of the source address for the traffic. The network address may be an IPv4 or IPv6 address.

• **Port 2 network address**

Port 2 will be used to receive the unidirectional network traffic sent from Port 1. The network address is used in the establishment of the destination address for the traffic. The network address may be an IPv4 or IPv6 address.

• **Select Stream**

Lets you browse a default file directory which holds Ixia port streams prepared through the use of IxExplorer, ScriptMate, or the TCL or C++ API.

The default directory is configured using the *File..Change User Settings...Directories* dialog, under the *Where to read hardware performance stream files:* setting.

The defaults for this and the next two settings is configured using the *File..Change User Settings...Hardware Performance Pair Defaults* dialog.

A number of VoIP and non-VoIP streams are pre-packaged with IxChariot. Refer to the *Ixia Streams* chapter in the *IxChariot Application Scripts* manual for details.

This script is applied by Port 1.

• **Override stream line rate**

Hardware pairs are capable of operating at hardware line rates which may overwhelm DUTs. The line rate specification allows you to optionally set the the data rate from the hardware ports to be controlled. If this option is not selected, the stream rate programmed into the selected stream is applied.

- **Measure hardware performance pair statistics and set filters**

  Ixia ports are capable of obtaining performance statistics and applying filters to incoming network traffic. Select one of the following options:

  - Select *Measure statistics and set filters* if you wish to measure additional Ixia-specific statistics (including latency) and also set hardware filters.

  - Select *Do not measure statistics and set filters* if you do not wish to measure additional Ixia-specific statistics and do not wish to set hardware filters.

  - Select *Set hardware filters only* if you wish to set hardware filters but do not wish to measure additional Ixia-specific statistics.

  Ixia ports have the ability to filter network traffic received on the port. Refer to the *Stack Manager User Guide* for more information.

  Ixia ports are capable of obtaining performance statistics. See *Examining Timing Records* on page 11-3. If you choose not to measure performance statistics, the port pair will only provide background traffic.

- **Pair Comment (optional)**

  A descriptive word or phrase that lets you easily identify each pair in the Test window.

Cloning Hardware Performance Pairs

Related Topics

Adding or Editing a Hardware Performance Pair on page 5-23

The process of cloning a hardware performance pair involves using the addresses of a non hardware performance pair in order to create a hardware performance pair. This can be done by selecting a non hardware performance pair in the Test Setup window and then selecting the *Edit ... Clone Hardware Performance Pair*. This will cause the *Clone Hardware Performance Pair* dialog to be displayed. Its contents and operation are as described in *Adding or Editing a Hardware Performance Pair* on page 5-23.

Replicating Pairs

Related Topics

Adding or Editing an Endpoint Pair on page 5-19

Replicating pairs you've already created can be a time saver. Highlight a pair, or multiple pairs, that you want to replicate and click **Replicate** on the Edit menu. If you highlight both a pair and a multicast group, the **Replicate** menu item is not available: you must replicate pairs and multicast groups separately.

- **Replication Count**

  Lets you specify the number of copies to make of the highlighted pair(s). The field defaults to a value of 1, which indicates that one copy will be made of each of the highlighted pair(s). To specify a number greater than one, enter the desired number of copies in the Replication Count field or use the up and down arrow keys to set the Replication Count field to the desired number. You are limited to the total number of pairs allowed by your IxChariot license.

After you have specified the number, click **OK** to add the replicated pair(s) to the end of the Test window. If you decide not to replicate the selected pair(s), click **Cancel**.

Sorting and Grouping Pairs

You can reorganize the way pairs are shown in the Test window according to a series of different criteria. Clicking **Sort** on the View menu opens the Sort dialog box, where you specify the first, second, and third criteria by which the pairs will be sorted:

*   **Sort by**

    Specifies the first sorting criterion. The list contains the parameters applicable for sorting the test data on the tab now shown in the Test window.

*   **Ascending**

    Sorts the data in order from smallest to largest values, or in alphabetical order if host names or group names are used for the endpoints.

*   **Descending**

    Sorts the data in order from largest to smallest values, or in reverse alphabetical order if host names or group names are used for the endpoints.

*   **Then by**

    Specifies the second and third sorting criteria.

The View menu also provides options to group the endpoint pairs and sort the groups:

*   **Group By**

    The Group By feature lets you organize the tests into groups, based on a selected criteria (such as grouping by Pair Comment). Once pairs are grouped, their results appear in these groups when you print or export. By default, the no-grouping category "All Pairs" is used. You can group the test pairs using any of the following criteria:

    *   Network Protocol – Group the pairs by network protocol.

    *   Script Filename – Group the pairs by the script filenames used in the test. For VoIP and Video pairs, the grouping will be based on the codec used.

    *   Endpoint 1 – Group the pairs by the Endpoint 1 IP address or port group name.

    *   Endpoint 2 – Group the pairs by the Endpoint 2 IP address or port group name.

    *   Service Quality – Group the pairs by the QoS templates used in the test.

    *   Pair Group Name – Group the pairs by pair group names.

    *   Pair Comment – Group the pairs by pair comment.

    *   Console to E1 Address – Group the pairs by the IP addresses specified in the *Console to Endpoint 1 Management* field of the test definition dialogs.

    *   E1 to E2 Address – Group the pairs by the IP addresses specified in the *Endpoint 1 to Endpoint 2 Management* field of the test definition dialogs.

    *   Run Status – Group by run status.

- Ixia Port on E1 – Group the pairs by the Endpoint 1 Ixia port addresses (these are IP address/card/port designations, such as 10.200.15.69/01/01). This is applicable only when you are using Ixia ports as endpoints, and you have assigned port group names (rather than specific addresses) to the test pair endpoints. When you have a test with many pairs allocated by DHCP, you can use this grouping option to easily identify which ports were used to allocate the IP addresses.

- Ixia Port on E2 – Group the pairs by the Endpoint 2 Ixia port addresses (these are IP address/card/port designations, such as 10.200.15.69/01/01). This is applicable only when you are using Ixia ports as endpoints, and you have assigned port group names (rather than specific addresses) to the test pair endpoints. When you have a test with many pairs allocated by DHCP, you can use this grouping option to easily identify which ports were used to allocate the IP addresses.

- **Group Sort Order**

    When you have endpoint pairs organized in groups, this option lets you sort the groups of pairs in either ascending or descending order.

Adding or Editing a Multicast Group

Related Topics

General information about IP Multicast is provided in *IP Multicast Testing* on page 10-9. To create a test emulating a multicast application, first create a multicast group; click **Add Multicast Group** on the Edit menu. Multicast group members are the Endpoint 2 computers designated as receivers of data from Endpoint 1 in IxChariot tests. When paired with the sending (Endpoint 1) computer, they are called *multicast pairs*. The procedure is similar for editing an existing multicast group.

By default, Endpoint 1 executes a test script, collects results, and reports them to the Console over the same network segment. Depending on the size of your test or the reliability of your test network, you may want to isolate test setup and results traffic from actual test traffic; doing so can increase the accuracy of results or decrease the likelihood that setup data will be lost, causing the test to fail.

If you want to use an alternate network for setup/results reporting, or a different network protocol or service quality between the Console and E1 and between the endpoints, press the **Edit Pair Setup >>** button and fill in the value for *Use Endpoint 1 address as management address*. The address by which endpoint 1 knows each of the multicast group members is set up by selecting the group member from the list and then pressing the **Edit Member Setup...** button. In the resulting dialog, fill in the values for *Use Endpoint 2 address as management address*

For example, you might use TCP to connect from the Console to E1, yet run a UDP test between E1 and E2. But keep in mind that IxChariot only uses connection-oriented protocols for setup communications between the Console and E1, and that TCP is required for setup communications between E1 and E2.

To use alternate networks, enter a different network address at which the Console will "know" E1 so that you can use separate networks for setup communications between the Console and E1 and for actual test traffic.

- **Group Name**

  A name that is unique within the test. If you do not enter a group name in this field, IxChariot creates a group name that is a combination of the IP address and port of the multicast group.

- **Group Comment**

  A descriptive word or phrase that helps you easily identify each multicast group in the Test window. Optional field.

- **Endpoint 1 to Multicast Group**

  - **Multicast Address**

    For IPv4, the Class D IP address to use for the IP Multicast test. Valid IP Multicast addresses are 224.0.0.0 through 239.255.255.255. We recommend using addresses beginning with 225.0.0.0 or higher because many addresses beginning with 224 are reserved for router usage.

    For IPv6, multicast addresses have a prefix of FF00::/8. Within the reserved multicast address range of FF00:: to FF0F::, the addresses listed in Table 5-3 are assigned to identify specific functions:

Table 5-3.   IPv6 Multicast Address Usage

Range	Usage
FF01::1	All nodes within the node-local scope.
FF02::1	All nodes on the local link.
FF01::2	All routers within the node-local scope.
FF02::2	All routers on the link-local scope.
FF05::2	All routers in the site.
FF02::1:FFXX:XXXX	Solicited-node multicast address, where XX:XXXX represents the last 24 bits of the IPv6 address of the node.

Refer to *Endpoint Support for IPv6* on page 5-10 for a list of endpoints that support IPv6.

Although IxChariot verifies that the IP Multicast address you enter is within the required range, it does not verify the reserved/unreserved status of the address you enter. See *Emulating IP Multicast Applications* on page 10-9 for more information on multicast addresses.

- **Multicast Port**

  The port number that the multicast group will use. The multicast port must uniquely identify the multicast group. You can enter values in the range from 1 to 65535. Avoid using well-known port numbers.

- **Endpoint 1 address**

  The "From" address for this multicast group. Enter an IP address, a hostname or an address in dotted notation. An example is 199.72.46.202. Endpoint 1 acts as the multicast sender.

- **Use Endpoint 1 address as management address**

  The Console will communicate with Endpoint 1 using the network address and protocol for E1 specified in the pair. If the Console needs to send test setup information, such as the address of E2, to E1 through an alternate network, clear the box and specify a setup address. Or clear the box to select a different protocol. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1. For example, if you choose the IPX protocol for the test, the default is to use SPX between the Console and Endpoint 1. For RTP and UDP, the default is TCP.

  The field below the check box holds the network address where the Console can contact E1. Lets you choose a different address for test setup and results flows between the Console and E1 than the endpoint address specified in the pair itself. Endpoint computers can have multiple network addresses. For example, a computer with multiple adapters may have multiple IP addresses.

  - If you are changing the network address at which E1 contacts E2 with setup information (see "**Use Endpoint 2 address as management address**," below), you'll probably need to change the address at which the Console contacts E1 as well.

  - In tests running streaming scripts (and in VoIP tests), be aware that E2 sends results back to E1. If the network is unreliable, you should specify a more reliable address for E1 here. It will also be used by E2 for results flows.

- **Multicast Group Members**

  Type an IP address (a hostname or address in dotted notation) or select one from the list. Then click **Add** to add the endpoint to the multicast group. To delete a multicast group member, select the group member in the list and click **Delete**.

- **Use Endpoint 2 address as management address**

  The network address and protocol used for test setup and results flows will be the same as those you specified when you created the endpoint pair. If E1 needs to send setup information, such as the application script, to E2 through an alternate network, clear the box and specify a setup address. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to E1. if you're using IPv6 and this box is checked, E1 uses TCP for IPv6 for this connection. If you clear this box and supply an alternate address, TCP for IPv4 is used.

The field below the checkbox holds the network address where E1 can contact E2. Lets you enter a different address for setup and results flows between the endpoints than is specified for test traffic in the pair itself. Endpoint computers can have multiple network addresses. Often, a more reliable network connection can be made between the endpoints at different addresses to ensure that setup and reporting information is received. In tests running streaming scripts (and in VoIP tests), E2 sends results back to E1. If the network is unreliable, you should specify a more reliable address for E2 here so that the endpoints use a reliable network for test setup and reporting.

• **Network Protocol**

The protocol to use when sending the IP Multicast data. Choose RTP or UDP. IP Multicast runs only over these protocols.

• **Service Quality (optional)**

The quality of service (QoS) to use in this IP Multicast test. If you've defined QoS templates, select one from the list.

• **Select Script**

Lets you select a streaming script for this multicast group. See "Streaming Scripts" in the *Application Scripts* guide for a list of available streaming scripts.

• **Edit this Script**

Lets you edit the script you've selected for the IP Multicast test. See the *Application Scripts* guide for information on editing streaming scripts.

Multicast Script Selection

Related Topics
Adding or Editing a Multicast Group on page 5-27

You can use a streaming script with a multicast group.

Replicating a Multicast Group

Related Topics
Replicating Pairs on page 5-25

Highlight the multicast group you want to replicate and click **Replicate** on the Edit menu to get the Replicate a Multicast Group dialog box. Only a single multicast group can be replicated at a time. If you highlight a combination of endpoint pairs and multicast groups, the Replicate menu item is not available. You must replicate pairs and a multicast group separately.

The dialog box shows the information for the multicast group that you highlighted, except for the **Group Name**, **Group Comment**, and **Multicast Port** fields. These fields are blank because they must be unique. Enter the information and modify any other fields that you want to change. See *Adding or Editing a Multicast Group* on page 5-27 for more information.

Stopping a Running Test

Related Topics

Understanding the Run Status on page 11-7

Sometimes your test may appear to be taking too much time to go from "Initializing" status to "Running" status. Usually, this indicates an error at the endpoints, or in your test configuration.

When you stop a test, the Console sends a message to Endpoint 1 of each pair, directing them to stop executing their scripts and to return any completed timing records that they haven't yet sent. Stopping a test can be accomplished quickly, except when one or more of the following occurs:

- A script is sending a large amount of data inside a transaction. The endpoint does not stop until it reaches either an `END_LOOP` or `END_TIMER` command.

- There are errors at an endpoint.

- There is excessive network congestion.

For either of these last two situations, you may wait indefinitely.

When you stop a test, IxChariot shows a dialog box with the progress: how many seconds have elapsed since you chose to stop. If you think you've waited far too long for a test to stop, click **Abandon Run**, which appears after 10 seconds. The Console has asked the Endpoint 1 computers to stop, but they haven't yet returned all their timing records.

Abandon Run is a severe action, to be taken rarely. It is possible that some endpoints are still finishing the execution of a script, or that they are trying to send their timing records to the Console. If you've abandoned a run, wait several minutes before starting another test to the same endpoints; on a subsequent run, if IxChariot encounters endpoint errors or an endpoint does not respond, you may need to restart each of the endpoints. The easiest, but most drastic, means of restarting the endpoint is to reboot the system that is running the endpoint. Otherwise, consult the *Stopping and Starting* subsections of each chapter of the *Performance Endpoints* guide for your endpoint types for instructions on how to stop and restart endpoints.

When a pair fails, IxChariot informs all the other pairs to stop after their initialization step unless you have specified on the Run Options for IxChariot not to do so. If your test appears to be "hung" in the "Initializing" state after a pair fails, it may because there is a network problem that keeps IxChariot from detecting the failure. In this case, you may need to stop the test manually. However, if you have selected the *Allow pair reinitialization* run option, IxChariot may require additional time to attempt to reinitialize any pairs that have failed. The amount of time depends on the number of pairs that IxChariot is attempting to reinitialize, and the reinitialization options that you have set. Refer to *Error Handling Tab* on page 7-14 for more information about options for handling failed pairs.

You can stop a running test by clicking **Stop** on the Run menu, or simply closing the Test window.

Running a
Traceroute

Related Topics

Traceroute Tab on page 6-35

To see information about the route data is taking between two endpoints, high-light a pair in the Test window and click **Run Traceroute** on the Run menu. Click **Run** to start the traceroute. Traceroutes may only be performed on one pair at a time.

Traceroute testing can only occur if Endpoint 1 in the selected pair is running on Windows (Windows 95 must have WinSock 2 loaded), HP-UX, IBM AIX, Linux, Sun Solaris for SPARC and x86, and Windows NT for DEC Alpha. Trac-eroute testing on some of these operating systems was not supported by versions earlier than 4.0 of the endpoint software. In addition, IxChariot only collects trac-eroute information for TCP/IP connections. A traceroute cannot be performed while a test is running, but it can be performed before or after a test run.

During a traceroute, the IxChariot Console initiates an ICMP (Internet Control Message Protocol) echo message that travels from the "**Source**" (Endpoint 1) to the first router, which sends it back. The time-to-live value on the message header is changed after each hop, and a record is kept of how long the message took to reach the "**Target**" and report back to the Source. Each time the datagram encounters a router, it sends information back to the Source.

If the message does not reach the Target within the timeout duration and number of router hops you specified during traceroute configuration, it is abandoned. Running a traceroute is therefore a good way to test a connection on your net-work or check a router that you suspect may be down.

• **Status**

  The status of the traceroute test. If an error occurs, the status is given as "Fin-ished with Error(s)."

• **Finished with Error(s)**

  Indicates that an error occurred during a traceroute.

• **Help for Message CHR***n*

  Provides an explanation of a traceroute error and advises how to avoid it.

• **Hop Count**

  The total number of hops encountered between the Source computer and the Target.

• **Hop Latency**

  Refers to the amount of time it took the ICMP message to clear a specific hop, in milliseconds.

• **Address**

  The IP network address of the hop.

• **Name**

  The resolved DNS name of a hop. By default, IxChariot resolves the numeric IP addresses of traceroute hops into DNS names. You can change this setting

on the Traceroute tab in the Change User Settings notebook to avoid adding latency to your time values.

• **** in Results Data**

Indicates that traceroute data was unavailable for that hop. Data may be unavailable because a firewall is blocking the ICMP message, because the message timed out, or because connectivity was lost.

• **HTML Results**

Shows your traceroute results formatted in HTML. The HTML report shows * if no data is available for a particular hop.

If you run a traceroute more than once between the same pair of endpoints, you may see some differences in the results. These differences may be the result of router behavior, or more specifically, of redirect instructions issued by the router. Routes can be changed for various reasons, but they are usually added to the routing table. You may see inconsistent results for a traced route if the route changes while the traceroute test is running. Therefore, you should run a test between the endpoints before running a traceroute. This prompts the router to update the routing table with the redirected route, which means that you see consistent results.

You must choose a pre-existing endpoint pair to run a traceroute from the Test window. To run a traceroute between any two computers, select **Run Trace-route** from the Tools menu.

Configure traceroute options in the Change User Settings notebook; see *Traceroute Tab* on page 6-35 for more information.

Bandwidth Considerations

Related Topics
Test Setup on page 5-33
Test Results on page 5-34

In some situations, you may need to consider the amount of data not directly related to the scripts used in an IxChariot test that will nevertheless be sent on your network during a test. This additional data includes the data required to set up or initialize a test and the data required to receive the results.

The following related topics provide data sizes for planning purposes. These sizes are for protocol payload only and do not include the overhead associated with the protocol or the transport. The actual sizes vary, depending on the end-point platform and number of script commands.

The numbers provided assume a Windows NT endpoint using the simple script Filesndl. The values are calculated for a single pair and should be multiplied by the number of pairs in your test.

Test Setup

Test setup creates data flows between the Console and Endpoint 1 and between Endpoint 1 and Endpoint 2. The sizes of these flows are summarized as follows:

Table 5-4.   Flow size summary

Test Setup Flows	Size
Between Console and Endpoint 1	1500 bytes
Between Endpoint 1 and Endpoint 2	1000 bytes

Test Results

Getting the results of the test involves sending timing records from Endpoint 1 to the Console. In a streaming test, these results are first sent from Endpoint 2 to Endpoint 1. The size of the timing record depends on the protocol used.

An additional overhead is associated with the results; this overhead is applied once per 500 timing records when batch reporting is selected (once per 300 timing records with VoIP pairs), or once per timing record when real-time reporting is selected.

To determine the full bandwidth required, multiply the size of the timing record by the number of timing records.

Table 5-5.   Bandwidth Requirements

Version	Timing Record Overhead	TCP and SPX pairs	UDP, IPX (non-streaming)	UDP, IPX (streaming)	RTP or VoIP pairs
Version 6.0+ with 6.0+ endpoints	20 bytes	22 bytes	46 bytes	84 bytes	132 bytes
Version 5.0, 5.20 and 5.40	20 bytes	20 bytes	44 bytes	82 bytes	130 bytes

For example, for IxChariot 6.0+ with 6.0+ endpoints consider the following test configuration:

*   Reporting type = BATCH

*   Protocol = UDP

*   # of timing records = 800 (2 batches, or 3 batches for VoIP pairs)

The bandwidth required is calculated as follows:

(20 + (500 * 46)) + (20 + (300 * 46)) = 36,840 bytes.
Multiply by 300 instead of 500 for VoIP pairs.

Creating Application Group Tests

IxChariot application groups enable the definition of tests comprising multiple pairs that:

- start and stop independent of one another, and

- are logically linked, such that an event in one connection triggers the suspension or resumption of execution in another connection.

Tests of this nature are referred to as *synchronized pair* tests. IxChariot supports synchronized pair testing for Regular pairs and VoIP pairs.

Application Groups Delivered with IxChariot

IxChariot provides a set of prebuilt application groups, each of which emulates an application that employs multiple, synchronized connections. These files are located in the following folders:

```
C:\Program Files\Ixia\IxChariot\Application Groups\
C:\Program Files\Ixia\IxChariot\Scripts\Gaming\
```

You can use these application groups to create synchronized pair tests.

Creating a Test from an Application Group

The procedure for creating a synchronized pair test differs from that of tests using individual (non-synchronized) pairs. Unlike individual scripts—which you select as part of a test definition—you must first *import* an application group into a test window. Once you have done that, the procedures for creating the test are nearly identical to any other test.

Importing an Application Group

To import an application group into a Test window:

1. Open a new Test window.

2. Select **File** > **Application Group** > **Import**.

3. Select the desired application group file (.iag file).

4. Click **Import**.

IxChariot opens the endpoint pairs defined in the application group.

Creating and Saving the Test

Once you import an application group into a Test window, the remaining procedures are similar to those of any other type of test. Typically, you will perform these actions to create and execute your test:

1. Assign addresses to the endpoint pairs, using the procedure defined in *Replacing Host Addresses in an Application Group* on page 5-36.

2. Make any desired modifications to the test scripts (such as modifying script variables).

3. Execute the test.

4. Save the test.

IxChariot saves the test as a .tst file. This test file operates and behaves just like any other test file: it is neither dependent upon, nor linked in any way, to the application group file from which you created it.

Replacing Host Addresses in an Application Group

The application groups delivered with IxChariot (as well as those that you create with IxProfile) use symbolic names for the endpoints in the group. For example, the SIP-VoIP application group uses endpoints names such as *caller* and *callee*. Therefore, when you import one of these application groups, you need to replace the symbolic names with the actual addresses that you are using in your test. IxChariot provides the Search and Replace Addresses dialog for this purpose.

To search for and replace addresses in an application group, follow these steps:

1. Click the **PG** button to display the application group names in the Group column of the Test window.

2. Select the desired application group.

3. Right-click the mouse to display the pop-up menu.

4. Select **Application Group** > **Search and Replace Addresses...** from the pop-up menu.

   IxChariot opens the Search and Replace Addresses dialog. Test addresses and management addresses are listed separately.

5. Select an address that you want to replace (a test or a management address).

6. Click **Modify**.

   IxChariot opens the Change Address dialog.

7. Enter the replacement address.

8. Click **OK** to close the Change Address dialog.

9. Click **OK** to close the Search and Replace Addresses dialog.

IxChariot locates each instance of the address and replaces it with the new address that you entered. The replacement is made in each of the pairs contained in the application group.

For Detailed Information

For detailed information about application groups:

- Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about creating and editing application groups.

- Refer to the *IxChariot Scripts and Streams Reference Guide* for detailed information about the application groups that are provided with IxChariot.

Using Test Scheduler

Related Topics

Before Running a Test on page 5-18
Setting Options for Initialization Failure on page 5-18
Initiating Test Execution on page 5-19

The IxChariot Test Scheduler provides an alternative to initiating a test run in real time. With Test Scheduler, you can:

• Set a date and time to start execution of a pre-configured IxChariot test.

• Save multiple iterations of the test results in separate, uniquely-named, files.

• Specify a recurrence pattern for tests that need to run repeatedly on a specific schedule.

The IxChariot Test Scheduler uses a visual calendar interface for all of its scheduling activities.

Scheduling a Test

To schedule a test (an IxChariot .tst file) to run on specific dates and times:

1. Start IxChariot Console.

2. Select **Run Test Scheduler** from the Tools menu.

   IxChariot opens the Test Scheduler calendar, as shown in Figure 5-4.

   You can also launch Test Scheduler directly from the Windows Start menu, rather than starting it from the IxChariot Console. The menu selection is Start > Programs > IxChariot > Test Scheduler.

Figure 5-4.   Test Scheduler Calendar
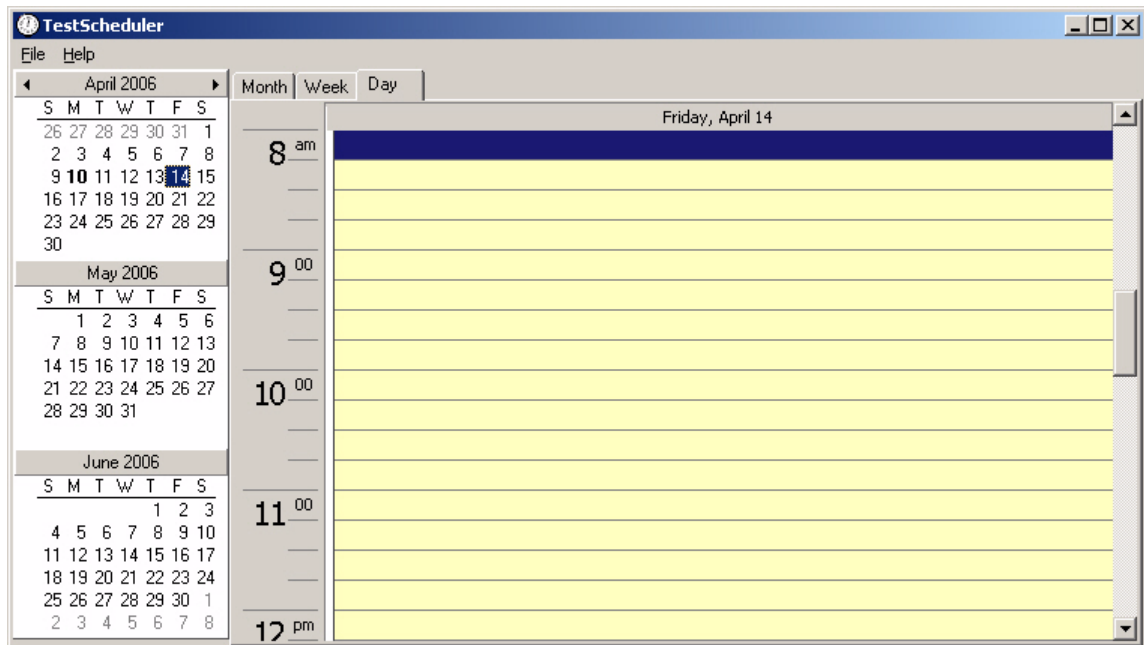
3. Select the desired calendar view: Month, Week, or Day.

4. Double-click an open time slot. This is the date and time on which you want to start the execution of a test.

   Test Scheduler opens the Test Scheduling window (an example of which is shown in Figure 5-5 on page 5-39).

5. Select the test that you want to schedule, and specify the desired test execution schedule, as described in Table 5-6.

Table 5-6.   Test Scheduling Parameters

Parameter	Description
General Tab:	
Test file	Specify the complete path to the test file (the IxChariot .tst file) that you are scheduling. You can use the Browse button to select the file.
Description	Optionally, specify a name that will appear in the calendar. By default, Test Scheduler uses the test name as the description.
Scheduling time	Specify when you want the test to start: • Select **Start immediately** if you want the test to start as soon as you click OK. • If you are scheduling a test that has the "Run for a fixed duration" run option set, specify the **Start** time. In this case, the End time and Estimated Duration fields are not available. • If you are scheduling a test that has not set the "Run for a fixed duration" run option, specify the **Start** time and either of the following: – the **End** time, or – the **Estimated duration**. When you specify the End time, the Test Scheduler adjusts the Estimated Duration accordingly. If you specify the Estimated Duration, the Test Scheduler adjusts the End time accordingly.
Completion actions	Optionally, specify the complete path for an application that will execute upon completion of the test. You can specify a different application for successful and unsuccessful test executions. If the path includes spaces, you need to enclose it within double-quote characters. Select the **Wait** checkbox if you want a waiting test to start execution only after the designated application has completed its execution.

Table 5-6.   Test Scheduling Parameters (Continued)

Parameter	Description
Copy test file	To save a copy of the test results in a directory other than your default Tests directory: • Select the **Copy test file** checkbox. • Specify the folder in which you wish to place the copy of the test results. You can use the Browse button to select the folder. Test Scheduler saves the test results to a file using the original test name with the date and time appended. This ensures a unique file name for each iteration of the test.
Recurrence Tab:	
Recurrence pattern	Specify a Recurrence Pattern: • One time only - No recurrence. • Daily - The test repeats daily at the time specified in the **Schedule on** field. • Weekly - The test repeats on a weekly basis. When you select Weekly, you need to specify the specific pattern: every *n* weeks, running on specific days of the week.
Range of Recurrence	If the recurrence pattern is Daily or Weekly, specify the Start date and the end conditions. You can specify No End Date, End After a specified number of completions, or End By a specified date.

Figure 5-5 shows an example of the Test Scheduling window.

Figure 5-5.   Test Scheduling Window Example
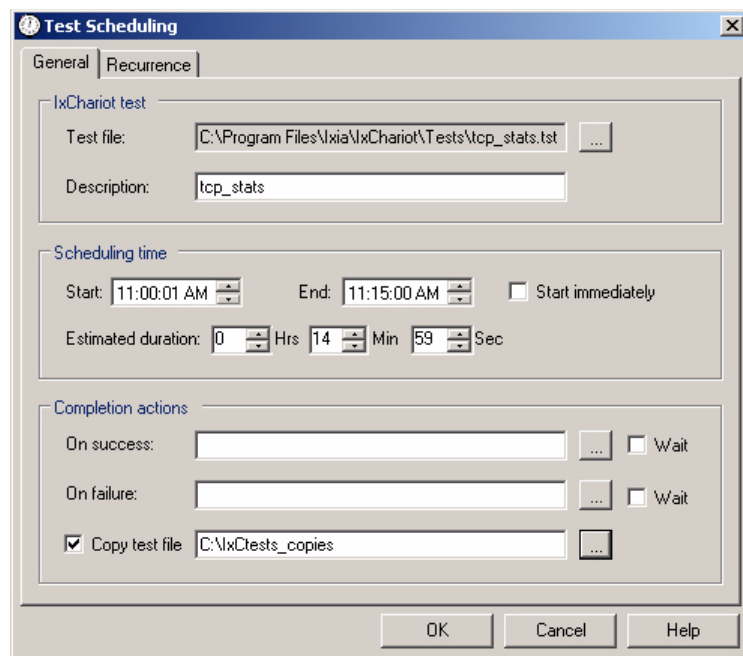
6. Once you have specified the schedule, click **OK** to save the settings.

   If you selected Start immediately, the test starts running as soon as you click OK. Otherwise, the Test Scheduler displays the scheduled test on the calendar. Refer to Table 5-7 for a description of the indicators that the Test Scheduler uses to indicate the status of scheduled tests.

7. Do *not* exit from the Test Scheduler.

   The Test Scheduler must be running at the time a test is scheduled for execution. Therefore, you should minimize the Test Scheduler whenever you have tests awaiting execution.

   When you minimize Test Scheduler, it places an icon in both the task bar and the Windows notification area (system tray), as shown in Figure 5-6.

   Figure 5-6.   Test Scheduler Toolbar Icon

8. Ensure that any test that is scheduled for execution is *not* open in the IxChariot Console when the Test Scheduler starts the test.

   If you have a test opened in the IxChariot Console at the time it is scheduled to start running, Test Scheduler will report errors when trying to save the test results to the .tst file.

Modifying a Test Schedule

To modify a test schedule:

1. Display the Test Scheduler calendar.

2. Select the month, week, or day view.

3. Double-click the entry that you want to change.

   Test Scheduler opens the Test Scheduling window.

4. Modify the schedule.

5. Click **OK** to save the changes.

Stopping a Test During Execution

To stop a test that is currently running:

1. Display the Test Scheduler calendar.

2. Select the Day view.

3. Scroll to the test that you want to stop.

   While a test is running, it displays a light orange status indicator on the calendar.

4. Right-click on the entry.

   Test Scheduler displays a pop-up menu.

5. Select **Stop** from the menu.

Test Scheduler terminates the test execution.

Removing an Entry from the calendar

To completely remove an entry from the scheduling calendar:

1. Display the Test Scheduler calendar.

2. Select the Day view.

3. Scroll to the scheduled test that you want to remove.

4. Right-click on the entry.

Test Scheduler displays a pop-up menu.

5. Select **Delete** from the menu.

Test Scheduler removes the entry from the schedule.

Viewing Messages for a Scheduled Test

To view the messages associated with a scheduled test:

1. Display the Test Scheduler calendar.

2. Select the Day view.

3. Scroll to the desired test.

4. Right-click on the entry.

Test Scheduler displays a pop-up menu.

5. Select **View messages** from the menu.

Test Scheduler opens the Test Scheduling Messages window, and displays any messages associated with the test. Test Schedule reports two types of messages:

• Test Scheduler Status Messages: These messages indicate the status of the selected test. For example, if the test has started execution, Test Scheduler displays the "STARTED" message and lists the date and time that the test started.

• IxChariot Error Codes: If the test has generated errors, Test Scheduler will display them here.

Only tests that are currently running or have completed their execution will have messages.

Obtaining Test Results

Once a test has run successfully, you can access the .tst file using the IxChariot Console to review the results.

Test Status Indicators

The Test Scheduler calendar uses color-coded indicators to show the status of tests that appear on the schedule. Table 5-7 describes these indicators.

Table 5-7.   Test Status Indicators

Status	Color	Description
Armed	light blue	The test is scheduled for execution.
Overdue	light gray	The time has elapsed and the test did not run. This will occur if the Test Scheduler is not running at the time a test is scheduled for execution. Refer to step 7 in *Scheduling a Test* for more information.
Success	light green	The test ran successfully.
Failure	light red	The test ran but generated errors. Note that if you have a test opened in the IxChariot console at the time it is scheduled to start running, Test Scheduler will report errors when trying to save the test results to the .tst file.
Running	light orange	The test is currently running.
Waiting	light yellow	The test is scheduled to begin but it is waiting for another test to complete its execution.
Stopped	light brown	The user explicitly stopped the test while it was running.

Comparing Test Results

The IxChariot Comparison window lets you compare results from multiple tests. Comparing as many as 9 tests in a single window can be extremely useful, particularly when you are comparing baseline results to results taken after equipment or configuration changes were made.

Figure 5-7. Comparison Window
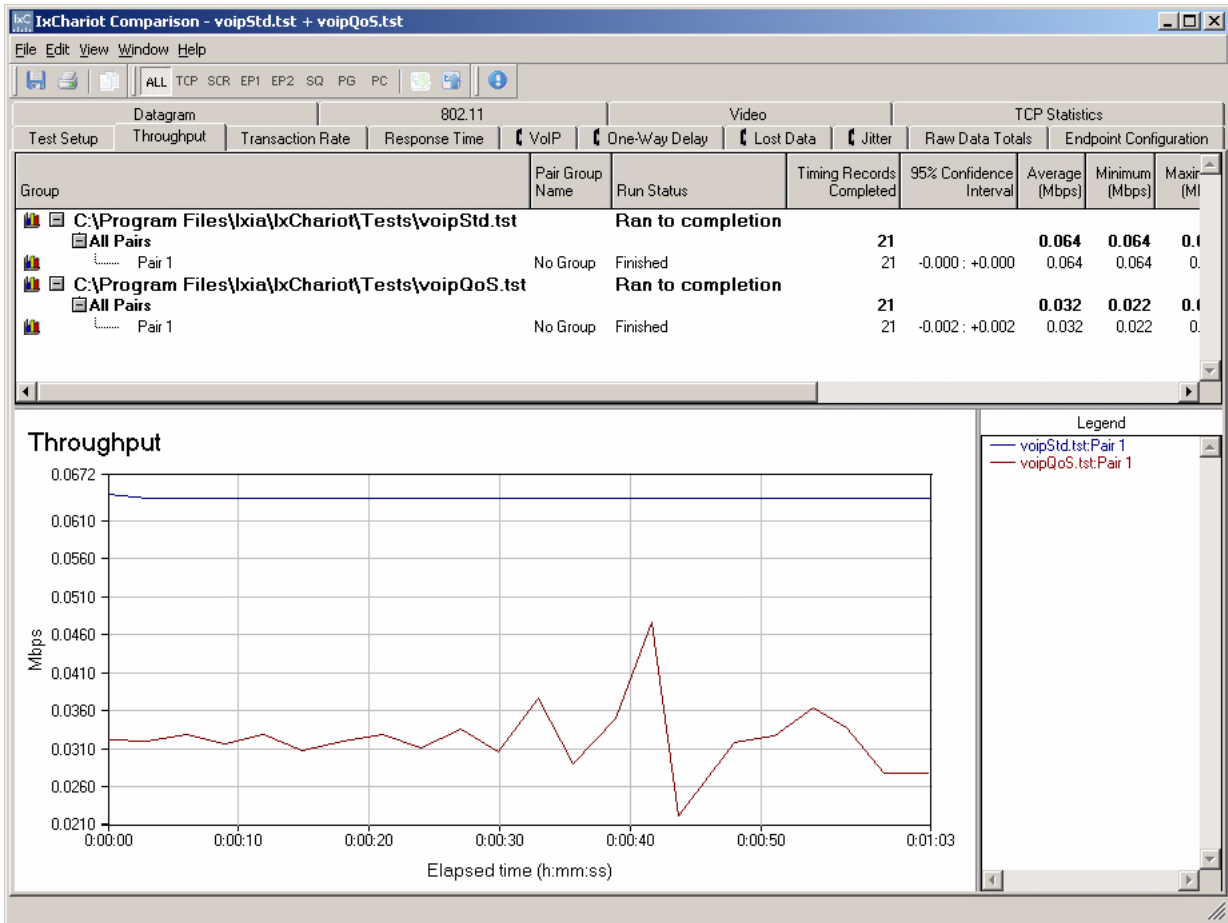
The Comparison window resembles the Test window, but it provides no options for creating new tests or for editing existing tests.

Comparing Tests

To compare the results of two or more tests:

1. Open each of the tests that you want to include in the comparison.

   You can compare as many as nine tests.

2. Close any test windows that you do not wish to include in the comparison, including the untitled.tst window.

3. Verify that each of the tests has been run.

4. Select **Compare Tests** from the Tools menu (from any of the open Test windows).

The Comparison window opens and displays the results of all open tests. See Figure 5-7 for an example.

The markings for graphing or expansion selected in the Test windows are not mirrored in the Comparison window, so you can manipulate results in the Comparison window without affecting the way the original tests are shown. Click **Mark Selected Items** on the Edit menu to select items for graphing. The resulting graph lets you compare the tests' overall results.

You can use similar techniques to compare specific groups or pairs across multiple tests. You can perform comparisons with raw data numbers by collapsing and expanding the appropriate tests and groups in the upper part of the Comparison window.

Only one Comparison window may be open at once per computer. You cannot modify tests or run tests from the Comparison window. To modify a test, access the individual Test window for that test.

To remove a test from a comparison, close that test from within its own test window. Use the Window menu to move between windows. When you return to the Comparison window, the test is no longer shown in the comparison.

If a test is currently running in a test window, the Comparison window shows "Running" as the run status, and the results are not updated in the Comparison window until the test completes. If an open test does not currently have results, "**n/a**" appears in the Results columns.

Just like the Test window, the Comparison window is partitioned into tabbed areas. The **Test Setup** tab lets you view information about how the test was configured, and the other tabs let you view the results of a test.

Saving a Comparison

To save a test comparison:

1. Select the Test Comparison window.

2. Select **Save Comparison** from the File menu.

IxChariot opens a Save Comparison dialog.

3. Enter a name for the comparison file.

When you save a comparison, IxChariot saves the following:

• all the titled tests' filenames (with directory location)

• the current settings for grouping, sorting, and graphing

• the current notebook tab

• the current throughput units being used.

Untitled tests are not stored in the comparison.

Save Comparison As

Saves a comparison for the first time, or saves an existing comparison under a different name. Enter a name for the comparison. You can also select an existing comparison name. The special characters *, \, and ? are not allowed in a comparison name. To save the comparison, click **OK**.

If you expand a test or mark pairs in the Comparison window, the **Save** menu item is disabled to indicate that the Comparison window does not save expanded tests or marked pairs. Saving a comparison a second time with another notebook tab selected overwrites the previously saved comparison.

Saving a comparison does not affect the associated test files and is entirely distinct from saving or modifying a test file.

Opening a Saved Comparison

You can open a previously saved comparison from the Comparison window. Comparisons are only accessible from this window and are not stored in the default Test directory.

To open a saved test comparison:

1. Select **Open Comparison** from the File menu.

   IxChariot opens the Open Comparison dialog box.

2. Select the name of the comparison you want to open from the list.

3. Click **OK** to open the selected comparison.

IxChariot closes all open test windows that are not part of the comparison, opens a test window for each test in the comparison, then opens the new Comparison window.

Comparison Window Menus and Shortcut Keys

The Comparison window menus and toolbars—which are describe below—are very similar to those used in the Test windows.

Access a shortcut Edit menu by right-clicking a selected pair. To see a shortcut menu containing the Graph Configuration and Throughput Units menu items, right-click over the graph section of the Comparison window.

Comparison Window - File Menu

The File menu in the Comparison window provides the following selections:

• **Open Comparison**

   Opens a saved comparison. Refer to *Opening a Saved Comparison* on page 5-45.

• **Save Comparison**

   Saves the test comparison properties to a file. Refer to *Saving a Comparison* on page 5-44.

- **Save Comparison as**

  Saves the test comparison properties to a file. Refer to *Save Comparison As* on page 5-45.

- **Open Test**

  Opens another Test window, and includes the test in the comparison.

- **Print**

  Prints test setup details and results. If you've entered preferences on the Output tab of the Change User Settings notebook, these are used.

- **Export**

  Exports test setup details and results to `HTML`, `.TXT`, or `.CSV` file format. If you've entered preferences on the Output tab of the Change User Settings notebook, these are used.

- **Exit**

  Exits the Comparison window.

Comparison Window - Edit Menu

The menu items on the Edit menu let you work with the items displayed in the Comparison window. You can copy pairs shown in the Comparison window and then paste the pairs into a Test window. You can also select and deselect multiple pairs from this window.

- **Copy**

  Copies an existing pair or group of pairs from the Comparison window to the Windows clipboard. First, select the pair(s) to be copied by clicking the individual pair. Once selected, the pair is highlighted. When copying three or more pairs, you can hold down the Shift key and click the first and last pairs to be copied. The two pairs you clicked, as well as all pairs in between, are highlighted.

  You can paste a pair you've copied into a Test window. You cannot, however, paste pairs into the Comparison window.

- **Select All**

  Selects all of the pairs in the Comparison window. All of the pairs in the Comparison window are highlighted to indicate that they are selected.

- **Deselect All**

  Deselects all of the highlighted pairs in the Comparison window. All pairs that were previously highlighted are no longer selected.

- **Mark Selected Items**

  Specifies a pair or pairs for inclusion in a graph or in a printed report. This command marks all pairs and groups that are currently selected in the Comparison window. A column to the left of the pairs displays a graph icon to indicate that pairs and groups are marked.

- **Unmark Selected Items**

  Excludes a pair or group from graphs or printed reports. Unmarks all pairs and groups currently selected in the Comparison window. The graph icon that once marked a pair or group is no longer displayed.

Comparison Window - View Menu

The selections available from the View menu are identical to those in the Test windows. Refer to *The View Menu* on page 3-9 for detailed information.

Comparison Window - Window Menu

The Window menu on the Comparison window lists all of the open windows. You can use this list to navigate among the open tests and the Comparison window.

The Help Menu

The Comparison window Help menu gives you instant access to the IxChariot online help. It includes the following menu options:

- Click **Contents and Index** to access the IxChariot Help files for the IxChariot Console, Performance Endpoints, IxChariot API, application scripts, and messages.

- Click **Current Window** to get descriptive information about the IxChariot window you are currently viewing.

- Click **Shortcut Keys** for a list of all shortcut keys and key combinations available for the current window.

- Click **About IxChariot** for details on the IxChariot version and build level, and for information about service and support. The About IxChariot dialog box contains copyright and release information.

  Click **Support Info** in the About IxChariot dialog box for IxChariot technical support information.

Shortcut Keys for the Comparison Window

You can use the following keys and key combinations in any IxChariot Comparison window, instead of using the mouse.

Table 5-8.   Shortcut Keys for the Comparison Window

Key or Key Combination	Command Invoked
F1	View context-sensitive help for the Comparison window.
F2	View the help Table of Contents and an index of all the available IxChariot help topics.

Table 5-8.   Shortcut Keys for the Comparison Window (Continued)

Key or Key Combination	Command Invoked
F9	Show the keys and key combinations available in a window.
F11	Open the About IxChariot dialog box, which shows version and build level and provides product support information.
Ctrl+A	Select all the pairs in a comparison.
Ctrl+C	Copy the test setup for one or more pairs to the Windows clipboard.
Ctrl+O	Open a previously-saved test.
Ctrl+S	Save this comparison. If the comparison is untitled, the Save As dialog box prompts you to choose a filename.
Alt+F4	This key combination can be used to close any window or dialog box. When used to close a dialog box, it has the same effect as clicking the **Esc** key or clicking **Cancel** with the mouse.

In addition to these keys, the **Alt** key can be used in combination with any under-scored letter to invoke a menu function. The menu function must be visible and not shown in gray. For example, clicking **Alt+F** shows the File menu.

Encrypting Setup Flows

Description

When you click the Run button in the Test window, IxChariot initiates the setup phase of a test. During the setup phase, the Console sends setup information to endpoint 1. Endpoint 1 parses the information and sends to Endpoint 2 its portion of the setup information.

Depending upon the release level of your IxChariot Console and the Performance Endpoints that you are using, you can choose to encrypt the setup data. Encryption of setup data is supported in IxChariot 6.30 and above (both the Console and the Endpoints).

When you enable encryption at both endpoints, the basic information flow is as follows:

1. The Console encrypts the setup data and sends it to Endpoint 1.

2. Endpoint 1 decrypts the data received from the Console.

3. Endpoint 1 extracts the setup data required by endpoint 2, encrypts it, and sends it to endpoint 2.

4. Endpoint 2 decrypts the data received from Endpoint 1.

Figure 5-8 illustrates the information flow when encryption is enabled (at both endpoints). Notice in this figure that the replies from the endpoints are not encrypted.

Figure 5-8.   Encrypted setup data (IxChariot 6.30 and above)
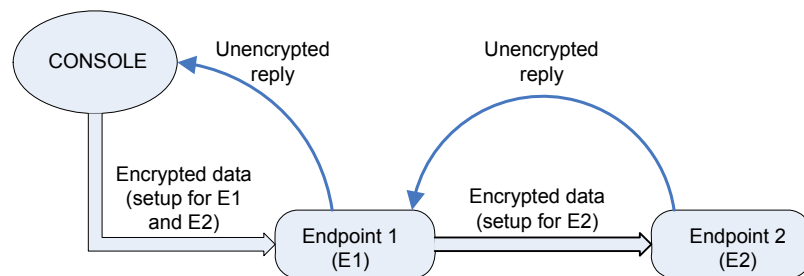
Configuring Encrypted Setup Flows

You configure setup flow encryption using the USE_ENCRYPTED_FLOWS parameter in the endpoint.ini file for one or both of the endpoints in a test. (Refer to the *IxChariot Performance Endpoints* guide for more information about endpoint.ini settings.)

During the setup phase of a test, the IxChariot Console determines whether or not the endpoints support encryption. If either of an endpoint pair supports encryption, the Console checks the setting in the Endpoint 1 endpoint.ini file to determine whether or not to encrypt the setup data before sending it. In similar fashion, Endpoint 1 checks the setting in the Endpoint 2 endpoint.ini file to determine whether or not to encrypt the setup data before sending it.

The USE_ENCRYPTED_FLOWS flag in the endpoint.ini file takes the following values:

- **OFF** – The endpoint will not accept encrypted data.

- **ON** – The endpoint will accept only encrypted data.

For Endpoint 1, this setting determines whether the endpoint will require encrypted data from the IxChariot Console. If the parameter is set to ON, then Endpoint 1 will reject unencrypted setup flows sent from the Console.

For Endpoint 2, this setting determines whether the endpoint will require encrypted data from Endpoint 1. If the parameter is set to ON, then Endpoint 2 will reject unencrypted setup flows sent from Endpoint 1,

Endpoint 1, however, can send either encrypted or unencrypted data to Endpoint 2, regardless of the setting of the USE_ENCRYPTED_FLOWS flag. The possible combinations are described in Table 5-9.

Table 5-9.   Effect of Encryption Settings

If Endpoint 1 USE_ENCRYPTED_ FLOWS Setting is:	And Endpoint 2 USE_ENCRYPTED_ FLOWS Setting is:	Then ...
OFF	OFF	Endpoint 1 accepts only unencrypted data from the Console, and sends unencrypted data to Endpoint 2.
ON	OFF	Endpoint 1 accepts only encrypted data from the Console, and sends unencrypted data to Endpoint 2.
OFF	ON	Endpoint 1 accepts only unencrypted data from the Console, and sends encrypted data to Endpoint 2.
ON	ON	Endpoint 1 accepts only encrypted data from the Console, and sends encrypted data to Endpoint 2.

Refer to Table 5-10 for a description of various combinations of USE_ENCRYPTED_FLOWS settings.

IxChariot Encryption Compatibility Scenarios

The encrypted setup flows feature is available in IxChariot release 6.30 and higher. It is important to note that both the IxChariot Console and the IxChariot endpoints must be running a supported release level for full feature support.

Table 5-10 summarized the various compatibility scenarios that you may encounter when using encrypted setup flows. This table assumes that the IxChariot Console is running at version 6.30 or higher.

Table 5-10.   Data Flow Encryption Scenarios

Console[a] to Endpoint 1 Data Flow	Endpoint 1 version	Endpoint 1 endpoint.ini flag setting	Endpoint 1 to Endpoint 2 Data Flow	Endpoint 2 version	Endpoint 2 endpoint.ini flag setting	Setup Result
unencrypted	Pre-6.30	N/A	unencrypted	Pre-6.30	N/A	Successful setup.
unencrypted	Pre-6.30	N/A	unencrypted	6.30 or higher	OFF	Successful setup.
unencrypted	Pre-6.30	N/A	unencrypted	6.30 or higher	ON	Setup fails. E2 will accept only encrypted data.
unencrypted	6.30 or higher	OFF	unencrypted	Pre-6.30	N/A	Successful setup.
encrypted	6.30 or higher	ON	unencrypted	Pre-6.30	N/A	Successful setup.
unencrypted	6.30 or higher	OFF	unencrypted	6.30 or higher	OFF	Successful setup.
unencrypted	6.30 or higher	OFF	encrypted	6.30 or higher	ON	Successful setup.
encrypted	6.30 or higher	ON	unencrypted	6.30 or higher	OFF	Successful setup.
encrypted	6.30 or higher	ON	encrypted	6.30 or higher	ON	Successful setup.

a. This table assumes that the IxChariot Console is running at version 6.30 or higher.

Notice in Table 5-10 that the setup fails for the scenario described in the third row in the table. This is, in fact, the expected behavior. The setup phase is designed to fail in any scenario where an endpoint requires encrypted data but receives unencrypted data.

If IxChariot Console is Pre-6.30

The setup phase will fail if the Console is running a pre-6.30 release of IxChariot and either Endpoint requires encrypted setup data. In this case, the Console will display error CHR0124. The setup phase will run successfully, however, if neither endpoint requires encrypted setup data.

Displaying Encryption Settings

The Endpoint Configuration tab in the Test window displays the encryption settings for both endpoints. In addition, if you right-click a pair in the Test window and select "Show endpoint configuration..." from the menu, IxChariot displays an Endpoint Configuration window for that pair. This window shows USE ENCRYPTED FLOWS as either ON or OFF.

Modifying the TCP Window Size

Obtaining optimal throughput across wide area network connections often requires adjustments to the *TCP window* buffers that are used for flow control. This section explains how to make these adjustments in IxChariot application scripts.

Purpose of the TCP Sliding Window

TCP uses the concept of a sliding window as its primary flow control mechanism. The sliding window allows the source TCP process to transmit multiple segments on a particular connection before waiting for an ACK from the TCP process on the destination device. The size of the window determines the number of bytes that will be transmitted without an acknowledgement. The TCP process at each end of the connection can dynamically adjust the size of the sliding window to avoid network congestion. A larger sliding window enables higher throughput.

The size of the buffers used for the sliding window have a significant impact on throughput. If the buffers are too large, the sender can overrun the receiving buffers, which will cause the receiver to shut down the send window. However, if the buffers are too small, you may end up using only a fraction of your available bandwidth. Careful adjustment of the sliding window is especially important for networks that have high latency and low bandwidth or low latency and high bandwidth. Satellite networks, WIFI networks, and 1 GB through 10 GB links can all benefit from well-tuned TCP window buffers.

The TCP Window Scale Option

TCP was originally developed when wide area links rarely exceeded 56 Kb/s connection speeds. Therefore, the TCP header provides only a 16 bit field for specifying the window size, which allows for a maximum buffer space of 64 KB.

To eliminate the 64 KB buffer size barrier, RFC 1323 specifies a TCP window scale option, which is negotiated at the opening of the connection. The TCP window scale option provides a means by which an operating system can use the 16 bit Window field in the TCP header to specify a value that scales the TCP window to over 1 MB. For example, if the scale value is 13, the TCP window size is computed as:

```
window = 65535 * 2^13 = 65535 * 8192 = 536,862,720 bytes
```

For TCP windows to scale beyond 64 KB, two things are required.

- The option for TCP window scaling must be active in the operating system. Some, but not all, operating systems enable this feature by default. For example, the option is on by default in Linux and off by default in Microsoft Windows. You will need to verify the requirements of your operating system prior to using the feature.

- The application must call `setsockopt` with the `SO_SNDBUF` and `SO_RCVBUF` arguments being passed with the number of bytes to allocate.

In IxChariot, setting the socket option is done by adding script variables directly after the CONNECT_INITIATE statement. *Setting the TCP Window Size in an IxChariot Script* on page 5-54 describes this in detail.

Buffer Types in IxChariot Application Scripts

IxChariot scripts use two types of buffers to manage the sending and receiving of data:

*   The SEND and RECEIVE commands define buffers that are internal to the endpoints. They are unrelated to the TCP sliding window service; rather, they determine how much data will be sent to the TCP stack for processing for each SEND and RECEIVE operation.

    For example, the send_buffer_size variable defines this buffer in the Ultra_High_Performance_Throughput.scr script:

    ```
    . . .
    12 SEND
    13  send = size_of_record_to_send (2147483647)
    14  buffer = send_buffer_size (1048576)
    . . .
    ```

*   The CONNECT_INITIATE and CONNECT_ACCEPT commands specify how much buffer space the operating system should allocate for the TCP sliding window service.

    For example, the conn_send_buffer_e1 and conn_rcv_buffer_e1 variables define these buffers for Endpoint 1 in the Ultra_High_Performance_Throughput.scr script:

    ```
    . . .
    3  CONNECT_INITIATE
    4   port = source_port (AUTO)
    5   send_buffer = conn_send_buffer_e1 (536870912)
    6   receive_buffer = conn_rcv_buffer_e1 (536870912))
    . . .
    ```

The buffers allocated via the The CONNECT_INITIATE and CONNECT_ACCEPT commands instruct the operating system to allocate buffers of a specific size for the TCP sliding window service. Note that there are four such variables: two for each endpoint. In the script noted above, the variables are:

*   conn_send_buffer_e1

*   conn_rcv_buffer_e1

*   conn_send_buffer_e2

*   conn_rcv_buffer_e2

Figure 5-9 illustrates the difference between these two types of buffers, and shows where they are defined within the IxChariot scripts.

Figure 5-9. Buffers in IxChariot Scripts
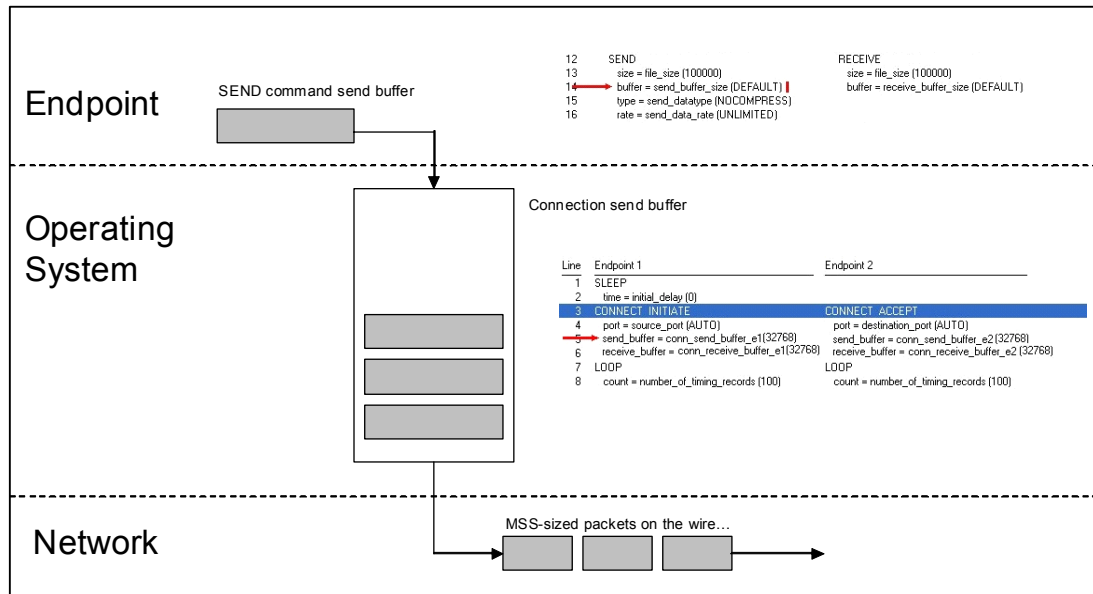
Calculating the Correct TCP Window Size

The optimal TCP window size is expressed as:

```
buffer size = 2 * bandwidth * delay
```

In this case, you will use the bandwidth of the slowest link in your path. You can use `ping` to obtain the round-trip time (RTT), in which case the formula becomes:

```
buffer size = bandwidth * RTT
```

For example, if you have two hosts with GigE cards communicating over a wide area connection where the round trip time (RTT) is 70 milliseconds, you can calculate the buffer size as follows:

```
buffer size = 1,000,000,000/8 * 70/1,000 = 8.75 MB
```

Based on this calculation, the typical default buffer size of 64 KB would be clearly inadequate for this connection.

Setting the TCP Window Size in an IxChariot Script

To set the TCP Window Scale option in IxChariot, you need to add script variables following the CONNECT_INITIATE statement in the script. This is illustrated in the Ultra_High_Performance_Throughput.scr script:

```
. . .
3   CONNECT_INITIATE
4    port = source_port (AUTO)
5    send_buffer = conn_send_buffer_e1 (536870912)
6    receive_buffer = conn_rcv_buffer_e1 (536870912))
. . .
```

To add the TCP Window Scale variables in the Script Editor:

1. Add a pair to your test.

2. Select the script, then click **Edit This Script** to open the Script Editor.

3. Double-click the `send_buffer` row that follows the CONNECT_INITIATE statement. (This is row 5 in the script example shown above.)

   The Script Editor opens the Edit Parameter dialog.

4. Select the **Variable** radio button.

5. Select one of the four parameters from the Parameter drop-down list. (**Send Buffer Size on E1** is the first parameter on the list.)

6. Enter the variable name corresponding to the selected parameter:

Parameter	Variable Name
Send Buffer Size on E1	conn_send_buffer_e1
Receive Buffer Size on E1	conn_rcv_buffer_e1
Send Buffer Size on E2	conn_send_buffer_e2
Receive Buffer Size on E2	conn_rcv_buffer_e2
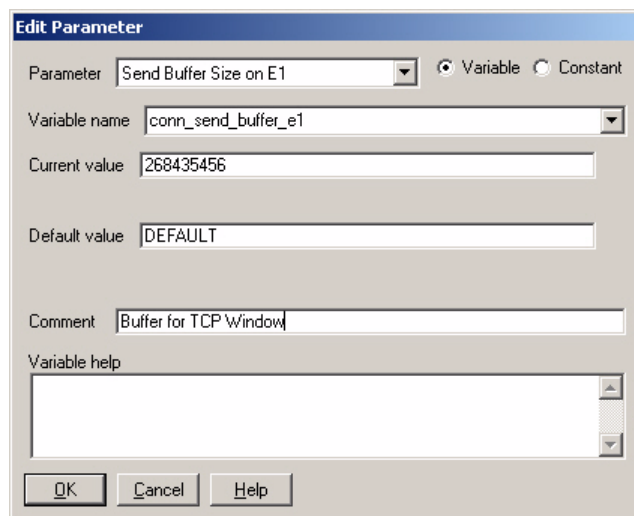
7. Enter the **Current value** and the **Default value**.

   These values will be based on the calculation shown in *Calculating the Correct TCP Window Size* on page 5-54.

8. Optionally, enter a **Comment** and **Variable Help** text.

   The Edit Parameter dialog should appear similar to the example shown in Figure 5-10.

   Figure 5-10. Create Variable for TCP Window Buffer

9. Click **OK** in the Edit Parameter dialog to save the new variable.

10. Repeat the above steps to create all four of the variables, using the variable names listed in step 6.

11. To save the script modifications for the current pair only, select **Save to Pair** from the File menu in the Script Editor.

   Refer to the *IxChariot Scripts Development and Editing Guide* for an explanation of all of the options for saving a script.

12. Select **Exit** from the File menu in the Script Editor.

Sample Bandwidth Delay Product Values

Bandwidth Delay Product (BDP) is the calculation used in *Calculating the Correct TCP Window Size* on page 5-54. Table 5-11 provides a matrix of BDP values that you can use when determining the optimal values for the TCP Window variables.

Table 5-11.   Sample Bandwidth Delay Products

One-Way Delay (ms)

Bandwidth (kbps)	1	10	15	30	50	100	500
256	64	640	960	1,920	3,200	6,400	32,000
512	128	1,280	1,920	3,840	6,400	12,800	64,000
768	192	1,920	2,880	5,760	9,600	19,200	96,000
1,024	256	2,560	3,840	7,680	12,800	25,600	128,000
1,536	384	3,840	5,760	11,520	19,200	38,400	192,000
2,048	512	5,120	7,680	15,360	25,600	51,200	256,000
3,192	798	7,980	11,970	23,940	39,900	79,800	399,000
4,096	1,024	10,240	15,360	30,720	51,200	102,400	512,000
8,192	2,048	20,480	30,720	61,440	102,400	204,800	1,024,000
10,000	2,500	25,000	37,500	75,000	125,000	250,000	1,250,000
34,000	8,500	85,000	127,500	255,000	425,000	850,000	4,250,000
100,000	25,000	250,000	375,000	750,000	1,250,000	2,500,000	12,500,000
1,000,000	250,000	2,500,000	3,750,000	7,500,000	12,500,000	25,000,000	125,000,000
10,000,000	2,500,000	25,000,000	37,500,000	75,000,000	125,000,000	250,000,000	1,250,000,000

Note that:

• The shaded values were not supported in IxChariot prior to release 6.20, due to the 64 KB buffer limitation.

• For optimal performance, choose a window size that is a multiple of the MSS.

Editing Script Variables

When you add an endpoint pair to a test, you can configure the variables used in the test script. This entails three steps:

- Open the script for editing in the IxChariot Script Editor, as described in *Opening the Script for Editing* on page 5-57.

- Modify the script variables, as required for your testing, as described in *Editing a Script Variable* on page 5-57.

- Save the modifications with the test, as described in *Saving the Script Modifications* on page 5-59.

Opening the Script for Editing

To open a script for editing:

1. In the Test window, double-click the pair that contains the script you wish to modify.

   IxChariot opens the Edit an Endpoint Pair dialog.

2. If you have not done so already, select the script for the test.

3. Click the **Edit This Script** button.

   IxChariot opens the script in the Script Editor

4. Continue with *Editing a Script Variable* on page 5-57.

Editing a Script Variable

You use the Edit Variable dialog box to modify the value assigned to a variable. To edit a script variable:

1. Select the variable from the Variable Name list in the lower area of the Script Editor window.

2. Select **Edit Variable** from the **Edit** menu.

   The Script Editor opens the Edit Variable dialog, as shown in Figure 5-11.

Figure 5-11. Edit Variable Dialog
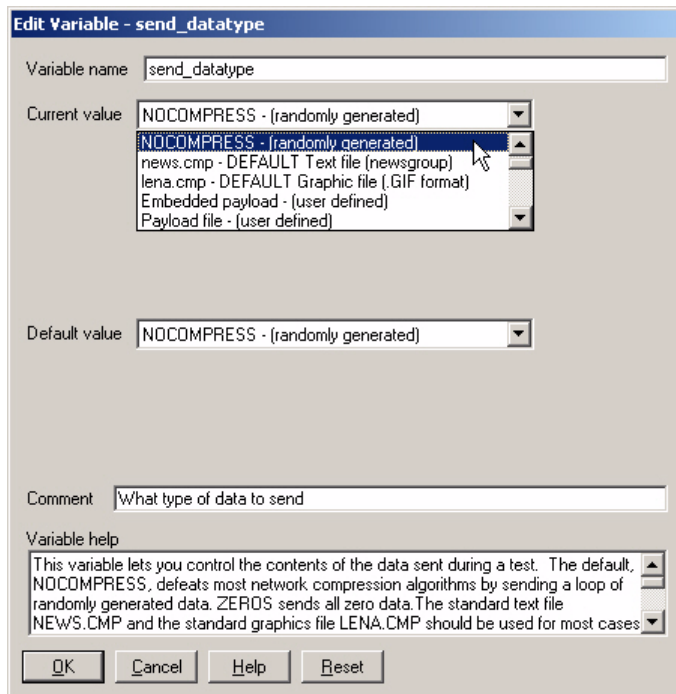
3. Make any desired changes.

   Table 5-12 describes the fields in the Edit Variable dialog.

4. Click **OK**.

5. Continue with *Saving the Script Modifications* on page 5-59.

Table 5-12.   Fields in the Edit Variable Dialog

Field	Description
Variable Name	A name for the variable that must be unique within the script, and cannot contain blanks; underscores are appropriate, however.
Current Value	The type of value to be selected. The available choices vary according to the parameter; for example, if you chose the Sleep Time parameter, the available choices are Constant Value and Uniform, Normal, Poisson, and Exponential Distribution. Do not use commas when entering values in this field. The type of variable used for the SLEEP command allows five values: *Constant Value, Uniform Distribution, Normal, Poisson,* and *Exponential*. For a Constant Value, one field is presented for the value. For a distribution, two fields let you enter the upper and lower distribution range. All values are in milliseconds. See "Setting Sleep Times" in the "Rules for Scripts" section of the *Application Scripts* guide for more information and pictures of the distributions. Click **Reset** to reset the value in the *Current value* field to the value showing in the *Default value* field the last time you exited the Edit Variable dialog box for this variable.
Default Value	Lets you specify the initial value for the variable, when the script is loaded from a file into a test. Accepts numbers to 9 digits. On some variable types, such as the buffer size on SEND and RECEIVE, you can use the terms "DEFAULT" or "AUTO." The DEFAULT value depends on the network protocol and the endpoints you are using AUTO, when entered for the "source_port" or "destination_port" variables, specifies that Endpoint 1 should choose the port number dynamically. Do not use commas when entering values in this field.
Variable Help	A description of the variable and how it operates in a script. You can customize the help text by entering information in this field.

Saving the Script Modifications

Once you have modified one or more variables in a script, you will save those modifications with the test. To save your script modifications:

1. Select **File** > **Save to Pair**.

IxChariot saves your modified script file to disk, using the original script file name. Your changes will apply only to the script used by the selected pair.

2. Select **File** > **Exit**

IxChariot closes the Script Editor and returns to the Test window.

For More Information

For detailed information about the script editor, refer to the *IxChariot Script Development and Editing Guide*.

Exporting and Printing Results

Related Topics

Custom Printing and Export Options on page 5-64
Output Tab on page 6-33
Output Templates on page 5-65

If you click **Print** or **Export** on the File menu in the Test window, you are given an opportunity to print or export any aspect of the test. You can export formatted results to any of the following formats:

• an ASCII text file

• a Web page file, in HTML format

• a comma-separated values file, in .csv format.

• a PDF file

See Appendix A, *IxChariot File Types*, for information on each file format.

Before choosing print or export options, remember that tests with many pairs and timing records can use a great deal of paper. Make sure that the pairs of interest are expanded in the Test window. If the group whose results you want to print or export is collapsed, click the plus sign next to that group to expand it. Details about collapsed pairs are not included when exporting or printing.

After selecting the pairs that you want to include in your report, next choose the level of detail that you want to see in your printed/exported output..

Printing Options

To print your test or various aspects of it, click the **Print** option in the **File** menu.

A Print Options dialog box opens, where the following options are available:

• **Output template**

Lets you choose an output template. To modify an output template, select it here. To create a new output template, enter the name of the new output template in this field. The special characters *,\, and ? are not allowed in an output template name.

• See *Output Tab* on page 6-33 for more information on selecting default output templates.

• See *Output Templates* on page 5-65 for information on creating output templates.

- **Summary report**

  Shows just the test setup and a summary of any results. You can also choose to include the information shown in each of the tabs and graphs in the printed or exported results.

  - Check "**Result Tables**" to include the information summarized in the Throughput[1], Transaction Rate, Response Time, Lost Data, and Datagram tabs.

  - Check "**Result Graphs**" to output the corresponding graphs. When exporting results, GIF image files are created for these graphs. This option is only available for HTML Export.

- **Complete report**

  Shows everything—all the setup information, all the scripts, all the results analysis, and all the individual timing records. This option is not recommended for printed reports unless you have small tests or plenty of paper.

- **Custom**

  Lets you choose precisely what to show in your report or display the selections from an output template you have created. Click **Select** to choose each option. See *Custom Printing and Export Options* on page 5-64 for more information on this dialog box.

- **Print all**

  Lets you print information about all groups and pairs in the test.

- **Print marked groups and pairs**

  Lets you print information only about selected groups and/or pairs. Before you can print them, you must mark pairs or groups in the Test window: click **Mark Selected Items** in the Edit menu.

- **Save options with test**

  Save the options you have selected in this dialog box along with the test. The next time you access the Print Options dialog box for this test, the options you have chosen are filled in.

- **Save Template**

  Saves an output template if you created a new one or modified an existing template. See *Output Templates* on page 5-65 for more information.

- **Select Printer...**

  Lets you select the printer for your output from among the printers currently defined on your computer. To change the characteristics of that printer, click **Properties** or **Job Properties**. For example, you may want to choose landscape printing—which may offer you a more readable view of your results. If you have long addresses or comments, use landscape printing to gain extra page width.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

At the bottom of the dialog box, IxChariot shows the approximate number of pages to be printed based on the options you have selected.

> Note: When printing to a file, if the target printer is set to **Pause Printing**, the IxChariot Test window will be disabled indefinitely. To continue, set **Pause Printing** to **Off**.

Export Options

To export your test or various aspects of it, click the **Export** option in the **File** menu.

The following export options become available:

- HTML...
- Text...
- CSV...
- PDF...

Export options for the HTML, Text, and PDF formats are common for all three, while the CSV format has its own set of export options.

The following options are available for HTML, Text, and PDF output:

- **Output template**

    Lets you choose an output template. To modify an output template, select it here. To create a new output template, enter the name of the new output template in this field. The special characters *,\, and ? are not allowed in an output template name.

    - See *Output Tab* on page 6-33 for more information on selecting default output templates.
    - See *Output Templates* on page 5-65 for information on creating output templates.

- **Summary report**

    Shows just the test setup and a summary of any results. You can also choose to include the information shown in each of the tabs and graphs in the printed or exported results.

    - Check "**Result Tables**" to include the information summarized in the Throughput[1], Transaction Rate, Response Time, Lost Data, and Datagram tabs.
    - Check "**Result Graphs**" to output the corresponding graphs. When exporting results, GIF image files are created for these graphs. This option is available for PDF and HTML Export.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

- **Complete report**

  Shows everything—all the setup information, all the scripts, all the results analysis, and all the individual timing records. This option is not recommended for printed reports unless you have small tests or plenty of paper.

- **Custom**

  Lets you choose precisely what to show in your report or display the selections from an output template you have created. Click **Select** to choose each option. See *Custom Printing and Export Options* on page 5-64 for more information on this dialog box.

- **Export all**

  Reports statistics on all the pairs in the test.

- **Export marked groups and pairs**

  Reports statistics on only the pairs marked with a graph symbol where they appear in the Test window.

- **Save Template**

  Saves an output template if you created a new one or modified an existing template. See *Output Templates* on page 5-65 for more information.

- **Save options with test**

  Save the options you have selected in this dialog box along with the test. The next time you access the Export to HTML/GIF, Export to Text File, or Export to PDF dialog box for this test, the options you have chosen are filled in.

Export Options for .CSV Files

Related Topics

Custom Printing and Export Options on page 5-64
Output Tab on page 6-33
Output Templates on page 5-65

IxChariot can export results in the widely-supported file format called `.csv` (for "comma-separated values"). In `.csv` format, values are separated from one another by commas, with the addition of double-quotes when needed. Popular spreadsheet programs, such as Microsoft Excel and Lotus 1-2-3, can open and save files in `.csv` format, making the `.csv` file format a good tool for coordinating the information you want to export from IxChariot.

Two characters in `.csv` format need special treatment: commas and double-quotation marks.

- If a comma is in a string, enclose the whole string in double quotes. For example, the comment field below contains two commas:

  ```
  "These commas, here, are part of the comment."
  ```

- Surround each double quotation mark in a string with its own pair of double quotation marks. For example, the comment field below contains two sets of double quotation marks:

  ```
  This endpoint is used in the """R&D1""" department.
  ```

The Export to `.csv` dialog box is shown when you click **Export to .CSV** on the File menu. You can select the type of content you want exported from the test. Clear the boxes for those options that you don't want to include in your report:

- **Test summary and run options**

    Provides a summary of any results and your run options

- **Pair summary**

    Provides information contained in the Test Setup tab

- **Pair details**

    Provides the timing records for the pairs in your test.

- **Export all**

    Reports statistics on all the pairs in the test

- **Export marked pairs**

    Reports statistics on only the pairs marked with a graph symbol where they appear in the Test window.

To save all the above selections as a default setting for `.csv` export, click **Change User Settings** on the Options menu and click the **Output** tab.

Custom Printing and Export Options

Related Topics

Exporting and Printing Results on page 5-60
Output Tab on page 6-33

In the Custom dialog box, you can specify the summary and detailed test information in your report.

To access the Custom dialog box, click to enable **Custom** reports and click **Select** in the Print Options/Export dialog boxes.

When you export to HTML, graphs are exported as separate files in `.gif` image format. A GIF file can be imported into your word-processing or graphics programs and can be linked to the Web page created by exporting results.

Clear the boxes for **Summary information** options that you do not want to include in your report.

- **Run options**

    Includes a summary of your run options in your results.

- **Test Setup (Console to Endpoint 1)**

    The network address, protocol, and quality of service used for the Console's communications with Endpoint 1.

- **Test Setup (Endpoint 1 to Endpoint 2)**

    The network address used for test setup flows between E1 and E2.

- **Test Execution**

    The network addresses, protocol, and service quality used for tests run between the endpoints.

Other options include results and/or graphs for Throughput, Transaction Rate, Response Time, RSSI, Video, TCP Statistics, VoIP, One-Way Delay, Jitter, Lost data, Datagram results, Raw data totals, and Endpoint configuration. For more information, see the topics grouped under *Printed/Exported Results* on page 11-39.

Because the results are configured in ranges, result tables are not available for Delay Variation or Consecutive Lost Datagrams. See *Jitter and Delay Variation* on page 10-52 and *The Lost Data Tab* on page 11-31 for more information.

Click to select the **Detailed Information** options to include in your report:

• **Pair configuration**

 Reports the script commands and variables used for your test

• **Endpoint configuration**

 Reports detailed information about the endpoints in your test. Click **Endpoint Configuration** on the View menu to see this information.

• **Timing records**

 Reports the individual timing records for the pairs in your test. Tests can contain a great many of these—tens of thousands—so we recommend checking the **Raw Data Totals** tab to find out how many you have. Click **Show Timing Records** in the View menu to see the timing records for a test.

Click **Select All** to choose all these options. Clear all the options by clicking **Deselect All**.

One or more of these Print or Export options may be dimmed, indicating that the option(s) is (are) not available to be reported for this particular test. An option is enabled only when information of that type exists for the test, or if your version of IxChariot supports that function. For example, jitter data is available only for tests using the RTP protocol, and Delay Variation data is available only for Consoles that support Voice over IP testing.

Output Templates

Related Topics
Custom Printing and Export Options on page 5-64
Output Tab on page 6-33
Exporting and Printing Results on page 5-60
Export Options for .CSV Files on page 5-63

Output templates let you save printing and exporting options so that when you want to print a test or export it to HTML/Text/PDF, you include only the results that you want to see. To set up your output templates, click **Edit Output Templates** in the Tools menu.

The Output Templates List dialog box lists any output templates you have already configured; there are no default templates.

You can add new templates, modify existing templates, copy and delete templates from this dialog box. Select and modify any options you want in the output template and click **OK** to save it. See *Exporting and Printing Results* on page 5-60 for more information on these options.

Once you have created an output template, you can set it as the default for the tests you print or export. See *Output Tab* on page 6-33 for more information.

Adding an Output Template

1. Click **Tools>Edit Output Templates....**The Output Templates List dialog opens.

2. Click **Add**. The Add Output Template dialog box opens.

3. In this dialog, enter a name for the new output template in the Template name field.

4. Go on to select the details to be included in the template:

   a: type of report

      i: Summary report: Result tables or Result graphs

      ii: Complete report

      iii: Custom

   b: details to export

      i: Export all

      ii: Export marked groups and pairs

5. Click **OK**. The new output template appears in the Output Templates List dialog window.

IMPORTANT:

The following fields are excluded from PDF Summary and Complete reports generated either from the GUI or using the FMTTST utility:

- Test Setup (Console to Endpoint 1)
- Test Setup (Endpoint 1 to Endpoint 2)
- Raw Data Totals
- Endpoint Configuration
- Detailed Information section (Pair Configuration, Endpoint Configuration, Timing Records)

If a particular template is used for one export format first (for example, HTML) and then for PDF, the above-mentioned fields do not appear in the generated PDF report. In turn, if a PDF template is used for exporting to another format, the above-mentioned fields appear as disabled.

Modifying an Output Template

1. Click **Tools>Edit Output Templates....**The Output Templates List dialog box opens.

2. Click **Modify**. The Modify Output Template - [Template Name] dialog opens.

3. Change the details you desire.

4. Click **OK**. The modified template appears in the Output Templates List dialog window.

Copying an Output Template

1. Click **Tools>Edit Output Templates....**The Output Templates List dialog box opens.

2. Click **Copy**. The Copy Output Template dialog opens.

3. Enter a name for the new template in the Template name field.

4. Click **OK**. The template appears in the Output Templates List dialog window.

Command-Line Programs

IxChariot contains four programs that produce command-line text output. These let you do many of the console functions (except creating and changing tests), from a command prompt at the computer where you installed the Console. You can combine these programs to execute tests from within batch files, intermixed, for example, with other programs.

Each of these commands writes information to the screen, using `stdout`. You can redirect this information to a file, using the > or >> operators. If you choose to redirect the output to a file, you can print the file or manipulate it with an ASCII text editor.

See the *API Guide* for information on the IxChariot API, which provides you with the ability to automate testing.

RUNTST: Running Tests

Related Topics

Using the Command Line to Run Large Tests on page 8-12
Command-Line Programs on page 5-67
Using RUNTST for Stress and Regression Testing on page 5-69

The program named `RUNTST` lets you run test files created by the IxChariot Console program. For tests with more than 5,000 pairs, you should use `RUNTST` instead of the Console to run your tests. See *Using the Command Line to Run Large Tests* on page 8-12 for more information.

Here's the syntax of the `RUNTST` command:

```
RUNTST test_filename [new_test_filename] [-t N] [-v]
```

Enter `RUNTST` at a command prompt on the computer you are using as the Console.

- As its first parameter, supply the filespec of an IxChariot test file.

- Its second parameter is optional; you can supply a separate filespec as a target for the test setup and results. If the second parameter, `new_test_filename`, is omitted, the results are written directly to the original test file.

- The `-t` (timeout) parameter is optional. If specified, this parameter causes RUNTST to stop running the test after *N* seconds. If the specified timeout value expires while a test is running, the test ends; otherwise, the test Run Options determine when the test ends.

- The `-v` (verbose mode) parameter is also optional. By default, RUNTST does not show results as it runs. The `-v` parameter causes RUNTST to send all run information to standard out as it is received.

As an example, here's how to run the IxChariot test contained in a file named `FILEXFER.TST`, and write the results back into that file:

```
RUNTST TESTS\FILEXFER.TST
```

The RUNTST program runs until the timing records are returned by all Endpoint 1 computers participating in the test.

Use the FMTTST command to read the binary results data in a test file and produce a formatted listing.

You can stop a running test by clicking **Ctrl+C** or **Ctrl+Break**; RUNTST will ask you if you really want to exit. If you answer with Y, RUNTST directs the endpoints to stop the test. If the stopping seems to take excessively long, click **Ctrl+C** or **Ctrl+Break** again to exit the program altogether.

RUNTST does not poll endpoints, even if polling is defined in the test file.

RUNTST logs to the file `runtst.log`. If your `runtst.log` file grows to more than 5 MB in size, it may affect IxChariot's performance. See *Keeping Log File Size to a Minimum* on page 8-12 for more information.

If RUNTST reads a test file from an older version, it writes out its test setup and results in its current version. For example, if you have a test that was created at version 3.x and you run the newest version of RUNTST, the file that's written will be in the newest version—which cannot be read by older versions of RUNTST and FMTTST. If you'd like to continue using older versions of RUNTST, be sure to make an extra copy of the test file or write to a `new_test_filename` so you still have a copy of your original.

The RUNTST and IxChariot Console programs can generally be loaded at the same time, but only one of them can be running a test at a time. If you've changed your reporting ports on the **Firewall Options** tab in the Change User Settings notebook to a value other than AUTO at the Console, RUNTST cannot run while the Console is loaded—expect to see message **CHR0264** at RUNTST.

The RUNTST program is installed at the Console in the `Ixia\IxChariot` program folder.

Using RUNTST for Stress and Regression Testing

`RUNTST` is the command-line program that executes the same IxChariot tests you run at the IxChariot Console. It can be a valuable tool for doing regression and stress tests. Using the IxChariot Console, you can create and save a variety of tests, which you can then run from `RUNTST`. Here is an example of a short batch file that continues executing three IxChariot tests, one after another.

```
:top
 RUNTST tests/test1.tst
 RUNTST tests/test2.tst
 RUNTST tests/test3.tst
 goto top
```

Because `RUNTST` is run from the command line, it is easy to combine it with other networking tools to build even larger, more complex tests. See *RUNTST: Running Tests* on page 5-67 for more information on its operation.

FMTLOG: Formatting Binary Error Logs

Related Topics
The Error Log Viewer on page 12-8

Whenever one of the console programs encounters a problem, it logs the problem information to an error log file at the Console. Similarly, whenever one of the endpoint programs encounters a problem it cannot report to the Console, it logs that problem to an error log file at the endpoint.

To view an error log, you can use the Error Log Viewer or the command-line program named `FMTLOG`. You can also use the Error Log Viewer to view `runtst.log` or `clonetst.log`. See *The Error Log Viewer* on page 12-8 for more information.

As the following table illustrates, each console program logs problems to a different log file with the extension `.log`:

Table 5-13.   Log file usage

Console Program	Log Filename
IxChariot Console	`Chariot.log`
`RUNTST`	`runtst.log`
`CHRAPI`	`chrapi.log`
`CLONETST`	`clonetst.log`
Endpoints	`endpoint.log`

In addition to `Chariot.log`, the `RUNTST`, `CHRAPI`, and `CLONETST` error logs at the Console are always written to a default directory: C:\Documents and Settings\User\My Documents\IxChariot, which can then be changed from **User Settings/Directories** (see *Directories Tab*).

For Windows endpoints, the endpoints' error logs are written to the directory where the endpoints are installed. For the location of the endpoints' error logs for

other types of endpoints, please see the corresponding endpoint chapter in the IxChariot Performance Endpoints guide.

The program `FMTLOG` reads from a binary log file, and writes its formatted output to `stdout`. Here is the syntax of the `FMTLOG` command:

```
FMTLOG log_filename >output_file
```

The `FMTLOG` program is installed at the Console in the `Ixia\IxChariot` program folder and at the endpoints. See the *Performance Endpoints* guide for additional information on running `FMTLOG` on the platform you are using.

FMTTST: Formatting Test Results

Related Topics

Output Templates on page 5-65

The command-line program named `FMTTST` lets you format your test results. It reads its input from an IxChariot test file. It can create output in four different forms: ASCII text, an HTML Web page, PDF, or a .CSV spreadsheet file. When `FMTTST` writes spreadsheet output, it appends a file extension of `.CSV` to the name of the IxChariot test file you specify.

Here is the syntax of the `FMTTST` command:

```
FMTTST test_filename [output_filename] [-v [-o] [-s]
[-d] | [-h [-c | -t template_name]]] | [-p [-c | -t
template_name]]] [-q]
```

The `tst_filename` parameter is the name of the IxChariot test file to be formatted. The `output_filename` is the file to which all test output is written. If no `output_filename` is supplied, output is directed to `stdout`. The template name parameter represents a template containing print/export options created at the IxChariot Console. See *Output Templates* on page 5-65 for more information on working with output templates.

Here are the `FMTTST` command-line options:

Table 5-14. FMTTST Command-Line Options

FMTTST option	FMTTST Option Description
-h	Creates HTML output. This flag controls the format of the output. If you use this flag, you cannot use the `-s` flag.
-v	Creates a comma-separated output (with file extension `.CSV`). You can select which aspects of the tests to export by specifying the specific `.CSV` options described below. If you use this flag without specifying `.CSV` specific flags, the entire contents of the test are used to create the output. If you use this flag, you cannot use the `-c` or `-t` flag to specify the print/options for the results.

Table 5-14.   FMTTST Command-Line Options (Continued)

FMTTST option	FMTTST Option Description
-c	Generate the output according to the export configuration last used in the IxChariot Console. The -c flag exports the test to TEXT format, or, in combination with the -h flag, to HTML and with the -p flag to PDF, using the custom configuration settings that were last selected at the IxChariot Console. This is useful in limiting the output to the exact data you are interested in. This flag controls what print/export options to use for the results. If you use this flag, you cannot use the -s flag.
-p	Creates PDF output. This flag controls the format of the output. The output_filename parameter is compulsory in this case.
-t	Creates output based on the print/export options saved in an output template. Enter the name of the output template after this parameter. This flag controls what print/export options to use for the results. If you use this flag, you cannot use the -c flag.
-q	Run in quiet mode. There is no confirmation for file overwrites.

- To generate HTML output, supply the -h flag

- To generate PDF output, supply the -p flag

- To generate spreadsheet output in the .CSV file format, supply the -v flag

- To generate ASCII text, run FMTTST with no options

- To generate output based on the print/export options saved in an output template, supply the -t flag and the name of the output template you want to use

For example, to see the formatted results for the IxChariot file Tests\Filexfer.tst on the screen—a screen at a time— enter the following:

```
FMTTST TESTS\FILEXFER.TST | more
```

You can also use FMTTST to create Web pages containing test results. The output files contain the HTML tags needed for any Web browser that supports tables. Use the -h flag to generate HTML output. For example:

```
FMTTST TESTS\FILEXFER.TST C:\WEB\XFER1.HTM -h
```

Graphs are exported and linked to the HTML page with the .gif file format. Files in .gif files are written to the current directory. The following filenames are used for the .gifs.

Table 5-15.   Filename Usage

Graph Type	Filename
throughput graph	<testname>_throughput.gif
transaction rate graph	<testname>_trans_rate.gif
response time graph	<testname>_resp_time.gif

You can use `FMTTST` to create spreadsheet output to the `.CSV` file format. This format can be read by modern spreadsheet programs, such as Excel or Lotus 1-2-3.

When you export to the `.CSV` file format, you can use the `.CSV` flags to specify which aspects of the tests you want to export. If you do not set one of these flags, all aspects of the test are exported. These flags can only be set when using the `-v` flag.

Here are the `FMTTST .CSV` options:

Table 5-16.   FMTTST .CSV Options

.CSV option	.CSV Option Description
`-o`	Provides a summary of any results and your run options.
`-s`	Provides information contained in the Test Setup tab of the Test window (Pair Summary table)
`-d`	Provides the timing records for the pairs in your test (Pair Details table)

To export all aspects of the test to `.CSV` file format, enter the following:

```
FMTTST TESTS\FILEXFER.TST TESTS\FILEXFER.CSV -v
```

In this example, the spreadsheet output is written to file `Tests\FILEXFER.CSV`.

To specify which aspects of the test to export to the `.CSV` file format, enter the flag for the aspect you want to export after the `-v` flag. For example, to export the Pair Summary table, enter the following:

```
FMTTST TESTS\FILEXFER.TST TESTS\FILEXFER.CSV -v -s
```

See *Export Options for .CSV Files* on page 5-63 for more information on the `.CSV` file format.

The `FMTTST` program is installed at the Console in the `Ixia\IxChariot` program folder.

CLONETST: Replicating Pairs in a Test

The command-line program named `CLONETST` lets you build complex tests with very little work. You also have the option to use the IxChariot Visual Test Designer to build complicated or large tests, but `CLONETST` lets you build them from the command line. See *Using CLONETST to Add Flexibility* on page 5-74 for suggestions on using `CLONETST`.

`CLONETST` recognizes the total number of pairs based on the sequential numbering, not the actual number of pairs. For example, you have 8 pairs, numbered 1-8, and you delete pair 5. If you attempt to clone pair 8, `CLONETST` returns an error stating that the pair doesn't exist. `CLONETST` counts the number of actual pairs, thus pair 8 to `CLONETST` is actually pair 7.

You start by building a test at the Console, adding a few pairs, and saving that test to a file. The `CLONETST` program takes two files as input:

- The test file you created at the Console;
- A text file containing a list of pair numbers and network addresses.

`CLONETST` reads the template pairs from your test file, and creates a third file. This third file is an IxChariot test file, created by replacing the network addresses in the first file with those it read from the second file.

Thus, the `CLONETST` program requires three parameters:

- Original test filename (binary IxChariot file)
- Clone list (ASCII text file)
- New test filename (binary IxChariot file)

For example:

```
CLONETST input.tst clone.lst output.tst
```

You specify three items in each line of the second file (named `clone.lst` in this example):

- The pair number to copy from the original test;
- The Endpoint 1 and 2 network addresses, used to replicate the original pair.

These three items must be specified together on a line within the input file, separated by spaces. Blank lines in this file are ignored.

The pair number is the sequential pair number in the list. If you have added or deleted pairs in the test, the pair number may not match the sequential pair number.

You cannot use `CLONETST` to build tests with multicast groups. Use the Console to build tests with multicast groups.

Here's an example line in a clone list file:

```
1 NewFromName NewToName
```

Here's another example, showing how you might create a test with two endpoint pairs, using `CLONETST` to produce a test file with four endpoint pairs. If the original test file contained two pairs, such as the following:

```
Pair 1 – 172.16.1.2 172.17.2.2 TCP Filercvl.scr …
Pair 2 – 172.16.1.3 172.17.2.3 TCP FTPput.scr …
```

and the `clone.lst` file contained the following:

```
1 192.168.33.10 10.99.98.97
1 192.168.33.11 10.99.98.96
1 192.168.33.12 10.99.98.95
2 192.168.66.20 10.88.87.86
2 192.168.66.21 10.88.87.85
2 192.168.66.22 10.88.87.84
```

the resulting `output.tst` file would look like this:

```
Pair 1 – 192.168.33.10 10.99.98.97 TCP Filercvl.scr …
Pair 2 – 192.168.33.11 10.99.98.96 TCP Filercvl.scr …
Pair 3 – 192.168.33.12 10.99.98.95 TCP Filercvl.scr …
Pair 4 – 192.168.66.20 10.88.87.86 TCP FTPput.scr …
Pair 5 – 192.168.66.21 10.88.87.85 TCP FTPput.scr …
Pair 6 – 192.168.66.22 10.88.87.84 TCP FTPput.scr …
```

We've used `CLONETST` to prune the UDP pair from the `output.tst` file above. A clever way to use `CLONETST` is to build an original file containing each of the different combinations of protocols and scripts you plan to use. The `clone.lst` file lets you then create tests on the fly, assigning addresses to pairs. You can omit the pairs you don't need for a given test.

The `CLONETST` program is installed at the Console in the `Ixia\IxChariot` program folder.

Using CLONETST to Add Flexibility

Use `CLONETST` to change network addresses in IxChariot tests. It lets you use one IxChariot test file as a template for creating another. If you have a small number of tests that you run, you can combine the different scripts, variables, and protocols in a "master" test file. Using this file as input to `CLONETST`, you can easily create large tests, without using the Console. You can even create new tests programmatically by creating `CLONETST` input files and then executing `CLONETST` from within your code. With batch programs and a command interpreter like Perl or REXX, you can automatically create a flexible collection of tests. See *CLONETST: Replicating Pairs in a Test* on page 5-72 for more information.

Visual Test Designer

The IxChariot Visual Test Designer helps you quickly get started running tests on your network and shows you a graphical representation of your endpoints, groups, pairs, and test connections.

Figure 5-12.   Visual Test Designer
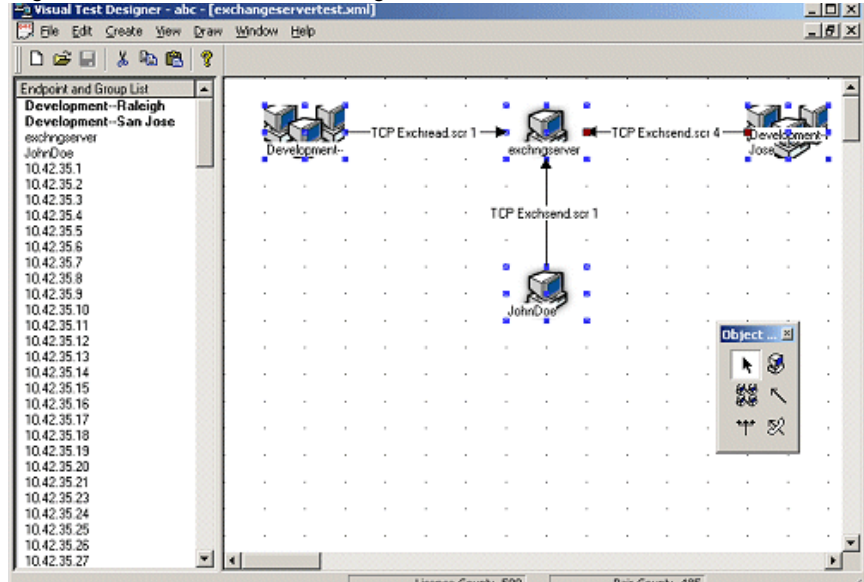
Use the Test Designer to build a very large test of as many as 10,000 pairs of endpoints. Or just pair up all the endpoints in a range of IP addresses to test a whole segment of your network. The Test Designer speeds up the process.

You must have version 5.0 or higher of Microsoft Internet Explorer installed to run the Visual Test Designer. A mouse is also required. Access the Test Designer by selecting **Design** from the File menu.

Getting Started

The Test Designer exists to help you create tests. Once you've added endpoints to a diagram of your test and used the Test Designer's drawing tools to pair them up, you simply export your test and run it. A **tutorial** explaining how to use the Test Designer is available in the Help menu.

The Test Designer has four main components:

• Endpoint and Group List

• Diagram

• Object Bar

• Menus

When you create new endpoints or groups of endpoints, their network addresses (or group names) appear in the Endpoint and Group List, which runs down the left side of the Test Designer window. If you click and drag an endpoint address or group name from the list to the Diagram on the right side of the window,

graphical representations of the endpoints appear. Create pairs of endpoints by using Connectors from the Object Bar.

The Test Designer's Object Bar offers six tools:

Table 5-17.   Test Designers Object Bar

Tool	Description
	Select a tool
	Create an endpoint
	Create an endpoint group
	Create a hardware performance endpoint
	Connect endpoints or groups (create endpoint pairs)
	Add a multicast connection
	Add a voice over IP connection

As a first step, you should add some endpoints to the Diagram that your test will use. Here's how to create an endpoint:

1. On the Object Bar, click the Endpoint tool, then click on the Diagram to drop it.

2. In the Create an Endpoint dialog box, choose the type of address you want to enter for the endpoint. From the **Address Type** list, choose Single IP Address. Or skip to step 4 if this computer has multiple virtual IP addresses.

3. Enter the network address of the endpoint in the Network Address field. Enter a DNS hostname, such as `www.ixiacom.com;` or the IPv4 address of the endpoint computer in dotted notation, such as `135.25.25.5;` or the IPv6 address of the endpoint computer in IPv6 notation, such as `2002:0:13FF:09FF::2.`

4. (Optional) If this particular endpoint computer contains multiple virtual addresses, choose **Range of Virtual IP Addresses** from the **Address Type** list. Enter the range of IP addresses in dotted notation in the fields labeled "**From**" and "**To**." Enter the endpoint's Console to Endpoint 1 address, the address where the Console will contact these virtual endpoints.

5. Click **OK** to save the endpoint. Its network address appears in the Endpoint and Group List, or, in the case of endpoints with virtual addresses, its Console to Endpoint 1 address appears. The endpoint's icon appears in the Diagram.

To remove an endpoint or endpoint group, highlight it in the List and click **Delete**, or click the icon you want to remove and click **Delete**; either option removes both the icon and the entry in the List. To remove an icon from the Diagram without removing its address from the List, click the icon to select it and click **Remove from Diagram** on the Edit menu.

Use Ixia Network Configurations

You can create, use, and clear Ixia network configurations from within the Visual Test Designer. Visual Test Designer provides the following options from the **Edit** menu:

• New Ixia Network Configuration…

 When you select New Ixia Network Configuration, IxChariot starts Stack Manager, allowing you to define a new Ixia network configuration. Once you have defined the configuration and assigned the ports, the port groups and the IP addressed belonging to those port groups appear in the Visual Test Designer Endpoint and Group List window.

• Select Ixia Network Configuration…

 When you select Select Ixia Network Configuration, IxChariot opens the Select an Ixia Network Configuration window, allowing you to load an Ixia network configuration file. Once you have loaded the configuration, the port groups and the IP addressed belonging to those port groups appear in the Endpoint and Group List window.

• Edit Ixia Network Configuration…

 When you select Edit Ixia Network Configuration, IxChariot starts Stack Manager, allowing you to modify the selected Ixia network configuration.

• Clear Ixia Network Configuration…

 When you select Clear Ixia Network Configuration, IxChariot clears the port groups and the IP addressed belonging to those port groups from the Endpoint and Group List window.

Create or Edit an Endpoint

When you use the Endpoint tool to add a new endpoint to the Diagram, Visual Test Designer opens the Create an Endpoint dialog box in which you specify the following parameters. You can modify these same parameters by editing the endpoint (by selecting **Edit** > **Edit…**).

• Endpoint Address

 An endpoint computer's network address, or a set of addresses assigned to that computer's network interface card (NIC). Enter a DNS hostname, such as `www.ixiacom.com`; or the IPv4 address of the endpoint computer in dotted notation, such as `135.25.25.5`; or the IPv6 address of the endpoint computer in IPv6 notation, such as `2002::2`.

• Address Type

 The network addressing scheme that applies to this endpoint computer's NIC. A single computer can be assigned multiple IP addresses for testing purposes. Choose Single IP Address or Range of Virtual IP Addresses from the list. Enter the address(es) in the Network Address or From and To fields.

• Increment by (range of addresses only–optional field)

This field lets you delineate separate subnets or classes of address within a single range. The following table contains some examples:

Table 5-18.   Example Address Ranges

Range of Addresses	Increment By	Addresses Created
From: 10.10.10.1 **To**: 10.10.10.10	0.0.0.1	10.10.10.1 10.10.10.2 10.10.10.3 10.10.10.4 10.10.10.5 10.10.10.6 10.10.10.7 10.10.10.8 10.10.10.9 10.10.10.10
From: 10.10.1.1 **To**: 10.10.10.10	0.0.1.0	10.10.1.1 10.10.2.1 10.10.3.1 10.10.4.1 10.10.5.1 10.10.6.1 10.10.7.1 10.10.8.1 10.10.9.1 10.10.10.1
From: 1.1.1.1 **To**: 1.1.2.2	0.0.1.1	1.1.1.1 1.1.1.2 1.1.2.1 1.1.2.2
From: 1.1.1.1 **To**: 1.5.10.5	0.2.0.2	1.1.1.1 1.1.1.3 1.1.1.5 1.3.1.1 1.3.1.3 1.3.1.5 1.5.10.1 1.5.10.3 1.5.10.5

• Setup Address

   The IPv4 or IPv6 network address at which the endpoint should be contacted with test setup information, if different from address specified above. See *Cloning Hardware Performance Pairs* on page 5-25 for more information. This is the address by which IxChariot identifies the endpoint containing the virtual addresses, if any are used.

• Contact Endpoint Using Network Address

   Instructs the IxChariot Console to use the endpoint's IP address to contact it when it acts as E1. Instructs E1 to use the endpoint's IP address to contact it when it acts as E2. If this box is checked, you don't need to enter an address below.

The option to enter a Range of Virtual IP addresses is included in case you want to create tests with many endpoint pairs, but you only want to use a few endpoint computers to represent those pairs. See *Configuring Virtual Addresses on Endpoint Computers* on page 8-15 for a full discussion.

Create or Edit a Hardware Performance Endpoint

When you use the Hardware Performance Endpoint tool to add a new hardware performance endpoint to the Diagram, Visual Test Designer opens the Create a Hardware Performance Endpoint dialog box in which you specify the following parameters. You can modify these same parameters by editing the hardware performance endpoint (by selecting **Edit** > **Edit…**).

*   Management address

    The network address at which the endpoint should be contacted with test management information. See *Cloning Hardware Performance Pairs* on page 5-25 for more information. This is the address by which IxChariot identifies the endpoint containing the virtual addresses, if any are used.

*   Network Address

    An endpoint computer's network address, or a set of addresses assigned to that computer's network interface card (NIC). Enter a DNS hostname, such as www.ixiacom.com; or the IPv4 address of the endpoint computer in dotted notation, such as 135.25.25.5; or the IPv6 address of the endpoint computer in IPv6 notation, such as 2002::2.

*   Address Type

    The network addressing scheme that applies to this endpoint computer's NIC. A single computer can be assigned multiple IP addresses for testing purposes. Choose Single IP Address or Range of Virtual IP Addresses from the list. Enter the address(es) in the Network Address or From and To fields.

*   Increment by (range of addresses only–optional field)

    This field lets you delineate separate subnets or classes of address within a single range. See Figure 5-18 on page 5-78 for some examples.

The option to enter a Range of Virtual IP addresses is included in case you want to create tests with many endpoint pairs, but you only want to use a few endpoint computers to represent those pairs. See *Configuring Virtual Addresses on Endpoint Computers* on page 8-15 for a full discussion.

Create or Edit a Group of Endpoints

When you use the Group tool to add a new endpoint group to the Diagram, Visual Test Designer opens the Create an Endpoint Group dialog box in which you specify the following parameters. You can modify these same parameters by editing the endpoint group (by selecting **Edit** > **Edit…**).

*   **Endpoint Group Name**

    The name of the group of endpoints you are creating. Identifies the icon that appears on the Diagram after you add endpoints to the group. Required field.

*   **Available Endpoints and Groups**

    A list of endpoints and endpoint groups that you've already created. Highlight the endpoint(s) or group(s) you want to add to the new group you are creating and click **Add** to add them to the group.

• **Remove**

Removes an endpoint or endpoint group from the endpoint group you are creating. Does not delete the endpoint from the Endpoint List.

• **Create New Endpoint**

Opens the Create an Endpoint dialog box. See *Create or Edit an Endpoint* on page 5-77.

• **Create Multiple Endpoints**

Opens the Create Multiple Endpoints dialog box.

Click **OK** to save the endpoints in the Endpoint List. Their group name appears in the Endpoint List, and a set of grouped icons appears on the Diagram.

Create or Edit Connectors

Related Topics

Create or Edit a VoIP Connector on page 5-83
Adding or Editing an Endpoint Pair on page 5-19
Normal Endpoints Pairs on page 5-80
Hardware Performance Endpoints Pairs on page 5-82

Connectors transform endpoints and groups of endpoints into IxChariot test pairs. Click and drag the connector tool from the Object Bar to draw a connector between two endpoints, between two groups of endpoints, or between an endpoint and a group in the Diagram.

When you add or edit a connector, you must specify an application script, the network protocol, and other optional parameters to use in tests with the pair(s) you are creating. When you export and run your test, IxChariot instructs all connected endpoints to run the selected scripts, using the protocols you selected.

When you add or edit a hardware performance connector, you must select a stream file to use in tests with the pair(s) you are creating.

Normal Endpoints Pairs

When a connector is used to create or edit an endpoint pair between non-hardware performance pairs, the following fields are available on the dialog. When a hardware performance pair is created, the dialog fields are as described in *Hardware Performance Endpoints Pairs*.

• **From Endpoint or Endpoint Group**

The endpoint or endpoint group where you began to draw the connector. This endpoint or group will serve as Endpoint 1 in your test pairs. Re-draw the connector to change this designation.

• **To Endpoint or Endpoint Group**

The endpoint or endpoint group where you attached the connector. This endpoint or group will serve as Endpoint 2 in your test. Re-draw the connector to change this designation.

- **Network Protocol**

  The network protocol to use in tests with these pairs. Click TCP, RTP, or UDP in the list or select one of the IPv6 protocols if you're running the IPv6 Test Module.

- **Service Quality (optional)**

  The quality of service (QoS) to use in tests with these pairs of endpoints. If you've defined QoS templates, click one in the list. Optional field.

- **Console to E1 protocol**

  The IP protocol to use for management communications (between the console and Endpoint 1). The choices are IPv4 and IPv6.

- **Select Script**

  Specifies the application script to run between the endpoint pairs in IxChariot tests. Click **Browse** to select from the folders of available application scripts.

- **Connector Count**

  - **Replications**

    Enter a value in this field to multiply the number of pairs you are creating with a single connector definition.

  - **Pair Count**

    Shows how many actual pairs you are creating when you change the number of replications. For example, if you are creating a group of 30 endpoints and are connecting them to a single endpoint 3 times (by choosing 3 Replications), you are creating 90 pairs.

- **Group Connector Properties**

  Lets you choose how many pairs are created with this connector. Only available if both of the objects being connected are endpoint groups, or endpoints with multiple virtual addresses assigned.

  - **All-to-All Pairing**

    Creates a full-mesh of pairs from every endpoint to every other endpoint. For example, if Endpoint Group A contains 20 endpoints and Endpoint Group B contains 10 endpoints, 200 pairs are created if the Replications count is 1.

  - **One-to-One Pairing**

    Connects each endpoint to another endpoint, using each endpoint in a single pair. If the groups contain different quantities of endpoints, the number of pairs created is equal to the number of endpoints in the smaller group. For example, if Endpoint Group A contains 20 endpoints and Endpoint Group B contains 10 endpoints, only 10 pairs are created if the Replications count is 1.

- **Connector Comment**

  - **Use Default Comment**

    Assigns the default comment to the connector. Default comments consist of the network protocol, the application script name, and the number of replications.

• **Comment**

Identifies the connector when it is shown in the Diagram.

Hardware Performance Endpoints Pairs

When a connector is used to create or edit a hardware performance endpoint pair, the following fields are available on the dialog. When a non-hardware performance pair is created, the dialog fields are as described in *Normal Endpoints Pairs*.

• **From Hardware Performance Endpoint**

The hardware performance endpoint where you began to draw the connector. This endpoint will serve as Endpoint 1 in your test pairs. Re-draw the connector to change this designation.

• **To Hardware Performance Endpoint**

The hardware performance endpoint where you attached the connector. This endpoint will serve as Endpoint 2 in your test. Re-draw the connector to change this designation.

• **Stream**

   • **Select Stream**

   Use this button to select the Ixia stream to be sent from Endpoint 1 to Endpoint 2.

   • **Override stream line rate**

   You may override the stream line rate for the selected stream by checking this box and filling in a percentage value between 0 and 100.

• **Measure hardware performance pair statistics**

Checking this box will cause the hardware performance pair to save additional statistics, including latency.

• **Connector Count**

   • **Replications**

   Enter a value in this field to multiply the number of pairs you are creating with a single connector definition.

   • **Pair Count**

   Shows how many actual pairs you are creating when you change the number of replications. For example, if you are creating a group of 30 endpoints and are connecting them to a single endpoint 3 times (by choosing 3 Replications), you are creating 90 pairs.

• **Connector Comment**

   • **Use Default Comment**

   Assigns the default comment to the connector. Default comments consist of the network protocol, the application script name, and the number of replications.

• **Comment**

Identifies the connector when it is shown in the Diagram.

Click **OK** to add the connector(s) to the Diagram. The number of pairs in your test shows on the Status bar in the lower right corner of the Diagram.

Create or Edit a VoIP Connector

Related Topics

Voice over IP Testing on page 10-34
Create or Edit Connectors on page 5-80
Codec Types on page 10-43
Adding or Editing a VoIP Endpoint Pair on page 10-38
Normal VoIP Endpoints Pairs on page 5-83.
VoIP Hardware Performance Endpoints Pairs on page 5-84.

Voice over IP connectors create VoIP endpoint pairs that emulate VoIP calls on your network. The Create a VoIP Connector dialog box lets you select the parameters that match your VoIP application. You can modify these same parameters by editing the VoIP connector (by selecting **Edit** > **Edit…**).

Not all configuration parameters available for VoIP endpoint pairs are available in the Test Designer. To configure a jitter buffer, for example, add the VoIP endpoint pair from the Test window.

For extensive information about the available parameters, see the topics under *Voice over IP Testing* on page 10-34.

Normal VoIP Endpoints Pairs

When a VoIP connector is used to make an endpoint pair between non-hardware performance pairs, the following fields are available on the dialog. When a hardware performance pair is created, the dialog fields are as described in *VoIP Hardware Performance Endpoints Pairs*.

• **From Endpoint or Endpoint Group**

The endpoint or group of endpoints that will act as Endpoint 1 in your VoIP pairs. E1 makes the VoIP call.

• **To Endpoint or Endpoint Group**

The endpoint or group of endpoints that will act as Endpoint 2 in your VoIP Replications

• **Number of concurrent calls**

The number of simultaneous calls IxChariot will emulate during tests between these endpoints. Values must be between 1 and 10,000, inclusive. Default value is 1.

• **Pair count**

Shows how many pairs were created when "Number of concurrent calls" was set.

- **Codec**

  The type of codec used on your network. Choose one of the supported codecs from the list. Refer to *Codec Types* on page 10-43 for more information.

- **Use silence suppression/Voice Activity Rate**

  Emulates the effects of silence suppression (also called voice activity detection) on the line during the VoIP test. Refer to *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information.

- **Override delay between voice datagrams**

  Determines the datagram size to be used in the VoIP test. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information. Optional field.

- **Timing record duration**

  The length of each timing record. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information.

- **Use IPv6 protocol**

  If checked, tells the Console to use the IPv6 version of RTP as the protocol for these VoIP endpoint pairs.

- **Service quality (optional)**

  Emulates the effects of a Quality of Service (QoS) scheme on call quality. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information. Optional field.

- **Group Connector Properties**

  Lets you choose how many pairs are created with this connector. See *Create or Edit Connectors* on page 5-80 for more information.

- **Connector Comment (optional)**

  Identifies the connector on the Test Designer Diagram.

- **Use default comment**

  By default, the Connector Comment is "VoIP" plus the type of codec, number of concurrent calls, and service quality selected.

VoIP Hardware Performance Endpoints Pairs

When a VoIP connector is used to make a hardware performance endpoint pair, the following fields are available on the dialog. When a non-hardware performance pair is created, the dialog fields are as described in *Normal VoIP Endpoints Pairs*.

- **From Hardware Performance Endpoint**

  The hardware performance endpoint that will act as Endpoint 1 in your VoIP pairs. E1 makes the VoIP call.

- **To Hardware Performance Endpoint**

  The hardware performance endpoint that will act as Endpoint 2 in your VoIP Replications

- **Concurrent voice streams**

   The number of simultaneous calls IxChariot will emulate during tests between these endpoints. Values must be between 1 and 10,000, inclusive. Default value is 1.

- **Pair count**

   Shows how many pairs were created when "Number of concurrent calls" was set.

- **Codec**

   The type of codec used on your network. Choose one of the supported codecs from the list. Refer to *Codec Types* on page 10-43 for more information.

- **Override delay between voice datagrams**

   Determines the datagram size to be used in the VoIP test. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information. Optional field.

- **Service quality (optional)**

   Emulates the effects of a Quality of Service (QoS) scheme on call quality. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information. Optional field.

- **Connector Comment (optional)**

   Identifies the connector on the Test Designer Diagram.

- **Use default comment**

   By default, the Connector Comment is "VoIP" plus the type of codec, number of concurrent calls, and service quality selected.

Create or Edit a Multicast Connection

Related Topics
IP Multicast Testing on page 10-9
Adding or Editing a Multicast Group on page 5-27

You can set up a Multicast Group with the Visual Test Designer by using the Multicast Connection icon on the Object Bar. Click on an endpoint or endpoint group, drag the connection icon to another endpoint or endpoint group, and click to connect them. The multicast connector that appears is identified by the multicast address and multicast port you entered.

For more information about IP Multicast testing with IxChariot, refer to *Emulating IP Multicast Applications* on page 10-9.

- **Test Parameters**

   Shows the addresses of the endpoints being connected or the name of the endpoint group.

- **Endpoint 1**

   The sender of the IP Multicast data. Must be a single endpoint.

- **Endpoint 2**

   The recipient or group of recipients of the IP Multicast data. Enter an endpoint address, then click **Add** to add it to the multicast group.

- **Network Protocol**

  The protocol to use when sending the IP Multicast data. Choose UDP or RTP.

- **Service Quality (optional)**

  The quality of service (QoS) to use in tests with these pairs of endpoints. If you've defined QoS templates, click one in the list. Optional field.

- **Select Script**

  Browses your Script directory, where IxChariot's application scripts are saved. You must use a streaming script for an IP Multicast test.

- **Multicast Address**

  For IPv4, this is the Class D address to use for the IP Multicast test. Valid IP Multicast addresses are 224.0.0.0 through 239.255.255.255. We recommend using addresses beginning with 225.0.0.0 or higher because many addresses beginning with 224 are reserved for router usage.

  For IPv6, multicast addresses have a prefix of FF00::/8. Within the reserved multicast address range of FF00:: to FF0F::, the addresses listed in Table 5-19 are assigned to identify specific functions:

Table 5-19.   IPv6 Multicast Address Usage

Range	Usage
FF01::1	All nodes within the node-local scope.
FF02::1	All nodes on the local link.
FF01::2	All routers within the node-local scope.
FF02::2	All routers on the link-local scope.
FF05::2	All routers in the site.
FF02::1:FFXX:XXXX	Solicited-node multicast address, where XX:XXXX represents the last 24 bits of the IPv6 address of the node.

- **Multicast Port**

  Identifies one of the possible destinations within a given host computer. IxChariot uses the combination of the multicast address and multicast port to uniquely identify a multicast group.

Create Multiple Endpoints

Related Topics

Create or Edit an Endpoint on page 5-77

On the Create menu, click **Multiple Endpoints** to open the Create Multiple Endpoints dialog box. When you fill in the fields provided, the new endpoints appear in the Endpoint List. To have them appear in the Diagram, click and drag them from the List to the Diagram.

- **Endpoint Addresses**

  The address range of the endpoints you want to add to the Diagram. In the fields labeled **From** and **To**, enter a range of valid IP addresses, using either IPv4 dotted notation (such as `135.25.25.5`) or IPv6 notation (such as `2002::2`).

- **Increment by (optional field)**

  Lets you delineate separate subnets or classes of address within a single range. See *Create or Edit an Endpoint* on page 5-77 for an example.

- **Endpoint Group Name (optional field)**

  Places all endpoints in the specified range into an endpoint group. Can be an existing group or a new group. If you are planning to build a very large test, it is easier to create pairs if you place endpoints in endpoint groups.

To modify the endpoint group, select it from the Endpoint List, then select **Edit** > **Edit…**

Saving Test Designer Definitions

The Test Designer saves designs in extensible markup language, XML. When you **Save** a test, the Test Designer saves your data with the file extension `.xml` so that you can view your Diagram in a Web browser. To save your data as an IxChariot test, click **Export to IxChariot Test**. The Test Designer then saves your data in a `.tst` file.

> **Note:** IxChariot test files cannot be loaded into the Test Designer.

If your test contains more than 1250 pairs, you should read the topics in Chapter 8, *Large-Scale Tests in IxChariot* before running your test.

Export and Run a Test

When you've finished setting up your test, you can save it and export it directly to IxChariot. The File menu offers two options: you can export the test and run it later (**Export to IxChariot Test**), or you can export and run it right away (**Export and Launch IxChariot**).

Once you've completed your Diagram, you can save it as an IxChariot test. Click **Export to IxChariot Test**. The Test Designer saves your data in a `.tst` file, which you can open in IxChariot. The Test Designer automatically saves the test with IxChariot's default Run Options. If you changed the default Run Options in the Change User Settings notebook, these are the options the Test Designer uses. It Is therefore a good idea to set these global options beforehand.

When you click **Export and Launch IxChariot**, you export your test to an IxChariot test (`.tst`) file and also open the IxChariot Test window. Your test appears in the Test window.

If your test contains more than 1250 pairs, you should read the topics in Chapter 8, *Large-Scale Tests in IxChariot* before running your test.

Viewing Diagram Objects, Tool Bars, and Windows

The IxChariot Visual Test Designer includes standard Windows options for viewing and organizing the endpoints and groups of endpoints you've added to the Diagram. You can also control how open windows are displayed.

- **Draw menu**

  Lets you change your diagram organization.

  - **Order**

    When your Diagram contains many endpoints, you can place their icons over each other by clicking and dragging. To view an endpoint or group icon that's been partially obscured, first click it to select it. Then click one of the following menu items:

    Table 5-20.Object Positioning Menu Items

Command	Description
Bring to Front	Places a selected icon in front of another icon.
Send to Back	Places a selected icon behind another icon.
Bring Forward	Places a selected window first in refresh order.
Send Backward	Places a selected window last in refresh order.

  - **Zoom In or Out**

    Gives a closer or broader view of the Diagram objects.

- **View menu**

  The View menu lets you hide the Object Bar or the Standard Windows Toolbar in the Test Designer window.

  - **Toolbars**

    Lets you select or deselect the toolbars you want to see.

  - **Status Bar**

    Lets you hide the Status bar, which is shown by default. It gives you extra help with each menu item and keeps a running total of the number of pairs you've added to your list.

- **Window menu**

  The Window menu lets you control how multiple tests are displayed. It contains standard Windows commands that determine how separate windows containing various test objects are arranged and viewed.

  - **Cascade**

    If you have multiple windows open, arranges them so that their title bars are visible.

  - **Tile**

    If you have multiple windows open, arranges them so that they share the diagram space.

- **Arrange Icons**

   If you have more than one window open and minimized, arranges window icons in the diagram space.

Shortcut Keys for the Test Designer

You can use the following keys and key combinations in the IxChariot Visual Test Designer instead of using the mouse.

Table 5-21.   Shortcut Keys for the Test Designer

Key or Key Combination	Command Invoked
F1	Get help for the Test Designer.
F2	View Object Bar.
F9	Show the keys and key combinations available in a window.
F11	Open the About IxChariot dialog box, which shows your version and build level, and gives you product support information.
Ctrl+B	Send selected object(s) to back.
Ctrl+C	Copy the test setup for one or more pairs to the clipboard.
Ctrl+D	Delete the highlighted endpoint or group. You are asked whether you are sure you want to delete that endpoint or group.
Ctrl+E	Create an endpoint.
Ctrl+F	Bring selected object(s) to front.
Ctrl+G	Create endpoint group.
Ctrl+H	Create an hardware performance endpoint.
Ctrl+M	Create multiple endpoints.
Ctrl+N	Start a new test in the Test Designer.
Ctrl+O	Open an existing Test Designer test file.
Ctrl+S	Save this test designer definition (with `.xml` extension).
Ctrl+V	Paste an endpoint or group from the Windows clipboard.
Ctrl+X	Cut an endpoint or group and place it on the Windows clipboard.
Alt+F4	Close any window or dialog box. Has the same effect as clicking the **Esc** key or clicking **Cancel** with the mouse.

6

IxChariot User Settings

This chapter provides a description of the IxChariot global user settings. The user settings gives you a great deal of control over how tests are run and how results are reported.

Topics in this chapter:

Accessing the User Settings

To access the Change User Settings window, select **Tools > Options > User Settings** from the main menu.

User settings include the following:

- Default parameters used in all tests you create
- Default units in which data is recorded and reported
- Warnings you might see while setting up or running tests
- Directories where IxChariot stores test and script files
- Global options for printing and exporting test results
- Settings specific to voice over IP testing
- Settings specific to application groups
- Settings specific to Ixia port configurations
- Test configuration for running tests through firewalls.

All of the user settings are applied globally to IxChariot. However, you can also define test-specific values for the following options: Run Options, Result Ranges, Datagram, and Ixia Port Configuration. Test-specific settings override the global settings. Refer to Chapter 7, *Test Run Options* for information about the test-specific run options.

Run Options Defaults Tab

Related Topics

This notebook page lets you change the default run options for your IxChariot tests. Note that you can modify all of the options that are presented on this tab on a per-test basis (using the Run Options tab accessed from the Run menu). Also note that the options set on the Run Options tab are test-specific and they take precedence over the default settings.

Performance Testing

Related Topics

The checkbox labeled **Set the test run options for performance testing** auto-matically configures testing options to achieve the highest performance results. Checking this box changes the default settings for the following test configuration parameters:

• How to report timings. The default setting lets you see the results in real time, as the test is running. For more accurate performance measurements, it is better to report in batch because batch reporting doesn't consume extra CPU and network resources during the test.

• Poll endpoints. The default setting specifies that the IxChariot Console will poll the endpoints once every minute during a test run. Polling also consumes network resources, creating extra data flows that interfere with performance measurements, so polling is disabled when you check Performance Testing.

Other default run options are already appropriate for performance testing. If you have changed the defaults for **How to end a test run**, **Collecting endpoint CPU utilization**, and **Collecting TCP statistics**, checking the Performance Testing checkbox changes your settings back to achieve higher performance. CPU collection, TCP statistics collection, Data Validation, and the New Random seed option are all automatically turned off for performance testing.

It Is highly recommended that you enable performance settings for tests of more than 500 pairs. See *Setting Run Options for Performance Testing* on page 8-4 for more information.

How to End a Test Run

Related Topics

This section of the Run Options notebook provides three ways to determine when a test run is complete. Some endpoint pairs run much faster than others, depending on the script variable values, endpoint computer CPU speeds, and network equipment. Unless you run a test for a fixed duration, you may find that some pairs have completed all their timings before other pairs have even reported once. You should experiment to get a good balance among pairs. The choices for ending a test run are as follows:

• **Run until any pair ends**

Stops the test run when any endpoint pair completes executing its script or ends with an error. Any timing records received after this first pair completes its script are discarded. This ensures that all timing records used in the calculations were generated while the other endpoint pairs were still executing scripts.

This setting is recommended for performance testing, particularly for highly utilized lines, because it shows measurements taken during times of maxi-

mum contention for shared bandwidth. As soon as a pair finishes its script, all pairs stop taking measurements, which means that no pair has an opportunity to communicate over an empty line.

Sometimes a pair running a streaming script will complete the test with fewer timing records than you specified in the script (in the script's number_of_timing_records parameter). This is because some data may have been lost. A timing record is complete when Endpoint 2 has received enough data to fill (and sometimes overflow) that record. However, one timing record may contain lost data as well as successfully received data. The disparity in the number of bytes sent and bytes received can cause the pair to complete before Endpoint 2 can complete the total number of timing records for which data was sent.

• **Run until all pairs end**

Allows all endpoint pairs to run until they have completed all the commands in their scripts or ended with errors.

Tests using this option may generate misleading data. The problem occurs as scripts finish and there is less competition for available bandwidth. In fact, the last executing script could have the network all to itself and report much better results than it would if other pairs were still executing scripts—the most likely situation under normal operating conditions.

This option is recommended only if the endpoint pairs do not share the same network resources—that is, if they use different routers and physical media.

• **Run for a fixed duration**

With this option, all endpoint pair scripts run for a fixed period of time, ignoring the number_of_timing_records value in their output loop. At the end of the period, the endpoints stop and Endpoint 1s return their results. You can choose values from 1 second to 99 hours, 59 minutes, and 59 seconds. Although the default setting is 1 minute, we recommend 2 to 5 minutes for most performance testing. It Is important to test for a sufficient length of time to generate at least 10 timing records; the records represent data samples, and you need enough samples to ensure test reliability.

You may set a much longer duration for stress testing. However, because application scripts generate roughly 50 timing records per minute on a LAN, setting the duration to long time periods can generate an enormous number of timing records—potentially exceeding the storage capacity of some console computers. We recommend using the Script Editor to tune the inner loops of your scripts so that they generate timing records less frequently. Experiment before running multi-hour, multi-pair stress tests. The run time duration is checked every time an END_LOOP or END_TIMER command executes. This may cause the actual run time to slightly exceed the run time you type here.

The "fixed duration" option for performance testing is useful because it avoids situations in which some endpoint pairs complete their scripts long before the others, leaving the last pairs a nearly empty line over which to communicate. But be careful not to overstress your network with extraordinarily long tests and many pairs.

How to Report Timings

Related Topics
Run Options Defaults Tab on page 6-2

The Run Options dialog box also gives you an opportunity to determine how results of a test will be reported. Because of the extra data flows created between each Endpoint 1 computer and the Console, the way the Endpoint 1 computers report the data in their timing records can affect test results. You can choose between two reporting methods:

- **Batch**

  With this setting, timing records are saved at Endpoint 1 and forwarded to the Console at the end of the test. Results are not displayed until the test completes, or until 500 timing records have been collected (or 300 for VoIP pairs). This keeps network traffic from Endpoint 1 to the Console from interfering with the actual performance measurements on your network. If you want to find out the progress of the test, click the Poll button.

  Batch reporting is always the recommended option for any performance testing.

- **Real-time**

  With this setting, every time a timing record is created, it is sent back to the Console. The Console updates the Test window as the timing records are received, letting you see how the test is progressing. While this is handy for verifying tests, real-time operation can have dramatic, negative effects on the test being run. Results are updated at least every 5 seconds. The specific amount of time between updates depends on the number of pairs in the test. Even if you choose real-time reporting, if you are running a loopback test containing multicast groups in real time, it may take several minutes for the timing records from the test to be shown in the Console.

  We strongly recommend doing performance measurements with batch timings, which avoid the extra network traffic of real-time operation. This extra traffic is doubled when executing a streaming script since the timing records are sent from Endpoint 2 to Endpoint 1, which then forwards the records to the Console. While real-time results look "cool," they consume resources in the network, at the endpoints, and at the Console. The worst-case behavior of real-time reporting occurs with many endpoint pairs, each generating timing records frequently.

- **Console behind Firewall**

  Select this option if there is a firewall between the IxChariot Console and endpoint 1. When you select this option, the IxChariot Console will initiate the connection with endpoint 1 for the duration of the test, thereby ensuring that endpoint 1 will be able to transmit timing records whenever needed.

  If you do not select this option, endpoint 1 initiates a connection with the Console whenever it has timing records to transmit. If there is no firewall, the connection is established successfully. However, if there is a firewall between endpoint 1 and the Console, it will prevent endpoint 1 from establishing the connection, in which case the test results will not be received by the Console.

  Refer to *Firewall Testing* on page 10-13 for more information about test environments that include firewalls.

Polling the
Endpoints

Related Topics

Performance Testing on page 6-2
Factors Affecting Results on page 11-52

When you click **Poll Endpoints Now** on the Run menu during a running test, the Console sends a request to each of the Endpoint 1 computers in the test to return the number of timing records they've generated so far.

There are two good reasons to poll the endpoints while a test is running:

*   The test is reporting results in Batch, and you want to retrieve either a count of timing records that have been collected, or retrieve the actual timing records so that you can view them in the table and on the graph, without waiting for the test to finish. (When reporting results in batch mode, IxChariot normally saves the timing records at Endpoint 1 and forwards them to the Console only at the end of the test or until a set number of timing records have been collected (500 records for regular pairs, 300 records for VoIP pairs).

*   You suspect that one or more Endpoint 1 computers can no longer be reached. If Endpoint 1 is powered off during a test, the Console never actually knows because the Console doesn't maintain a connection with Endpoint 1 while a test is running. Polling forces the Console to initiate contact with each Endpoint 1 computer.

Polling can, however, adversely affect your test results. As a general guideline, you should disable the **Poll endpoints** option for a test that has 500 or more pairs. If polling is enabled for a test with a large number of pairs, IxChariot may return a CHR 0245 error. Further, you should refrain from selecting **Poll Endpoints Now** from the Run menu during the execution of a test that has a large number of pairs.

You can set the following options to control IxChariot polling behavior:

*   **Poll endpoints**

    Select this option to allow IxChariot to poll endpoints during a test. De-select this option to disable polling.

*   **Interval**

    This option allows you to specify the automatic polling interval, in minutes, when the test is run in real-time mode.

*   **Retrieve Timing Records**

    When this option is selected, IxChariot returns the actual timing records when you click **Poll Endpoints Now** on the Run menu. When it is not selected, IxChariot returns only a count of timing records when you click **Poll Endpoints Now** on the Run menu.

    This selection is applicable only to tests run in batch mode. Tests that run in real-time mode always retrieve actual timing records

The more endpoints involved in a test, the less often IxChariot refreshes status changes in the Test window. This reduces the overhead of updating records at the Console. For large tests, this refresh period can take quite a long time. The actual

status of the endpoint (polling or running) may not be evident for a while, even though the run status has changed while the test is running.

Clock synchronization

Related Topics

Ixia Port Access Restriction on page 4-4
One-Way Delay on page 10-47
The One-Way Delay Tab on page 11-27

The clock synchronization options allows you to specify the timing source for the endpoints on a test network. There are three potential timing sources:

• Ixia hardware timestamps:

An Ixia chassis backplane can provide the timing source for all the ports in a chassis chain. In this case, an endpoint obtains an Ixia hardware timestamp each time it sends a time value over the network. The receiving node also obtains an Ixia hardware timestamp, thereby enabling IxChariot to perform the necessary time calculations using the hardware timestamp clock as the reference system. Using Ixia hardware clock synchronization produces the most reliable method of reporting one-way delays.

If you are setting up a test that uses a virtual chassis chain based on Ixia AFD1 GPS clocking, you must select *Ixia hardware timestamps* as the clock synchronization option. (Refer to *Using an AFD1 in an IxChariot Test Network* on page 4-23 for more information.)

• External device:

Clocking for Linux endpoints can be provided by external Network Time Protocol (NTP) servers that are synchronized to the Global Positioning System (GPS).

> **Note:** IxChariot provides external clock synchronization support for Linux endpoints only.

External clock synchronization requires properly-configured NTP servers. IxChariot does not verify that the external timing source is operational or accurate. Consequently, if an NTP server is misconfigured or inoperable, test results will be invalid.

You can use either of the following configurations when using NTP servers and the GPS as the external clocking source:

• Each IxChariot Linux endpoint is configured with Performance Endpoint software, a GPS device, and a properly-configured NTP server.

• Each IxChariot Linux endpoint is configured with Performance Endpoint software, but is not configured with a GPS device and an NTP server. Instead, a GPS device is connected to a separate network node on which the NTP server is running. In this case, the IxChariot endpoints synchronize to the NTP server over a local (very low latency) network. This configuration generally yields lower accuracy than the first configuration.

Using external synchronization as the timing source is generally more reliable than using the endpoint internal timers, but less reliable than using Ixia hardware timestamps. External synchronization is preferred over endpoint inter-

nal timers on high latency networks, whereas endpoint internal timers are recommended for endpoints that are on the same LAN.

• Endpoint internal timers:

When neither Ixia hardware timestamps nor an external device can be used as the clocking source, Endpoint 1 and Endpoint 2 synchronize their time using their internal timers. This synchronization occurs during test initialization. The endpoints periodically synchronize their clocks based on the estimated clock deviation computed from previous synchronizations. (You can force a clock synchronization for each test run by setting the FORCE_CLOCKSYNC keyword in the endpoint.ini files.)

The Run Options provide two clock synchronization options: *Use Ixia hardware synchronization*, and *External synchronization*. You can select both, either, or neither of these to specify the clock synchronization method you will use for your test network. Table 6-1 describes the effect of setting these options.

Table 6-1.   Effect of Clock Synchronization Settings

Options:			
Use Ixia hardware clock synchronization	*External synchronization*	**Preferred Clock Source:**	**Fallback Clock Source:**
Yes	Yes	Ixia hardware timestamps.	External device.
Yes	No	Ixia hardware timestamps.	Endpoint internal timers.
No	Yes	External device.	N/A
No	No	Endpoint internal timers.	N/A

As shown in Table 6-1, the timing source that each endpoint pair uses in a test depends upon the selected options and the availability of that specific source:

• **Use Ixia hardware clock synchronization**

When this option is selected, all pairs attempt to use the Ixia hardware time-stamp as the timing source on the test network. If the Ixia port hardware timestamp is not available to a specific endpoint pair, that pair will use:

• the external clock source, if **External synchronization** is selected, or

• the endpoint internal timers, if **External synchronization** is not selected.

This provides support for tests in which Ixia pairs and non-Ixia pairs are both used.

If available, Ixia hardware clock synchronization is always preferred over external synchronization, which, in turn, is always preferred over endpoint internal timers.

• **External synchronization**

When only this option is selected, all Linux endpoint pairs obtain their timing values from the external timing source. IxChariot does not validate the pres-

ence of or the reliability of the external timing source. Rather, IxChariot assumes that the clocking is provided by an external system.

Management Quality of Service

Related Topics
IxChariot Network Topology on page 2-2
IxChariot Test Process Overview on page 2-5
Chapter 9, *Quality of Service Testing*

The Management Quality of Service section of the Run Options dialog allows you to specify QoS settings for IxChariot management traffic. There are two distinct management QoS settings:

• **Console Service Quality**

  This setting specifies the QoS template that IxChariot will use for management traffic sent from the console to Endpoint 1. This traffic includes the pre-setup and setup data that the console sends to Endpoint 1 prior to the start of the test.

• **Endpoint Service Quality**

  This setting specifies the QoS template that IxChariot will use for all other management traffic: management traffic from Endpoint 1 to the console, and management traffic between Endpoint 1 and Endpoint 2. This traffic includes the pre-setup and setup data that Endpoint 1 sends to Endpoint 2 prior to the start of the test, the test results that Endpoint 2 sends to Endpoint 1 during the test execution, and the test results that Endpoint 1 sends to the console during the test execution.

To specify IxChariot default settings for either or both of these, select the desired QoS templates in the fields on the *Run Options Defaults* tab in the Change User Settings window.

To specify Management QoS settings for a specific test, first open the test, then select the desired QoS templates in the fields on the *Run Options* tab in the Run Options window. The per-test settings override the default settings.

The following points summarize the operation of the Management QoS feature:

• QoS values that are to be used for all management traffic will be set through the IxChariot console—or through the use of IxChariot APIs—and transmitted to the endpoints in the setup phases, before the test begins running. Therefore, the same QoS values will be used for management traffic by all the pairs in a test. Different tests can use different values or even disable the use of QoS for management entirely.

• When a QoS value is selected for management traffic, that value will be used for all management traffic throughout the test, whether or not different pairs use separate logical networks for this traffic.

• There are two separate settings for QoS management traffic: Console Service Quality and Endpoint Service Quality.

• Only TOS and DSCP templates are supported for management traffic.

- The TOS and DSCP QoS templates will be available for use with management traffic only if all the stations involved in the communication (console and endpoints) support them.

- QoS values for management traffic will be used during the sending of all commands: pre-setup, setup, stop, and so forth. Refer to *QoS During the Three-Way Handshake* on page 6-10 for a description of the QoS values used during the initial three-way handshake.

- Management QoS settings will be used for all timing record transmissions and for clock synchronization.

The following limitations apply to the Management QoS feature:

- QoS for management traffic is not supported on the following:

  - hardware performance pairs

  - VoIP hardware performance pairs

  - IPX/SPX (used as the protocol for the management network)

  - IPv6 (used for management traffic between the Console and Endpoint 1).

- Only TOS and DSCP templates are supported for management traffic.

The default Management QoS templates defined for IxChariot at any given time are saved in the registry, and loaded from the registry when the IxChariot Console starts up. Once saved in the registry, the default QoS templates are used in every new test created from that console (unless you select different QoS templates for a specific test).

The QoS settings that you set as run options affect only the current test and are therefore saved in the test file (tst file extension). Loading a particular test file will also load its saved run options, including the preferred values for management QoS. Furthermore, QoS values loaded as run options will override any previous default settings applied to the test.

However, both the registry and the test file contain only the name of the template used for management QoS. The actual template is stored in the servqual.dat file, as are test traffic QoS templates. Therefore, for a template to be applied properly, the description corresponding to its name must be present in this file. This is especially important when relocating tests from one machine to another. If a test uses a custom named template, it is necessary to either copy servqual.dat onto the new machine along with the test, or rebuild the template with known values before running the test.

QoS During the Three-Way Handshake

During the TCP three-way handshake connection establishment process, IxChariot sets a QoS value on the accept socket of the connection. Setting the QoS values for the pre-setup flow requires some processing that is not required for other traffic flows. Specifically, it requires the use of the INITIAL_MANAGEMENT_TOS value specified in the endpoint.ini file. During pre-setup, the console sends a buffer to Endpoint ,1, and Endpoint 1 sends a buffer to Endpoint 2. These buffers contain the QoS values that will be used for

all management traffic during the test. However, the endpoints must parse those buffers to learn the QoS values. Once the pre-setup flow is complete, the endpoints use the QoS values that are specified for all the connections, including the three-way handshake.

During the pre-setup flow, if the console is the TCP sender, the QoS settings are handled as follows:

1. The console (sender) initiates the TCP connection, sending the initial SYN packet. The console will always use the QoS values set in Run Options.

2. Endpoint 1 (the receiver) responds with SYN and ACK. This is where the INITIAL_MANAGEMENT_TOS value from the endpoint.ini file is used. The receiver must use this value because it will not know the Run Options QoS value until it receives the buffer and parses it.

3. The console (sender) responds with an ACK. The connection is established.

4. The console (sender) sends the management buffer containing the QoS value.

5. Endpoint 1 (the receiver) parses the buffer and gets the QoS value. From this point on, the endpoints will use the Run Option QoS value to communicate with the sender.

   Note: For Linux and Unix endpoints, the receiving side sends an acknowledge packet as soon as the buffer is received (before parsing starts). Therefore, this ACK packet will use the QoS marking from endpoint.ini file. (However, if there was a previous run, the QoS markings will be inherited from that run.)

6. The receiver sends a response to the sender. The sender acknowledges. They close the connection. All these use the proper QoS values from run options.

   Notes:
   - When an endpoint is the sender, it will initiate the connection only after it has parsed the needed value from a previously sent buffer. Therefore, it does not require the use of the INITIAL_MANAGEMENT_TOS value from the endpoint.ini file.
   - If subsequent tests are run using different QoS settings, and the endpoints are not reset, the values from the previous run will be used during the pre-setup phase. The QoS values will be reset when the endpoint parses the buffer.
   - On Windows endpoints, if you change the Endpoint QoS value such that no template is specified (or a null QoS value is specified), the endpoint will not be able to reset the QoS value that was set during pre-setup. Therefore, the receiving side will use the value from the previous test throughout the pre-setup flow.

Refer to Chapter 9, *Quality of Service Testing*, for detailed information about creating QoS templates, and for procedures for using Quality of Service for application traffic.

Miscellaneous Run
Options

Related Topics

Polling the Endpoints on page 6-6
Performance Testing on page 6-2

In addition to controlling options for ending a test run, reporting results, and handling initialization failures, the Run Options notebook lets you configure a number of other test parameters:

•   **Collect endpoint CPU utilization**

Instructs IxChariot to collect CPU utilization data for the endpoint computers executing a test. CPU utilization is the percentage of available CPU time spent executing all the currently active processes on a computer. It Is a good indication of available network resources and the degree to which they may be overtaxed. The CPU Utilization percentage is shown in the Percent CPU Utilization of E1 column and the Percent CPU Utilization of E2 column on the Raw Data Totals Tab of the Test window. These columns are only shown if this box is checked. CPU Utilization values are shown only after a pair completes; while a test is still running, "n/a" is shown.

This percentage is an approximation based on CPU utilization samples taken during the test. Sampling starts during the first CONNECT or ACCEPT command in the script and continues until the script is complete. After the script completes, the average of the samples is calculated and reported to the Console.

Tests that collect endpoint CPU utilization should run for longer than 1 second in order to collect meaningful CPU utilization numbers. The endpoint samples the CPU utilization every 500 milliseconds and requires at least two valid samples in order to calculate the percentage of utilization.

For computers with more than one CPU, IxChariot calculates the CPU utilization percentage by adding together the percentages for each CPU and then dividing this amount by the number of CPUs. For example, if a computer contains two CPUs and one CPU is 50% utilized and the second CPU is idle, the CPU utilization of the process is calculated as (50% +0%)/2 =25% CPU utilization.

CPU Utilization is not supported by the endpoints on all operating systems. For Novell NetWare computers that contain more than one CPU, IxChariot returns the CPU Utilization of the first processor only. Consult "Endpoint Capabilities" in the *Performance Endpoints* guide for more information about operating-system support.

•   **Collect TCP statistics**

If you are running tests on an Ixia chassis, you can select "Collect TCP statistics" to instruct IxChariot to collect a set of TCP statistics for the endpoints executing a test. These statistics are collected for TCP packets that include SYN, FIN, ACK, and RST messages, as well as TCP connections, TCP retransmissions, and timeouts.

TCP statistics collection requires IxOS 3.80 or higher and is available only on load modules with a Power-PC 405 or Power-PC 750 processor: TXS8, TXS4, STXS4, SFPS4, ALM-T8, ELM-ST2, TXS2, LM10GE700F1B-P, LM622MR, OLM1000STXS24.

Note also that complete IPv6 TCP statistics are only available in IxOS 4.10 and above. In IxOS 4.0 (and earlier), the following IPv6 TCP statistics are not reported:

- SynReceived
- FinReceived
- FinAckReceived
- ResetReceived

These are marked "N/A" in the TCP statistics tab.

Refer to *Collecting TCP Statistics* on page 10-82 for more information about collecting TCP statistics.

- **Validate data upon receipt**

Instructs all the endpoints in a test to validate each byte they receive. In some test environments, you may not be sure if the data is being transferred correctly from one endpoint to another. Endpoints can validate that what they receive is what they expected to receive by comparing payload bytes to the bytes specified in the script for the `send_datatype` and `control_datatype` variables.

Obviously, validation slows the performance measured at the endpoints. This function should be used for network stress testing and for testing of new hardware and software.

- **Use a new seed for random variables on every run**

The sleep or `transaction_delay` script variable tells the endpoints to pause. The send and receive `buffer_size` variables specify the sizes of the buffers the endpoints use when sending data. If you change the default "Constant Value" to one of the four random distributions—Uniform, Normal, Poisson, or Exponential—the sleep durations and buffer sizes are based on random numbers, which are generated from a "seed." When IxChariot uses the same seed on consecutive runs, the random sleep durations or buffer sizes are generated in the same sequence.

You should have IxChariot use the same seed when you are trying to get the same values for sleep durations or buffer sizes, run after run. Check this box if you want the sequence of randomly selected values to be different on each run.

- **Use fewer connections for test setup**

Instructs the Console to use the fewest possible connections to contact each Endpoint 1 computer during the initialization phase of a test. Recommended for large tests. This option is automatically selected when you add the 501st pair to a test, unless you clear this box. For tests with >1250 pairs, this option cannot be disabled. Test initialization may take slightly longer when this box is checked.

If you change your mind, click **Undo** to reset all the fields in the Run Options notebook to the values you had before you made any changes.

- **Enable Ixia hardware timestamps**

    Select this option when creating a test that uses video pairs. The Ixia hardware timestamps measure the delay factor with much greater precision on IXIA hardware.

- **Number of overlapped sends**

    Enter the number of overlapped sends that you want your scripts to use during test execution. You can enter any value from 2 through 999999. IxChariot will save this setting to the Registry as the default value for new tests.

    Overlapped I/O is a Microsoft Windows feature that allows sending (and receiving) multiple buffers in parallel. This can yield higher throughput and reduce CPU utilization.

Error Handling Defaults Tab

Related Topics
Run Options Defaults Tab on page 6-2
Factors Affecting Results on page 11-52
Performance Testing on page 6-2
How to End a Test Run on page 6-3
How to Report Timings on page 6-4
Pair Reinitialization and Graphs on page 11-7

Three options you can configure in the Run Options dialog box give you some control in failure-prone networks:

- **Stop run on initialization failure**

    Initialization occurs when you click the Run button: the IxChariot Console contacts all Endpoint 1 computers, which in turn contact their Endpoint 2 partners. When you start a test, you are never completely sure whether all the endpoints can be reached. But if your test involves many different endpoints, you may want to run the test even if some of the endpoints are unavailable.

    Checking the **Stop on initialization failure** box stops the run when any endpoint cannot pass all the initialization steps. If you leave the box cleared, the test will be run if at least one endpoint pair can be initialized. Those endpoints that cannot be initialized are omitted from the results and show errors.

- **Connect timeout during test**

    You may be testing in noisy networks, where long connections are frequently dropped. IxChariot retries its connection attempts for the number of minutes you specify here. If that amount of time elapses and a connection still cannot be established, IxChariot declares a connection failure and issues the appropriate error message.

    A connection attempt by an endpoint may consist of more than one Sockets `Connect` call, even if the timeout is set to zero. If the connection failure is due to a transient condition, such as network congestion, the endpoint will issue a fixed number of Sockets `Connect` calls per attempt. If the failure is due to a permanent condition, such as insufficient memory resources, the endpoint will issue one Sockets `Connect` call per attempt. This information applies to TCP connections only. See "Mapping Communication Commands

to APIs" in the *IxChariot Script Development and Editing Guide* for more information about endpoint operating systems using TCP.

A value of 0 minutes means that connection failure is declared after the first unsuccessful series of connection attempts by the endpoints. The timeout option tracks errors encountered on `CONNECT_INITIATE` commands in a script; errors that occur on `SEND`, `RECEIVE`, or other commands will still cause a running test to stop. Thus, this timeout is most helpful in scripts with short connections.

Accepted values are in the 0-999 range.

- **TCP Receive timeout during test**

When this option is enabled, the test will stop if any pair stops due to TCP retransmission timeouts. When you enable this option, you must specify the TCP retransmission timeout interval, in minutes.

Accepted values are in the 0-999 range.

- **Stop test after *x* running pairs fail**

You may want to let some pairs fail while the remainder of the pairs continue executing their scripts. IxChariot implements this option when the test enters the running state. Once in running state, IxChariot lets the specified number of pairs fail before terminating the test.

Accepted values are in the 0-9999 range.

> **Note**: When the Run until any pair ends radio button is selected in the How to end a test run pane in the Run Options Defaults tab, the Stop test after x running pairs fail option is disabled.

- **Allow pair reinitialization for setup**

When you select this option, IxChariot will attempt to reinitialize an endpoint pair that fails during the initialization phase of the test.

The reinitialization feature provides flexibility in test planning and execution. When running large-scale tests (up to 100,000 pairs), it is often desirable—if not necessary—for a test to proceed even if some of the pairs fail during initialization, and to allow IxChariot to reinitialize the pairs that fail.

When you select the *Allow pair reinitialization for setup* option, you must also set the following parameters:

- Try reinitializing *n* times

Specify the number of times that IxChariot should attempt to reinitialize the failed endpoint pair. The default is three attempts.

If IxChariot is not successful in reinitializing the pair after the specified number of attempts, initialization of the pair is considered to have failed. At this point, the pair will be included in the count of failed pairs (refer to the *Stop test after x running pairs fail* parameter above).

Accepted values are in the 1-99999 range.

- Retry reinitializing after *n* milliseconds

Specify the number of milliseconds to wait between reinitialization attempts. The default is to wait ten milliseconds between attempts. Accepted values are in the 1-99999 range.

• **Allow pair reinitialization at runtime**

When you select this option, IxChariot will attempt to reinitialize an endpoint pair that fails during the execution of the test.

The reinitialization feature provides flexibility in test planning and execution. When running large-scale tests (up to 100,000 pairs), it is often desirable—if not necessary—for a test to proceed even if some of the pairs fail during test execution, and to allow IxChariot to reinitialize the pairs that fail. As another example, in many WLAN tests it is not uncommon for a pair to fail if the performance endpoint moves out of the coverage area of the access point (AP) while the test is in progress. By using appropriate reinitialization options, you can allow IxChariot to reinitialize the pairs that fail.

When you select the *Allow pair reinitialization at runtime* option, you must also set the following parameters:

• Try reinitializing *n* times

Specify the number of times that IxChariot should attempt to reinitialize the failed endpoint pair. The default is three attempts.

If IxChariot is not successful in reinitializing the pair after the specified number of attempts, initialization of the pair is considered to have failed. At this point, the pair will be included in the count of failed pairs (refer to the *Stop test after x running pairs fail* parameter above).

Accepted values are in the 1-99999 range.

• Retry reinitializing after *n* milliseconds

Specify the number of milliseconds to wait between reinitialization attempts. The default is to wait ten milliseconds between attempts. Accepted values are in the 1-99999 range.

Datagram Tab

Related Topics

You specify values for the IxChariot default datagram properties on the Datagram tab of the Change User Settings window. Datagram testing refers to testing with the connectionless protocols UDP, IPX, and RTP. The IxChariot Console itself provides for the retransmission of lost data in test with these protocols. IxChariot does not retransmit when using a streaming script.

Expect to do some experimentation to find the best combination of settings for these options. If you change your mind, click **Undo** to reset all the fields to the values you had before you made any changes.

Non-Streaming Script Options

The first three options on the Datagram notebook page apply only to non-streaming scripts:

- **UDP Window Size**

  The number of bytes that can be sent from an endpoint to its partner without an acknowledgment. Calculate the number of datagrams sent in a window by taking the Window Size, dividing by the script's `send_buffer_size`, and rounding this value up. Default value is 1500.

- **Retransmission Timeout Period**

  The number of milliseconds the sender will wait, after sending for the first time or retransmitting a block of data, to receive an acknowledgment that the block was received. Default is 200 ms.

- **Number of Retransmits before Aborting**

  The number of times the sender will re-send a block of data for which an acknowledgment is not received. Default number of retransmissions is 50.

Streaming Script Options

The next two options apply only to streaming scripts.

- **Receive Timeout**

  The number of milliseconds the receiver waits before determining that the streaming script has ended. The default timeout value is 10000 ms. If Endpoint 2 is running on Windows, the minimum receive timeout value is 500 ms.

- **Multicast Time To Live (TTL)**

  Controls the forwarding of IP Multicast packets. Set this value based on how far you want the data forwarded.

  This field defaults to a value of 1 hop. However, the packets cannot cross a router if the TTL value is 1. If you run a test with a TTL less than the number of routers between the endpoints, the test fails and you receive the error message **CHR0216**. You'll need to adjust the TTL to run a test with a multicast group that crosses a router.

Options for RTP Traffic

There is one option that applies only to scripts that generate RTP traffic.

- **Use extension headers for RTP timestamps**

  When you select this option, IxChariot will generate RTP packets with the header extension described in *RTP Header Timestamp* on page 5-8.

  When you de-select this option, IxChariot will generate RTP packets with IxChariot legacy timestamps.

Data Rate Optimization Options

The four data rate optimization options apply only to streaming pairs. These options allow you to enable optimization algorithms for sending streaming traffic at a fixed rate. There are two algorithms: one to reduce jitter, the other to avoid throughput spikes by limiting the data rate.

- **Low sender jitter**

  When checked, this option enables an optimization algorithm that uses very precise timers to reduce jitter. When this algorithm is enabled, the datagrams are sent by Endpoint 1 at more precise intervals than when the standard algorithm is in effect.

- **Limit data rate**

  When checked, this option enables an optimization algorithm that limits the data rate (throughput) measured on intervals much smaller than a timing record interval.

  The option is enabled only when the *Low sender jitter* option is enabled.

- **Data rate limit**

  The option is enabled only when the *Limit data rate* option is enabled.

  Use this field to specify the data rate limit for the streaming pairs in the test. The data rate limit is expressed as a percent of the required data rate defined in the script (the send_data_rate variable). For example, a value of 100 means that the limit is equal to 100% of the required data rate. The valid range of values is from 100 through 200.

- **Measured interval**

  The option is enabled only when the *Limit data rate* option is enabled.

  Use this field to set the measured interval for all the streaming pairs in the test. This is the interval (in milliseconds) over which IxChariot enforces the data rate limit. The valid range of values is from 1 through 999,999 ms.

  The purpose of this optimization algorithm is to prevent IxChariot from exceeding the required data rate (which may happen with the standard IxChariot algorithm). Therefore, you will typically set this value to match the interval over which the network devices check for throughput spikes.

When To Use Data Rate Optimization

The standard IxChariot algorithm for sending traffic at a fixed rate is designed to maintain the required data rate (as defined by the send_data_rate variable in the script) on average over the entire timing record interval. For streaming scripts, this method can potentially lead to two undesirable outcomes:

- The datagrams will not always be sent at fixed intervals. In some cases, the send jitter will be high.

- The algorithm may exceed the required data rate over a small interval in order to maintain the average data rate on the timing record interval. Some network devices will treat this as a throughput burst and drop the traffic.

You can reduce or eliminate these problems by enabling the data rate optimization options. There are three combinations of settings that you can set:

- Uncheck *Low sender jitter*. In this case, IxChariot uses the standard algorithm for sending traffic at a fixed rate. The optimization algorithm is disabled.

- Check *Low sender jitter* and uncheck *Limit data rate*. In this case IxChariot will use the high precision timers and sleeps to ensure low sender jitter, but

will not enforce any new throughput limits (the regular limit for the entire timing record interval will remain in effect).

- Check both *Low sender jitter* and *Limit data rate*. In this case, IxChariot will use the high precision timers and will also enforce the data rate limit over the requested interval.

Using a Data Rate Limit Greater Than 100

Note that enabling the *Limit data rate* option can result in a throughput rate that is lower than that required data rate (measured over the timing record interval). For example, if the first several send operations in a data stream transmit at a rate significantly lower than the send_data_rate value, subsequent send operations cannot exceed the send_data_rate in an effort to make up for the initial lower transmission rate. In contrast, the standard IxChariot algorithm for sending traffic at a fixed rate will, if necessary, exceed the send_data_rate to maintain the required data rate. Setting the *data rate limit* value to 100% results in a trade-off: it eliminates spikes at the cost of throughput.

If a *data rate limit* value of 100% results in unacceptable loss of throughput in a test, you can increase the value (up to 200%) to increase throughout while keeping the throughput spikes to an acceptable level.

Data Rate Optimization Limitations

Note the following limitations on the use of the data rate optimization options:

- The data rate optimization options are only applicable for the following Performance Endpoints: Windows 32-bit, Windows 64-bit, and Linux On PowerPC.

- The data rate optimization options require Endpoint 1 (the sender) to be running a Performance Endpoint software release of 6.70 or higher.

- The data rate optimization options are intended for tests that use fewer than ten streams. Internal tests at Ixia show that the settings are most effective when between four and seven streams are used.

Endpoint Pair Defaults Tab

Related Topics
VoIP Pair Defaults Tab on page 6-21
Cloning Hardware Performance Pairs on page 5-25
Adding or Editing an Endpoint Pair on page 5-19

The **Endpoint Pair Defaults** notebook page lets you set a default network protocol, service quality, and script. Whenever you add a new endpoint pair or multicast group, these values are initially selected in the **Network protocol** and **Endpoint 1** and **Endpoint 2 Service quality** lists in the Add Pair dialog box and

in the **Network protocol** and **Service quality** lists in the Add Multicast Group dialog box.

•   **Endpoint 1 to Endpoint 2: Network protocol, Endpoint 1 Service quality and Endpoint 2 Service quality**

Lets you select a default network protocol to use in testing. Although IxChariot uses TCP by default, the *Network protocol* field lets you choose the network protocol that you plan to use most frequently between endpoint pairs.

In the *Endpoint 1 Service quality* and *Endpoint 2 Service quality* fields, choose the quality of service (QoS) template that will serve as a default for the application traffic generated by your tests . Different QoS templates cand be selected for either endpoint. Service Quality is an optional parameter in IxChariot tests.

•   **Console to Endpoint 1 Management**

Lets you select a default network protocol to use in testing. Although IxChariot uses TCP by default, the Network Protocol field lets you choose the network protocol that you plan to use most frequently between endpoint pairs. Choose the quality of service (QoS) template that should serve as a default for your tests in the Service Quality field. Service Quality is an optional parameter in IxChariot tests.

Lets you specify a different network protocol and/or service quality to use in communications between the Console and the Endpoint 1 computers. Clear "**Use Endpoint 1 values from pair**" if you want to use different protocol and service quality settings for setup flows between the Console and Endpoint 1 and for communications between the endpoints. See *Cloning Hardware Performance Pairs* on page 5-25 for more information.

•   **Endpoint 1 to Endpoint 2 Management**

Lets you specify an alternate network address for communications between the endpoints. Clear "**Use Endpoint 2 address as management address"** if you want Endpoint 1 to be able to contact Endpoint 2 at a different address. "address as management address" are the protocol and address you specify when you create the endpoint pair. See *Cloning Hardware Performance Pairs* on page 5-25 for more information.

•   **Default Script**

Lets you choose which script is initially selected when you add an endpoint pair. If you do not choose a script in this dialog box, the default setting for each test will be no script at all, and IxChariot will remind you to choose a script as you add each pair or multicast group. If you do select a default script, a brief description of it appears just below the script field. Click **Clear Default Script** if you decide that you no longer want a default script.

See *Adding or Editing an Endpoint Pair* on page 5-19 for more information about these fields and how they affect a test.

If you change your mind, click **Undo** to reset all the fields to the values you had before you made any changes.

VoIP Pair Defaults Tab

Related Topics

Endpoint Pair Defaults Tab on page 6-19
Codec Types on page 10-43
Jitter Buffers on page 10-52
Timing Records and Graphs on page 11-6
Voice over IP Testing on page 10-34

The **VoIP Pair Defaults** tab in the Change User Settings notebook lets you set or change the defaults for all your voice over IP test pairs. The settings you select here appear in the Add a VoIP Endpoint Pair dialog box in the Test window.

- **Codec**

  The type of codec to use. Click the appropriate codec in the list. The default is G.711u. See *Codec Types* on page 10-43 for descriptions of each codec.

- **Packet Loss Concealment**

  If you are using the G.711u or G.711a codec, click to enable Packet Loss Concealment (PLC). See *Codec Types* on page 10-43 for more information. Most G.711 codecs don't currently implement PLC.

- **Use silence suppression**

  Emulates the effects of silence suppression (also called voice activity detection) on the line during the VoIP test. See *Silence Suppression* on page 10-47 for more information.

- **Voice activity rate**

  An indicator of the percentage of time during the call that talking is occurring. Determines how much actual voice data the simulated call contains for your tests. See *Silence Suppression* on page 10-47 for more information. Only enabled if "Use silence suppression" is checked.

- **Override delay between voice datagrams**

  Determines the datagram size to be used in the VoIP test. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for a full explanation.

- **Timing record duration**

  The length of each timing record. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for a full explanation.

- **Number of timing records**

  The number of timing records to generate for a test.

- **Use IPv6 protocol**

  Instructs the Console to use the IPv6 protocol in VoIP testing.

- **Service quality (optional)**

  Emulates the effects of a quality of service (QoS) scheme on call quality. Any QoS templates you've already created are available in the list. The **VoIP QoS** template is recommended; it emulates the QoS used by most voice over IP applications.

- **Initial delay value**

  Adds a `SLEEP` at the beginning of the application script. Recommended for emulating the effects of multiple users accessing the network at staggered intervals. Choose **Constant value** to enter an exact time setting in milliseconds, or choose one of the four mathematical distributions to let IxChariot select a series of values. Ensure that this delay is smaller than any timeout delay set in your script; too large a value may result in an error message. See "Setting Sleep Times" in the *Application Scripts* guide for more information.

- **Upper limit**

  Determines the value of the delay if you've chosen Constant value, or sets the upper limit of the mathematical distribution if you've chosen a mathematical distribution. Initial delays must be between 0 and 2147483647 ms.

- **Lower limit**

  Sets the lower limit of the mathematical distribution if you've chosen a mathematical distribution for your delay. Initial delays must be between 0 and 2147483647 ms. Not needed for a Constant delay value.

- **Additional fixed delay**

  Delay from a known, constant source. See *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information. Range is 0-300 ms.

- **Jitter buffer**

  Emulates the effects of jitter buffering on your VoIP network. Select *Jitter buffer* if you want IxChariot to emulate the effects of jitter buffering, and then specify either the absolute value (number of voice datagrams) or the frame-based value (number of milliseconds). De-select *Jitter buffer* if you do not want IxChariot to emulate jitter buffering in your test.

  Refer to *Jitter Buffers* on page 10-52 for more information.

- **Source and destination port numbers**

  Sets the port number used for the source (Endpoint 1) and the destination (Endpoint 2) in the voice transmission. The default value for both the source and the destination ports is AUTO. Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about setting source and destination port numbers.

- **Payload**

  Specifies the VoIP payload source. When you choose *Random payload*, the VoIP pair emulates a pair of endpoint computers that are transmitting voice traffic using the selected codec types (such as G7.11u). Alternatively, you can specify an external file that provides the payload. For more information about supplying payload from external files, refer to the *IxChariot Scripts Development and Editing Guide*.

HPP Defaults Tab

Related Topics

Adding or Editing a Hardware Performance Pair on page 5-23
Examining Timing Records on page 11-3

The **HPP Defaults** notebook page lets you set a default stream, line rate, and use of statistics for hardware performance pairs. Whenever you add a new hardware performance endpoint pair, these values are initially selected in the **Select Stream, Line Rate** and **Measure hardware performance pair statistics** elements in the Add Hardware Performance Pair dialog box.

• **Select Stream**

Lets you browse a default file directory which holds Ixia port streams prepared through the use of IxExplorer, ScriptMate, or the TCL or C++ API.

The default directory is configured using the *File..Change User Settings...Directories* dialog, under the *Where to read hardware performance stream files:* setting.

• **Override stream line rate**

Hardware pairs are capable of operating at hardware line rates which may overwhelm DUTs. The line rate specification allows you to optionally set the the data rate from the hardware ports to be controlled. If this option is not selected, the stream rate programmed into the selected stream is applied.

• **Measure hardware performance pair statistics**

Ixia ports are capable of obtaining performance statistics. Checking this box will make them available in the run results. See *Examining Timing Records* on page 11-3. If this box is not checked, the port pair will only provide background traffic.

VoIP HPP Defaults Tab

Related Topics

Endpoint Pair Defaults Tab on page 6-19
Codec Types on page 10-43
Jitter Buffers on page 10-52
Timing Records and Graphs on page 11-6
Voice over IP Testing on page 10-34

The **VoIP HPP Defaults** tab in the Change User Settings notebook lets you set or change the defaults for all your voice over IP test pairs. The settings you select here appear in the Add a VoIP Endpoint Pair dialog box in the Test window.

• **Codec**

The type of codec used on your network. Choose one of the supported codecs from the list. Refer to *Codec Types* on page 10-43 for more information.

- **Override delay between voice datagrams**

    Determines the datagram size to be used in the VoIP test. VoIP applications break voice data into datagrams based on delay, or the amount of time between successive datagrams. Default value is 20 ms. Values must be between 10 and 200 ms. IxChariot may adjust values slightly after you've entered them so that no partial buffers are sent.

- **Service quality**

    Emulates the effects of a Quality of Service (QoS) scheme on call quality. Only the template named **VoIPQoS** is available for VoIP hardware performance pairs.

- **Concurrent voice streams**

    The number of concurrent voice streams that will be generated by the hardware performance pair.

- **Source port number**

    The UDP source port to be used for the generated traffic.

- **Destination port number**

    The UDP destination port to be used for the generated traffic.

Video Pair Defaults Tab

Related Topics
Adding or Editing a Video Endpoint Pair on page 10-54
Adding or Editing a Video Multicast Group on page 10-58

The **Video Pair Defaults** tab in the Change User Settings notebook lets you set or change the defaults for the Endpoint 1 to Endpoint 2 traffic in all your video test pairs. The settings you select here appear in the Add a Video Endpoint Pair dialog box in the Test window.

- **Network protocol**

    The network protocol used to send the test traffic over the network. The valid choices are: UDP, UDP-IPv6, RTP, and RTP-IPv6.

- **Service Quality**

    The quality of service (QoS) template to use in tests with these pairs of endpoints. If you have defined QoS templates, select the template that you want to establish as the default. Optional field.

- **Video - Encoding**

    The Video encoding algorithm that the video endpoint pairs will simulate. Valid choices ar: MPEG2 and Custom. Note that the encoding method impacts the size of the UDP/RTP frames that are generated. For example, MPEG2 media frames are 188 bytes each.

- **Video - Frames per DG**

    The number of media frames per datagram. All signed integers values are valid. The default value for Ethernet is 7.

- **Video - Bitrate**

  The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 4 and 15 Mbps. The default is 3.75 Mbps.

  You can set the bitrate units to Mbps (1024 kilo bytes), Kbps (1024 bytes), or kbps (1000 bytes).

- **Timing Record duration**

  The approximate duration of a timing record. This is an indication only, rather than an exact setting. Timing records in IxChariot are based on the volume of data transferred, rather than on timing measurements. If packet loss occurs, the timing record duration will be larger.

- **Number of Timing Records**

  The number of timing records to generate for a test.

  The default value is 50, and the valid range of values is from 1 to 2,147,483,647.

- **Source port number**

  The UDP source port for the test traffic. The default is Auto.

- **Destination port number**

  The UDP destination port for the test traffic. The default is Auto.

- **Initial delay - Value**

  This parameter adds a SLEEP at the beginning of the application script. It is recommended for emulating the effects of multiple users accessing the network. Select "Constant value" if you want to set an exact value for the delay. With the other options, IxChariot generates values for the delay corresponding to the selected mathematical distribution (Uniform, Normal, Poisson, Exponential). In this case, you can set the lower and upper limits for these distributions. The default is "Constant value."

- **Initial delay - Upper limit**

  This is either the value for the delay if you have chosen either "Constant value"; or the upper limit for the values generated by one of the mathematical distribution. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

- **Initial delay - Lower limit**

  This parameter is visible only if you selected one of the four mathematical distributions. It represents the lower limit for the values generated by the selected distribution. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

IPTV Defaults Tab

The **IPTV Defaults** tab lets you set the default values for the IPTV channels and IPTV receiver groups. As with other user settings, the IPTV defaults are loaded from the registry when IxChariot is started and are saved when IxChariot exits.

Table 6-2 describes the parameters for which you can set default values:

Table 6-2.   IPTV Defaults in the Change User Settings Window

Parameter	Description
IPTV Traffic Characteristics:	
Encoding	The desired video encoding for this video stream. The choices are MPEG2 or Custom.
RTP Payload Type	If you select *Custom* encoding, you must specify an RTP Payload type. (If you select *MPEG-2* encoding, this field will be read-only.)
Media Frames per DG	The number of media frames per datagram. All signed integer values are valid. The default value for Ethernet is 7.
Media Frame Size	The media frame size for the selected encoding.
Bitrate	The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 3 and 15 Mbps. The default is 3.75 Mbps.
Bitrate unit of measure	The unit of measure for the bit rate. You can set the bitrate units to Gbps (1024 megabytes), Mbps (1024 kilobytes), Kbps (1024 bytes), or kbps (1000 bits).
Source port number	The source port number for the video stream.
Send Buffer Size	The socket buffer size used for sending the data streams. The default value is determined by the operating system on which the Performance Endpoint is running.
Test Network:	
Network Protocol	The network protocol to use for this video stream. The options are RTP and UDP.
Service Quality	Optional. The name of the QoS template that will define the service quality for this video stream.
IPTV Receiver Settings – Channel Control:	
Number of TR	The number of timing records that will be generated for each join/leave cycle.
TR duration (sec)	The duration of each timing record (in seconds).

Table 6-2.   IPTV Defaults in the Change User Settings Window (Continued)

Parameter	Description
IPTV Receiver Settings – Receiver Settings:	
Number of iterations	The number of times the receiver will switch channels (leave one channel and join another).
Receive Buffer Size	The socket buffer size used for receiving the data streams. The default value is determined by the operating system on which the Performance Endpoint is running.
Switch delay (ms)	The number of milliseconds that will elapse between leaving one channel and joining another.

Application Groups Tab

Related Topics

Creating Application Group Tests on page 5-35

The **Application Groups** tab in the Change User Settings notebook lets you set or change the defaults for the application groups you use in your tests.

• **Validation algorithm**

   You can choose between the following two validation options:

   • Use Fast Validation – The fast validation algorithm requires less processing time than the comprehensive algorithm, but it may identify a valid application group as invalid. (However, it is equally reliable as the comprehensive algorithm when reporting a valid application group.)

   • Use Comprehensive Validation – The comprehensive validation algorithm requires more processing time than the fast algorithm, but it is more precise in its error reporting.

• **Warnings**

   You can choose to display or suppress the following warning messages:

   • Deleting application group from test – When this option is enabled, IxChariot presents a warning and requests confirmation if you attempt to delete an application group. Note that when you delete an application group, IxChariot deletes all of the pairs contained within the group.

   • Application group has no events defined – When this option is enabled, IxChariot presents a warning and requests confirmation if you attempt to delete all events from an application group, or if you create an application group without defining any events.

Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about application groups.

Ixia Port Configuration Tab

Related Topics
Stack Manager User Guide
The Tools Menu on page 3-10

Use the **Ixia Port Configuration** tab to set the following options:

*   **Connect to hardware using Socks**

    Select this option to enable or disable automatic connection through the Socks server. When enabled, IxChariot will automatically and transparently communicate with the endpoint through the use of the Socks server on the chassis.

*   **Always perform discovery at start of test**

    Select this option if you want IxChariot to always perform an ARP discovery at the start of every test run. This allows the tests to run successfully if network or device configurations change between tests.

    When you first start a hardware performance pair, the port CPU sends out an ARP or IPv6 Router Discovery packet to set everything up. However, if the "Always perform discovery at start of test" option is not selected, there will be no further Discovery packets on subsequent runs of the test. If the router configuration changes or the router is rebooted, then the subsequent runs of the hardware performance pair will fail since they have no discovery information.

*   **Apply only Endpoint DoD package**

    Select this option if you want to apply the endpoint DoD package only, without applying the configuration.

> **NOTE**: Enabling the **Apply only Endpoint DoD package** option disables the **Deconfigure ports and release ownership after the test ends** option.

*   **Deconfigure ports and release ownership after the test ends**

    Select this option to deconfigure the Ixia ports used in your test(s) and release ownership on them as soon as the test(s) end. After installing IxChariot for the first time on your computer (and proceeding to configure an IxChariot test), the checkbox is selected by default.

*   **Use default IxTCLServer**

    If you select the Use default IxTCLServer option, IxChariot uses the TCL Server on the default Master chassis from the configuration, (the first chassis in the list).

    If you de-select this option, the User defined Ix-TCLServer text box is enabled. In this case, you need to specify the location of the specific TCL Server to use.

- **TestServer**

  Select "Launch local TestServer process" to launch the default Middleware TestServer (which is a local process).

  If you de-select this option, the "User defined TestServer" text box is enabled. In this case, you need to enter the name of the specific Middleware TestServer to use.

Result Ranges Tab

Related Topics

The Jitter Tab on page 11-29
The Lost Data Tab on page 11-31
VoIP Pair Defaults Tab on page 6-21
Jitter and Delay Variation on page 10-52

User-defined result ranges are provided so that you can more accurately determine call quality based on hardware and network thresholds, such as the size of the jitter buffer at the receiving endpoint and the relative standards in place to judge call quality. Refer to *The Jitter Tab* on page 11-29 and *The Lost Data Tab* on page 11-31 for more information about how results are presented.

Jitter (Delay Variation)

These ranges indicate the number of datagrams (expressed as a percent of the total number of datagrams sent) that experienced jitter. Delay variations, in milliseconds, are placed in groups, or ranges, so that you can compare them to such benchmarks as the size of the jitter buffer and the standards for call quality established for the hardware you are using.

- **Range**

  By default, IxChariot configures delay variation jitter data in five ranges. Clear the boxes next to any ranges you don't want to see in your results.

- **Minimum (ms)**

  Sets the lower limit of the range, in milliseconds. Automatic setting.

- **Maximum (ms)**

  Sets the upper limit of the range, in milliseconds. Leave the value in place or enter a new value to change the default ranges. Ranges must be contiguous. As you change each range, the value for the next range minimum is automatically filled in.

  Default settings for the five Jitter (delay variation) ranges are as follows:

  - 0-10 ms

  - 11-20 ms

  - 21-30 ms

  - 31-50 ms

  - 51 + ms

Consecutive Lost Datagrams

Consecutive Lost Datagrams indicates the call quality based on the amount of the transmission that experienced noticeable loss. Also helps to determine the relative burstiness of datagram loss during the voice transmission.

- **Range**

  By default, IxChariot configures data on lost datagrams in 5 ranges. Clear the boxes next to any ranges you don't want to see in your results.

- **Minimum**

  Sets the lower limit of the range, in number of datagrams. Automatic setting.

- **Maximum**

  Sets the upper limit of the range, in number of datagrams. Leave the value in place or enter a new value to change the default ranges. Ranges must be contiguous. As you change each range, the value for the next range minimum is automatically filled in.

Default settings for the five Consecutive Lost Datagram ranges are as follows:

- 1-1
- 2-3
- 4-5
- 6-10
- 11 +

Throughput Units Tab

Related Topics

The Throughput Tab on page 11-17

The **Throughput Units** tab on the Change User Settings notebook lets you choose one of six ways to tailor the throughput[1] numbers in your results to units that reflect your test environment. In reading these values, remember that an uppercase "K" represents *1,024*, while a lowercase "k" represents *1,000*. Similarly, an uppercase "B" represents *bytes*, while a lowercase "b" represents *bits*.

Table 6-3.   Throughput Units

Unit	Description
KBps	1,024 Bytes per second
kBps	1,000 Bytes per second
Kbps	1,024 bits per second (that is, 128 Bytes per second)
kbps	1,000 bits per second (that is, 125 Bytes per second)

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

Table 6-3.   Throughput Units (Continued)

Unit	Description
Mbps	1,000,000 bits per second (that is, 125,000 Bytes per second—the default setting)
Gbps	1,000,000,000 bits per second (that is, 125,000,000 Bytes per second)

We do not recommend changing your throughput units from Mbps. Differing throughput units can cause unexpected confusion when comparing results. Be especially careful that you are using the same units when cutting and pasting exported values from different files.

These units do not affect other numbers, like transaction rate, response time, or relative precision.

Click **Undo** to reset your units to the units you had before you made changes.

Directories Tab

Related Topics
Appendix A, *IxChariot File Types*

The **Directories** tab in the Change User Settings notebook lets you change the default directory paths that IxChariot uses to save and retrieve test files, script files, error logs, and Aptixia files. These directory locations are important for the IxChariot console:

• **Where to read and write test files**

The drive, path, and directory used when you open and save test files.

• **Where to read script files**

Tells IxChariot where script files are to be found and loaded when you click Open a Script File while editing an endpoint pair.

• **Where to read hardware performance stream files**

Tells IxChariot where hardware performance stream files (.sdf files) are to be found and loaded when you click the Select Stream button from the Add a Hardware Performance Pair dialog.

• **Where to write Console error logs**

This is the directory where log files are created. All error files and assert.err are included in this directory. When an error occurs during a run or while cloning a test, IxChariot writes an entry to the error log file. At the Console, this file is `Chariot.log`, `Chrapi.log`, `Clonetst.log`, or `Runtst.log`, depending on which program you were running when the error occurred. The error log file is written to the drive, path, and directory listed in this field. See *The Error Log Viewer* on page 12-8 for more information.

•   **Where to import Aptixia files**

This is the directory from which IxChariot will load Ixia Network Configuration files when you select for use with a test.

•   **Where to export Aptixia files**

This is the directory to which IxChariot will save Ixia Network Configuration files that you create.

When you make changes to any of these directories, IxChariot looks to see if the directory exists. You might choose to have any of these fields point to a directory on a LAN drive, for example, even if you are not attached to that drive when you are updating these fields. When prompted, answer **OK** to the question; IxChariot uses what you've entered, even if it cannot find the directory when you enter it. But make sure that the drive is accessible and the directory exists before actually running a test.

•   **Display the *n* most recently accessed tests**

Lets you decide how many test files should remain accessible from the File menu. You can set this value from 0 to 9.

If you change your mind, click **Undo** to reset all the fields to the values you had before you made any changes.

Firewall Options Tab

Related Topics
Firewall Testing on page 10-13
Firewalls and Fixed Ports on page 8-13

The **Firewall Options** tab in the Change User Settings notebook lets you specify options for testing through firewalls. See *Firewall Testing* on page 10-13 for a full discussion and information on configuring your firewall.

Endpoint 1 through Firewall to Console Reporting

IxChariot uses the *Endpoint 1 through firewall to Console Reporting* setting to determine the destination port numbers for test results sent from Endpoint 1 to the Console. Table 6-4 describes the options.

Table 6-4.   Endpoint 1 through Firewall to Console Reporting

Setting	Description
TCP	These three parameters specify the IxChariot Console destination port numbers that Endpoint 1 uses when sending test results to the Console.
SPX	
Hardware Performance Pair Statistics	Your choices are: • **Auto**: If you select Auto, IxChariot will dynamically choose the destination port numbers. • **Port**: If you enter specific port numbers, IxChariot will use those as the destination ports. In each case, the default setting is *Auto*, which allows IxChariot to choose a random port number.

Endpoint 1 through firewall to Endpoint 2

IxChariot uses the *PAT device support between E1 and E2* setting to determine the correlation method to use for user-defined ports on TCP. In this context, IxChariot needs to know if Endpoint 2 port numbers are modified by a NAT device that implements PAT (Port Address Translation).

Your choices are:

- Select **PAT device support between E1 and E2** if there is a NAT device between Endpoint 1 and Endpoint 2 and that NAT device is using PAT.

- Deselect **PAT device support between E1 and E2** if the device uses NAT without PAT.

Management ports

IxChariot uses the *Endpoint management* setting to determine the destination port number for test setup data sent from the Console to Endpoint 1 and from Endpoint 1 to Endpoint 2. This port number is used for clock synchronization, as well.

Your choices are:

- **Auto**: If you select *Auto,* IxChariot will choose the port number (port 10115).

- **Port**: If you enter a specific port number, IxChariot will use it as the management port. Typically, you will enter a specific port number when you already have a port open (such as port 80) and you do not want to open and close additional ports each time you run a test.

Management Port Notes:

- This parameter is applicable only when the test transport protocol is TCP. For SPX, the management port number is always 10117.

- If you specify a port number for the Console, you must set the same port number on every endpoint that you will use in the test. (You specify the management port for the endpoints by setting the MANAGEMENT_PORT keyword in the endpoint.ini files. Refer to the Performance Endpoint Guide for details.)

- If you use a packet inspection firewall and you set port 80 as the management port, the firewall may reject packets carrying the management traffic.

- If you specify a port that is used by another application (such as an IxChariot script), the endpoints will report errors.

- The management port cannot be the same as any of the reporting ports (specified in *Endpoint 1 through Firewall to Console Reporting* on page 6-32).

Clicking **Undo** resets all the fields to the values they showed before you made any changes.

Output Tab

The **Output** tab in the Change User Settings notebook lets you select default output template and HTML, PDF, Text, or CSV file format options for exporting test results.

With output templates, you can save printing options. IxChariot lets you choose a default output template or select a template each time you print or export test results. The name of your default template appears in the **Output Template** field in the Print/Export dialog box for each new test. For information on template options, see *Custom Printing and Export Options* on page 5-64.

In each of the first three fields, click **[None]** if you want to use the print options specified in the test.

• **Print Configuration default**

The output template you want to appear in the Print Options dialog box when you click **Print** in the File menu in the Test window. See *Export Options* on page 5-62.

• **Export to HTML Configuration default**

The output template you want to appear in the Export to HTML/GIF dialog box when you click **Export to HTML** in the File menu in the Test window. See *Export Options* on page 5-62.

• **Export to PDF Configuration default**

The output template you want to appear in the Export to PDF dialog box when you click **Export to PDF** in the File menu in the Test window. See *Export Options* on page 5-62.

• **Export to Text Configuration default**

The output template you want to appear in the Export to Text File dialog box when you click **Export to Text** in the File menu in the Test window. See *Export Options* on page 5-62.

• **CSV Export**

Settings to be used as the default in the Export to CSV File dialog box. IxChariot lets you export test information to a spreadsheet output in the `.csv` file format. See *Export Options for .CSV Files* on page 5-63 for information on the `.csv` file format.

CSV Export - Contents Settings:

• Test summary and run options

Exports a summary of any results and the Run Options you configured.

• Pair summary

Exports pair information contained in the **Test Setup** and results tabs of the Test window.

• Pair details

Exports the timing records for the pairs in your test.

CSV Export - Scope Settings:

• Export all

Reports results on all the pairs when you export results to `.csv` format.

• Export marked pairs

Reports results on marked pairs only when you export results to `.csv` format. Marked pairs have a graph symbol in the far left column in the Test window.

Traceroute Tab

Related Topics

Running a Traceroute on page 5-32

You can run a traceroute on an endpoint pair by highlighting the pair in the Test window and choosing **Traceroute** from the Run menu. The **Traceroute** tab in the Change User Settings notebook lets you configure your traceroute options beforehand.

- **Maximum hop count**

  Specifies the number of router hops an IxChariot traceroute test can make before abandoning the run. If the maximum hop count is less than two, the ICMP (Internet Control Message Protocol) Echo message IxChariot sends out cannot cross a router. We require a minimum value of 2 so that the message will at least leave the local subnet. The default value in this field is 30, but you may want to set it much lower. Enter a value between 2 and 40 in this field.

- **Maximum per-hop timeout**

  Specifies the maximum time the message can take to reach each hop on its way to Endpoint 2 (the Target) before the traceroute test is abandoned. The default value in this field is 3000 ms. Enter a value between 1 and 10000 in this field.

- **Resolve DNS names**

  By default, IxChariot resolves the DNS names you enter for endpoint computers into numeric IP addresses. Clear the box labeled Resolve DNS names if you don't want intermediate hop addresses to be resolved to names. Letting IxChariot resolve the hops into names adds latency to your time values.

  See *Running a Traceroute* on page 5-32 for more information.

Warnings Tab

Related Topics

Finished Test Warnings on page 12-6

Some users like to be warned before they take actions with non-trivial side-effects. Other users don't want the annoyance of warning messages that keep popping up. The **Warnings** notebook page in the Change User Settings notebook lets you decide which warnings you see. The default setting enables all warnings. Clear the check boxes next to the warnings you want to disable:

- **Stopping a test**

  You clicked **Stop** while a test was running. The warning asks, Do you want to stop this running test?

- **Clearing the results of a test**

  You clicked **Clear Results** on the File menu. The warning asks, Are you sure you want to clear these results?

- **Another operation clears the results of a test**

  You tried to perform an operation that will clear the results of a test. The warning asks, Do you want to change the test setup, which will cause the results to be erased?

- **Deleting pairs from a test**

  You clicked **Delete** while pairs were highlighted. The warning asks, Do you want to delete these endpoint pairs?

- **Abandoning a run**

  A test could not be stopped in the usual way, so you have clicked **Abandon Run**. This is a fairly drastic action to take and can cause endpoint errors. The warning asks, Do you want to abandon the test, even though the endpoints may still be running a script?

- **Printing more than 25 pages**

  Your printed results will take up more than 25 pages. The warning asks, Do you want to print this many pages?

- **Running a test with more than 500 timing records for a single pair**

  You've set up a test that could potentially create a huge number of timing records, which could make the Console very sluggish or cause the system to run out of memory. The warning asks, Do you want to run this test?

- **Running a test with more than 10000 pairs**

  You've set up a test with more than 10,000 pairs and are preparing to run it from the IxChariot Console graphical user interface (GUI). Poor GUI performance may occur. We recommend using RUNTST; see *RUNTST: Running Tests* on page 5-67. The warning asks, Do you want to continue?

- **Overwriting a previously-saved test**

  You clicked **Save** after running a test you'd run before. Your results will be overwritten. The warning asks, Do you want to continue this Save operation and overwrite the existing test file and its results?

- **Saving a test file in an upgraded version**

  You tried to save a test file taken from an earlier version of IxChariot. The warning asks, Do you want to continue this Save operation and save the test results in an old format?

- **Saving a test file in an older version**

  You are about to save a test file in a format compatible with an older version of IxChariot. The warning asks, Are you sure you aren't losing any capabilities that you may need in future testing?

- **Saving a script file in an upgraded version**

  You opened a script file from an earlier version of IxChariot. The warning asks, Are you sure you want to save it as a newer version, which cannot be opened by older versions of IxChariot?

- **Saving a script file in an older version**

  You are about to save a script file in a format compatible with an older version of IxChariot. The warning asks, Are you sure you aren't losing any capabilities that you may need in future testing?

- **Creating a regular pair to simulate a VoIP pair**

  You are using a previously saved copy of the IxChariot `VoIPs` script to run a voice over IP test with an endpoint pair. The warning asks, Are you sure you don't want to create a VoIP endpoint pair for this test?

- **A pair's E1 address differs from its Console to E1 address**

  The Endpoint 1 address you entered differs from the address you entered in the Pair Setup dialog box. This may cause the test to fail. The warning asks, Are you sure that the Console and Endpoint 1 addresses are correct?

- **A pair's E2 address differs from its E1 to E2 address**

  The Endpoint 2 address you entered differs from the address you entered in the Pair Setup dialog box. This may cause the test to fail. The warning asks, Are you sure that the E2 addresses in this dialog box and in the pair itself are correct?

- **Running IxChariot with a log file larger than 5 MB**

  Your `.log` or `runtst.log` file exceeds 5 MB in size. A log file larger than 5 MB may affect the performance of the IxChariot GUI. The warning advises you to delete the log file or transfer it to another computer.

- **E1 knows E2 is used, but Console knows E1 is not used**

  You entered an address for **E1 knows E2** in the Pair Setup dialog box, but you did not enter an address for **Console knows E1**. The warning asks, Are you sure you want to use the E1 address in the pair for setup flows?

- **Detect outside changes in payload**

  When you save a script or a test that uses embedded or referenced payload files, IxChariot performs a check to detect changes made to the payload file after it was added to the script. The warning gives you the opportunity to continue with the save or cancel the save operation.

- **Finished Test Warnings**

  The three items at the bottom of the Warnings notebook page control Finished Test Warnings, warnings that are generated after a completed test run if the test may have generated data that is invalid or non-significant. As with other warnings, clear the box beside a warning to disable it. See *Finished Test Warnings* on page 12-6 for more information.

7

Test Run Options

This chapter provides a description of the run options that you can set for individual IxChariot tests.

Topics in this chapter:

Accessing the Run Options

To access the Run Options window, select **Run > Set Run Options** from the main menu.

The Run Options window is organized into five tabs:

IxChariot gives you a great deal of control over how tests are run and how results are reported. Particularly when you are concerned about network performance, you should plan to change some run options to yield the most accurate measurements.

All of these run options are applied on a per-test basis. Therefore, you can vary these run options from test to test. They are saved in the test file, along with the endpoint pair information.

Run Options Tab

The Run Options tab is organized into the following categories of options:

- *Performance Testing* on page 7-2
- *How to End a Test Run* on page 7-3
- *How to Report Timings* on page 7-4
- *Polling the Endpoints* on page 7-5
- *Clock synchronization* on page 7-6
- *Management Quality of Service* on page 7-8
- *Miscellaneous Run Options* on page 7-11

The following topics present a description of each run option. If you change your mind, click **Undo** to reset all the fields to the values you had before you made any changes.

Performance Testing

Related Topics
Designing IxChariot Performance Tests on page 10-84
Polling the Endpoints on page 7-5
How to End a Test Run on page 7-3
How to Report Timings on page 7-4
How Long Should a Performance Test Run? on page 10-86

The checkbox labeled **Set the test run options for performance testing** automatically configures testing options to achieve the highest performance results. Checking this box changes the default settings for the following test configuration parameters:

- How to report timings. The default setting lets you see the results in real time, as the test is running. For more accurate performance measurements, it is better to report in batch because batch reporting doesn't consume extra CPU and network resources during the test.

- Poll endpoints. The default setting specifies that the IxChariot Console will poll the endpoints once every minute during a test run. Polling also consumes network resources, creating extra data flows that interfere with performance measurements, so polling is disabled when you check Performance Testing.

Other default run options are already appropriate for performance testing. If you have changed the defaults for **How to end a test run**, **Collecting endpoint CPU utilization**, and **Collecting TCP statistics**, checking the Performance Testing checkbox changes your settings back to achieve higher performance. CPU collection, TCP statistics collection, Data Validation, and the New Random seed option are all automatically turned off for performance testing.

It Is highly recommended that you enable performance settings for tests of more than 500 pairs. See *Setting Run Options for Performance Testing* on page 8-4 for more information.

How to End a Test Run

Related Topics

Run Options Tab on page 7-2
Miscellaneous Run Options on page 7-11
Polling the Endpoints on page 7-5
How to Report Timings on page 7-4

This section of the Run Options notebook provides three ways to determine when a test run is complete. Some endpoint pairs run much faster than others, depending on the script variable values, endpoint computer CPU speeds, and network equipment. Unless you run a test for a fixed duration, you may find that some pairs have completed all their timings before other pairs have even reported once. You should experiment to get a good balance among pairs. The choices for ending a test run are as follows:

- **Run until any pair ends**

  Stops the test run when any endpoint pair completes executing its script or ends with an error. Any timing records received after this first pair completes its script are discarded. This ensures that all timing records used in the calculations were generated while the other endpoint pairs were still executing scripts.

  This setting is recommended for performance testing, particularly for highly utilized lines, because it shows measurements taken during times of maximum contention for shared bandwidth. As soon as a pair finishes its script, all pairs stop taking measurements, which means that no pair has an opportunity to communicate over an empty line.

  Sometimes a pair running a streaming script will complete the test with fewer timing records than you specified in the script (in the script's number_of_timing_records parameter). This is because some data may have been lost. A timing record is complete when Endpoint 2 has received enough data to fill (and sometimes overflow) that record. However, one timing record may contain lost data as well as successfully received data. The disparity in the number of bytes sent and bytes received can cause the pair to complete before Endpoint 2 can complete the total number of timing records for which data was sent.

- **Run until all pairs end**

  Allows all endpoint pairs to run until they have completed all the commands in their scripts or ended with errors.

  Tests using this option may generate misleading data. The problem occurs as scripts finish and there is less competition for available bandwidth. In fact, the last executing script could have the network all to itself and report much better results than it would if other pairs were still executing scripts—the most likely situation under normal operating conditions.

  This option is recommended only if the endpoint pairs do not share the same network resources—that is, if they use different routers and physical media.

• **Run for a fixed duration**

With this option, all endpoint pair scripts run for a fixed period of time, ignor-ing the `number_of_timing_records` value in their output loop. At the end of the period, the endpoints stop and Endpoint 1s return their results. You can choose values from 1 second to 99 hours, 59 minutes, and 59 seconds. Although the default setting is 1 minute, we recommend 2 to 5 minutes for most performance testing. It Is important to test for a sufficient length of time to generate at least 10 timing records; the records represent data samples, and you need enough samples to ensure test reliability.

You may set a much longer duration for stress testing. However, because application scripts generate roughly 50 timing records per minute on a LAN, setting the duration to long time periods can generate an enormous number of timing records—potentially exceeding the storage capacity of some console computers. We recommend using the Script Editor to tune the inner loops of your scripts so that they generate timing records less frequently. Experiment before running multi-hour, multi-pair stress tests. The run time duration is checked every time an `END_LOOP` or `END_TIMER` command executes. This may cause the actual run time to slightly exceed the run time you type here.

The "fixed duration" option for performance testing is useful because it avoids situations in which some endpoint pairs complete their scripts long before the others, leaving the last pairs a nearly empty line over which to communicate. But be careful not to overstress your network with extraordi-narily long tests and many pairs.

How to Report Timings

Related Topics

The Run Options dialog box also gives you an opportunity to determine how results of a test will be reported. Because of the extra data flows created between each Endpoint 1 computer and the Console, the way the Endpoint 1 computers report the data in their timing records can affect test results. You can choose between two reporting methods:

• **Batch**

With this setting, timing records are saved at Endpoint 1 and forwarded to the Console at the end of the test. Results are not displayed until the test com-pletes, or until 500 timing records have been collected (or 300 for VoIP pairs). This keeps network traffic from Endpoint 1 to the Console from interfering with the actual performance measurements on your network. If you want to find out the progress of the test, click the Poll button.

Batch reporting is always the recommended option for any performance test-ing.

• **Real-time**

With this setting, every time a timing record is created, it is sent back to the Console. The Console updates the Test window as the timing records are received, letting you see how the test is progressing. While this is handy for

verifying tests, real-time operation can have dramatic, negative effects on the test being run. Results are updated at least every 5 seconds. The specific amount of time between updates depends on the number of pairs in the test. Even if you choose real-time reporting, if you are running a loopback test containing multicast groups in real time, it may take several minutes for the timing records from the test to be shown in the Console.

We strongly recommend doing performance measurements with batch timings, which avoid the extra network traffic of real-time operation. This extra traffic is doubled when executing a streaming script since the timing records are sent from Endpoint 2 to Endpoint 1, which then forwards the records to the Console. While real-time results look "cool," they consume resources in the network, at the endpoints, and at the Console. The worst-case behavior of real-time reporting occurs with many endpoint pairs, each generating timing records frequently.

* **Console behind Firewall**

Select this option if there is a firewall between the IxChariot Console and endpoint 1. When you select this option, the IxChariot Console will initiate the connection with endpoint 1 for the duration of the test, thereby ensuring that endpoint 1 will be able to transmit timing records whenever needed.

If you do not select this option, endpoint 1 initiates a connection with the Console whenever it has timing records to transmit. If there is no firewall, the connection is established successfully. However, if there is a firewall between endpoint 1 and the Console, it will prevent endpoint 1 from establishing the connection, in which case the test results will not be received by the Console.

Refer to *Firewall Testing* on page 10-13 for more information about test environments that include firewalls.

Polling the Endpoints

Related Topics
Performance Testing on page 7-2
Factors Affecting Results on page 11-52

When you click **Poll Endpoints Now** on the Run menu during a running test, the Console sends a request to each of the Endpoint 1 computers in the test to return the number of timing records they've generated so far.

There are two good reasons to poll the endpoints while a test is running:

* The test is reporting results in Batch, and you want to retrieve either a count of timing records that have been collected, or retrieve the actual timing records so that you can view them in the table and on the graph, without waiting for the test to finish. (When reporting results in batch mode, IxChariot normally saves the timing records at Endpoint 1 and forwards them to the Console only at the end of the test or until a set number of timing records have been collected (500 records for regular pairs, 300 records for VoIP pairs).

* You suspect that one or more Endpoint 1 computers can no longer be reached. If Endpoint 1 is powered off during a test, the Console never actually knows because the Console doesn't maintain a connection with Endpoint 1 while a test is running. Polling forces the Console to initiate contact with each Endpoint 1 computer.

Polling can, however, adversely affect your test results. As a general guideline, you should disable the **Poll endpoints** option for a test that has 500 or more pairs. If polling is enabled for a test with a large number of pairs, IxChariot may return a CHR 0245 error. Further, you should refrain from selecting **Poll Endpoints Now** from the Run menu during the execution of a test that has a large number of pairs.

You can set the following options to control IxChariot polling behavior:

- **Poll endpoints**

  Select this option to allow IxChariot to poll endpoints during a test. De-select this option to disable polling.

- **Interval**

  This option allows you to specify the automatic polling interval, in minutes, when the test is run in real-time mode.

- **Retrieve Timing Records**

  When this option is selected, IxChariot returns the actual timing records when you click **Poll Endpoints Now** on the Run menu. When it is not selected, IxChariot returns only a count of timing records when you click **Poll Endpoints Now** on the Run menu.

  This selection is applicable only to tests run in batch mode. Tests that run in real-time mode always retrieve actual timing records

The more endpoints involved in a test, the less often IxChariot refreshes status changes in the Test window. This reduces the overhead of updating records at the Console. For large tests, this refresh period can take quite a long time. The actual status of the endpoint (polling or running) may not be evident for a while, even though the run status has changed while the test is running.

Clock synchronization

Related Topics

The clock synchronization options allows you to specify the timing source for the endpoints on a test network. There are three potential timing sources:

- Ixia hardware timestamps:

  An Ixia chassis backplane can provide the timing source for all the ports in a chassis chain. In this case, an endpoint obtains an Ixia hardware timestamp each time it sends a time value over the network. The receiving node also obtains an Ixia hardware timestamp, thereby enabling IxChariot to perform the necessary time calculations using the hardware timestamp clock as the reference system. Using Ixia hardware clock synchronization produces the most reliable method of reporting one-way delays.

  If you are setting up a test that uses a virtual chassis chain based on Ixia AFD1 GPS clocking, you must select *Ixia hardware timestamps* as the clock synchronization option. (Refer to *Using an AFD1 in an IxChariot Test Network* on page 4-23 for more information.)

• External device:

Clocking for Linux endpoints can be provided by external Network Time Protocol (NTP) servers that are synchronized to the Global Positioning System (GPS).

> **Note:** IxChariot provides external clock synchronization support for Linux endpoints only.

External clock synchronization requires properly-configured NTP servers. IxChariot does not verify that the external timing source is operational or accurate. Consequently, if an NTP server is misconfigured or inoperable, test results will be invalid.

You can use either of the following configurations when using NTP servers and the GPS as the external clocking source:

• Each IxChariot Linux endpoint is configured with Performance Endpoint software, a GPS device, and a properly-configured NTP server.

• Each IxChariot Linux endpoint is configured with Performance Endpoint software, but is not configured with a GPS device and an NTP server. Instead, a GPS device is connected to a separate network node on which the NTP server is running. In this case, the IxChariot endpoints synchronize to the NTP server over a local (very low latency) network. This configuration generally yields lower accuracy than the first configuration.

Using external synchronization as the timing source is generally more reliable than using the endpoint internal timers, but less reliable than using Ixia hardware timestamps. External synchronization is preferred over endpoint internal timers on high latency networks, whereas endpoint internal timers are recommended for endpoints that are on the same LAN.

• Endpoint internal timers:

When neither Ixia hardware timestamps nor an external device can be used as the clocking source, Endpoint 1 and Endpoint 2 synchronize their time using their internal timers. This synchronization occurs during test initialization. The endpoints periodically synchronize their clocks based on the estimated clock deviation computed from previous synchronizations. (You can force a clock synchronization for each test run by setting the FORCE_CLOCKSYNC keyword in the endpoint.ini files.)

The Run Options provide two clock synchronization options: *Use Ixia hardware synchronization*, and *External synchronization*. You can select both, either, or neither of these to specify the clock synchronization method you will use for your test network. Table 7-1 describes the effect of setting these options.

Table 7-1.   Effect of Clock Synchronization Settings

Options:			
Use Ixia hardware clock synchronization	*External synchronization*	**Preferred Clock Source:**	**Fallback Clock Source:**
Yes	Yes	Ixia hardware timestamps.	External device.
Yes	No	Ixia hardware timestamps.	Endpoint internal timers.
No	Yes	External device.	N/A
No	No	Endpoint internal timers.	N/A

As shown in Table 7-1, the timing source that each endpoint pair uses in a test depends upon the selected options and the availability of that specific source:

- **Use Ixia hardware clock synchronization**

    When this option is selected, all pairs attempt to use the Ixia hardware time-stamp as the timing source on the test network. If the Ixia port hardware timestamp is not available to a specific endpoint pair, that pair will use:

    - the external clock source, if **External synchronization** is selected, or

    - the endpoint internal timers, if **External synchronization** is not selected.

    This provides support for tests in which Ixia pairs and non-Ixia pairs are both used.

    If available, Ixia hardware clock synchronization is always preferred over external synchronization, which, in turn, is always preferred over endpoint internal timers.

- **External synchronization**

    When only this option is selected, all Linux endpoint pairs obtain their timing values from the external timing source. IxChariot does not validate the pres-ence of or the reliability of the external timing source. Rather, IxChariot assumes that the clocking is provided by an external system.

Management Quality of Service

Related Topics

IxChariot Network Topology on page 2-2
IxChariot Test Process Overview on page 2-5
Chapter 9, *Quality of Service Testing*

The Management Quality of Service section of the Run Options dialog allows you to specify QoS settings for IxChariot management traffic. There are two dis-tinct management QoS settings:

- **Console Service Quality**

    This setting specifies the QoS template that IxChariot will use for manage-ment traffic sent from the console to Endpoint 1. This traffic includes the pre-

setup and setup data that the console sends to Endpoint 1 prior to the start of the test.

- **Endpoint Service Quality**

    This setting specifies the QoS template that IxChariot will use for all other management traffic: management traffic from Endpoint 1 to the console, and management traffic between Endpoint 1 and Endpoint 2. This traffic includes the pre-setup and setup data that Endpoint 1 sends to Endpoint 2 prior to the start of the test, the test results that Endpoint 2 sends to Endpoint 1 during the test execution, and the test results that Endpoint 1 sends to the console during the test execution.

To specify IxChariot default settings for either or both of these, select the desired QoS templates in the fields on the *Run Options Defaults* tab in the Change User Settings window.

To specify Management QoS settings for a specific test, first open the test, then select the desired QoS templates in the fields on the *Run Options* tab in the Run Options window. The per-test settings override the default settings.

The following points summarize the operation of the Management QoS feature:

- QoS values that are to be used for all management traffic will be set through the IxChariot console—or through the use of IxChariot APIs—and transmitted to the endpoints in the setup phases, before the test begins running. Therefore, the same QoS values will be used for management traffic by all the pairs in a test. Different tests can use different values or even disable the use of QoS for management entirely.

- When a QoS value is selected for management traffic, that value will be used for all management traffic throughout the test, whether or not different pairs use separate logical networks for this traffic.

- There are two separate settings for QoS management traffic: Console Service Quality and Endpoint Service Quality.

- Only TOS and DSCP templates are supported for management traffic.

- The TOS and DSCP QoS templates will be available for use with management traffic only if all the stations involved in the communication (console and endpoints) support them.

- QoS values for management traffic will be used during the sending of all commands: pre-setup, setup, stop, and so forth. Refer to *QoS During the Three-Way Handshake* on page 7-10 for a description of the QoS values used during the initial three-way handshake.

- Management QoS settings will be used for all timing record transmissions and for clock synchronization.

The following limitations apply to the Management QoS feature:

- QoS for management traffic is not supported on the following:

    - hardware performance pairs

    - VoIP hardware performance pairs

    - IPX/SPX (used as the protocol for the management network)

- IPv6 (used for management traffic between the Console and Endpoint 1).

- Only TOS and DSCP templates are supported for management traffic.

The default Management QoS templates defined for IxChariot at any given time are saved in the registry, and loaded from the registry when the IxChariot Console starts up. Once saved in the registry, the default QoS templates are used in every new test created from that console (unless you select different QoS templates for a specific test).

The QoS settings that you set as run options affect only the current test and are therefore saved in the test file (tst file extension). Loading a particular test file will also load its saved run options, including the preferred values for management QoS. Furthermore, QoS values loaded as run options will override any previous default settings applied to the test.

However, both the registry and the test file contain only the name of the template used for management QoS. The actual template is stored in the servqual.dat file, as are test traffic QoS templates. Therefore, for a template to be applied properly, the description corresponding to its name must be present in this file. This is especially important when relocating tests from one machine to another. If a test uses a custom named template, it is necessary to either copy servqual.dat onto the new machine along with the test, or rebuild the template with known values before running the test.

QoS During the Three-Way Handshake

During the TCP three-way handshake connection establishment process, IxChariot sets a QoS value on the accept socket of the connection. Setting the QoS values for the pre-setup flow requires some processing that is not required for other traffic flows. Specifically, it requires the use of the INITIAL_MANAGEMENT_TOS value specified in the endpoint.ini file. During pre-setup, the console sends a buffer to Endpoint ,1, and Endpoint 1 sends a buffer to Endpoint 2. These buffers contain the QoS values that will be used for all management traffic during the test. However, the endpoints must parse those buffers to learn the QoS values. Once the pre-setup flow is complete, the endpoints use the QoS values that are specified for all the connections, including the three-way handshake.

During the pre-setup flow, if the console is the TCP sender, the QoS settings are handled as follows:

1. The console (sender) initiates the TCP connection, sending the initial SYN packet. The console will always use the QoS values set in Run Options.

2. Endpoint 1 (the receiver) responds with SYN and ACK. This is where the INITIAL_MANAGEMENT_TOS value from the endpoint.ini file is used. The receiver must use this value because it will not know the Run Options QoS value until it receives the buffer and parses it.

3. The console (sender) responds with an ACK. The connection is established.

4. The console (sender) sends the management buffer containing the QoS value.

5. Endpoint 1 (the receiver) parses the buffer and gets the QoS value. From this point on, the endpoints will use the Run Option QoS value to communicate with the sender.

   Note: For Linux and Unix endpoints, the receiving side sends an acknowledge packet as soon as the buffer is received (before parsing starts). Therefore, this ACK packet will use the QoS marking from endpoint.ini file. (However, if there was a previous run, the QoS markings will be inherited from that run.)

6. The receiver sends a response to the sender. The sender acknowledges. They close the connection. All these use the proper QoS values from run options.

   Notes:
   - When an endpoint is the sender, it will initiate the connection only after it has parsed the needed value from a previously sent buffer. Therefore, it does not require the use of the INITIAL_MANAGEMENT_TOS value from the endpoint.ini file.
   - If subsequent tests are run using different QoS settings, and the endpoints are not reset, the values from the previous run will be used during the pre-setup phase. The QoS values will be reset when the endpoint parses the buffer.
   - On Windows endpoints, if you change the Endpoint QoS value such that no template is specified (or a null QoS value is specified), the endpoint will not be able to reset the QoS value that was set during pre-setup. Therefore, the receiving side will use the value from the previous test throughout the pre-setup flow.

Refer to Chapter 9, *Quality of Service Testing*, for detailed information about creating QoS templates, and for procedures for using Quality of Service for application traffic.

Miscellaneous Run Options

Related Topics
Accessing the Run Options on page 7-1
Polling the Endpoints on page 7-5
Performance Testing on page 7-2

In addition to controlling options for ending a test run, reporting results, and handling initialization failures, the Run Options notebook lets you configure a number of other test parameters:

- **Collect endpoint CPU utilization**

  Instructs IxChariot to collect CPU utilization data for the endpoint computers executing a test. CPU utilization is the percentage of available CPU time spent executing all the currently active processes on a computer. It Is a good indication of available network resources and the degree to which they may be overtaxed. The CPU Utilization percentage is shown in the Percent CPU Utilization of E1 column and the Percent CPU Utilization of E2 column on the Raw Data Totals Tab of the Test window. These columns are only shown if this box is checked. CPU Utilization values are shown only after a pair completes; while a test is still running, "n/a" is shown.

  This percentage is an approximation based on CPU utilization samples taken during the test. Sampling starts during the first CONNECT or ACCEPT com-

mand in the script and continues until the script is complete. After the script completes, the average of the samples is calculated and reported to the Console.

Tests that collect endpoint CPU utilization should run for longer than 1 second in order to collect meaningful CPU utilization numbers. The endpoint samples the CPU utilization every 500 milliseconds and requires at least two valid samples in order to calculate the percentage of utilization.

For computers with more than one CPU, IxChariot calculates the CPU utilization percentage by adding together the percentages for each CPU and then dividing this amount by the number of CPUs. For example, if a computer contains two CPUs and one CPU is 50% utilized and the second CPU is idle, the CPU utilization of the process is calculated as (50% +0%)/2 =25% CPU utilization.

CPU Utilization is not supported by the endpoints on all operating systems. For Novell NetWare computers that contain more than one CPU, IxChariot returns the CPU Utilization of the first processor only. Consult "Endpoint Capabilities" in the *Performance Endpoints* guide for more information about operating-system support.

- **Collect TCP statistics**

  If you are running tests on an Ixia chassis, you can select "Collect TCP statistics" to instruct IxChariot to collect a set of TCP statistics for the endpoints executing a test. These statistics are collected for TCP packets that include SYN, FIN, ACK, and RST messages, as well as TCP connections, TCP retransmissions, and timeouts.

  TCP statistics collection requires IxOS 3.80 or higher and is available only on load modules with a Power-PC 405 or Power-PC 750 processor: TXS8, TXS4, STXS4, SFPS4, ALM-T8, ELM-ST2, TXS2, LM10GE700F1B-P, LM622MR, OLM1000STXS24.

  Note also that complete IPv6 TCP statistics are only available in IxOS 4.10 and above. In IxOS 4.0 (and earlier), the following IPv6 TCP statistics are not reported:

  - SynReceived
  - FinReceived
  - FinAckReceived
  - ResetReceived

  These are marked "N/A" in the TCP statistics tab.

  Refer to *Collecting TCP Statistics* on page 10-82 for more information about collecting TCP statistics.

- **Validate data upon receipt**

  Instructs all the endpoints in a test to validate each byte they receive. In some test environments, you may not be sure if the data is being transferred correctly from one endpoint to another. Endpoints can validate that what they receive is what they expected to receive by comparing payload bytes to the bytes specified in the script for the `send_datatype` and `control_datatype` variables.

Obviously, validation slows the performance measured at the endpoints. This function should be used for network stress testing and for testing of new hardware and software.

- **Use a new seed for random variables on every run**

  The sleep or `transaction_delay` script variable tells the endpoints to pause. The send and receive `buffer_size` variables specify the sizes of the buffers the endpoints use when sending data. If you change the default "Constant Value" to one of the four random distributions—Uniform, Normal, Poisson, or Exponential—the sleep durations and buffer sizes are based on random numbers, which are generated from a "seed." When IxChariot uses the same seed on consecutive runs, the random sleep durations or buffer sizes are generated in the same sequence.

  You should have IxChariot use the same seed when you are trying to get the same values for sleep durations or buffer sizes, run after run. Check this box if you want the sequence of randomly selected values to be different on each run.

- **Use fewer connections for test setup**

  Instructs the Console to use the fewest possible connections to contact each Endpoint 1 computer during the initialization phase of a test. Recommended for large tests. This option is automatically selected when you add the 501st pair to a test, unless you clear this box. For tests with >1250 pairs, this option cannot be disabled. Test initialization may take slightly longer when this box is checked.

  If you change your mind, click **Undo** to reset all the fields in the Run Options notebook to the values you had before you made any changes.

- **Enable Ixia hardware timestamps**

  Select this option when creating a test that uses video pairs. The Ixia hardware timestamps measure the delay factor with much greater precision on IXIA hardware.

- **Number of overlapped sends**

  Enter the number of overlapped sends that you want your scripts to use during test execution. You can enter any value from 2 through 999999. IxChariot will save this setting to the Registry as the default value for new tests.

  Overlapped I/O is a Microsoft Windows feature that allows sending (and receiving) multiple buffers in parallel. This can yield higher throughput and reduce CPU utilization.

Error Handling Tab

Related Topics

Run Options Tab on page 7-2
Factors Affecting Results on page 11-52
Performance Testing on page 7-2
How to End a Test Run on page 7-3
How to Report Timings on page 7-4
Pair Reinitialization and Graphs on page 11-7

Three options you can configure in the Run Options dialog box give you some control in failure-prone networks:

• **Stop run on initialization failure**

Initialization occurs when you click the Run button: the IxChariot Console contacts all Endpoint 1 computers, which in turn contact their Endpoint 2 partners. When you start a test, you are never completely sure whether all the endpoints can be reached. But if your test involves many different endpoints, you may want to run the test even if some of the endpoints are unavailable.

Checking the **Stop on initialization failure** box stops the run when any endpoint cannot pass all the initialization steps. If you leave the box cleared, the test will be run if at least one endpoint pair can be initialized. Those endpoints that cannot be initialized are omitted from the results and show errors.

• **Connect timeout during test**

You may be testing in noisy networks, where long connections are frequently dropped. IxChariot retries its connection attempts for the number of minutes you specify here. If that amount of time elapses and a connection still cannot be established, IxChariot declares a connection failure and issues the appropriate error message.

A connection attempt by an endpoint may consist of more than one Sockets `Connect` call, even if the timeout is set to zero. If the connection failure is due to a transient condition, such as network congestion, the endpoint will issue a fixed number of Sockets `Connect` calls per attempt. If the failure is due to a permanent condition, such as insufficient memory resources, the endpoint will issue one Sockets `Connect` call per attempt. This information applies to TCP connections only. See "Mapping Communication Commands to APIs" in the *IxChariot Script Development and Editing Guide* for more information about endpoint operating systems using TCP.

A value of 0 minutes means that connection failure is declared after the first unsuccessful series of connection attempts by the endpoints. The timeout option tracks errors encountered on `CONNECT_INITIATE` commands in a script; errors that occur on `SEND`, `RECEIVE`, or other commands will still cause a running test to stop. Thus, this timeout is most helpful in scripts with short connections.

The accepted values are in the 0-999 range.

• **Stop test after *x* running pairs fail**

You may want to let some pairs fail while the remainder of the pairs continue executing their scripts. IxChariot implements this option when the test enters the running state. Once in running state, IxChariot lets the specified number of pairs fail before terminating the test.

The accepted values are in the 0-9999 range.

> **Note**: When the Run until any pair ends radio button is selected in the How to end a test run pane in the Run Options tab, the Stop test after x running pairs fail option is disabled.

• **Allow pair reinitialization for setup**

When you select this option, IxChariot will attempt to reinitialize an endpoint pair that fails during the initialization phase of the test.

The reinitialization feature provides flexibility in test planning and execution. When running large-scale tests (up to 100,000 pairs), it is often desirable—if not necessary—for a test to proceed even if some of the pairs fail during initialization, and to allow IxChariot to reinitialize the pairs that fail.

When you select the *Allow pair reinitialization for setup* option, you must also set the following parameters:

• Try reinitialization *n* times

Specify the number of times that IxChariot should attempt to reinitialize the failed endpoint pair. The default is three attempts.

If IxChariot is not successful in reinitializing the pair after the specified number of attempts, initialization of the pair is considered to have failed. At this point, the pair will be included in the count of failed pairs (refer to the *Stop test after x running pairs fail* parameter above).

Accepted values are in the 1-99999 range.

• Try reinitializing after *n* milliseconds

Specify the number of milliseconds to wait between reinitialization attempts. The default is to wait ten milliseconds between attempts. Accepted values are in the 1-99999 range.

• **Allow pair reinitialization at runtime**

When you select this option, IxChariot will attempt to reinitialize an endpoint pair that fails during the execution of the test.

The reinitialization feature provides flexibility in test planning and execution. When running large-scale tests (up to 100,000 pairs), it is often desirable—if not necessary—for a test to proceed even if some of the pairs fail during test execution, and to allow IxChariot to reinitialize the pairs that fail. As another example, in many WLAN tests it is not uncommon for a pair to fail if the performance endpoint moves out of the coverage area of the access point (AP) while the test is in progress. By using appropriate reinitialization options, you can allow IxChariot to reinitialize the pairs that fail.

When you select the *Allow pair reinitialization at runtime* option, you must also set the following parameters:

• Try reinitialization *n* times

Specify the number of times that IxChariot should attempt to reinitialize the failed endpoint pair. The default is three attempts.

If IxChariot is not successful in reinitializing the pair after the specified number of attempts, initialization of the pair is considered to have failed. At this point, the pair will be included in the count of failed pairs (refer to the *Stop test after x running pairs fail* parameter above).

Accepted values are in the 1-99999 range.

• Try reinitializing after *n* milliseconds

Specify the number of milliseconds to wait between reinitialization attempts. The default is to wait ten milliseconds between attempts. Accepted values are in the 1-99999 range.

Result Ranges Tab

Related Topics

The Jitter Tab on page 11-29
The Lost Data Tab on page 11-31
Video Pair Defaults Tab on page 6-24
Jitter and Delay Variation on page 10-52

User-defined result ranges are provided so that you can more accurately determine call quality based on hardware and network thresholds, such as the size of the jitter buffer at the receiving endpoint and the relative standards in place to judge call quality. Refer to *The Jitter Tab* on page 11-29 and *The Lost Data Tab* on page 11-31 for more information about how results are presented.

Jitter (Delay Variation)

These ranges indicate the number of datagrams (expressed as a percent of the total number of datagrams sent) that experienced jitter. Delay variations, in milliseconds, are placed in groups, or ranges, so that you can compare them to such benchmarks as the size of the jitter buffer and the standards for call quality established for the hardware you are using.

• **Range**

By default, IxChariot configures delay variation jitter data in five ranges. Clear the boxes next to any ranges you don't want to see in your results.

• **Minimum (ms)**

Sets the lower limit of the range, in milliseconds. Automatic setting.

• **Maximum (ms)**

Sets the upper limit of the range, in milliseconds. Leave the value in place or enter a new value to change the default ranges. Ranges must be contiguous. As you change each range, the value for the next range minimum is automatically filled in.

Default settings for the five Jitter (delay variation) ranges are as follows:

• 0-10 ms

- 11-20 ms
- 21-30 ms
- 31-50 ms
- 51 + ms

Consecutive Lost Datagrams

Consecutive Lost Datagrams indicates the call quality based on the amount of the transmission that experienced noticeable loss. Also helps to determine the relative burstiness of datagram loss during the voice transmission.

- **Range**

  By default, IxChariot configures data on lost datagrams in 5 ranges. Clear the boxes next to any ranges you don't want to see in your results.

- **Minimum**

  Sets the lower limit of the range, in number of datagrams. Automatic setting.

- **Maximum**

  Sets the upper limit of the range, in number of datagrams. Leave the value in place or enter a new value to change the default ranges. Ranges must be contiguous. As you change each range, the value for the next range minimum is automatically filled in.

Default settings for the five Consecutive Lost Datagram ranges are as follows:

- 1-1
- 2-3
- 4-5
- 6-10
- 11 +

Datagram Run Options

Related Topics
Setting Datagram Run Options on page 10-4
UDP Throughput Testing on page 10-4

On the **Datagram** tab in the Run Options notebook, you can configure options for datagram testing (i.e., testing with the connectionless protocols UDP, IPX, and RTP). The IxChariot Console itself provides for the retransmission of lost data in test with these protocols. IxChariot does not retransmit when using a streaming script.

Expect to do some experimentation to find the best combination of settings for these options. If you change your mind, click **Undo** to reset all the fields to the values you had before you made any changes.

Non-Streaming Script Options

The first three options on the Datagram notebook page apply only to non-streaming scripts:

- **UDP Window Size**

  The number of bytes that can be sent from an endpoint to its partner without an acknowledgment. Calculate the number of datagrams sent in a window by taking the Window Size, dividing by the script's `send_buffer_size`, and rounding this value up. Default value is 1500.

- **Retransmission Timeout Period**

  The number of milliseconds the sender will wait, after sending for the first time or retransmitting a block of data, to receive an acknowledgment that the block was received. Default is 200 ms.

- **Number of Retransmits before Aborting**

  The number of times the sender will re-send a block of data for which an acknowledgment is not received. Default number of retransmissions is 50.

Streaming Script Options

The next two options apply only to streaming scripts.

- **Receive Timeout**

  The number of milliseconds the receiver waits before determining that the streaming script has ended. The default timeout value is 10000 ms. If Endpoint 2 is running on Windows, the minimum receive timeout value is 500 ms.

- **Multicast Time To Live (TTL)**

  Controls the forwarding of IP Multicast packets. Set this value based on how far you want the data forwarded.

  This field defaults to a value of 1 hop. However, the packets cannot cross a router if the TTL value is 1. If you run a test with a TTL less than the number of routers between the endpoints, the test fails and you receive the error mes-

sage **CHR0216**. You'll need to adjust the TTL to run a test with a multicast group that crosses a router.

RTP Options

There is one option that applies only to scripts that generate RTP traffic.

• **Use extension headers for RTP timestamps**

When you select this option, IxChariot will generate RTP packets with the header extension described in *RTP Header Timestamp* on page 5-8.

When you de-select this option, IxChariot will generate RTP packets with IxChariot legacy timestamps.

Data Rate Optimization Options

The four data rate optimization options apply only to streaming pairs. These options allow you to enable optimization algorithms for sending streaming traffic at a fixed rate. There are two algorithms: one to reduce jitter, the other to avoid throughput spikes by limiting the data rate.

• **Low sender jitter**

When checked, this option enables an optimization algorithm that uses very precise timers to reduce jitter. When this algorithm is enabled, the datagrams are sent by Endpoint 1 at more precise intervals than when the standard algorithm is in effect.

• **Limit data rate**

When checked, this option enables an optimization algorithm that limits the data rate (throughput) measured on intervals much smaller than a timing record interval.

The option is enabled only when the *Low sender jitter* option is enabled.

• **Data rate limit**

The option is enabled only when the *Limit data rate* option is enabled.

Use this field to specify the data rate limit for the streaming pairs in the test. The data rate limit is expressed as a percent of the required data rate defined in the script (the send_data_rate variable). For example, a value of 100 means that the limit is equal to 100% of the required data rate. The valid range of values is from 100 through 200.

• **Measured interval**

The option is enabled only when the *Limit data rate* option is enabled.

Use this field to set the measured interval for all the streaming pairs in the test. This is the interval (in milliseconds) over which IxChariot enforces the data rate limit. The valid range of values is from 1 through 999,999 ms.

The purpose of this optimization algorithm is to prevent IxChariot from exceeding the required data rate (which may happen with the standard IxChariot algorithm). Therefore, you will typically set this value to match the interval over which the network devices check for throughput spikes.

When To Use Data Rate Optimization

The standard IxChariot algorithm for sending traffic at a fixed rate is designed to maintain the required data rate (as defined by the send_data_rate variable in the script) on average over the entire timing record interval. For streaming scripts, this method can potentially lead to two undesirable outcomes:

• The datagrams will not always be sent at fixed intervals. In some cases, the send jitter will be high.

• The algorithm may exceed the required data rate over a small interval in order to maintain the average data rate on the timing record interval. Some network devices will treat this as a throughput burst and drop the traffic.

You can reduce or eliminate these problems by enabling the data rate optimization options. There are three combinations of settings that you can set:

• Uncheck *Low sender jitter*. In this case, IxChariot uses the standard algorithm for sending traffic at a fixed rate. The optimization algorithm is disabled.

• Check *Low sender jitter* and uncheck *Limit data rate*. In this case IxChariot will use the high precision timers and sleeps to ensure low sender jitter, but will not enforce any new throughput limits (the regular limit for the entire timing record interval will remain in effect).

• Check both *Low sender jitter* and *Limit data rate*. In this case, IxChariot will use the high precision timers and will also enforce the data rate limit over the requested interval.

Using a Data Rate Limit Greater Than 100

Note that enabling the *Limit data rate* option can result in a throughput rate that is lower than that required data rate (measured over the timing record interval). For example, if the first several send operations in a data stream transmit at a rate significantly lower than the send_data_rate value, subsequent send operations cannot exceed the send_data_rate in an effort to make up for the initial lower transmission rate. In contrast, the standard IxChariot algorithm for sending traffic at a fixed rate will, if necessary, exceed the send_data_rate to maintain the required data rate. Setting the *data rate limit* value to 100% results in a trade-off: it eliminates spikes at the cost of throughput.

If a *data rate limit* value of 100% results in unacceptable loss of throughput in a test, you can increase the value (up to 200%) to increase throughout while keeping the throughput spikes to an acceptable level.

Data Rate Optimization Limitations

Note the following limitations on the use of the data rate optimization options:

• The data rate optimization options are only applicable for the following Performance Endpoints: Windows 32-bit, Windows 64-bit, and Linux On PowerPC.

• The data rate optimization options require Endpoint 1 (the sender) to be running a Performance Endpoint software release of 6.70 or higher.

- The data rate optimization options are intended for tests that use fewer than ten streams. Internal tests at Ixia show that the settings are most effective when between four and seven streams are used.

Ixia Port Configuration Tab

Related Topics
Stack Manager User Guide
The Tools Menu on page 3-10

The Ixia Port Configuration tab provides the following run options:

- **Apply only Endpoint DoD package**

  Select this option if you want to apply the endpoint DoD package only, without applying the configuration.

> **NOTE**: Enabling the **Apply only Endpoint DoD package** option disables the **Deconfigure ports and release ownership after the test ends** option.

- **Deconfigure ports and release ownership after the test ends**

  Select this option to deconfigure the Ixia ports used in your test(s) and release ownership on them as soon as the test(s) end.

- **Use default IxTCLServer**

  If you select this option, IxChariot will use the TCL Server on the default Master chassis from the configuration, (the first chassis in the list).

  If you de-select this option, the **User defined Ix-TCLServer** text box is enabled. In this case, you need to specify the location of the specific TCL Server to use.

8 *Large-Scale Tests in IxChariot*

IxChariot is a highly-scalable test architecture, allowing you to create and run tests using as few as one endpoint pair to as many as 100,000 endpoint pairs. This chapter focuses on the requirements for creating large-scale tests.

Topics in this chapter:

Maximum Number of Pairs

There are two aspects to determining the maximum number of pairs that you can use in an IxChariot test:

- *Total Number of Pairs Per Test*
- *Maximum Number of Pairs Per Endpoint*

Total Number of Pairs Per Test

IxChariot supports a total of 100,000 pairs in a test, assuming that your test network has the capacity to support this maximum and your IxChariot software license permits it. This limit is independent of the specific Performance Endpoints that you are using.

For example, if you are running Windows Vista on your endpoint computers, you can design a test that uses 66 endpoint computers: 33 endpoint 1 computers, and 33 endpoint 2 computers. Each of the 33 endpoint 1/endpoint 2 pairs could support 3,000 concurrent TCP connections, for a total of 99,000 TCP connections.

Maximum Number of Pairs Per Endpoint

The maximum number of pairs that IxChariot supports for an individual IxChariot Performance Endpoint depends upon the following factors:

• The specific Performance Endpoints that you are using.

• The layer 3 and layer 4 protocols that your test requires.

• The amount of RAM installed in your endpoint machines.

• The specific operating system that you are running in your endpoint machines.

• The Run Options that you have set for your test.

• The Ixia Load Modules that you are using in your test (applicable to the IxOS Performance Endpoint only).

For example, the HP-UX Performance Endpoint, running on an endpoint machine with 1 GB of RAM, will support of maximum of 200 TCP pairs and a maximum of 150 UDP pairs.

For More Information

Refer to the "Endpoint Pair Capacity" topic in the *IxChariot Performance Endpoints* guide for a comprehensive matrix listing the memory requirements for each Performance Endpoint and the number of pairs supported for each protocol.

IxChariot Console Memory Requirements

If you are planning to run large-scale tests, you need to ensure that your IxChariot Console PC has sufficient memory to support your requirements. Table 8-1 identifies the minimum requirements and recommended configuration for various pair counts.

Table 8-1.   IxChariot Console RAM Requirements for Large Tests

Number of Pairs	Minimum Requirements	Recommended Configuration
1,000	Pentium III, 90 MB free RAM	Pentium 4, 256 MB RAM
5,000	Pentium 4, 140 MB free RAM	Pentium 4, 512 MB RAM
10,000	Pentium 4, 204 MB free RAM	Pentium 4, 512 MB RAM
50,000	Pentium 4, 732 MB free RAM	Pentium 4, 2 GB RAM
100,000	Pentium 4, 1.5 GB free RAM	Pentium 4, 2 GB RAM

Another factor to consider when evaluating memory requirements for your IxChariot Console PC is the number of tests that you can open at the same time. Table 8-2 on page 8-3 identifies the *approximate* number of tests that you will be able to open concurrently, based on the number of pairs in the test, the number of timing records per pair, the number of timing records, and the configuration of

the PC. The results shown were obtained by Ixia using a PC with a Pentium 4 processor with 2 GB of RAM.

Table 8-2.   RAM Requirements for Opening Multiple Tests

Number of Pairs PerTest	Number of Timing Records Per Test	Number of Timing Records Per Pair	Number of tests Opened Without Exhausting Available Memory
1,000	50,000	50	72
10,000	500,000	50	10
20,000	1,000,000	50	5
50,000	500,000	10	2
100,000	1,000,000	10	1

The tests from which the data in Table 8-2 were derived included test results.

Testing with 500 or More Pairs

Broadly speaking, we use the term "large-scale test" to refer to any test that approaches the maximum pair limitations for the Performance Endpoints that you are using. Using this definition, a test using Linux x86 Performance Endpoints with 300 pairs and a test using Windows CE Performance Endpoints with 85 pairs are both large-scale, in the sense that they can stress the Performance Endpoint and the endpoint machines.

This section, however, focuses on adjustments that IxChariot makes—and that you may need to make—once you define a test with 500 or more pairs. It is organized into the following topics:

* *Graphing Adjustments* on page 8-3

* *Results Grouping* on page 8-4

* *Setting Run Options for Performance Testing* on page 8-4

* *Additional Recommended Run Options* on page 8-6

* *Configuration Changes for Linux and Windows* on page 8-6

Graphing Adjustments

Because each endpoint pair is represented by an individual line in IxChariot graphs, a graph of more than 500 pairs cannot be displayed effectively. Unless you take steps to limit the number of elements in a graph, you will see the following message in the graphing area once the results are in:

> Too many pairs are selected to graph by pair

By default, IxChariot will attempt to graph each endpoint pair individually. When your test includes a large number of pair, you should consider placing the endpoints in groups, and graphing the groups rather than graphing the individual pairs. Organizing pairs in groups means you'll be able to see and use your results much better. Once endpoints are grouped, click **Graph Configuration** on the

View menu. Under the heading Graph Content, click **Groups** to graph your end-point pairs by their group name. IxChariot's ability to graph by group depends on the number of timing records in the test. Organizing endpoints into multiple groups instead of a single large group improves GUI performance.

You can disable graphing altogether by clicking **Disable Graphing** on the View menu. The graph legend is automatically disabled when graphing is disabled.

Another graphing option is to unmark some endpoint pairs for graphing until you get below the 501-pair cutoff point. The graph icon next to a pair in the Test window indicates that it will be graphed. To unmark a pair, highlight it and click **Unmark selected item(s)** on the Edit menu.

Results Grouping

When you create a test with 1,001 or more pairs, the IxChariot Console automatically groups the pairs in blocks of 500, as shown in Figure 8-1.

Figure 8-1.   Automatic Grouping of Pairs
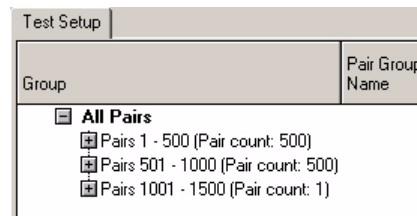
Setting Run Options for Performance Testing

When you create a test with 500 or more pairs on the same endpoint, it is recommended that you enable the *Set the test run options for performance testing* run option for the test.

When you enable the *Set the test run options for performance testing* option, IxChariot chooses the following settings for your test:

*   **How to report timings**: *Batch.*

    Report results in batch mode, after the test run completes. Batch reporting does not consume extra CPU resources during the test. It Is extremely difficult for your network to report test results in real time during a large test. Each endpoint can generate hundreds of timing records during a single short test, and every record must be returned to the IxChariot Console. If they are reported in real time, timing records could overwhelm the link between the Console and the Endpoint 1 computers.

*   **Poll endpoints**: *Disabled.*
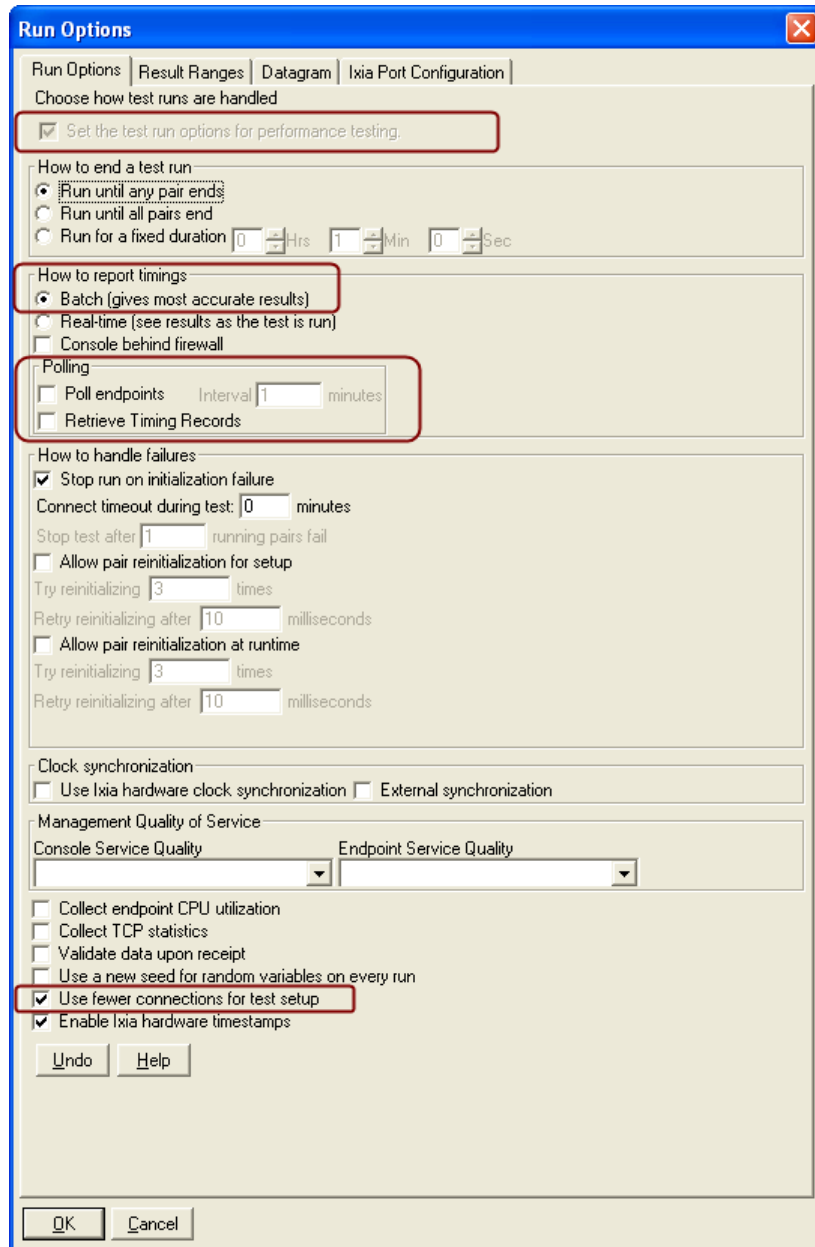
    Polling consumes network resources, creating extra data flows that interfere with measurements.

*   **Use fewer setup connections**: *Enabled.*

    For very large tests, it is essential for the Console to use fewer connections to send test setup information to the endpoints.

These options are shown in Figure 8-2 on page 8-5.

Figure 8-2.   Run Options for Performance Testing

Additional Recommended Run Options

When you are running a large test that may stress the IxChariot Console, it is doubly important to select settings that will yield optimal IxChariot GUI performance and accurate test results.

Many of the default Run Options are already appropriate for performance testing. Following is a set of recommended run option setting that you should consider for each large test that you create:

- **How to end a test run:** The default setting is "Run until any pair ends." With large tests, it may be more appropriate to run the test for a fixed duration.

- **How to handle failures:** The default setting is "Stop run on initialization failure." With large tests, it is more likely that a single pair may fail during test initialization or test execution, in which case the test will stop if you are using the default setting. You can use the **Allow pair reinitialization** settings to allow an endpoint to reinitialize a pair that fails, either during test initialization or test execution.

- **Clock synchronization Hardware Timestamps**: Disabled

- **Collect endpoint CPU utilization**: Disabled

- **Collect TCP statistics**: Disabled

- **Validate Data on Receipt**: Disabled

- **Use a New Seed for Random Variables on Every Run**: Disabled.

- **Enable Ixia hardware timestamps**: Disabled

The Clock synchronization, CPU collection, TCP statistics collection, Data Validation, and New Random seed options are all automatically turned off for performance testing because they create extra flows of data on the network and require unnecessary processing overhead.

Configuration Changes for Linux and Windows

When using the 32-bit and 64-bit Windows and Linux Performance Endpoints, you may need to make some configuration changes to ensure that they can accommodate the maximum number of concurrent network connections.

Linux Configuration Changes

Depending upon your Linux distribution and release level, you may need to increase the maximum number of open files descriptors when using Linux computers as endpoints. To do so, use the following command:

```
ulimit -n descriptors
```

where *descriptors* is the number of file descriptors. The recommended value for ulimit is approximately:

```
2 * number-of-pairs + 100
```

Therefore, for 300 pairs the value would be approximately 700.

Windows Configuration Changes

When running a test that uses Windows computers as endpoints and requires a large number of pairs (at or near the maximum), it is recommend that you increase the TcpMaxDataRetransmissions. If you leave the TCP maximum retransmissions value set to the default, your test may fail due to a TCP timeout.

Microsoft provides detailed instructions for modifying the TCP/IP stack parameters to allow for more TCP retransmissions; the instructions are available from this web site: http://support.microsoft.com/kb/170359/EN-US/.

Ixia tests have shown that changing the TcpMaxDataRetransmissions parameter to 16 (from the default of 5) allows a 500-pair test to execute successfully. Setting this parameter to 30 should be sufficient for a test that includes 3,000 pairs (if you are using Windows Vista).

For More Information

Refer to the "Endpoint Pair Capacity" topic in the *IxChariot Performance Endpoints* guide for a list of the memory requirements for the 32-bit and 64-bit Linux and Windows Performance Endpoints.

Modifying Application Scripts

As you increase the number of pairs used in a test, it is important that you make appropriate modifications to the scripts to accommodate the increased load on the network. The topics that follow explain why this is necessary and provide recommendations for modifying the scripts:

- *Adjusting for Maximum Timing Records* on page 8-7
- *Scripts Suitable for Large-Scale Testing* on page 8-11

Adjusting for Maximum Timing Records

Because there are a number of interrelated factors that determine the rate and volume of timing records returned to the IxChariot Console, there are various ways that you can modify your application scripts to ensure that the endpoints will not need to buffer timing records and potentially cause a test to fail. These modifications are needed to control one or the other of these two factors:

- The *rate* at which timing records are returned to the console: If the timing records are generated at too fast a rate, the endpoint will need to buffer them, potentially leading to buffer overflows on the endpoint computer.
- The *volume* of data that is generated: If Endpoint 1 transmits more records per second than the console can accept, the endpoint will need to buffer them, potentially leading to buffer overflows on the endpoint computer.

Reducing the Volume

The most direct way to reduce the total number of timing records that are returned to the Console is by changing the number_of_timing_records variable in

the script. Table 8-3 shows an example in which the number of timing records is reduced from 100 to 10. This reduces the total number of timing records by a factor of 10 (in this example).

Table 8-3.   Change number_of_timing_records In a Script

Number of Pairs	Number of Timing Records Per Pair	Total Number of Timing Records for the Test
10,000	100	1,000,000
10,000	**10**	100,000

Reducing the number of timing records has the following effects:

• It reduces the amount of data (total number of timing records) that is transmitted from Endpoint 1 to the Console.

• It reduces the granularity of the test results. Reducing the granularity of the test results is not always a desirable result, but it is the trade-off for reducing the number of timing records.

Controlling the Rate

Another way to avoid buffer overflows is to control the rate at which the timing records are generated and returned to the Console. There are two basic approaches:

• *Change the File Size* on page 8-8

• *Change the Number of Transactions* on page 8-9

Change the File Size

One method for controlling the rate at which records are generated and returned to the Console is by changing the File Size variable in the script, possibly in conjunction with reducing the number of timing records generated per pair. Table 8-4 shows an example in which both the number of timing records and the file size (number of bytes) are modified.

Table 8-4.   Change file_size and number_of_timing_records in a script

Number of Bytes (file_size)	Number of timing Records Per Pair	Total Number of Bytes for the test	Test Duration (in minutes)	Timing Records per Second
100,000	100	10,000,000	54 seconds	1.919
100,000	**10**	1,000,000	6 seconds	1.923
1,000,000	**10**	10,000,000	54 seconds	0.193

Table 8-4 shows how changes to the file size and the number of timing records per pair impact the transaction rate (number of timing records transmitted per second):

- The first row shows an example of a test that requires 54 seconds to run through to completion. This test transmits 1.919 timing records per second.

- The second row shows that reducing the number of timing records by a factor of 10 reduces the test duration by the same factor. As with the first row, this test generates approximately 1.9 timing records per second.

- The third row shows that reducing the number of timing records by a factor of 10 while increasing the file size by a factor of 10 allows the test to attain the original test duration and generate the original volume of data (10,000,000 bytes). However, this test generates 0.193 timing records per second, as compared to 1.923 timing records for the test in row 2.

The duration of a test must also be considered. For example, if a test cuts 10 timing records per pair in 54 seconds (as shown in Table 8-4 on page 8-8), it will cut approximately 600 timing records in one hour and 59,400 timing records in 99 hours. If the test has 100,000 pairs it will cut 5,940,000,000 timing records total per test. In a case such as this, you may need to reduce the number of timing records per pair to limit to total number of timing records that will be generated for the test.

As with the example in Table 8-3 on page 8-8, these adjustments may result in a reduction in the granularity of the test data. Your test objectives will determine which trade-offs you will need to make in your test.

Change the Number of Transactions

Another way to reduce the rate at which timing records are returned to the Console is to increase the number of transactions per timing record, while reducing the number of timing records generated.

For example, Table 8-5 compares two pairs from a test. They both transmit the same volume of data within the same amount of time, but pair 2 requires approximately twice as much time to transmit a single record (because each record contains twice as much data).

Table 8-5.   Effect of Changing Transactions Per Record

Pair Number	Number of Timing Records	Transactions Per Record	Bytes Sent by E1 per Timing Record
1	100	1	100,000
2	50	2	200,000

Figure 8-3 shows a comparison of these timing records. The elapsed time for the first record for pair 1 is 0.524 seconds, whereas the elapsed time for the first record for pair 2 is 1.046 seconds.

Figure 8-3. Comparison of Timing Records

Reducing Buffer Sizes

Another factor in successfully running a test that approaches the maximum number of pairs is to adjust the sizes of the SEND and RECEIVE buffers. In some scripts, the default value for these buffers is DEFAULT, which allows the operating system to set the value. However, because the buffers are allocated for each pair, your buffers may exhaust all available memory unless you set the buffers to a smaller size.

In general, you should consider reducing the buffer sizes as you increase the number of pairs used in a test. This is especially important if you enable the "Validate Data on Receipt" run option.

Other Script Modifications

Refer to *Firewalls and Fixed Ports* on page 8-13 for information about other script changes that may be required if you are testing in a network with an active firewall.

Scripts Suitable for Large-Scale Testing

Some of the IxChariot application scripts are especially suitable for large tests.

Recommended Scripts for Large-Scale Testing

Following is a description of the scripts that are suitable for large tests, with recommendations for scaling them up for larger tests:

* Throughput.scr and High_Performance_Throughput.scr

  These scripts test network throughput by sending a large file from Endpoint 1 to Endpoint 2. Endpoint 2 receives and acknowledges the file. The High_Performance_Throughput.scr script differs in the size of the file that is sent (10,000,000 bytes versus 100,000 bytes) and in the setting of the send_buffer_size. Refer to *Adjusting for Maximum Timing Records* on page 8-7 for a description of the types of modifications you may want to consider for these scripts.

* Large_Test_Response_Time.scr

  This is a modified version of the Inquiryl benchmark script. It emulates 50,000 users each performing 10 inquiry transactions per hour. The script uses a randomized transaction_delay with a normal distribution in a range of 10 to 26 seconds. Use this script when a test calls for a large number of users performing short transactions. Depending on what transaction_delay value is chosen, the test should be run (if using run for duration) for a longer duration than the delay, otherwise you may not get very many timing records.

* Large_Test_Throughput.scr

  This is a modified version of the Filesndl benchmark script. It transfers a file at a send_data_rate of 2 KBps. It provides for a good test of many users, all establishing connections and transferring data at a given rate. Each pair represents 1 user, although the script could be modified to emulate multiple users. See *Emulating Multiple Users* on page 10-92. This test works for any number of pairs (up to maximum supported by your Performance Endpoint). You might modify the send_data_rate for smaller tests, but be careful: you can easily flood the network. This script is good for testing the throughput capacity of your network ("filling the pipes").

Creating a Stepped-Load Test

If you want to create a stepped-load test, in which a large number of pairs (500 or more) gradually increases in increments of, for example, 500, start by creating the maximum number of endpoint pairs that will be running simultaneously. (Be careful not to exceed the number of pairs allowed by your license.) Select Large_Test_Response_Time, or another script that uses long connections and sends a small amount of data. Then edit the script to use a different initial_delay for each group. The initial_delay variable determines how long an E1 will wait before it begins executing the script. Rename the script each time you edit it so that you can keep track of which pairs started their scripts first and which ones started later, when the network was already busy.

Tips for Running Large-Scale Tests

The IxChariot Console supports tests with thousands of simultaneous endpoint pairs, but the operating systems of the Console and endpoint computers may create some problems unless you take steps in advance. Depending on your hardware and operating systems, you may need to limit the number of pairs you configure on each endpoint computer, even when the endpoints are multiprocessor computers. See the following topics for details.

- *Reducing the Number of Threads* on page 8-12

- *Keeping Log File Size to a Minimum* on page 8-12

- *Using the Command Line to Run Large Tests* on page 8-12

- *Device Drivers* on page 8-13

- *Firewalls and Fixed Ports* on page 8-13

- *Data Retransmission Timeouts* on page 8-15

- *Synattack Protection* on page 8-15

- *Configuring Virtual Addresses on Endpoint Computers* on page 8-15

Reducing the Number of Threads

If an endpoint computer contains many virtual addresses or multiple adapters, use a single unique setup value for all addresses on that computer. This allows IxChariot to use fewer threads during test setup, and it means you can have multiple unique pair combinations with fewer computers. Configure test setup values by selecting pairs in the Test window and clicking **Edit Pair Setup** on the Edit menu. See *Cloning Hardware Performance Pairs* on page 5-25 and *Configuring Virtual Addresses on Endpoint Computers* on page 8-15 for more information.

Keeping Log File Size to a Minimum

Upon startup, IxChariot displays a popup warning if your error log file exceeds 5 MB in size. Log files greater than 5 MB may inhibit performance, particularly when you are working with large numbers of endpoint pairs. Running tests with 500 or more pairs can rapidly increase the size of your `Chariot.log` or `runtst.log` files even if only a small percentage of all the pairs fail. Massive log files are hard to read and consume extra CPU resources.

If you see this popup message, you should delete your log file and let IxChariot begin logging again with an empty file. If you want to preserve the log file, make a backup copy. IxChariot's Error Log Viewer does not support deleting individual log entries, but you can open and read the log file before you decide whether to delete it; see *The Error Log Viewer* on page 12-8 for a full discussion. Refer to *FMTLOG: Formatting Binary Error Logs* on page 5-69 for more information about IxChariot's error logs.

Using the Command Line to Run Large Tests

Related Topics
RUNTST: Running Tests on page 5-67
FMTTST: Formatting Test Results on page 5-70

If your test contains more than 10,000 pairs, you should not plan to run it using the IxChariot GUI, which will probably not run as well as you'd prefer. Instead, you should run it from the command line. Refer to *RUNTST: Running Tests* on page 5-67 for more information about RUNTST.

By default, RUNTST runs in "quiet mode." This means that you'll see less output while it is running. If you want to see the output while RUNTST is running, use the -v flag with RUNTST:

```
runtst tst_filename [new_test_filename] [-t N] -v
```

This returns RUNTST to verbose mode; however, this is not recommended for large tests.

RUNTST is also optimized for Telnet so that you can easily run it from a remote location.

To export your results, use the command-line program FMTTST. For large tests, exporting to HTML is not recommended; HTML files will be extremely large. In addition, your formatted output might not show results for all pairs. That's because pairs in exported output are either collapsed or expanded, depending on the settings you selected in the Test window. See *Sorting and Grouping Pairs* on page 5-26 for more information.

We recommend exporting results for large tests to .CSV format.

Export your results to .CSV format using the -v flag:

```
FMTTST tst_filename [output_filename] -v
```

By default, all pairs/groups are marked for export, which potentially creates a very large .CSV file. FMTTST does not offer an option to unmark the pairs you don't want to include in the exported file. Load the test into the IxChariot GUI and unmark desired pairs and/or groups by clicking **Unmark Selected Item(s)** on the Edit menu. Then export the test using FMTTST.

FMTTST is discussed in *FMTTST: Formatting Test Results* on page 5-70.

Device Drivers

To achieve maximum endpoint performance during large tests, it is necessary to disable any software or hardware device drivers that may be filtering or monitoring network traffic. Some examples of programs you should close are protocol analyzers, such as Microsoft's NetMon or Sniffer Pro, or Ixia IxProfile.

Firewalls and Fixed Ports

Related Topics
Firewall Testing on page 10-13
Firewall Options Tab on page 6-32

Testing with user-defined ports, which you might do if your network has an active firewall, involves some advance configuration when your tests are larger than 1250 pairs. Specifically, the operating systems may not allow the endpoints to process thousands of connections running to the same port at the same time. Some pairs may indicate that no endpoint program is installed on their partner

computer (CHR0204), or that the connection attempt timed out (CHR0200). Others may report that they cannot complete their scripts because the assigned port is already in use (message CHR0206).

To resolve these problems, take some or all of the following steps:

1. Edit the application script you are using. If you are running many endpoint computers to one server using the same port number, set the `initial_delay` variable to use a Uniform random distribution. Set the Lower Limit to 0 seconds and the Upper Limit to 5 seconds or so. This gives the Endpoint 2 computers a chance to process incoming connections before too many other requests arrive and are rejected by the operating system. A randomized delay avoids flooding the server queue.

2. Set the "Connect timeout during test" Run Option to a value greater than 0 minutes. This will cause the endpoint to retry test connections if a failure is detected.

3. If you are an experienced user, you can change a Registry setting for your Windows operating system. For Windows 2000/2003, the parameter in question is located in the Registry under

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
   \Tcpip\Parameters\
   ```

   Create a new `DWORD` value, `TcpMaxConnectRetransmissions`. The valid range is 0-255 (decimal). The Default is 2.

   This parameter determines the number of times that TCP retransmits a connect request (`SYN`) before aborting the attempt. The retransmission timeout is doubled with each successive retransmission in a given connect attempt. The initial timeout is controlled by the `TcpInitialRtt` Registry value.

4. Running any Windows *Server* operating system (such as Windows 2000 Advanced Server) as Endpoint 2 will usually avoid problems at the endpoints when running large tests with the same port number. All of the Windows *Server* operating systems allow more incoming connections to be processed at one time.

5. Some operating systems limit the number of TCP/IP ports that applications can use. For example, Windows 2000/2003 use the port range 1025-5000. You can change this with the following Registry setting:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
   \Tcpip\Parameters\
   Parameter: TcpMaxUserPort
   ```

   Try this if you receive message CHR0206 during a test run. It increases the number of ports made available for the endpoints.

The **Firewall Options** tab in the Change User Settings dialog lets you decide which port the Console uses to contact the Endpoint 1 computers and whether to use a fixed port for communications between Endpoint 1 and Endpoint 2. For best results when running large tests through a firewall, set the option to "**Use Endpoint 1 fixed port**." That way, no extra data correlator is added to the header in test flows, meaning there's less overhead. This option doesn't work for firewalls configured to provide network address translation (NAT). See *Firewall Options Tab* on page 6-32 for more information.

Data Retransmission Timeouts

As a general rule, instead of using just a few multiprocessor computers to represent thousands of endpoints, consider using more endpoint computers with fewer pairs per computer to run very large tests. Under heavy stress, TCP/IP may abort the connections if data is retransmitted for a given time without an `ACK`. If this happens, the operating system may leave large numbers of threads in `TCP_Receive` state after a test completes instead of freeing them for future connections. This situation may cause the test to hang and make it impossible to stop it. Or you might see the message CHR0245.

Try changing a Registry setting to instruct the stack to try to retransmit the data for a longer time period. In Windows 2000/2003, add a Registry `DWORD` value, `TcpMaxDataRetransmissions`:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
  \Tcpip\Parameters\
```

The valid range is 0-4294967295 (decimal). The Default is 5. We have typically used a value of 30 for large tests, but you may need to do some experimenting.

Synattack Protection

If you are planning to run tests with a large number of pairs using the same port number, you may encounter some problems. The endpoint operating systems have a limited queue for incoming connections to the same port. When an endpoint computer cannot process the connections fast enough, the message CHR0204 is returned. To alleviate this problem, you have several options:

If a Windows 2000/2003 endpoint computer is configured for *synattack protection*, problems could arise when running tests with large numbers of connections using the same port on the same computer. Synattack protection limits the number of simultaneous connections that can be established by delaying the allocation of route cache entry resources. Check your Windows 2000/2003 Registry settings for the following to see if synattack protection is active on your computer:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:
  \Tcpip\Parameters\

Parameter: SynAttackProtect
```

If the value is set to anything above 0, synattack protection is active.

Configuring Virtual Addresses on Endpoint Computers

"Virtual" addresses, which serve as multiple unique IP addresses on a single computer, let you run more pairs on fewer computers. Creating virtual addresses for use in testing is straightforward in the Windows operating systems. You can follow similar steps to configure virtual addresses for a UNIX operating system, but the steps vary for each type of UNIX. Ixia provides a utility, `SetAddr.exe`, to help you build virtual addresses in Windows 2000 and Windows Server 2003; it is available free on our Web site at http://www.ixiacom.com/support/chariot/utils/ and now ships with the Windows endpoints. See the *Performance Endpoints* guide for more information.

Below are instructions for manually creating virtual test addresses in Windows 2000 / Windows Server 2003, Red Hat Linux, and Sun Solaris.

Configuring Virtual Addresses on Windows

Follow these steps to create virtual test addresses in Windows 2000 and Windows Server 2003:

1. Right-click **Network Neighborhood** and choose **Properties**.

2. Select the connection for the device you are configuring. Right-click and choose **Properties**.

3. Select **TCP/IP** from the list and click **Properties**.

4. In the Internet Protocol (TCP/IP) Properties dialog, click **Advanced**.

5. In the Advanced TCP/IP Settings dialog, click **Add**.

6. Enter the first virtual address, then click **Add**.

7. Repeat 6 as many times as necessary.

8. Click **OK** to exit the dialog and save your changes.

9. Reboot the computer.

The virtual addresses you use will depend on the addressing scheme on your network; consult a network administrator. Windows supports hundreds of virtual addresses per NIC. But be careful. The TCP/IP stack has to work harder whenever packets arrive: checking to see if a packet is destined for the computer requires more work if the list of addresses is long. In our own testing, we've created as many as 2500 addresses on Windows 2000/2003 computers with 512 MB of RAM. The amount of RAM is very important.

Configuring Virtual Addresses on Linux

Red Hat Linux ships with some scripts that help you create virtual addresses. These scripts are located in `/etc/sysconfig/network-scripts`.

The `ifup-aliases` script contains instructions on how to create multiple address aliases for the computer.

You'll need to create a series of files called `ifcfg-eth0-range[n]`, where *n* designates the number you've assigned to each ranges. You need a separate file for each Class C block of addresses. Here's a sample file:

```
IPADDR_START=10.41.20.1
IPADDR_END=10.41.20.254
```

This file adds a range of addresses from 10.41.20.1 to 10.41.20.254 as aliases. If you are creating more than 1 range of addresses you will need to add the `CLONENUM_START=x` statement to the `ifcfg-eth0-range` files. The `CLONENUM_START` command indicates the interface clone number to use for this range. The interface numbers start with 0 and increment until the last address in the range. So, for example, if you have a range of `10.41.20.1` to `10.41.20.254` the interface number would start at 0 and end at 253. If you have another range of `10.41.21.1` to `10.41.21.254` you would need to set the `CLONENUM_START` to 254, which is the next available interface after the first range. After creating the alias files, run the `ifdown eth0 and ifup eth0`

scripts (where `0` designates the interface number) to apply the aliases to the running configuration. In our testing, we've created as many as 1,000 addresses on Red Hat Linux version 6.2.

Configuring Virtual Addresses on Sun Solaris

Solaris 2.x allows you to assign more than one IP address per interface, using `ifconfig`. Before v2.5, this feature is undocumented. Here's the syntax:

```
ifconfig IF:N plumb
ifconfig IF:N ip-address up
```

where "`IF`" is an interface (e.g., `le0`) and `N` is a number between `1` and `<MAX>`. Removing the pseudo interface and associated address is done with

```
ifconfig IF:N 0.0.0.0 down
```

In newer release you must use the following command, but beware that this unplumbs your real interface on older releases, so try the above command first:

```
ifconfig IF:N unplumb
```

As with physical interfaces, all you need to do is make the appropriate `/etc/hostname.IF:X` file.

The maximum number of virtual interfaces, `<MAX>` above, is 255 in Solaris releases prior to 2.6. Solaris 2.6 and Solaris 2.5.10 with the Solaris Internet Server Supplement (SISS) allow you to set this value with `ndd`, up to a hard maximum of 8192. Here's how to do it:

```
/usr/sbin/ndd -set /dev/ip ip_addrs_per_if 4000
```

There is no limit inspired by the code; so if you bring out `adb` you can increase the maximum even further.

9

Quality of Service Testing

This chapter provides a description of the options and procedures for incorporating Quality of Service into your IxChariot tests.

Topics in this chapter:

QoS for Management and Application Traffic

This chapter describes procedures for creating and managing QoS templates, and for selecting and applying QoS templates to the *application traffic* in a test. Beginning with IxChariot 6.50 Service Pack 2, you can also apply QoS templates to the *management traffic* generated by an IxChariot test.

Refer to *Management Quality of Service* on page 7-8 for more information about the application of QoS templates for management traffic.

QoS Overview

In IP networks, Quality of Service (QoS) is a set of standards and mechanisms that identify and handle application traffic according to defined service levels. Implementing QoS helps to ensure that bandwidth allocation is based on the specific requirements of the type of traffic traversing the network. Without QoS, all traffic is treated equally. With QoS, network devices can give priority to mission-critical traffic and media-rich traffic, and thereby minimize network delays and data loss.

IxChariot tests support the following QoS models:

*   Layer 2 QoS:
    *   Microsoft Generic QoS (*Microsoft GQoS*)
    *   *Layer 2 Priority QoS (Class of Service)*
*   Layer 3 QoS:
    *   *IP TOS*
    *   *DiffServ*
    *   Microsoft Vista qWave (*Microsoft qWave*)
    *   Microsoft Generic QoS (*Microsoft GQoS*)
*   Microsoft Windows CE 6.0 WMM (*QoS on Microsoft Windows CE and Microsoft Windows Mobile*)

Each of the QoS models is described below.

Layer 2 Quality of Service

IxChariot provides support for Layer 2 Quality of Service on Windows and Linux:

*   on Windows via Generic QoS - *Microsoft GQoS*
*   on Linux via L2 Priority QoS (Class of Service) - *Layer 2 Priority QoS (Class of Service)*

Microsoft GQoS

Generic QoS (GQos) is a Microsoft API that software developers can use to create QoS-aware applications. IxChariot can use the GQoS support provided by WinSock 2 on Windows XP and newer. You access this support when you select one of the GQoS template names from the Service Quality field when adding RTP, TCP or UDP pairs. Each QoS template name tells the network what kind of service the connection requires. Predefined QoS templates are part of the "Windows QoS Service Provider" software.

Setting GQoS Template Values

Use the procedures below to set the Generic QoS template values via Microsoft Windows Group policy for both Layer 2 (Layer 2 Priority QoS/Class of Service) and Layer 3 (DSCP).

Setting GQoS Template Values for Layer 2 Priority/Class of Service

The values of the predefined Generic QoS templates/service types in IxChariot can be set as follows:

1. Enter *gpedit.msc* in the command prompt;

2. In the Group Policy window that opens, click **Computer Configuration>Administrative Templates**;

3. Click **Network>QoS Packet Scheduler**;

4. Click **Layer-2 priority value**;

5. Set the properties of one of the available predefined templates/service types:

   a: Non-Conforming packets template:

   i: Double-click **Non-conforming packets**; the Non-conforming packets Properties window opens;

   ii: In the Setting tab, select the Enabled radio button;

   iii: Leave the default value (**1**) in the *Priority value* field or change it to match your testing needs;

   iv: Click **OK**.

   b: Best effort service type template:

   i: Double-click **Best effort service type**; the Best effort service type Properties window opens;

   ii: In the Setting tab, select the Enabled radio button;

   iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

   iv: Click **OK**.

   c: Controlled load service type template:

   i: Double-click **Controlled load service type**; the Controlled load service type Properties window opens;

   ii: In the Setting tab, select the Enabled radio button;

   iii: Leave the default value (**4**) in the *Priority value* field or change it to match your testing needs;

   iv: Click **OK**.

   d: Guaranteed service type template:

   i: Double-click **Guaranteed service type**; the Guaranteed service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**5**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

e: Network control service type template:

i: Double-click **Network control service type**; the Network control service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**7**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

f: Qualitative service type:

i: Double-click **Qualitative service type**; the Qualitative service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

Setting GQoS Template Values for Layer 3 (DSCP)

The values of the predefined Generic QoS templates/service types in IxChariot can be set as follows:

DSCP Conforming Packets

1. Enter *gpedit.msc* in the command prompt;

2. In the Group Policy window that opens, click **Computer Configuration>Administrative Templates**;

3. Click **Network>QoS Packet Scheduler**;

4. Click **DSCP-value of conforming packets**;

5. Set the properties of one of the available predefined templates/service types:

a: Best effort service type template:

i: Double-click **Best effort service type**; the Best effort service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *DSCP value* field or change it to match your testing needs;

i: Click **OK**.

b: Controlled load service type template:

    i: Double-click **Controlled load service type**; the Controlled load service type Properties window opens;

    ii: In the Setting tab, select the Enabled radio button;

    iii: Leave the default value (**24**) in the *DSCP value* field or change it to match your testing needs;

    iv: Click **OK**.

c: Guaranteed service type template:

    i: Double-click **Guaranteed service type**; the Guaranteed service type Properties window opens;

    ii: In the Setting tab, select the Enabled radio button;

    iii: Leave the default value (**40**) in the *DSCP value* field or change it to match your testing needs;

    iv: Click **OK**.

d: Network control service type template:

    i: Double-click **Network control service type**; the Network control service type Properties window opens;

    ii: In the Setting tab, select the Enabled radio button;

    iii: Leave the default value (**48**) in the *DSCP value* field or change it to match your testing needs;

    iv: Click **OK**.

e: Qualitative service type:

    i: Double-click **Qualitative service type**; the Qualitative service type Properties window opens;

    ii: In the Setting tab, select the Enabled radio button;

    iii: Leave the default value (**0**) in the *DSCP value* field or change it to match your testing needs;

    iv: Click **OK**.

DSCP Non-Conforming Packets

1. Enter *gpedit.msc* in the command prompt;

2. In the Group Policy window that opens, click **Computer Configuration>Administrative Templates**;

3. Click **Network>QoS Packet Scheduler**;

4. Click **DSCP-value of non-conforming packets**;

5. Set the properties of one of the available predefined templates/service types:

a: Best effort service type template:

    i: Double-click **Best effort service type**; the Best effort service type Properties window opens;

    ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *DSCP value* field or change it to match your testing needs;

i: Click **OK**.

b: Controlled load service type template:

i: Double-click **Controlled load service type**; the Controlled load service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

c: Guaranteed service type template:

i: Double-click **Guaranteed service type**; the Guaranteed service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

d: Network control service type template:

i: Double-click **Network control service type**; the Network control service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

e: Qualitative service type:

i: Double-click **Qualitative service type**; the Qualitative service type Properties window opens;

ii: In the Setting tab, select the Enabled radio button;

iii: Leave the default value (**0**) in the *Priority value* field or change it to match your testing needs;

iv: Click **OK**.

Please see below an example of the output generated after setting the values for a GQoS template both for Layer 2 and for Layer 3 (it is a Wireshark capture). A Priority value of **4** was set for L2 and DSCP value of **18** was set for L3.

```
⊟ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 2525
    100. .... .... .... = Priority: 4
    ...0 .... .... .... = CFI: 0
    .... 1001 1101 1101 = ID: 2525
    Type: IP (0x0800)
⊟ Internet Protocol, Src: 192.168.99.1 (192.168.99.1), Dst: 192.168.99.2 (192.168.99.2)
    Version: 4
    Header length: 20 bytes
 ⊞ Differentiated Services Field: 0x60 (DSCP 0x18; Class Selector 3; ECN: 0x00)
    Total Length: 1356
    Identification: 0x41d2 (16850)
 ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 255
    Protocol: UDP (0x11)
 ⊞ Header checksum: 0x2d1a [correct]
    Source: 192.168.99.1 (192.168.99.1)
    Destination: 192.168.99.2 (192.168.99.2)
⊞ User Datagram Protocol, Src Port: 55000 (55000), Dst Port: 55000 (55000)
```

Layer 2 Priority QoS (Class of Service)

Layer 2 Priority QoS can also be defined as best-effort QoS (Quality of Service) or CoS (Class of Service) at Layer 2 and is implemented in network adapters and switches without involving any reservation setup.

IEEE 802.1p establishes eight levels of priority. The highest priority is seven, which might go to network-critical traffic such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) table updates. Values five and six might be for delay-sensitive applications such as interactive video and voice. Data classes four through one range from controlled-load applications such as streaming multimedia and business-critical traffic - carrying SAP data, for instance - down to "loss eligible" traffic. The zero value is used as a best-effort default, invoked automatically when no other value has been set.

IxChariot supports Layer 2 Priority QoS (Class of Service) on:

• Linux 32-bit

• Linux 64-bit

• Linux on ARM

• Linux on Lexra

• Linux on OpenWRT

Layer 3 Quality of Service

IxChariot provides support for Layer 3 Quality of Service as follows:

• on all operating systems on IPv4 and on Linux and Sun Solaris on IPv6 via IPTOS/DiffServ

• on Windows via Generic QoS

• Microsoft Windows CE 6.0 WMM

IP TOS

The type-of-service (TOS) byte in an IP header specifies traffic precedence and type of service (as defined in RFC 791 and RFC 1349). Figure 9-1 shows the TOS byte in the IP header.

Figure 9-1.   TOS Byte In the IP Header
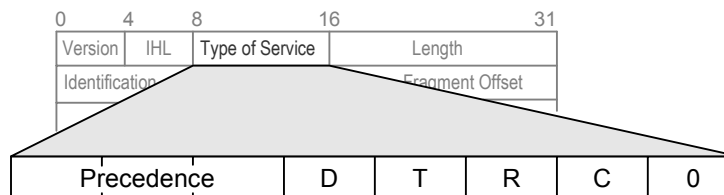
The precedence field comprises the first three bits and supports eight levels of priority. The lowest priority is 0, the highest is 7 (values 6 and 7 are reserved for network control packets). Table 9-1 describes the settings.

Table 9-1.   Precedence Field Settings

Precedence	Bit Setting	Description
Network Control	111	Intended for intranetwork use only. Lowest drop preference.
Internetwork Control	110	Intended for use by gateway control originators.
CRITIC/ECP	101	A priority setting for voice data.
Flash Override	100	Network-specific setting; generally means "maximize throughput."
Flash	011	Network-specific setting.
Immediate	010	Network-specific setting; generally means "maximize reliability."
Priority	001	Network-specific setting; generally means "low maximum delay," for real-time traffic.
Routine	000	Roughly corresponds to Best Effort Service. Highest drop preference.

The four bits following the precedence field specify the type of service. Only one of these bits can be enabled at one time. Each bit defines the desired type of service:

- **D** – The *delay* bit instructs network devices to choose high speed to minimize delay.

- **T** – The *throughput* bit specifies high capacity links to ensure high throughput.

- **R** – The *reliability* bit specifies that reliable links should be used to minimize data loss.

- **C** – The *cost* bit specifies that data transmission should be accomplished at minimal cost.
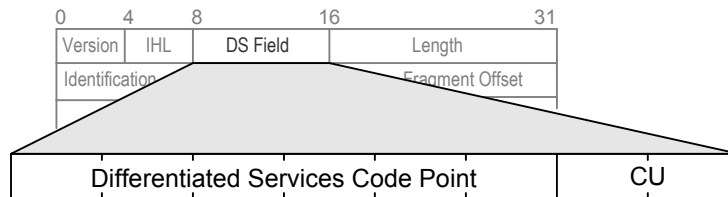
The last bit in the TOS byte is reserved and is always set to 0.

> **NOTE**: On Microsoft Windows Server 2008 (32- and 64-bit) and on Microsoft Windows Vista (32- and 64-bit), IP TOS can be set only via qWave templates.

DiffServ

Differentiated Services (DiffServ) is a QoS model defined by the IETF for IP networks (refer to RFC 2474). This model is designed to be scalable and to provide consistent service classes independent of application. DiffServ redefines the TOS byte of the IP header (see Figure 9-1) as the *DS field*. Figure 9-2 shows the DS field in the IP header.

Figure 9-2.   DS Field in the IP Header

The first six bits of the DS field are used as a differentiated service code point (DSCP), and the last two bits are currently unused (CU).

In the DiffServ QoS model, traffic is classified by marking the DS field with a DSCP value. Queuing mechanisms provide differentiated forwarding of the traffic at each hop, based on the DSCP value.

> **NOTE**: On Microsoft Windows Vista (32- and 64-bit) and Microsoft Windows Server 2008 (32- and 64-bit), DiffServ can be set only via qWave templates.

Per-Hop Behavior (PHB)

The DSCP selects the Per-Hop Behavior (PHB) that a packet experiences at each node. RFC 2474 defines PHB as the externally observable forwarding behavior applied at a DiffServ-compliant node to a DiffServ Behavior Aggregate.

There are currently four standard PHBs:

- Default ("Best Effort") PHB (as defined in RFC 2474)
- Class-Selector PHB (as defined in RFC 2474)
- Assured Forwarding (AF) PHB (as defined in RFC 2597)
- Expedited Forwarding (EF) PHB (as defined in RFC 2598).

Default PHB

RFC 2474 recommends codepoint 000000 as the Default PHB.

Class-Selector PHBs

RFC 2474 defines 21 codepoints, including the Class-Selector PHBs listed in Table 9-2.

Table 9-2.   DSCP Codepoints - Class Selector PHBs

Class Selector Name	DS Field Bit Value
CS0	000 000
CS1	001 000
CS2	010 000
CS3	011 000
CS4	100 000
CS5	101 000
CS6	110 000
CS7	111 000

Assured Forwarding (AF) PHB

The Assured Forwarding (AF) PHB group provides delivery of IP packets in N independently forwarded AF classes. Within each AF class, an IP packet can be assigned one of M different levels of drop precedence. Currently, four classes with three levels of drop precedence in each class are defined for general use. (Additional AF classes or levels of drop precedence may be defined for local use.)

RFC 2597 recommends the codepoints described in Table 9-3 for the four general use AF classes.

Table 9-3.   Recommended Codepoints for AF Classes

	AF Class 1	AF Class 2	AF Class 3	AF Class 4
Low Drop Prec	001010	010010	011010	100010
Medium Drop Prec	001100	010100	011100	100100
High Drop Prec	001110	010110	011110	100110

This table shows each of the AFMN bit values, where M is the AF class and N is the drop precedence. In list form, these are:

AF11 = 001010

AF12 = 001100

AF13 = 001110

AF21 = 010010

AF22 = 010100

AF23 = 010110

AF31 = 011010

AF32 = 011100

AF33 = 011110

AF41 = 100010

AF42 = 100100

AF43 = 100110

Expedited Forwarding (EF) PHB

RFC 2598 defines the EF PHB as a as a PHB that can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DS domains.

RFC 2598 recommends codepoint 101110 as the default EF PHB.

Microsoft qWave

Microsoft Windows Vista and newer Windows versions address QoS requirements for traffic that travels within the home, traffic that travels within an enterprise environment, and traffic that travels between a home and an enterprise over broadband connections.

For home network scenarios, Windows Vista provides a QoS framework referred to as Quality Windows Audio Video Experience (qWave). qWave addresses home audio-video (A/V) streaming scenarios that involve real-time, high-priority traffic that shares a single network with best-effort, low-priority traffic. It supports both Ethernet and wireless (Wi-Fi) home networks.

qWave marks A/V streaming packets with appropriate Layer 2 tags (802.1p or 802.1e) and Layer 3 tags (DSCP).

Refer to the Microsoft QWave web page (http://www.microsoft.com/whdc/device/stream/qWave.mspx) for detailed information.

QoS on Microsoft Windows CE and Microsoft Windows Mobile

IxChariot supports QoS on:

• Windows CE 5.2 and Windows CE 6.0

• Windows Mobile 5.0AKU3 and Windows Mobile 6.

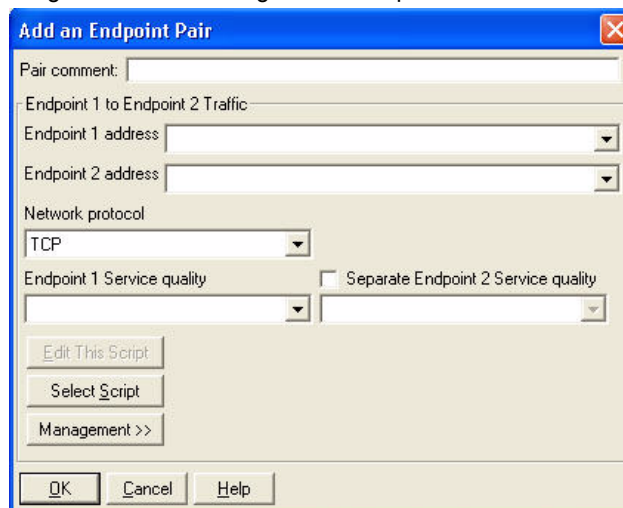# Selecting a QoS Template for Test Application Traffic

This topic describes the procedure for selecting and applying a QoS template for the layer 7 traffic generated by a test. You apply QoS templates to individual pairs in a test. QoS is available for all pair types except for hardware performance pairs and VoIP hardware performance pairs.

To select a QoS template for a pair:

1. Add the pair to the test (or edit an existing pair).

2. Select the desired QoS template for Endpoint 1 and Endpoint 2 from the corresponding Endpoint 1 Service Quality and Endpoint 2 Service Quality drop-down list, as shown in Figure 9-3.

   Refer to *IxChariot QoS Template Descriptions* on page 9-13 for descriptions of the IxChariot QoS templates.

   Figure 9-3.   Selecting a QoS Template

During test execution, IxChariot generates IP packets with the QoS coding that you selected.

IxChariot QoS Template Descriptions

IxChariot provides a set of QoS templates that you can use for your tests. You can use them unchanged, modify them, and make copies of them. Each QoS template is either predefined on the endpoints or is created at the Console and distributed to the endpoints when a test is started.

The following sections describe these templates:

IxChariot QoS Templates - IP TOS and DiffServ

Table 9-4 describes the IP TOS and DiffServ QoS templates that are delivered with IxChariot.

Table 9-4.   IxChariot QoS Templates - IP TOS and DiffServ

Definition	Template Name	QoS Type	Default Bit Pattern
User defined	TrafficTypeBackground	DiffServ	00 10 00 00
Predefined	TrafficTypeBestEffort	DiffServ	00 00 00 00
User defined	TrafficTypeVideo	DiffServ	10 10 00 00
User defined	TrafficTypeVoice	DiffServ	11 10 00 00
User defined	VoIPQoS	IP TOS	101 110 00

IxChariot QoS Templates - GQoS

Table 9-5 lists the Generic QoS templates that are delivered with IxChariot. These GQoS templates are provided with Win32 operating systems and appear in the Service Quality list when you add an endpoint pair to a test:

Table 9-5.   IxChariot QoS Templates - GQoS

GQoS Template Name	Description	Bit Pattern
G711	Pulse code modulation (PCM), a common method of voice encoding. **Note:** Do not use the G711 Generic QoS template for VoIP testing. Instead, use the `VoIPQoS` template that appears in the Service quality list in the Add a VoIP Endpoint Pair dialog box.	10 10 00 00
G723.1	Dual-rate 6.3/5.3-kbps voice encoding scheme.	10 10 00 00

Table 9-5.   IxChariot QoS Templates - GQoS (Continued)

GQoS Template Name	Description	Bit Pattern
G729	Voice encoding scheme that produces high quality at a low data rate.	10 10 00 00
H261CIF	Common video codec used with image sizes of 352 x 288 pixels.	01 10 00 00
H261QCIF	Common video codec used with communications channels that are multiples of 64 kbps and image sizes of 352 x 288 pixels.	01 10 00 00
H263CIF	Common video codec used with image sizes of 176 x 144 pixels.	01 10 00 00
H263QCIF	Common video codec used with communication channels that are multiples of 64 kbps and image sizes of 176 x 144 pixels.	01 10 00 00

All of the GQoS templates are predefined on the Performance Endpoints.

IxChariot QoS Templates - qWave

Table 9-6 lists the qWave QoS templates that are delivered with IxChariot.

Table 9-6.   IxChariot QoS Templates - qWave

qWave Traffic Type	Description	Bit Pattern	Class Selector
BestEffort	This service type requests the same network priority as regular traffic.	00 00 00 00	default
Background	This service type requests a network priority lower than traffic of type QOSTrafficTypeBestEffort. For example, this service could be used for applications performing data backups.	00 10 00 00	CS1
ExcellentEffort	This service type requests a network priority higher than QOSTrafficTypeBestEffort. This service type should be used for data traffic that is more important than standard end-user traffic.	10 10 00 00	CS5
AudioVideo	This service type should be used for audio-video streaming traffic, such as MPEG2 streaming.	10 10 00 00	CS5

Table 9-6.   IxChariot QoS Templates - qWave (Continued)

qWave Traffic Type	Description	Bit Pattern	Class Selector
Voice	This service type should be used for realtime voice streams, such as VoIP.	11 10 00 00	CS7
Control	This service type should only be used for the most critical data. For example, you might use it for data carrying user inputs for audio-video application controls, such as play, pause, and so forth. (The audio-video traffic, however, should use QOSTrafficTypeAudioVideo.)	11 10 00 00	CS7

Refer to Table 9-2 for a list of the standard DiffServ class selectors.

IxChariot QoS Templates - WMM

Table 9-6 lists the Windows CE 6.0 WMM QoS templates that are delivered with IxChariot.

Table 9-7.   IxChariot QoS Templates - Windows CE 6.0 WMM

DSCP Traffic Type	Description	Bit Pattern	Class Selector
DSCPBestEffort	This service type requests the same network priority as regular traffic.	00 00 00 00	default
DSCPBackground	This service type requests a network priority lower than traffic of type DSCPBestEffort. For example, this service could be used for applications performing data backups.	00 10 00 00	CS1
DSCPExcellentEffort	This service type requests a network priority higher than DSCPBestEffort. This service type should be used for data traffic that is more important than standard end-user traffic.	10 10 00 00	CS5
DSCPVideo	This service type should be used for video streaming traffic.	10 10 00 00	CS5

Table 9-7.   IxChariot QoS Templates - Windows CE 6.0 WMM

DSCP Traffic Type	Description	Bit Pattern	Class Selector
DSCPAudio	This service type should be used for realtime voice streams, such as VoIP.	11 10 00 00	CS7
DSCPControl	This service type should only be used for the most critical data.	11 10 00 00	CS7

Refer to Table 9-2 for a list of the standard DiffServ class selectors.

QoS Template File

QoS templates are saved in CSV format in the servqual.dat file on the IxChariot Console computer. (The servqual.dat file is located in the directory where the IxChariot Console is installed.) If you want to run tests from a different IxChariot console using a custom QoS template, take a copy of your servqual.dat file and extract the portions you need.

QoS Template Support on Endpoints

Table 9-8 lists the type of QoS supported for each of the Performance Endpoints. Note that none of the Performance Endpoints support every QoS type.

Table 9-8.   Performance Endpoint QoS Support

Operating System	IP TOS	DiffServ	GQoS	CoS
Apple Macintosh	**Yes**	No	No	No
IxOS Load Module	**Yes**	**Yes**	No	No
Linux (all)	**Yes**	**Yes**	No	**Yes**
UNIX (all)	**Yes**	**Yes**[a]	No	No
Microsoft Windows NT	**Yes**	No	No	No
Microsoft Windows 2000	**Yes**	**Yes**	**Yes**[b]	No
Microsoft Windows XP (32-bit and 64-bit editions)	**Yes**	**Yes**	**Yes**[b]	No
Microsoft Windows Server 2003 (32-bit and 64-bit editions)	**Yes**	**Yes**	**Yes**[b]	No
Microsoft Windows Vista (32-bit and 64-bit editions)[c]	**Yes**[d]	**Yes**[d]	**Yes**[b]	No

Table 9-8.   Performance Endpoint QoS Support

Operating System	IP TOS	DiffServ	GQoS	CoS
Microsoft Windows Server 2008 (32-bit and 64-bit editions)[c]	**Yes**[d]	**Yes**[d]	**Yes**[b]	No
Microsoft Windows 7 (32-bit and 64-bit editions)	**Yes**[d]	**Yes**[d]	**Yes**[b]	No
Microsoft Windows Server 2008 R2	**Yes**[d]	**Yes**[d]	**Yes**[b]	No
Microsoft Windows CE 5.0	No	**Yes**	No	No
Microsoft Windows Embedded CE 6.0	No	**Yes**	No	No

a.integrated into 2.4 kernel
b.Generic QoS template values can be set on Windows via the procedure described in the *IxChariot User Guide*, *Quality of Service Testing* chapter, *Setting GQoS Template Values*.
c.for Windows 2008 and Vista, only qWave and GQoS are supported. As such, you cannot directly set the bits for marking the packets, but must use instead presets provided by Microsoft. The following patterns can be set: 111000, 101000, 001000, 000000.
d.supported via qWave

Testing with Vista and Non-Vista Endpoints

You can run tests between Windows Vista and other operating systems using the same qWave template for both endpoints. The Vista endpoint will interpret the template values as qWave, while the other operating system will interpret the template values as DSCP.

You can also run tests between Windows Vista and other operating systems using the same DSCP template for both ends. However, on the Vista endpoint, the traffic will be marked only if the DSCP value maps to one of the values supported by qWAVE.

Testing with WMM and Non-WMM Endpoints

You can run tests between Windows CE 6.0 (and Windows Mobile 5.0) and other operating systems using the same DSCP template for both endpoints. The Windows CE 6.0 endpoint will interpret the template values as WMM, while the other operating system will interpret the template values as DSCP.

You can also run tests between Windows CE 6.0 and other operating systems using the same DSCP template for both ends. However, on the Windows CE 6.0 endpoint, the traffic will be marked only if the DSCP value maps to one of the values supported by WMM.

Creating and Modifying Custom QoS Templates

IxChariot provides tools that enable you to create custom QoS templates. The procedures for creating templates are presented in the following topics:

• *Creating IP TOS Templates* on page 9-18

• *Creating DiffServ Templates* on page 9-19

• *Creating qWave and WMM Templates* on page 9-19

• *Creating GQoS Templates* on page 9-20

• *Creating Layer 2 Priority Templates* on page 9-22

• *Modifying QoS Templates* on page 9-23

• *Copying QoS Templates* on page 9-23

When you create a custom QoS template, you must assign it a name. Template names are case-sensitive. If there is a matching template found with this name on an endpoint computer, the values configured in your custom template override the values defined on the endpoint.

You can use your custom QoS templates for both the *application traffic* and the *management traffic* generated by an IxChariot test. Note, however, that only TOS and DSCP templates can be used for management traffic (these correspond to IP TOS, DiffServ, and qWave templates). Generic QoS and Layer 2 Priority templates can only be used for test traffic.

Creating IP TOS Templates

To create a new IP TOS QoS template:

1. Select **Tools** > **Edit QoS Templates…**

    IxChariot opens the QoS Template List dialog.

2. Click **Add**.

    IxChariot opens a floating menu listing the QoS template types.

3. Select **IP TOS…** from the floating menu.

    IxChariot opens the Add TOS Template dialog.

4. Define the QoS settings:

    a: Enter a unique name for your new template.

    b: Select the desired IP precedence bits from the drop-down list.

    c: Select the checkbox corresponding to the IP TOS bit that you wish to enable.

    As an alternative to steps b and c, you can directly set the six IP TOS QoS bits by clicking the bit representations shown in Bit Field Settings. This toggles each bit between 0 and 1.

5. Click **OK** to save your new template.

The new template is immediately listed in the QoS Template List.

Creating DiffServ Templates

To create a new DiffServ QoS template:

1. Select **Tools** > **Edit QoS Templates…**

   IxChariot opens the QoS Template List dialog.

2. Click **Add**.

   IxChariot opens a floating menu listing the QoS template types.

3. Select **DiffServ…** from the floating menu.

   IxChariot opens the Add DiffServ Template dialog.

4. Define the QoS settings:

   a: Enter a unique name for your new template.

   b: Either select a predefined DiffServ codepoint from the drop-down list, or directly set the six QoS bits by clicking the bit representations shown in the Bit Field Settings. When you directly set the bits, IxChariot selects the User-defined DiffServ Codepoints radio button.

5. Click **OK** to save your new template.

The new template is immediately listed in the QoS Template List.

Creating qWave and WMM Templates

To create a new Windows Vista qWave QoS template or a new Windows CE 6.0 WMM QoS template:

1. Select **Tools** > **Edit QoS Templates…**

   IxChariot opens the QoS Template List dialog.

2. Click **Add**.

   IxChariot opens a floating menu listing the QoS template types.

3. Select **DiffServ…** from the floating menu.

   IxChariot opens the Add DiffServ Template dialog (which contains the option for Windows Vista and Windows CE 6.0 traffic types).

4. Define the QoS settings:

   a: Enter a unique name for your new template.

   b: Either select a predefined Windows Vista or WinCE 6.0 traffic type from the drop-down list, or directly set the six QoS bits by clicking the bit representations shown in the Bit Field Settings.

   The custom bit settings must match one of the predefined traffic types.

5. Click **OK** to save your new template.

The new template is immediately listed in the QoS Template List.

Creating GQoS
Templates

To create a new GQoS template:

1. Select **Tools** > **Edit QoS Templates…**

IxChariot opens the QoS Template List dialog.

2. Click **Add**.

IxChariot opens a floating menu listing the QoS template types.

3. Select **Generic QoS…** from the floating menu.

IxChariot opens the Add QoS Template dialog.

4. Define the QoS settings by entering or selecting values for all of the GQoS properties, as described in Table 9-9 on page 9-20.

5. Click **OK** to save your new template.

The new template is immediately listed in the QoS Template List.

Table 9-9.   GQoS Template Properties

Property	Description
Template Name	A unique name for the template.
Service Type	The appropriate level of service for the data flow. Refer to Table 9-10 for a description of the available service types.
No Traffic Control	Disables traffic control, in which case QoS flows will not be used.
Token Rate (bytes/sec)	Defines the burst rate. If packets are sent out uniformly at the token rate, the bucket remains empty. Each outgoing packet is matched by one token. If a packet is sent without a matching token, the packet may be dropped. If the transmission rate is less than the token rate, the unused tokens accumulate up to the token bucket size. The Token Rate field is expressed in bytes/second. A value of No Rate, to indicate that no rate-limiting is enforced, is obsolete and isn't allowed on newer operating systems that allow GQoS. If you've loaded a template from a version of IxChariot older than 4.3, No Rate is still supported, but you'll be required to enter a rate and a Token Bucket Size for any new templates you create.
	To transmit without losing packets, use the following settings:
	• Set the **Token Rate** field at or above the average transmission rate. 1000 bytes/sec is an average rate.
	• Set the **Token Bucket Size** field to be large enough to accommodate the largest expected burst of data. 1000 bytes is an average size.
	If data is sent at a low rate for a period of time, the sending computer can send a large burst of data all at once until it runs out of tokens. Afterwards, data may only be sent at the token rate until its data burst is exhausted.

Table 9-9. GQoS Template Properties (Continued)

Property	Description
Token Bucket Size (bytes)	Controls the size of data bursts in bytes, but not the transmission burst rate. If packets are sent too rapidly, they may block other applications' access to the network for the duration of the burst. This value designates the largest typical frame size in your application, expressed in bytes. In constant rate applications, the Token Bucket Size is chosen to accommodate small variations. In video applications, the token rate is typically the average bit rate peak-to-peak. In constant-rate applications, the Token Rate should equal the Maximum Transmission Rate.
Latency (microseconds)	The maximum acceptable delay in microseconds between transmission of a packet by the sender and its receipt by the intended receiver or receivers. Precise interpretation of this number depends on the service type specified in the QoS request.
Delay Variation (microseconds)	The difference, in microseconds, between the maximum and minimum possible delay that a packet experiences. Determines the amount of buffer space needed at the receiving side to restore the original data transmission pattern.
Maximum Transmission Rate (bytes/sec)	The rate at which packets may be sent back-to-back, expressed in bytes/second. Lets certain intermediate routers allocate their resources efficiently. In this field, enter the maximum packet size in bytes that is used in the traffic flow.
Maximum Transmission Size (bytes)	The largest packet size, in bytes, permitted in your network. QoS-enabled routers use this MTU size in policing multicast network traffic.
Minimum Policed Size (bytes)	The minimum packet size in bytes that will be given the level of service requested.

Table 9-10 describes the GQoS service types:

Table 9-10. GQoS Service Types

Service Type	Service Type Description
No Traffic	In either the sending or receiving QoS specification, this value indicates that there will be no traffic in this direction. On duplex-capable media, this signals underlying software to set up unidirectional connections only.
Best Effort	The service provider takes the QoS specification as a guideline and makes reasonable efforts to maintain the level of service requested, without making any guarantees on packet delivery. The network routers do not guarantee prioritization of the data.

Table 9-10.   GQoS Service Types (Continued)

Service Type	Service Type Description
Controlled Load	The network router gives priority to the data, and it operates as if the data is the only data on the network at the time. Thus, this service may assume:
	A high percentage of transmitted packets will be successfully delivered by the network to the receiving end nodes. (Packet loss rate should closely approximate the basic packet error rate of the transmission medium.)
	Transit delay experienced by a high percentage of the delivered packets will not greatly exceed the minimum transit delay experienced by any successfully delivered packet.
Guaranteed	This service type value is designed for applications that require a precisely known quality of service but would not benefit from better service, such as real-time control systems. The service provider implements a queuing algorithm that isolates the flow from the effects of other flows as much as possible and guarantees the application the ability to propagate data at the token rate for the duration of the connection. If the sender sends faster than the token rate, the network may delay or discard the excess traffic. If the sender does not exceed the token rate over time, latency is also guaranteed.
	NOTE: To set the values for these templates, see *Setting GQoS Template Values* on page 9-3.
General Information	All service types are supported for this traffic flow.
No Change	In either sending or receiving QoS specifications, this level of service requests that the quality of service in the corresponding direction is not changed. Select **No Change** when requesting a QoS change in one direction only, or when requesting a change only in the Provider-Specific part of a QoS specification and not in the Sending or Receiving Flowspec.
Qualitative	For this service type, the network determines how the data flow is treated. Generally offers medium quality, non-quantifiable guarantees.

Creating Layer 2 Priority Templates

To create a new Layer 2 Priority template:

1. Select **Tools** > **Edit QoS Templates…**

    IxChariot opens the QoS Template List dialog.

2. Click **Add**.

    IxChariot opens a floating menu listing the QoS template types.

3. Select **Layer 2 Priority (Linux only)…** from the floating menu.

    IxChariot opens the Add Layer 2 Priority Template dialog.

4. Enter a name for the template in the *Template name* field.

5. Define the QoS settings by selecting a priority value from the Priority drop-down list (available values range from 0 to 7).

6. Click **OK** to save your new template.

The new template is immediately listed in the QoS Template List.

Modifying QoS Templates

To modify a QoS template:

1. Select **Tools** > **Edit QoS Templates…**

   IxChariot opens the QoS Template List dialog.

2. Select the template that you want to modify.

3. Click **Modify**.

   IxChariot opens the Modify Template dialog that is specific to the type of template you are modifying.

4. Modify the QoS settings.

5. Click **OK** to save your modified template.

Copying QoS Templates

To copy an QoS template:

1. Select **Tools** > **Edit QoS Templates…**

   IxChariot opens the QoS Template List dialog.

2. Select the template that you want to copy.

3. Click **Copy**.

   IxChariot opens the Copy Template dialog that is specific to the type of template you are copying.

4. Enter a name for the new template in the Template name field.

5. Click **OK.**

IxChariot copies the template.

Configuring Endpoints for Testing with a Service Quality

This section describes configuration changes that you may need to make on end-point computers to support the various QoS templates.

UNIX and Linux

All the UNIX and Linux endpoints support testing with IP TOS bit settings and DiffServ as the Service Quality.

GQoS Client Configuration Requirements

Endpoint computers that need to support GQoS must be configured as follows:

•   The client NIC driver must support 802.1p.

•   802.1p must be enabled on the NIC driver.

•   The `DisableUserTOSSetting` registry key must be is added, as described in *Windows 2000/2003, and XP Registry Configuration* on page 9-24.

•   The QoS Packet Scheduler service must be installed, as described in *Windows QoS Packet Scheduler* on page 9-24.

•   The group policy snap-in must be used to enable and map the settings for 802.1p and the DSCPs.

Windows QoS Packet Scheduler

Windows 2000 and Windows Server 2003 have TCP/IP QoS support, but it may not be installed as part of the basic Windows installation. Windows 2000/2003 provides a component called the QoS Packet Scheduler. If this component is installed, QoS tests can take advantage of Microsoft's GQoS support for Diff-Serv marking. To install the QoS Packet Scheduler, do the following:

1. Right-click **My Network Places** and select **Properties**.

2. Right-click **Local Area Connection** and select **Properties**.

3. Click **Install**.

4. Select **Service** and click **Add**.

5. A list appears. Highlight the list item **QoS Packet Scheduler** and click **OK**. The QoS Packet Scheduler will now appear in the list of installed components.

6. Restart your computer to ensure that the QoS Packet Scheduler is properly initialized.

Windows 2000/ 2003, and XP Registry Configuration

Windows 2000/2003 and Windows XP allow for testing with IxChariot's TOS and DiffServ templates for TCP, UDP, and RTP after you make an addition to the Registry. Add the following DWORD value at the endpoints:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcp
ip\Parameters\DisableUserTOSSetting
```

In this case, set the REG_DWORD value to **0**.

Be sure to restart the endpoint computers after you edit their Registry settings. Then run the test.

Windows Vista Registry Configuration

For Windows Vista, DiffServ QoS DSCP marking is enabled by modifying or creating the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\RTC\Transport\
QoSEnabled
```

In this case, set the REG_DWORD value to **1**.

Be sure to restart the endpoint computers after you edit their Registry settings. Then run the test.

Configuring Routers for Testing with a Service Quality

To use GQoS RSVP support, you must make sure that the RSVP protocol is enabled on the router interface. For information on how to enable the RSVP protocol, refer to the documentation for your router.

We have tested QoS with Cisco routers running IOS version 11.3.5. If you are testing QoS with Cisco routers, we recommend that you use this version or later of the IOS software. If you do not have version 11.3.5, the following bug fixes are necessary for QoS testing:

- CSCdk28283: non-SBM aware routers running only RSVP should not process SBM messages.

- CSCdk27983: RSVP should not drop messages with `10xxxxxx` class objects.

- CSCdk27475: RSVP should not drop policy object in `RESV` message.

- CSCdk29610: RSVP should not reject `RTEAR` message without flowspec.

- CSCdk38002: Router crashes while forwarding `RSVP Path-Err` message.

- CSCdk38005: Router does not forward `RESV` messages with Guaranteed service flowspec. This fixes the main problem we encountered, which prevented tests with Guaranteed service types from working.

We have found in our testing that if a router in the path between two endpoints rejects the QoS request after the test has started running, the endpoint does not recognize this, and the test will continue to run. The error messages are returned asynchronously from the router and the traffic will be treated as Best Effort. This situation can occur if a router in the path does not have the necessary bandwidth to fulfill the request.

10 *Concepts*

Related Topics

The following topics provide in-depth, conceptual support for specialized testing situations. IxChariot is an extremely powerful, flexible tool, with a large array of configuration options. To get the most accurate and reliable results, and to understand the circumstances that may affect test results, read the following topics in advance, and follow the advice provided.

- *Fixed-Duration Testing* on page 10-2 describes the requirements and procedures for configuring tests that run for a fixed duration, versus tests that are configured to generated a specific number of timing records.

- *UDP Throughput Testing* on page 10-4 explains how to configure a throughput test using one of the available UDP protocol implementations in IxChariot.

- *IP Multicast Testing* on page 10-9 explains how to set up and run an IP Multicast test.

- *Firewall Testing* on page 10-13 explains how to configure IxChariot for testing with a firewall between the Console and Endpoint 1, or between the endpoint computers.

- *Voice over IP Testing* on page 10-34 contains theoretical knowledge and practical advice for users of IxChariot's VoIP Test Module.

- *Video Stream Testing* on page 10-54 describes all of the parameters available for setting up tests for Video Pairs and Multcast Video Groups, and describes the Media Delivery Index results produced by the tests.

• *IPTV Testing* on page 10-66 describes the underlying concepts and the required procedures for setting up IPTV channel switching tests.

• *Tips for Testing* on page 10-83 gives general advice for doing accurate stress and performance testing, for reducing the number of timing records generated by a test, and for testing with the TCP `close_type` parameter.

Fixed-Duration Testing

When you configure a test, you specify when and how the test will end. Your choices are:

• The test will stop when any one pair completes its execution.

• The test will stop only when every pair has completed its execution.

• The test will run for a fixed duration.

When you configure a test to run for a fixed duration, every endpoint pair in the test runs for the amount of time that you specify. In this case, the scripts ignore the number_of_timing_records value in their outer loop. Instead, IxChariot runs as many transactions during that time as it can. At the end of the test period, the endpoints stop and each Endpoint 1 returns its results to the IxChariot Console.

When a test is set to run for a fixed duration, the run time duration is checked every time an END_LOOP or END_TIMER command executes. This may cause the actual run time to slightly exceed the run time that you specified.

How Long Can a Test Run?

You can set a test to run for any period of time from 1 second to 99 hours, 59 minutes, and 59 seconds. Although the default setting is 1 minute, Ixia recommends 2 to 5 minutes for most performance testing. It is important to test for a sufficient length of time to generate at least 10 timing records. The records represent data samples, and you will need a sufficient number of samples to ensure test reliability.

Impact of Test Duration on Timing Record Generation

Because application scripts generate a large number of timing records per minute on a LAN, setting the duration to long time periods can generate an enormous number of timing records, potentially exceeding the storage capacity of some console computers.

For example, if you run a test that generates 100 timing records in 20 seconds, and you then run the same test with the fixed duration set to one hour, IxChariot generates approximately 18,000 timing records. When you run a test for a fixed duration, Ixia strongly recommends increasing the transactions_per_record values in the scripts such that you will decrease the test's total number of timing records to less than 10,000.

Reducing Timing Records in a Nonstreaming Script

To reduce the number of timing records for a nonstreaming script:

1. Open the script in the Script Editor.

2. Edit the transactions_per_record variable to increase the Current Value by a factor of 10.

   For example, if transactions_per_record has a Current Value of 50, change the number to 500.

3. Select **Save to Pair** from the Script Editor **File** menu, then exit the Script Editor.

4. Select S**et Run Options…** from the **Run** menu.

5. Select the **Run for a fixed duration** radio button, enter a value of 1 minute, then click **OK**.

6. Run the test and view the results in the test window.

   Select the **Raw Data Totals** tab. The Timing Records Completed column shows the total number of timing records generated in one minute. For a good statistical sample, you will want to see 50 to 100 records per pair.

7. If the results contain more than a total of 10,000 timing records, further increase the transactions_per_record value, then repeat step 6.

8. Increase the transactions_per_record to match the duration of your test, using this formula:

   (new transactions_per_record value) =
   (minutes to run) x (transactions_per_record value in the 1 minute test)

For example, if you set the transactions_per_record value to 500 for a 1 minute test, you would calculate the transactions_per_record value for a 48-hour test as follows:

   transactions_per_record = 1,440,000

   (60 minutes per hour) x (48 hours) x (500 transactions per record)

Reducing Timing Records in a Streaming Script

Streaming scripts do not use the transactions_per_record variable. Rather, a timing record is generated at the end of each SEND command. To reduce the number of timing records, increase the file_size variable on the SEND command. The file_size variable controls the amount of data per transaction.

Use the following formula to determine the value to use for the file_size variable:

   file_size =
   (duration of timing record)
   / (number of timing records * (send_data_rate * 8))

If you specify a send_data_rate other than UNLIMITED, you can use the following equations to estimate the number of timing records you will receive:

   duration of timing record = (file_size * 8) / send_data_rate

For example, if you use the Realaud.scr script with the script default parameters, you can expect each timing record to take about 1.39 seconds:

1.39 = (14,040 bytes * 8) / 80,736 bps.

For More Information

Refer to Chapter 8, *Large-Scale Tests in IxChariot* for more information about timing record maximums and memory consumption.

UDP Throughput Testing

To ensure that you gain maximum value from your UDP throughput tests, you need to consider a number of factors in your test definition, including:

• The specific UDP protocol implementation that you plan to use.

• The script variables that set the send and receive buffer sizes.

• The Datagram Run Options that control window size, retransmissions, and timeouts.

Choosing the UDP Protocol

IxChariot uses three implementations of the UDP protocol, as described in *UDP Configuration* on page 5-6. Table 10-1 summarizes the available choices for UDP protocol.

Table 10-1. UDP Protocol Options Per Type of Script

	Streaming Script	Non-Streaming Script
IxChariot Reliable UDP	No	Yes
IxChariot UDP for Streaming Scripts	Yes	No
RFC 768 UDP	Yes	No

When you are using a streaming script, you can choose the IxChariot-proprietary or the industry-standard UDP implementation. This choice impacts the type of statistics that you can collect, as well as the point at which they are presented in the Test window. Refer to *Comparison of UDP Statistics Measures* on page 5-7 for more information.

Setting Datagram Run Options

To configure datagram run options for a test:

1. Select **Run** > **Set Run Options…**

2. Select the **Datagram** tab in the Run Options window.

3. Configure the values as described in:

   • *Setting the UDP Window Size* on page 10-4

   • *Setting Retransmission Values* on page 10-5

   • *Setting the Streaming Options for a Test* on page 10-7

Setting the UDP Window Size

You set the UDP window size on the Datagram tab in the Run Options window. The UDP window size is used only with non-streaming scripts. It is the number

of bytes that can be sent from an endpoint to its partner without an acknowledgment.

This parameter applies to the protocol stack; it is meant to avoid flooding the network or an endpoint with datagrams. Sending more datagrams than the network can handle results in lost datagrams, and lost datagrams can cause timeouts and retransmissions, which in turn mean poor performance. Sending too few datagrams means data is much less likely to be lost, but you may experience poor performance.

Setting this value too low hurts performance by forcing the sender to wait for acknowledgments and by increasing the number of ACK packets flowing through the network. For example, if you set the window size to 4096, the sender seeks an acknowledgment from the receiver after each block of 4,096 bytes it has sent. You can set the Window Size to be larger, resulting in fewer pauses for acknowledgments, but in the event of a connection failure, more bytes will have to be resent.

If the Window Size is too large, more datagrams may need to be retransmitted. In addition, memory usage is a concern. When a test begins, both endpoints in each connection using a datagram protocol allocate enough memory to hold a complete window.

The UDP window size has a direct bearing on throughput, given that the smaller the window size, the greater the number of acknowledgements that must be received and processed. In many cases, the default UDP window size of 1500 bytes is too small and, therefore, adds considerable overhead to data transmissions. If the application being emulated lets you set datagram parameters, use the same values for the Window Size that it is using. Otherwise, choose a Window Size large enough for several datagrams (or more/less, depending on the expected load on the network and the individual hosts) to be sent before an acknowledgment is required. One approach to setting an optimal UDP window size is to first run a test using the default UDP window size, and note the throughput values. Then set the UDP window size to the highest maximum value (9,999,999 bytes), and re-run the test. The throughput should increase for the second test.

Setting Retransmission Values

You set the datagram retransmission values on the Datagram tab in the Run Options window. The datagram retransmission values are used only with non-streaming scripts. The values are:

• *Retransmission Timeout Period*

• *Number of Retransmits before Aborting*

It is important to note that standard (RFC 768) UDP is an unreliable protocol and, therefore, does not use acknowledgements and retransmissions. You use the two datagram retransmission parameters to manage the graceful termination of tests for which acknowledgements are not received or for which an excessive number of retransmission occur.

Retransmission Timeout Period

Retransmission Timeout Period is the number of milliseconds the sender will wait, after sending for the first time or retransmitting a block of data to the receiver, to receive an acknowledgment that the block has been received.

Datagrams can be delayed rather than lost. If the timeout parameter is set too low, an endpoint can behave as if a datagram has been lost when acknowledgments aren't received quickly enough. For example, if you set the Retransmission Timeout Period to 1000 ms and an acknowledgment isn't received, the sender will resend its original block and wait 1 second to hear from the receiver. Enter values from 1 to 99999 ms (about 100 seconds) for the Retransmission Timeout Period.

If you set this value too low for the network you are using, the sender will re-send blocks of data that the receiver has actually received. This unnecessary traffic can create bottlenecks. Conversely, if the Retransmission Timeout is set too high, the endpoint will wait too long before retransmitting. This can significantly degrade performance in networks where packets are lost frequently.

The timeout period should be at least as long as the average round-trip time between the endpoints, minus the overhead for processing the data at the endpoints. For example, if the endpoints are connected via satellite, use 2 * 270 ms plus some overhead, depending on the speed of the endpoint computers, to represent the average round-trip propagation delay for satellite transmissions. A value of 10 or 20 ms may be appropriate for endpoints directly connected on a LAN, such as an Ethernet.

We recommend setting the Retransmission Timeout to twice the propagation delay of your network. This is equivalent to the response time measured by the Inquiry, Long Connection (`Inquiryl`) script.

In general, the more traffic on the network, the higher the timeout period. A low timeout period can cause many endpoints to retransmit simultaneously, increasing network congestion.

Number of Retransmits before Aborting

Number of Retransmits before Aborting is the number of times the sender will re-send a block of data for which an acknowledgment is not received. The default number of retransmissions is 50. If this number is too high, the sender may generate needless traffic when there's been a real failure. However, if it is too low, the sender may declare a connection failure and end the test during momentary congestion, when the connection is otherwise okay. Enter values from 1 to 999 for the Number of Retransmits before Aborting.

To calculate how long the endpoint waits before ending a test, multiply the Number of Retransmits before Aborting by the Retransmission Timeout Period. On an extremely congested network, an application may not be able to complete even one transaction because datagrams are lost in the network. In this case, it is best

for the application to abort as an indication that the network is too congested and needs to be repaired.

Setting the Streaming Options for a Test

You set the datagram streaming option on the Datagram tab in the Run Options window. The streaming options are used only with streaming scripts and they apply to all pairs in a test. The options are:

• Receive Timeout

• Multicast Time To Live.

Set these parameters to resemble the multicast application you are emulating as closely as possible (see *Datagram Run Options* on page 7-18). If you are unsure what those values should be, here are some general guidelines to follow:

• **Receive Timeout**

The number of milliseconds the endpoint issuing a `RECEIVE` command waits before determining that a script has ended. If the data has not been received in this amount of time, endpoints notify the sender that the data was not received.

Receive Timeout is used for both multimedia pairs and multicast groups. If you set this number too low and the transmission encounters normal network delays, the receiver may time out while the data is still transmitting. If you set it too high, the receiver may spend unnecessary time waiting for a transmission that has failed.

If Endpoint 2 is running on Windows, the minimum receive timeout is 500 ms.

• **IP Multicast Time To Live (TTL)**

Controls the forwarding of IP Multicast packets. (See *IP Multicast Testing* on page 10-9 for more information.) Set this value based on how far you want the data forwarded. Expect to do some experimentation to find the best combination of variable settings.

This field defaults to 1. A value of 1 means that the packet does not leave the sender's local subnet. If you want to route the packet across a router, you must set the value of this field to at least 2. A general rule is to set the TTL to one more than the number of router hops to the farthest endpoint in the multicast group.

With Microsoft's TCP/IP stacks on Windows, a TTL value of 0 still lets multicast packets leave the local host. Thus, if you run a loopback test with one computer, you may impact the performance of your network because the packets are broadcast on the local subnet.

See *Datagram Run Options* on page 7-18 for information on the Receive Timeout and IP Multicast TTL values.

Setting Buffer Sizes

The send and receive buffers determine the size of the datagram that is sent over the network. In general, you should always set these to the same size. In an IxChariot script, the `send_buffer_size` variable is configured for the SEND command and the `receive_buffer_size` variable is configured for the RECEIVE command.

The default value for both of these variables is DEFAULT. When the buffers are set to a value of DEFAULT, the endpoints use buffers that are the default size for the network protocol and platform being used.

Refer to the *IxChariot Script Development and Editing Guide* for:

• a list of the default buffer sizes for each platform and protocol.

• instructions for emulating applications that use variable buffer sizes.

Fragmentation and Reassembly

The network packet size, sometimes called the frame size or maximum transmission unit (MTU), is the maximum amount of data a network adapter card can transmit at one time. When a datagram exceeds the packet size, the protocol stack breaks the datagram into pieces. This is called *datagram fragmentation*. The process of *datagram reassembly* takes place at the destination computer.

Because IPX doesn't support datagram fragmentation and reassembly, each datagram can be no larger than the largest datagram supported over the entire route between endpoints. Therefore, if IPX computers on one LAN segment support 1467-byte datagrams, but only 512-byte datagrams are supported on the next hop, any datagram over 512 bytes is dropped.

TCP/IP performs fragmentation and reassembly for higher-layer protocols such as UDP and TCP. If the TCP/IP stack can reassemble datagram fragments faster than the application software (or the endpoints) can issue API Send calls, your tests can run faster if you set the `send_buffer_size` to be as large as possible.

On the other hand, a large number of datagram fragments may increase the congestion in a network and, therefore, the likelihood that a fragment may be dropped. When that occurs, the entire datagram must be retransmitted, causing poorer performance.

To avoid fragmentation, the datagram size must be smaller than the packet size. Performance Endpoint datagram support uses a 9-byte header. In addition, UDP has an 8-byte header; IP has a 20-byte header; and IPX has a 30-byte header. So, for example, if you are using UDP, the datagram size is 9+8+20+`send_buffer_size`; for IPX, it is 9+30+ `send_buffer_size`.

A network's packet size is limited by its physical type: Ethernet's packet size is around 1500 bytes, and Token Ring's is about 4096 bytes. And the packet size also depends on the network software and the protocols used. For example, some versions of IPX don't support more than 512 bytes, even on Ethernet.

Some platforms don't reassemble fragmented datagrams correctly. In these cases, setting the `send_buffer_size` to a value less than the `frame_size` usually resolves the problem.

When you are running tests of 200 or more pairs and using streaming scripts, you may see some unusual spikes in the throughput results near the end of a test run. See *Throughput Spikes in Streaming Tests* on page 12-15 for more information.

IP Multicast Testing

Related Topics

Adding or Editing a Multicast Group on page 5-27
Emulating IP Multicast Applications on page 10-9
Setting Up Your Hardware and Software for IP Multicast on page 10-12
Setting the Streaming Options for a Test on page 10-7

IxChariot supports the testing of IPv4 and IPv6 Multicast delivery. IP Multicast uses RTP or UDP to deliver data from one sender to multiple receivers. Any computer designated as Endpoint 1 can send data to a group of multiple Endpoint 2 computers with a single UDP or RTP data stream. The sender does not guarantee delivery of the data to the receivers.

IP Multicast receivers must subscribe to the multicast group prior to receiving data. The multicast group is identified with an IP Multicast address and port. The IP Multicast address specifies the multicast group to which data should be delivered. This Class D IP address falls within a specified range. The IP Multicast port identifies one of the possible destinations within a given host computer. IxChariot uses the combination of the multicast address and multicast port to uniquely identify a multicast group.

To emulate an IP Multicast application, select one of the streaming scripts. You can set some of the test parameters, such as how far an IP Multicast packet is forwarded before it is dropped. See *Setting the Streaming Options for a Test* on page 10-7 for more information.

Emulating IP Multicast Applications

Related Topics

Adding or Editing a Multicast Group on page 5-27
IP Multicast Testing on page 10-9
Setting Up Your Hardware and Software for IP Multicast on page 10-12

Endpoints use the IP Multicast address and port to send data to all members in a multicast group. In tests containing multicast groups, Endpoint 1 acts as the sender, and the computers in the role of Endpoint 2 act as the receivers. You can set the TTL used for the IP Multicast packets. You can also change the default timeout used by the Endpoint 2 computers. This value emulates buffering of multicast data by receivers; after a certain length of time, the receivers have to recognize that the sender is no longer sending.

To emulate an IP Multicast application, first set up a multicast group at the IxChariot Console. Click **Add Multicast Group** on the Edit menu in the Test window. Within a test, you can configure multiple groups to emulate different applications, sending data to multiple sets of addresses. The IP Multicast address and port combination must be unique for each multicast group in a test. See *Adding or Editing a Multicast Group* on page 5-27 for more information.

Multicast Addresses

For IPv4, the valid range of IP Multicast addresses is 224.0.0.0 through 239.255.255.255 (the *class D IP addresses*). Some Class D addresses are reserved and should be avoided. For example, addresses between 224.0.0.0 and 224.0.0.255 are reserved for routing protocols and other low-level topology discovery or maintenance protocols. IxChariot lets you specify any IP Multicast address. However, when testing with a reserved IP Multicast address, be aware that other applications, hosts, or routers may be transmitting data to this address. Unexpected test results may occur. We recommend addresses beginning with 225.0.0.0 or higher because many addresses beginning with 224 are reserved for router usage.

For IPv6, multicast addresses have a prefix of `FF00::/8`. Table 10-2 lists the multicast addresses (from the reserved range of `FF00::` to `FF0F::`) that are reserved for specific functions:

Table 10-2.   IPv6 Multicast Address Usage

Range	Usage
FF01::1	All nodes within the node-local scope.
FF02::1	All nodes on the local link.
FF01::2	All routers within the node-local scope.
FF02::2	All routers on the link-local scope.
FF05::2	All routers in the site.
FF02::1:FFXX:XXXX	Solicited-node multicast address, where XX:XXXX represents the last 24 bits of the IPv6 address of the node.

Refer to *Endpoint Support for IPv6* on page 5-10 for a list of endpoints that support IPv6.

Although IxChariot verifies that the IP Multicast address you enter is within the valid range, it does not verify the reserved/unreserved status of the address you enter.

Because UDP and RTP are unreliable protocols, some data may be lost. At the end of a test, the Endpoint 2 computers report the number of bytes and datagrams lost during the transmission. Lost data does not cause a pair to fail during a test run, however.

Here's a simple example of an IP Multicast test, with one multicast group consisting of three computers (each labeled Endpoint 2, below).

Figure 10-1.   Simple IP Multicast Test
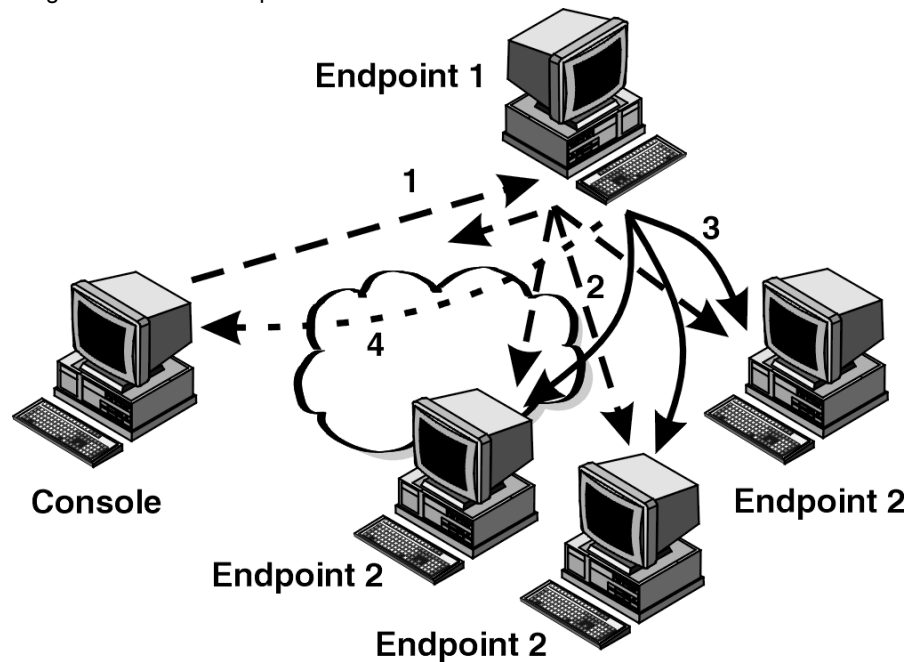
The data flows in the above picture are numbered and described below.

1. You create a test at the IxChariot Console and click **Run** on the Run menu. The Console sends the setup information to Endpoint 1, using a TCP connection. The setup data includes the following:

   • The application script (a streaming script)

   • The specific IP address of each Endpoint 2 in the multicast group

   • The protocol to use when connecting to E2 (UDP or RTP)

   • The Quality of Service (QoS) template to use (if any)

   • The multicast group address (a Class D IP address)

   • The multicast port number that each E2 should use while the test is running

   • How long to run the test

   • How to report results.

2. E1 keeps its half of the application script and forwards the other half to each E2 in the multicast group, using a TCP connection. It also sends the multicast group address and port number at which they should receive data while the test is running. The E2 computers subscribe to (or join) the multicast group. When all E2 computers have acknowledged they are ready, E1 replies to the Console on its TCP connection. The Console directs all pairs to start.

3. E1 executes its multimedia script as the UDP or RTP sender, and the E2 computers receive the data. E2 computers collect timing records and information on lost data.

4. E1 returns timing records and results to the Console.

For an IP Multicast test, you should use the application script that most closely matches the application you want to emulate. You must use a streaming script for IP Multicast testing. See the *Application Scripts* guide for more information on the streaming scripts. If the streaming scripts provided do not meet your needs, you can create a new streaming script.

Avoid using a single computer as E2 more than once in a single multicast group. This includes multiple IP addresses or multiple host names that actually represent the same computer. The endpoint attempts to bind to the same port number, which will result in a communications error during the test setup.

Due to the nature of IP Multicast, data packets from a test containing multicast groups could disrupt other applications running on the network. The test may time out in some cases or stop before completion in other cases. Try increasing the Datagram Receive Timeout value in the Run Options notebook; see *Setting the Streaming Options for a Test* on page 10-7 for more information.

If you want the endpoints to verify that the data received matches the data sent, click the **Validate data upon receipt** checkbox on the **Run Options** tab of the Run Options dialog box. See *Miscellaneous Run Options* on page 7-11 for more information.

Setting Up Your Hardware and Software for IP Multicast

Before you run a test containing multicast groups, you must do the following to prepare your hardware and software for IP Multicast:

• **Verify That Your TCP/IP Protocol Stack Supports IP Multicast**

All of the active Performance Endpoints (and many of the archived endpoints) provide IP Multicast support on platforms with the necessary protocol stack support. To run IP Multicast, install the latest Performance Endpoints on any of the supported operating systems (please refer to the *IxChariot Performance Endpoints* manual for a list of the supported operating systems).

Refer to *Endpoint Support for IPv6* on page 5-10 for a list of the endpoints that support IPv6 (as well as IPv4).

• **Configure Routers to Enable IP Multicast Support**

Many of today's routers have IP Multicast support built into their operating system. However, this support is not automatically enabled. To run tests containing multicast groups across a router, you must first configure the IP Multicast support, which enables IP Multicast data to be forwarded by your router.

For information on how to enable and configure your router's IP Multicast support, refer to its documentation. The router should be updated with the latest `ROM`, `EEPROM`, `BIOS`, or microcode revision level.

• **Verify that Routers Have Enough RAM**

Providing support for IP Multicast routing increases the amount of RAM required by a router. Routers maintain additional routing tables (discussed below) to decide how to forward IP Multicast packets. See the documentation for your router to determine the amount of RAM required for IP Multicast

applications. If necessary, add additional RAM before running tests containing multicast groups.

- **Choose the Right Routing Table Algorithms**

  There are three families of IP Multicast routing algorithms: `DVMRP`, `M-OSPF`, and `PIM`. Your network administrator must decide which routing algorithms to implement for the routers in your network so that your routers can communicate IP Multicast routing information with each other. See the table below for more information:

Table 10-3. Routing Table Algorithms

Routing Table Algorithm	Description
Distance-Vector Multicast Routing Protocol (DVMRP)	Forwards packets based on the source of the subnetwork's location. This is the only algorithm that has its own "unicast" routing protocol.
Multicast Extensions to Open Shortest Path First (M-OSPF)	Uses "Intra-area routing" inside an OSPF. Uses source-based trees.
Protocol-Independent Multicast—Dense Mode (PIM-DM)	Communicates IP Multicast information to groups that are located in a small geographic area.
Protocol-Independent Multicast— Sparse Mode (PIM-SM)	Communicates IP Multicast information to sparsely distributed groups. One of the goals of this algorithm is to limit traffic by only providing data to routers interested in the information.

- **Implement IGMP Snooping**

  Older architecture may not handle multicast traffic particularly well. If your switch is not configured to filter multicast datagrams, you could flood your network with unwanted traffic, which could be sent to segments of your network where you aren't conducting IxChariot tests. Newer switches, on the other hand, offer the possibility of Internet Group Management Protocol (`IGMP`) snooping, which detects multicast traffic and routes it only to stations that have requested it. Talk to your vendor or consult your product documentation before you begin testing with IP Multicast.

Firewall Testing

Related Topics

Network Traffic

Before we discuss the steps necessary to implement testing through a firewall or other device that performs NAT (network address translation), it is necessary to review how IxChariot operates in several areas.

The following figure and ensuing discussion clarify the type of network traffic used by IxChariot and IxChariot tests. Only IP traffic is shown in the diagram, although IPX is mentioned in the discussion; the numbers shown are destination port numbers. Although TCP management port 10115 is shown in the figure, this port is configurable in IxChariot 6.50 and higher.

Figure 10-2.　　　IxChariot Network Traffic and Destination Port Numbers

n = specified in user settings, firewall tab
s = specified in test script
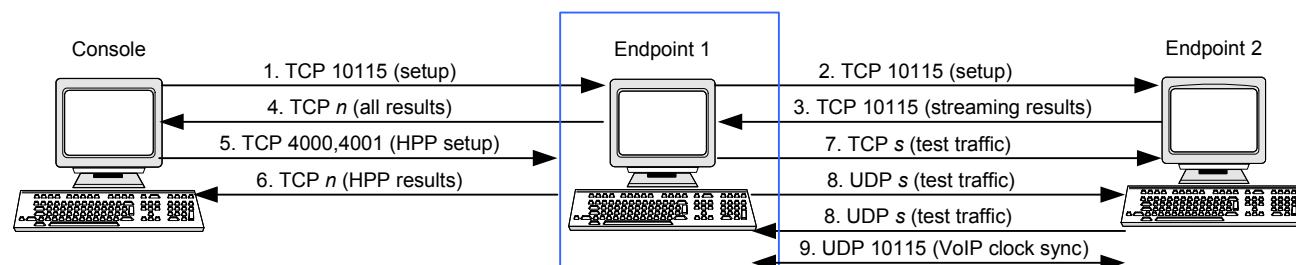
Each of the nine traffic flows shown in Figure 10-2 are explained below:

1. **Console to Endpoint 1 setup traffic**

   The console uses one of the following destination ports to setup Endpoint 1:

   • TCP traffic: Either 10115 (the default) or a user-designated port.
   • SPX traffic: 10117

   Whether TCP or SPX is used is determined in the port pair setup dialog in the *Endpoint 1 - Network Protocol* field.

2. **Endpoint 1 to Endpoint 2 setup traffic**

   Endpoint 1 uses one of the following destination ports to setup Endpoint 2:

   • TCP traffic: Either 10115 (the default) or a user-designated port.
   • SPX traffic: 10117

   Whether TCP or SPX is used is determined in the port pair setup dialog in the *Endpoint 1 - Network Protocol* field.

3. **Endpoint 2 to Endpoint 1 results for streaming pairs, VoIP pairs, and Video pairs**

   When streaming pairs, VoIP pairs, or Video pairs are used, test results are sent back from Endpoint 2 to Endpoint 1 using either 10115 (the default) or a user-designated port.

4. **Endpoint 1 to Console streaming results**

   All test results are sent back from Endpoint 1 to the Console using the port number specified in the *User Settings - Firewall* tab on the *TCP* or *SPX* line. The *Auto* checkbox allows IxChariot to choose the port. TCP or SPX usage is determined in the port pair setup dialog in the *Endpoint 1 - Network Protocol* field.

5. **Console to Endpoint 1 HPP setup**

   When Ixia Hardware Performance Pairs are used, Console to Endpoint 1 setup is performed by TCP connections on ports 4000 and 4001.

6. **Endpoint 1 to Console HPP results**

   When Ixia Hardware Performance Pairs are used, results are returned to the Console using TCP on a port specified in the *User Settings - Firewall* tab on the *Hardware Performance Pair Statistics* line. The *Auto* checkbox allows IxChariot to choose the port.

7. **Endpoint 1 to Endpoint 2 TCP test traffic**

   An IxChariot script specifies whether it uses TCP or UDP for its network traffic in the *Endpoint 2 - Network Protocol* field of the port pair setup dialog. If a script uses TCP for its test protocol, then Endpoint 1 to Endpoint 2 traffic will flow to the destination port on Endpoint 2 as determined in Endpoint 2's *CONNECT_ACCEPT*'s *port* parameter within the script.

8. **Endpoint 1 to Endpoint 2 UDP test traffic**

   An IxChariot script specifies whether it uses TCP or UDP for its network traffic in the *Endpoint 2 - Network Protocol* field of the port pair setup dialog. Two UDP flows might be used:

   • Endpoint 1 to Endpoint 2 traffic will flow to the destination UDP port on Endpoint 2 as determined in Endpoint 2's *CONNECT_ACCEPT*'s *port* parameter within the script.

   • Endpoint 2 to Endpoint 1 traffic will flow to the **source** UDP port on Endpoint 1 as determined in Endpoint 1's *CONNECT_INITIATE*'s *port* parameter within the script.

9. **Endpoint 1 to/from Endpoint 2 Clock Synchronization**

   For VoIP/RTP tests, it is necessary that Endpoint 1 and Endpoint 2 keep their clocks in synchronization. If specified, the alternate *Use Endpoint 1 address as management address* and *Use Endpoint 2 address as management address* addresses are used for this communications. This is accomplished by exchanging UDP messages on either 10115 (the default) or a user-designated port.

Endpoint 2 Destination Port Sharing and Data Correlation

Related Topics
Firewall Options Tab on page 6-32
Firewall Testing on page 10-13
Firewall and NAT on page 10-16
Testing with Firewalls on page 10-20

At times it's desirable that a host (or Ixia port) serves as an Endpoint 2 for multiple pairs. At times it's necessary that a particular TCP port number on an Endpoint 2 be used. For example, HTTP tests will typically require the use of port 80 on Endpoint 2. This is also necessary when a firewall is positioned between the endpoints.

Scripts which use an automatically generated destination address are not affected by this discussion. Scripts which use UDP may not share a destination port.

An Endpoint 2 may be used by multiple pairs to run scripts that accept connections on the same port. This is shown in Figure 10-3.

Figure 10-3.   Endpoint 2 Destination Port Sharing
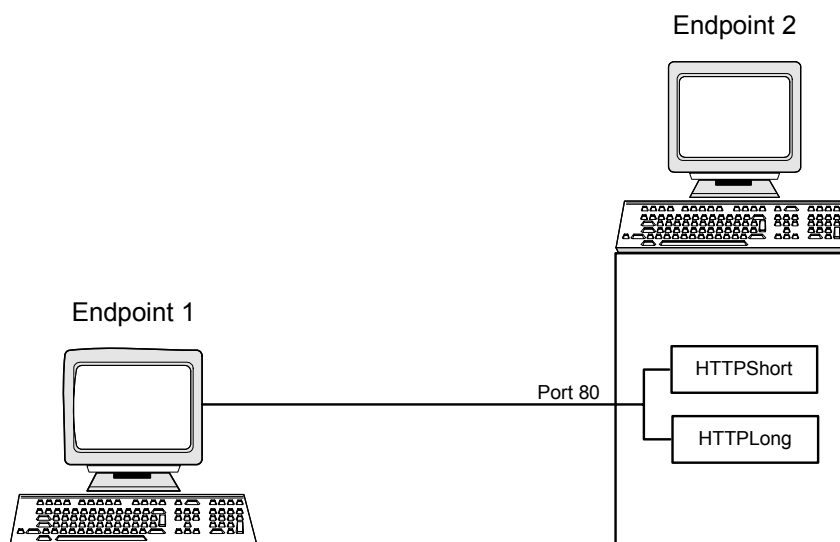
When a single pair of hosts is used to run two endpoint pairs with different scripts that attempt to connect to the same port on the same Endpoint 2, Endpoint 2 must be able to distinguish the pairs from each other. This is necessary in order to send the connection to the correct script – *HTTPShort* and *HTTPLong*, in Figure 10-3.

Endpoint 2 uses two techniques to differentiate the Endpoint 1's:

- **Source Port Identification.**

    Endpoint 2 looks at the source port number of the traffic that it receives in order to distinguish the pairs. We'll see that this can't be used in some cases when a firewall or other NAT'ing device sits between the endpoints.

- **Packet Correlator**

    The first four bytes of the first packet of traffic from Endpoint 1 to Endpoint 2 contains special data, called a correlator, which allows Endpoint 2 to distinguish the pairs. This technique, however, can't be used when specific content is included in the payload portion of the traffic. This includes the use of embedded script payloads and use of .cmp files.

When a NAT'ing device is used with payload contents, we'll see that there is no way for Endpoint 2 to distinguish the source of the packet. In this case, only one script may be used per particular Endpoint 2 IP port. In addition, only one connection should be used within the inner loop, so as to ensure even distribution.

Firewall and NAT   **Related Topics**
Firewall Options Tab on page 6-32
Firewall Testing on page 10-13
Endpoint 2 Destination Port Sharing and Data Correlation on page 10-15

A firewall is a programmable network device that controls the network traffic that is allowed to pass through it. Programming of a firewall is typically done by specifying a table of rules, each of which consists of the follow entries:

• Source address range. The range of source address to which this rule applies.

• Source port range. The range of source port numbers to which this rule applies.

• Protocol. The protocol(s) to which this rule applies, including IP-TCP and IP-UDP.

• Destination address range. The range of destination addresses to which this rule applies.

• Destination port range. The range of destination port numbers to which this rule applies.

• Action. If the rule matches a packet being inspected:

   • Drop / reject. The packet is blocked at the firewall.

   • Pass. Allow the packet to pass through the firewall.

If no rule applies, a default *match all-drop* rule is used.

When the first TCP packet in a session is 'passed' through, the firewall sets up an temporary rule that allows return traffic to pass through as well. For example, consider a packet with a source address of 200.10.10.1, using TCP on port 1025 and with a destination address of 198.1.1.1, port 80. If this packet matches a 'pass' rule, then the firewall will also set up a temporary rule that allows packets from 198.1.1.1, port 80 to pass through the firewall to 200.10.10.1, port 1025.

UDP packets generally do not get this type of treatment. Each UDP direction is treated as a separate flow, requiring a rule to handle it.

NAT (Network Address Translation) comes into play because firewalls are normally configured as shown in Figure 10-4.

Figure 10-4.   Normal Firewall/NAT Configuration
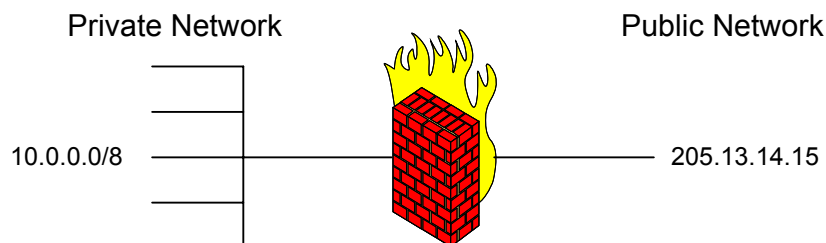
That is, an internal, network exists on the *private* side of the firewall. Non-Internet routeable networks, like 10.0.0.0/8, are typically used. On the other side of the firewall, a single Internet routeable network address is used. The firewall may use more than one address on its external network, if desired.

When an internal host, 10.0.0.2 for example, attempts to access a host on the Internet it is necessary for the firewall to translate the internal host's IP address and port into a new source address. For example,

- Internal host 10.0.0.2 uses source port 1025 to access an HTTP server at 205.100.100.2 on port 80.

- The firewall replaces the source address and port on the packet with its own address (205.13.14.15) and a dynamically generated source port; 1300 for example. This network address translation is saved in a firewall NAT table. The packet is then sent to the HTTP server.

- The HTTP server sends response packets on the TCP connection to 205.13.14.15:1300.

- The firewall looks at its NAT table and finds the entry for:

     10.0.0.2:1025 - 205.13.14.15:1300

and reverses the translation it did earlier. That is, it changes the destination address of the packet to port 1025 on 10.0.0.2 - thereby sending the response back to the private network host.

A similar process happens in reverse when an host on the public network attempts to access a server within the private network. For example,

- Although the firewall is often a computer that could support an HTTP server, this is not typically done. Instead, an internal server is used to host the HTTP server; let's say 10.0.0.42 in this example. This requires that the firewall have a permanent NAT table entry that translates all public reference to 205.13.14.15:80 to 10.0.0.42:80.

- External host 205.14.0.4, using source port 1400 attempts to access a web server at 205.13.14.15 on port 80.

- Using this NAT table entry, the destination address and port are re-written on the packet and sent to the internal HTTP server. A temporary firewall rule is installed that allows the return traffic to pass through the firewall. That is, 10.0.0.42:80 traffic sent to 205.14.0.4:1400 will be passed.

- When the internal HTTP server responds to the traffic, the temporary firewall rule allows the traffic to pass.

In summary, a firewall performs two basic functions:

- Allows or disallows traffic to pass between the private network(s) and public network(s).

- Performs network and port address translation so as to allow routing across networks.

Limitations on NAT Usage

There are some limitations on the use of NAT in IxChariot test networks:

- You cannot use NAT between the console and an Ixia chassis.

- You cannot use a dual-PAT network for IxChariot tests. This is a network in which PAT is used on both the private side and the public side of the test network, as shown in Figure 10-5.

Figure 10-5.   Dual-PAT Configuration
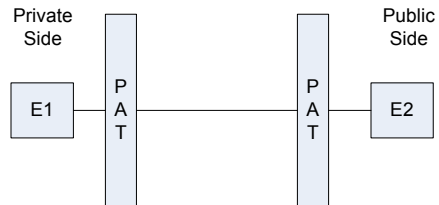
Testing with Firewalls

When a firewall is involved in a test, the firewall may appear in one of eight possible positions with respect to the Console, Endpoint 1, and Endpoint 2, as shown in Figure 10-6.

Figure 10-6. Firewall Positioning
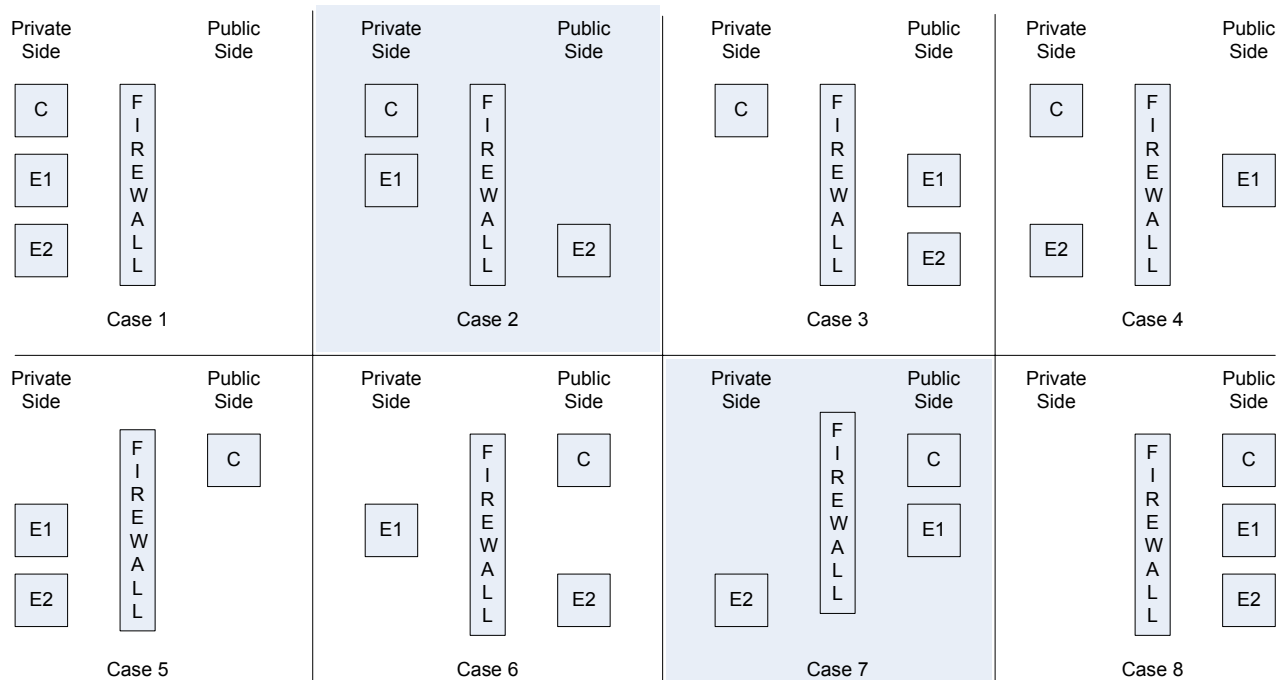
Unless otherwise specified, the following assumptions are made:

- The firewall has been previously programmed to allow all TCP and UDP connections originating from the private side of the network to the public side of the network. The firewall automatically allows return packets on the TCP connection to flow.

- All of the configurations specified must be performed on each Endpoint 1 to Endpoint 2 pair.

- The addresses used in pair setup are the same as one would otherwise use if a firewall were not involved.

- As shown in Figure 10-7, the settings in the *Change User Settings - Firewall Options* tab should be:

  - *Endpoint 1 through firewall to Console Reporting* - all *Auto* boxes checked.

• *PAT device support between E1 and E2* - unchecked.

Figure 10-7.   Firewall Option Settings
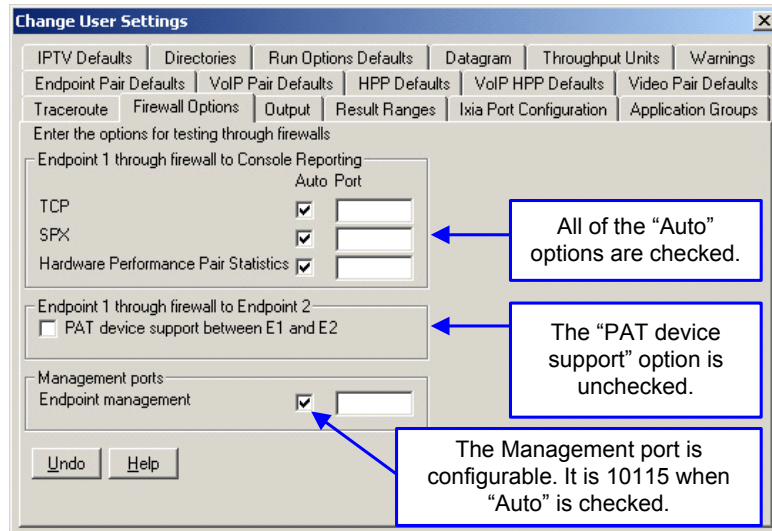
Each of the eight cases illustrated in Figure 10-6 on page 10-20 are explained in the following sections. Note that the two most common cases (case 2 and case 7) are shaded in the diagram.

Case 1 - Console, Endpoint 1 and Endpoint 2 Private

Because there is no firewall between any of the components, there are no special considerations.

Case 2 - Console and Endpoint 1 Private, Endpoint 2 Public

In case 2, the Console and Endpoint 1 are on the private side of the firewall, while Endpoint 2 is on the public side. Figure 10-8 illustrates this configuration.

Figure 10-8.   Firewall Configuration 2
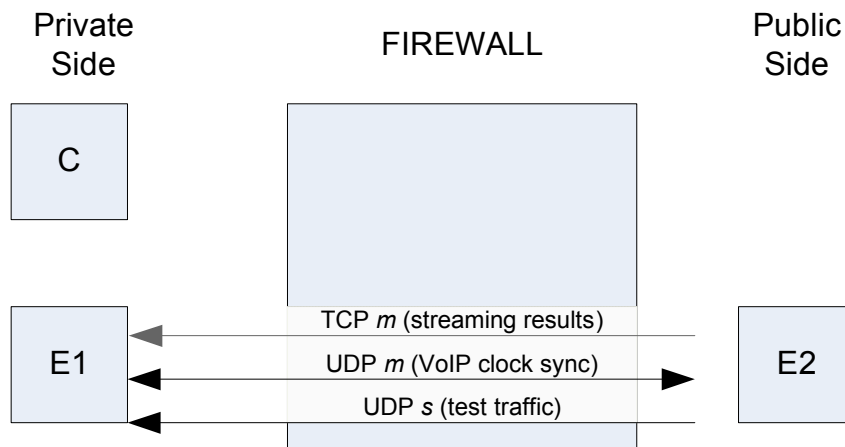
The port designations in Figure 10-8 are as follows:

• **TCP *m*** and **UDP *m*** refer to the port number specified in the *Endpoint Management* field in the *User Settings – Firewall Options* tab.

• **UDP *s*** refers to the port number (or numbers) specified in the script, if the script sends UDP messages from Endpoint 2 to Endpoint 1.

Case 2 Firewall Configuration

Your test network firewall must be configured as follows:

• **TCP *m* (streaming results)** – for streaming tests only. TCP *m* should be allowed through from the public to the private side and NAT'd to E1.

   Important! This implies that there can be only one E1 host or that each E1 has a unique public address.

• **UDP *m* (VoIP clock sync)** – for VoIP/RTP tests only. UDP on port *m* should be allowed through and NAT'd to E1.

• **UDP *s* (test traffic)** – if the test script(s) use UDP messages sent from Endpoint 2 to Endpoint 1, then the UDP port(s) used should be allowed through the firewall and NAT'd to E1.

Case 2 Pair Setup

To support the test network illustrated in Figure 10-8, configure your test pairs as follows:

• *Endpoint 1 address* – must be set to *localhost*. This causes Endpoint 2 to use the source address and port of the received packets from E1 (from the firewall) in order to talk back to Endpoint 1.

- *Use Endpoint 1 address as management address* – should be unchecked.
- *Use Endpoint 1 address as management address* – should be the real private network address of Endpoint 1.

Case 2 User Settings - Firewall Options

To support the test network illustrated in Figure 10-8, configure your firewall options as follows:

- *PAT device support between E1 and E2* – should be checked.

   Important! If payloads are used in any of the scripts that share a TCP port, then only one script may be used on any particular E2.

The test configuration settings required for Case 2 in the IxChariot Console are shown in Figure 10-9.

Figure 10-9.   IxChariot Settings for Firewall Case 2
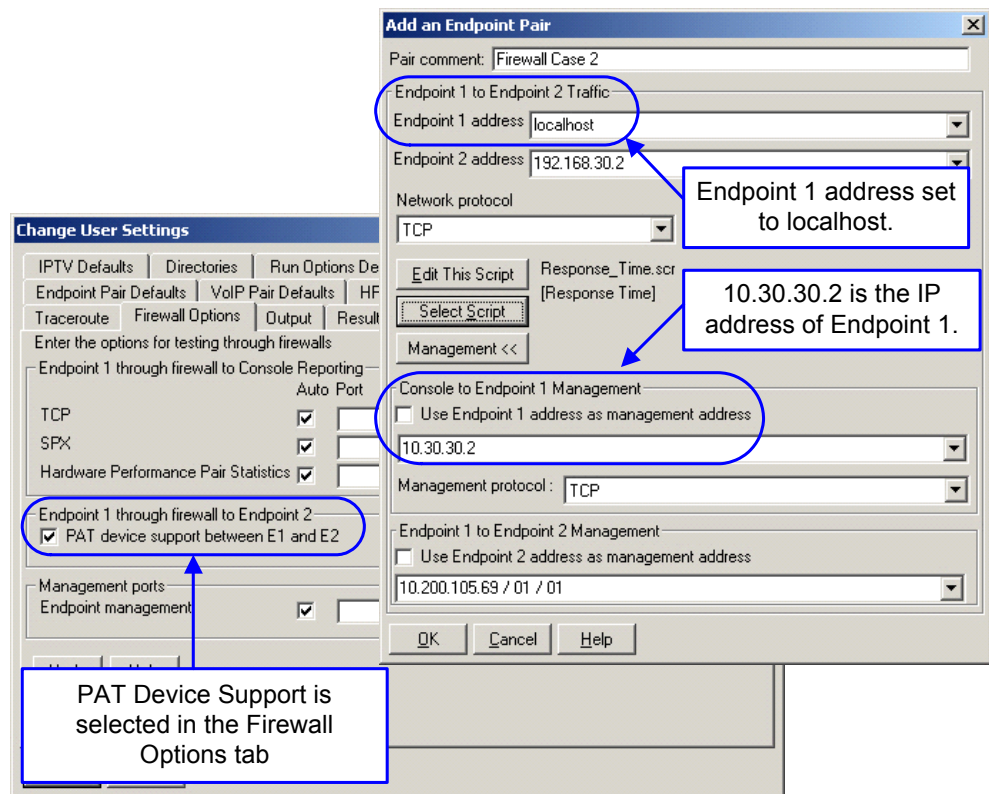
Case 3 - Console Private, Endpoint 1 and 2 Public

In case 3, the Console is on the private side of the firewall, while both Endpoint 1 and Endpoint 2 are on the public side. Figure 10-10. illustrates this configuration.

Figure 10-10.   Firewall Configuration 3
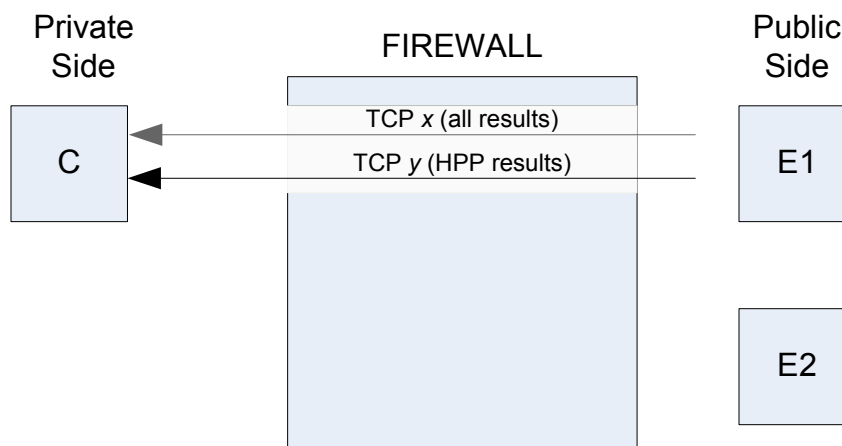
The port designations in Figure 10-10 are as follows:

- **TCP *x*** refers to the port number specified in the *Endpoint 1 through fire-wall to Console Reporting* field in the *User Settings – Firewall Options* tab.

- **TCP *y*** refers to the port number specified in the *Endpoint 1 through fire-wall to Console Reporting – Hardware Performance Pair Statistics* field in the *User Settings – Firewall Options* tab.

Case 3 Firewall Configuration

Your test network firewall must be configured as follows:

- **TCP *x* (all results)** – TCP *x* should be allowed through from the public to the private side and NAT'd to the Console.

- **TCP *y* (HPP results)** – only used when E1 is an Ixia Hardware Performance Port. TCP *y* should be allowed through from the public to the private side and NAT'd to the Console.

Case 3 Pair Setup

There are no specific pair setup requirements for this case.

Case 3 User Settings - Firewall Options

To support the test network illustrated in Figure 10-10, configure your firewall options as follows:

- *Console through firewall to E1 - TCP.* The *Auto* checkbox should be un-checked and a specific port should be set. A port that is not in use by the fire-wall and Console should be selected.

- *Console through firewall to E1 - Hardware Performance Pair Statistics*.
When E1 is an Ixia Hardware Performance Port only, the *Auto* checkbox
should be un-checked and a specific port should be set. A port that is not in
use by the firewall and Console should be selected.

The test configuration settings required for Case 3 in the IxChariot Console are
shown in Figure 10-11.

Figure 10-11.   IxChariot Settings for Firewall Case 3
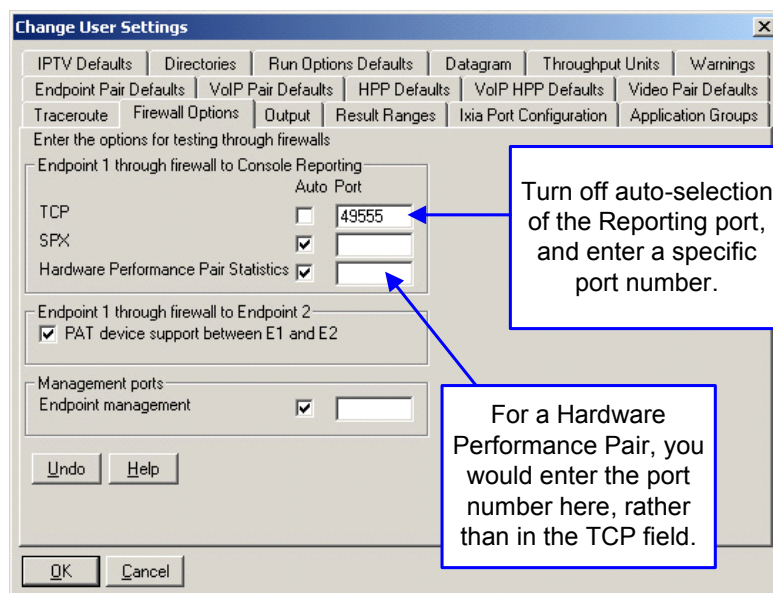
Case 4 - Console and Endpoint 2 Private, Endpoint 1 Public

In case 4, the Console and Endpoint 2 are on the private side of the firewall,
while Endpoint 1 is on the public side. Figure 10-12 illustrates this configuration.

Figure 10-12.   Firewall Configuration 4
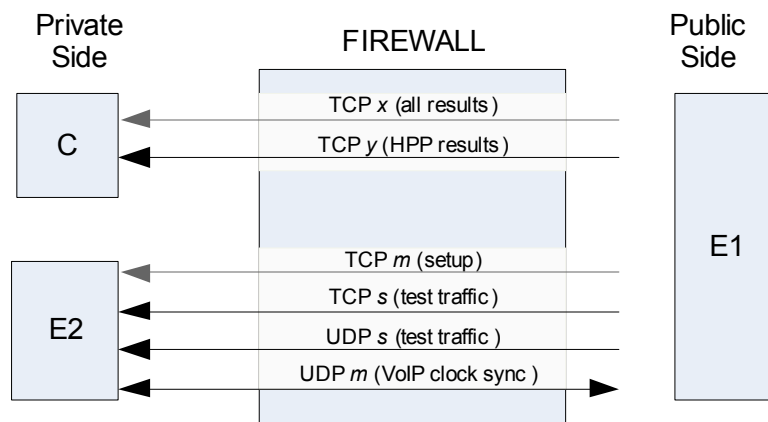
The port designations in Figure 10-12 are as follows:

- **TCP** *m* and **UDP** *m* refer to the port number specified in the *Endpoint Management* field in the *User Settings – Firewall Options* tab.

- **TCP** *x* refers to the port number specified in the *Endpoint 1 through firewall to Console Reporting* field in the *User Settings – Firewall Options* tab.

- **TCP** *y* refers to the port number specified in the *Endpoint 1 through firewall to Console Reporting – Hardware Performance Pair Statistics* field in the *User Settings – Firewall Options* tab.

- **UDP** *s* refers to the port number (or numbers) specified in the script, if the script sends UDP messages from Endpoint 2 to Endpoint 1.

Case 4 Firewall Configuration

Your test network firewall must be configured as follows:

- **TCP *x* (all results)** – TCP *x* should be allowed through from the public to the private side and NAT'd to the Console.

- **TCP *y* (HPP results)** – only used when E1 is an Ixia Hardware Performance Port. TCP *y* should be allowed through from the public to the private side and NAT'd to the Console.

- **TCP *m* (setup)** – TCP *m* should be allowed through from the public to the private side and NAT'd to E2.

  Important! This implies that there can only be one Endpoint 2 host used per firewall public address.

- **TCP *s* (test traffic)** – TCP traffic for the port used in the test script(s) must be allowed from the public to the private side and NAT'd to E2.

- **UDP *s* (test traffic)** – if the test script(s) use UDP messages send from Endpoint 1 to Endpoint 2, then the UDP port(s) used should be allowed through the firewall and NAT'd to E2.

- **UDP *m* (VoIP clock sync)** – for VoIP/RTP tests only. UDP on port *m* should be allowed through and NAT'd to E2. Remember that the *Use Endpoint 2 address as management address* is used for this traffic.

Case 4 Pair Setup

To support the test network illustrated in Figure 10-12, configure your test pairs as follows:

- *Endpoint 2 address* – must be set to the public address of E2.

Case 4 User Settings - Firewall Options

To support the test network illustrated in Figure 10-12, configure your firewall options as follows:

- *Endpoint 1 through firewall to Console Reporting – TCP.* The *Auto* checkbox should be un-checked and a specific port should be set. A port that is not in use by the firewall and Console should be selected.

- *Endpoint 1 through firewall to Console Reporting – Hardware Performance Pair Statistics.* When E1 is an Ixia Hardware Performance Port only, the *Auto* checkbox should be un-checked and a specific port should be set. A port that is not in use by the firewall and Console should be selected.

The test configuration settings required for Case 4 in the IxChariot Console are shown in Figure 10-13.

Figure 10-13.   IxChariot Settings for Firewall Case 4

Case 5 - Console Public, Endpoints 1 and 2 Private

In case 5, the Console is on the public side of the firewall, while Endpoint 1 and Endpoint 2 are on the private side. Figure 10-14 illustrates this configuration.

Figure 10-14.   Firewall Configuration 5
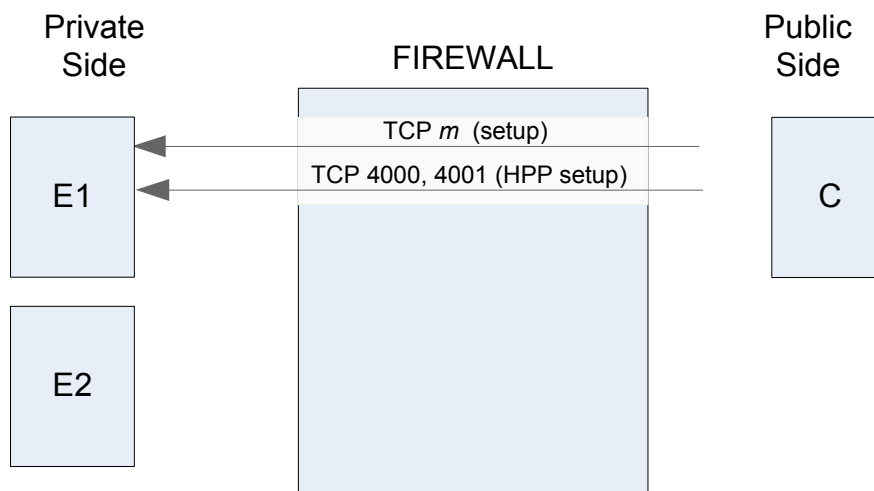
The port designations in Figure 10-14 are as follows:

- **TCP *m*** refers to the port number specified in the *Endpoint Management* field in the *User Settings – Firewall Options* tab.

- **TCP 4000, 4001** are the non-configurable destination port numbers used by hardware performance pairs for sending setup traffic from the Console to Endpoint 1.

Case 5 Firewall Configuration

Your test network firewall must be configured as follows:

- **TCP *m*** – TCP *m* should be allowed through from the public to the private side and NAT'd to E1.

- **TCP 4000, 4001 (HPP setup)** – only used when E1 is an Ixia Hardware Performance Port. TCP 4000 and 4001 should be allowed through from the public to the private side and NAT'd to E1.

Case 5 Pair Setup

To support the test network illustrated in Figure 10-14, configure your test pairs as follows:

- *Use Endpoint 1 address as management address* – should be unchecked.

- *Use Endpoint 1 address as management address* – should be the public address of the firewall.
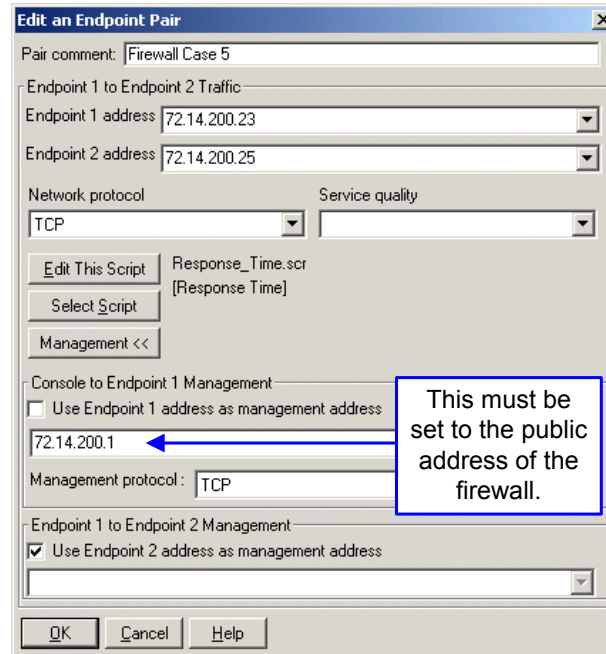
    Important! This implies that there can only be one host used for Endpoint 1 per public firewall address.

Case 5 User Settings - Firewall Options

This case has no specific firewall option settings.

The test configuration settings required for Case 5 in the IxChariot Console are shown in Figure 10-15.

Figure 10-15.   IxChariot Settings for Firewall Case 5

Case 6 - Console and Endpoint 2 Public, Endpoints 1 Private

In case 6, the Console and Endpoint 2 are on the public side of the firewall, while Endpoint 1 is on the private side. Figure 10-16 illustrates this configuration.

Figure 10-16.   Firewall Configuration 6
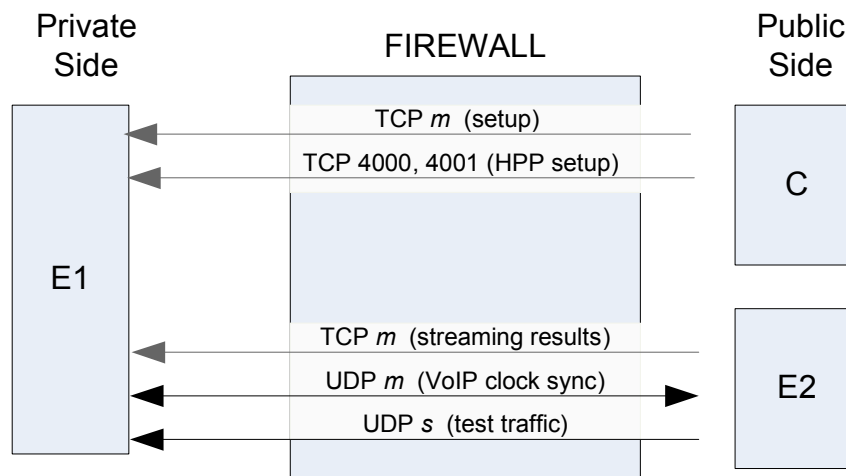
The port designations in Figure 10-16 are as follows:

- **TCP *m*** and **UDP *m*** refer to the port number specified in the *Endpoint Management* field in the *User Settings – Firewall Options* tab.

- **TCP 4000, 4001** are the non-configurable destination port numbers used by hardware performance pairs for sending setup traffic from the Console to Endpoint 1.

- **UDP *s*** refers to the port number (or numbers) specified in the script, if the script sends UDP messages from Endpoint 2 to Endpoint 1.

Case 6 Firewall Configuration

Your test network firewall must be configured as follows:

- **TCP *m* (setup)** – TCP *m* should be allowed through from the public to the private side and NAT'd to E1.

  Important! This implies that there may only be one E1 host in this configuration.

- **TCP 4000, 4001 (HPP setup)** – if E1 is an Ixia Hardware Performance Port only. TCP 4000 and 4001 should be allowed through from the public to the private side and NAT'd to E1.

- **TCP *m* (streaming results)** – for streaming tests only. TCP *m* should be allowed through from the public to the private side and NAT'd to E1. This is the same rule as the first bullet in this section.

- **UDP *m* (VoIP clock sync)** – for VoIP/RTP tests only. UDP on port *m* should be allowed through and NAT'd to E1.

- **UDP *s* (test traffic)** – if the test script(s) use UDP messages sent from End-
point 2 to Endpoint 1, then the UDP port(s) used should be allowed through
the firewall and NAT'd to E1.

Case 6 Pair Setup

To support the test network illustrated in , configure your test pairs
as follows:

- *Endpoint 1 address* – must be set to *localhost*. This causes Endpoint 2 to use
the source address and port of received packets (from the firewall) in order to
talk back to Endpoint 1.

- *Use Endpoint 1 address as management address* – should be unchecked.

- *Use Endpoint 1 address as management address* – should be E1's public IP
address.

Case 6 User Settings - Firewall Options

To support the test network illustrated in , configure your firewall
options as follows:

- *NAT support between E1 and E2* – should be checked.

  Important! If payloads are used in any of the scripts that share a TCP port,
  then only one script may be used on any particular E2.

The test configuration settings required for Case 6in the IxChariot Console are
shown in .

Figure 10-17.   IxChariot Settings for Firewall Case 6
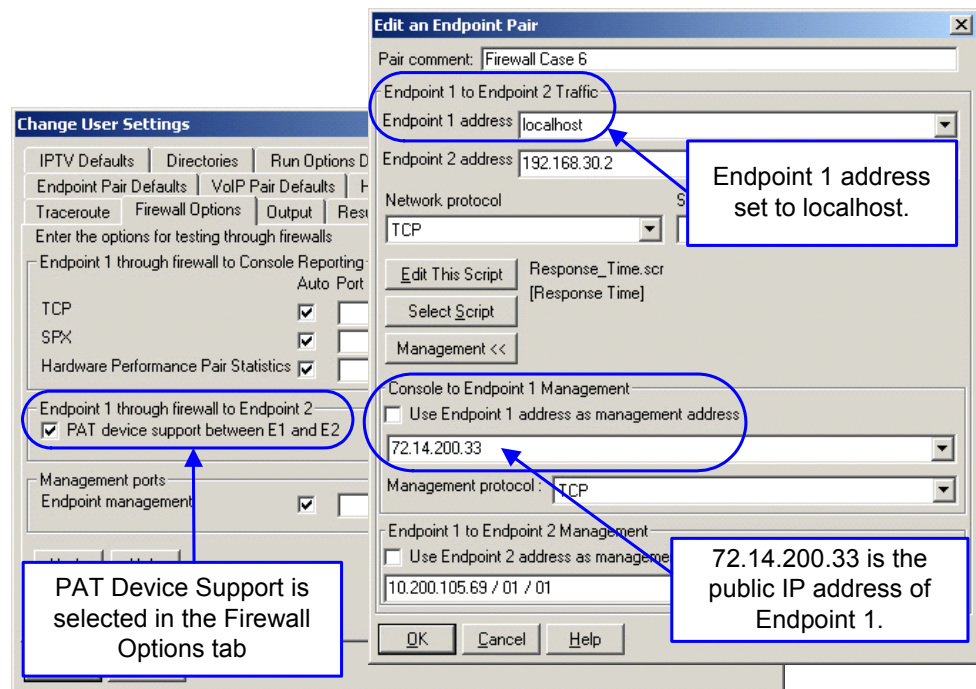
Case 7 - Endpoint 2 Private, Console and Endpoint 1 Public

In case 7, Endpoint 2 is on the private side of the firewall, while the Console and Endpoint 1 are on the public side. Figure 10-18 illustrates this configuration.

Figure 10-18.　　Firewall Configuration 7
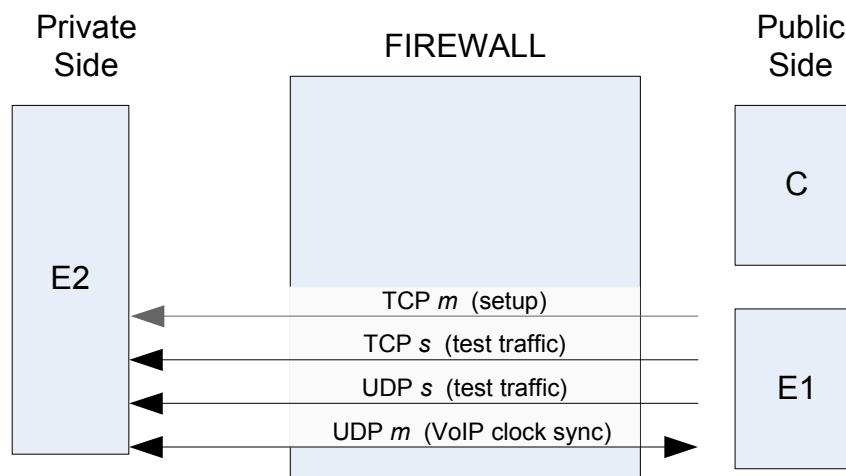
The port designations in Figure 10-18 are as follows:

- **TCP *m*** refers to the port number specified in the *Endpoint Management* field in the *User Settings – Firewall Options* tab.

- **TCP *s*** and **UDP *s*** refer to the port number (or numbers) specified in the scripts.

Case 7 Firewall Configuration

Your test network firewall must be configured as follows:

- **TCP *m* (setup)** – TCP *m* should be allowed through from the public to the private side and NAT'd to E2.

- **TCP *s* (test traffic)** – TCP traffic for the port used in the test script(s) must be allowed from the public to the private side and NAT'd to E2.

- **UDP *s* (test traffic)** – if the test script(s) use UDP messages sent from End-point 1 to Endpoint 2, then the UDP port(s) used should be allowed through the firewall and NAT'd to E2.

- **UDP *m* (VoIP clock sync)** – for VoIP/RTP tests only. UDP on port *m* should be allowed through and NAT'd to E2.

Case 7 Pair Setup

To support the test network illustrated in Figure 10-18, configure your test pairs as follows:

- *Endpoint 2 address* – must be set to the public address of E2.

Important!

- This implies that there can be only one Endpoint 2 host or each E2 must have a unique public IP address.

- Only the traffic types marked OK in the table below are supported in this topology

Traffic Type	IP translation (NAT)	IP translation + port translation (NAT + PAT)
TCP	OK (bind to ANY_ADDR)	Failure (bind to E2's port)
UDP reliable	Failure (bind to E2's address)	Failure (bind to E2's port)
Streaming (unicast and multicast)	OK (bind to ANY_ADDRESS)	Failure (bind to E2's port)

Case 7 User Settings - Firewall Options

There are no specific firewall option settings required for this configuration.

The test configuration settings required for Case 7 in the IxChariot Console are shown in Figure 10-19.

Figure 10-19.   IxChariot Settings for Firewall Case 7
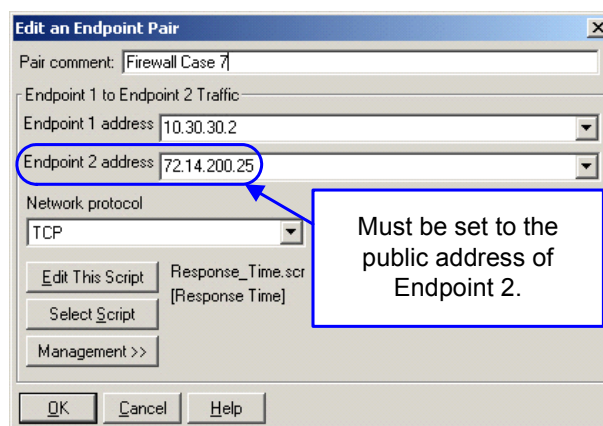
Case 8 - Console, Endpoint 1 and Endpoint 2 Public

Because there is no firewall between any of the components, there are no special considerations.

Voice over IP Testing

Related Topics

If you are thinking about expanding your use of voice over IP (VoIP) applications, you should plan to use IxChariot to determine how much voice traffic can be added to an existing network without affecting business-critical applications. The following topics offer advice to aid you when you are preparing to roll out a new VoIP application or when you are adding VoIP to an existing network. See *VoIP Test Module Features* on page 10-35 for more information.

The objective of VoIP is usually to lower telecommunications costs by switching voice traffic to under utilized WAN and LAN links. However, voice traffic needs special treatment on your network. If you are thinking about implementing a bandwidth-management scheme, you need to find out how many voice channels your network can currently support and how the voice traffic might affect business applications.

It is also entirely possible that your network is not configured to handle VoIP traffic. Voice traffic is uniquely time-sensitive. It cannot be queued, and if datagrams are lost, the conversation will be choppy or even incomprehensible. NAT-enabled firewalls, slow or excessively congested links, or improperly implemented QoS schemes are just a few of the factors that could inhibit or even prevent VoIP traffic from crossing your network in a recognizable form.

IxChariot offers the following features to help you run voice traffic on your IP network and determine its quality:

- Five different codec types, emulating different compression algorithms, data rates, and datagram sizes.

- Flexibility in changing datagram sizes.

- The ability to use silence suppression and set the voice activity rate.

- A jitter buffer you can configure.

- Quality of service.

- Performance and quality metrics that include jitter (measured two ways), lost data, consecutive lost datagrams, one-way (network) delay, and a Mean Opinion Score.

VoIP Test Module Features

The VoIP test modules includes these features:

Table 10-4.   VoIP Test Module Features

Feature Categories	VoIP Test Module Features
Testing Options	VoIP endpoint pairs
	VoIP hardware performance pairs
	VoIP Pair Defaults tab
	6 codecs
	VoIP connectors (Visual Test Designer)
	Result Ranges tab
Test Window Tabs	VoIP tab
	One-Way Delay tab
	Consecutive Lost Datagrams tab
	Maximum Consecutive Lost Datagrams tab
	Jitter (Delay Variation) tab
	Jitter (Delay Variation) Maximum tab
Performance Metrics	Delay variation jitter
	Delay variation jitter maximum
	Consecutive lost datagrams
	Maximum consecutive lost datagrams
	Jitter buffer lost datagrams
	One-way (network) delay
	End-to-end delay
	Mean Opinion Score estimate

Planning a Series of VoIP Tests

Related Topics

Voice over IP Testing on page 10-34
Adding or Editing a VoIP Endpoint Pair on page 10-38
Jitter Buffers on page 10-52
Codec Types on page 10-43
VoIP Pair Limits on page 10-36

VoIP applications are very throughput- and CPU-intensive. They will almost certainly disrupt the other traffic on your network once they're in use. You also need to know how many voice channels you need to make high-quality calls on your network. It is a good idea, therefore, to baseline several business-critical applications with IxChariot before rolling out voice over IP.

Before you set up a VoIP test or develop a test plan, you need to understand several issues. Your network's handling of VoIP traffic depends on the following factors:

• What type of codec will you be using?

• At what rate does your codec send VoIP data?

• How many voice channels will you have?

In creating a series of tests that exactly fit the requirements of your VoIP implementation, IxChariot's VoIP support makes most of the choices for you. Simply

choose the codec, and the correct compression algorithm and bit rate are selected automatically. The codec's bit rate determines how much bandwidth the voice traffic requires. See *Codec Types* on page 10-43 for more information.

Will your VoIP equipment be sending full-duplex (G711) 64 kbps traffic, or lower bit-rate voice encoding (G723 or G729) at 5.3 or 6.3 kbps? The default values in an IxChariot VoIP test emulate a unidirectional voice stream. If you are emulating a full-duplex bi-directional voice stream like G711, set up two pairs using the G711 codec for each voice channel being emulated.

Also consider the type of topology you are emulating. Will all channels be going over dedicated WAN links between PBXs, or will they reach desktops via the LAN? If you are trying to determine how much voice traffic can be supported using excess WAN capacity, multiply the data rate by the number of voice channels being emulated. For example:

```
10 64 kbps channels = 10 X 64 kbps = 640 kbps IxChariot
data rate
```

To emulate a PBX or campus configuration that includes multiple connections, increase the number of test pairs to equal the number of channels:

```
 1 Full-Duplex 64 kbps VoIP = 2 pairs
10 Full-Duplex 64 kbps VoIP = 20 pairs
```

Before you begin VoIP testing with IxChariot, run some baseline performance tests, using the `Throughput` and `Response_Time` scripts, or scripts emulating mission-critical applications. Keep records of the throughput your users have come to expect. Then add voice traffic to the baseline test in increments.

Select the `initial_delay` for the VoIP pairs so that those pairs begin executing their scripts after the pairs representing normal business traffic have been running for a while. Compare the results of the baseline test with the baseline plus voice test in the Comparison window.

Continue adding pairs to the test until you see a performance degradation from the business applications. This gives you an estimated upper limit on the number of voice channels that can be supported before you start hindering other traffic. The MOS estimates for the VoIP pairs should also demonstrate the quality of the calls.

VoIP Pair Limits

The following guidelines suggest the maximum number of simultaneous VoIP pairs you can run without sacrificing call quality.

In our own testing, we used two Pentium 800 computers with 256 MB of RAM on Windows 2000, SP 1 for the endpoints. Although the codec used is the most significant performance factor, we've discovered that the brand of NIC card installed in the endpoints is also significant.

Table 10-5.   Two-Way Voice Flow Test

Codec	Number of Pairs
G711u/G711a (64 kbps)	200
G.726 (32 kbps)	250
G729 (8 kbps)	150
G723m (6.3 kbps)	250
G723a (5.3 kbps)	250

Table 10-6.   One-Way Voice Flow Test

Codec	Number of Pairs
G711u/G711a (64 kbps)	200
G.726 (32 kbps)	250
G729 (8 kbps)	150
G723m (6.3 kbps)	200
G723a (5.3 kbps)	200

If the number of VoIP pairs is too large for the transmitting endpoint to handle (or too much is happening on the endpoint), the timestamp on the packet may 'slip'. Although RFC 3095 paragraph 4.5.3 calls for the timestamp to increase by a constant value for each packet, IxChariot inserts the actual transmit time in the packet so that it can accurately calculate the packet's delay from endpoint 1 to endpoint 2. You'll need to decrease the number of pairs if evenly spaced timestamps are a requirement for your equipment. Note that this 'slip' does not occur when using VoIP Hardware Performance Pairs.

As part of its VoIP testing, endpoint 1 will transmit 5 pseudo-VoIP packets at the end of the test in order to tell endpoint 2 that the test is complete. These packets look like they are RTP packets, but violate RFC 1889 by not incrementing the sequence number or time stamp.

Setting up a Voice over IP Test

Related Topics
Planning a Series of VoIP Tests on page 10-35
Adding or Editing a VoIP Endpoint Pair on page 10-38
Jitter Buffers on page 10-52
Codec Types on page 10-43

Before you get started testing your network to determine voice over IP readiness, make sure you are familiar with your voice over IP equipment and the application(s) you are going to be testing. Keep track of the following parameters:

• The codec your VoIP network will use.

• The size, in milliseconds, of the datagrams your VoIP application sends over the network.

• The size of any jitter buffer at the receiving stations.

•	The way your application handles silence suppression.

•	The quality of service that will be applied to the voice traffic by any interven-ing routers.

IxChariot lets you configure all of these parameters to exactly match the actual traffic voice over IP will send across your network.

To create a test of a VoIP implementation, open a new test file. On the Edit menu, click **Add VoIP Pair**. Refer to *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for more information about each configuration parameter. When you run a VoIP test for the first time, you'll probably notice that it takes a bit longer to initialize than a test with regular endpoint pairs. That's because for VoIP testing, the endpoints in each pair must synchronize their clocks to measure one-way (network) delay. Refer to *One-Way Delay* on page 10-47 for more infor-mation.

Adding or Editing a VoIP Endpoint Pair

Related Topics
Codec Types on page 10-43
Setting up a Voice over IP Test on page 10-37
Jitter Buffers on page 10-52
Silence Suppression on page 10-47
Adding or Editing an Endpoint Pair on page 5-19

To create a voice over IP test, click **Add VoIP Pair** on the Edit menu. Choose from the following configuration parameters. Click **Advanced** to edit the appli-cation script to be used in your test.

For VoIP testing, by default, Endpoint 1 executes a test script, receives results from Endpoint 2, and reports them to the Console over the same network seg-ment. Depending on the size of your test or the reliability of your test network, you may want to isolate test setup and results traffic from actual test traffic; doing so can increase the accuracy of results or decrease the likelihood that setup data will be lost, causing the test to fail.

If you want to use an alternate network for setup/results reporting, or a different network protocol or service quality between the Console and E1 and between the endpoints, fill in the values for *Use Endpoint 1 address as management address* and *Use Endpoint 2 address as management address*. For example, you might use TCP to connect from the Console to E1, yet run a UDP test between E1 and E2. But keep in mind that IxChariot only uses connection-oriented protocols for setup communications between the Console and E1, and that TCP is required for setup communications between E1 and E2.

•	**Endpoint 1 to Endpoint 2 Traffic**

  •	**Endpoint 1 address**

    The network address of a computer playing the role of Endpoint 1 in an IxChariot test. The IxChariot Console contacts Endpoint 1 computers directly, sending them the test setup information you entered. Endpoint 1 acts as the sender of voice data. Enter a network address that matches the protocol you are using. Refer to *Adding or Editing an Endpoint Pair* on page 5-19 for more information.

- **Endpoint 2 address**

  A computer playing the role of Endpoint 2 in an IxChariot test. Endpoint 1 computers contact the Endpoint 2 computers with test setup information. Endpoint 2 acts something like the receiver of voice data in a VoIP call.

- **Codec**

  The type of codec used on your network. Choose one of the supported codecs from the list. Refer to *Codec Types* on page 10-43 for more information.

- **Packet Loss Concealment (PLC)**

  For G.711u and G.711a codecs only. See *Codec Types* on page 10-43 for more information. PLC should improve your MOS estimate, but it is only factored into the calculation if your results include some data loss. Most G.711 codecs today do not support PLC.

- **Use silence suppression**

  Emulates the effects of silence suppression (also called voice activity detection) on the line during the VoIP test. Refer to *Silence Suppression* on page 10-47 for more information.

- **Voice activity rate**

  An indicator of the percentage of time during the call that talking is occurring. Determines how much actual voice data the simulated call contains for your tests. You can set any rate from 1% to 100% to indicate the amount of voice data transmitted during a call. Default value is 50%. Only enabled if "Use silence suppression" is checked.

- **Override delay between voice datagrams**

  Determines the datagram size to be used in the VoIP test. VoIP applications break voice data into datagrams based on delay, or the amount of time between successive datagrams. Default value is 20 ms. Values must be between 10 and 200 ms. IxChariot may adjust values slightly after you've entered them so that no partial buffers are sent.

- **Timing record duration**

  The length of each timing record. Timing records are used to take measurements during an IxChariot test; see *Understanding Timing* on page 11-1 for more information. The length of a timing record determines the number of datagrams whose measurements are included in each timing record your test generates. The default value is 3 seconds, or about 150 VoIP datagrams if the datagram size is 20 ms (see "Override delay between voice datagrams," above). The time you set is used to calculate the number of datagrams that will be received in that time period if there is no lost data or delay. The endpoint uses the number of datagrams to be received when determining when a timing record ends. It is therefore possible to have timing records longer than the duration specified.

- **Service quality (optional)**

  Emulates the effects of a Quality of Service (QoS) scheme on call quality. IxChariot's pre-configured VoIP template and any QoS templates you have already created are available in the list. The template named **VoIPQoS** is recommended for VoIP testing. However, it is not supported

by all endpoint operating systems. Refer to "Endpoint Capabilities" in the *Performance Endpoints* guide for a complete listing of endpoints that support testing with IP TOS templates. Optional field.

The following fields are available by pushing the **Management>>** button.

* **Console to Endpoint 1 Management**

  * **Use Endpoint 1 address as management address**

    The Console will communicate with Endpoint 1 using the network address and protocol for E1 specified in the pair. If the Console needs to send test setup information, such as the address of E2, to E1 through an alternate network, clear the box and specify a setup address. Or clear the box to select a different protocol. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1. For example, if you choose the IPX protocol for the test, the default is to use SPX between the Console and Endpoint 1. For RTP and UDP, the default is TCP.

    The field below the check box holds the network address where the Console can contact E1. Lets you choose a different address for test setup and results flows between the Console and E1 than the endpoint address specified in the pair itself. Endpoint computers can have multiple network addresses. For example, a computer with multiple adapters may have multiple IP addresses.

    * If you are changing the network address at which E1 contacts E2 with setup information (see "**Use Endpoint 2 address as management address**," below), you'll probably need to change the address at which the Console contacts E1 as well.

    * In tests running streaming scripts (and in VoIP tests), be aware that E2 sends results back to E1. If the network is unreliable, you should specify a more reliable address for E1 here. It will also be used by E2 for results flows. Endpoint 2

  * **Management Protocol**

    The protocol that the Console will use to contact Endpoint 1 for the purpose of setup.

* **Endpoint 1 to Endpoint 2 Management**

  * **Use Endpoint 2 address as management address**

    The network address and protocol used for test setup and results flows will be the same as those you specified when you created the endpoint pair. If E1 needs to send setup information, such as the application script, to E2 through an alternate network, clear the box and specify a setup address. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to E1. if you're using IPv6 and this box is checked, E1 uses TCP for IPv6 for this connection. If you clear this box and supply an alternate address, TCP for IPv4 is used.

    The field below the check box holds the network address where E1 can contact E2. Lets you enter a different address for setup and results flows between the endpoints than is specified for test traffic in the pair itself.

Endpoint computers can have multiple network addresses. Often, a more reliable network connection can be made between the endpoints at different addresses to ensure that setup and reporting information is received. In tests running streaming scripts (and in VoIP tests), E2 sends results back to E1. If the network is unreliable, you should specify a more reliable address for E2 here so that the endpoints use a reliable network for test setup and reporting.

The following fields are only available after pressing the **Advanced>>** button.

- **Initial delay value**

    Adds a SLEEP at the beginning of the application script. Recommended for emulating the effects of multiple users accessing the network. Choose Constant value to enter an exact time setting in milliseconds, or choose one of the four mathematical distributions to let IxChariot select a series of values. See "Setting Sleep Times" in the *Application Scripts* guide for more information.

- **Upper limit**

    Determines the value of the delay if you've chosen Constant Value, or sets the upper limit of the mathematical distribution if you've chosen a mathematical distribution. Initial delays must be between 0 and 2147483647 ms.

- **Lower limit**

    Sets the lower limit of the mathematical distribution if you've chosen a mathematical distribution for your delay. Initial delays must be between 0 and 2147483647 ms. Not needed for a Constant delay value.

- **Additional Fixed Delay**

    Lets you add delay to emulate the effects of known equipment impairments, or any other known, fixed source of delay on your network. For example, if you are testing equipment that adds 10 milliseconds of delay to each datagram, enter 10 ms here. Values are in milliseconds; range is 0-300 ms. Optional field.

- **Jitter buffer**

    Emulates the effects of jitter buffering on your VoIP network. Jitter buffers may be configured based on time (called an "absolute" jitter buffer) or based on number of datagrams (a "frame-based" jitter buffer). Refer to *Jitter Buffers* on page 10-52 for more information.

- **Source and destination port numbers**

    Sets the port number used for the source (Endpoint 1) and the destination (Endpoint 2) in the voice transmission.The default value for both the source and the destination ports is AUTO. Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about setting source and destination port numbers.

- **Payload**

    The Payload option allows you to include payload data with your VoIP endpoint pairs. You can use either of the following sources for the data:

    - Random payload: This option generates random data to simulate VoIP traffic. In this case, the size of the payload is calculated based on timing record duration and codec data rate.

- Payload file: With this option you select an external file containing pay-load data for the pair (in the Filename field). The data must already be encoded with the correct codec (G.711, G.729, and so forth). Because the timing record duration will depend on the file size and codec data rate, the timing record controls will be disabled.

When you select Payload file, you must specify whether the payload should be embedded in the script or stored externally in Referenced payload files.

For more information about the use of payload files, refer to the *IxChariot Scripts Development and Editing Guide*.

The SYNCHRONIZATION**>>** button becomes active only when the VoIP is added to a application group. For detailed instructions, refer to the *IxChariot Scripts Development and Editing Guide*.

Adding or Editing a VoIP Hardware Performance Pair

Related Topics

A VoIP hardware endpoint pair utilizes a pair of Ixia test ports as VoIP end-points. It includes the network and management addresses of the two computers.

- **Port 1 and Port 2 management addresses**

  Ixia ports each have an IPv4 management address by which the IxChariot Console controls the hardware. This management address is of the form:

  <base octet 1>.<base octet 2>.<card #>.<port #>

  - The first two octets are associated with the *base address* of the Ixia chassis, which is normally 10.0.0.0. When two or more chassis are used in an IxChariot test, all of the chassis need to have different base addresses. These base addresses may be observed and modified using Stack Manager.

  - The third octet is the card number for the port.

  - The fourth octet is the port number on the card.

  The management addresses for both ports must be entered.

- **Port 1 network address**

  Port 1 will be used to generate unidirectional network traffic destined for Port 2. The network address is used in the establishment of the source address for the traffic. The network address may be an IPv4 or IPv6 address.

- **Port 2 network address**

  Port 2 will be used to receive the unidirectional network traffic sent from Port 1. The network address is used in the establishment of the destination address for the traffic. The network address may be an IPv4 or IPv6 address.

- **Codec**

  The type of codec used on your network. Choose one of the supported codecs from the list. Refer to *Codec Types* on page 10-43 for more information.

- **Override delay between voice datagrams**

  Determines the datagram size to be used in the VoIP test. VoIP applications break voice data into datagrams based on delay, or the amount of time between successive datagrams. Default value is 20 ms. Values must be between 10 and 200 ms. IxChariot may adjust values slightly after you've entered them so that no partial buffers are sent.

- **Service quality**

  Emulates the effects of a Quality of Service (QoS) scheme on call quality. Only the template named **VoIPQoS** is available by default for VoIP hardware performance pairs. Additional templates may be crated from the main IxChariot screen's *Tools ... Edit QoS Templates...* menu choice.

- **Concurrent voice streams**

  The number of concurrent voice streams that will be generated by the hardware performance pair.

- **UDP source port number**

  The UDP source port to be used for the generated traffic. The source port is **not** incremented for each call.

- **UDP destination port number**

  The UDP destination port to be used for the generated traffic.

- **Pair Comment (optional)**

  A descriptive word or phrase that lets you easily identify each pair in the Test window.

Cloning VoIP Hardware Performance Pairs

Related Topics

Adding or Editing a VoIP Endpoint Pair on page 10-38

The process of cloning a hardware performance VoIP pair involves using the addresses of a non hardware performance pair in order to create a hardware performance pair. This can be done by selecting a non hardware performance VoIP pair in the Test Setup window and then selecting the *Edit ... Clone Hardware Performance VoIP Pair*. This will cause the *Clone Hardware Performance VoIP Pair* dialog to be displayed. Its contents and operation are as described in *Adding or Editing a VoIP Endpoint Pair* on page 10-38.

Codec Types

Related Topics

VoIP Score Calculation on page 10-48
Adding or Editing a VoIP Endpoint Pair on page 10-38

In a voice over IP transmission, the codec samples the sound and determines the data rate. You can perform voice over IP testing with IxChariot using 6 codec types. In the descriptions below, the "**packetization delay**" refers to the delay this codec introduces as it converts a signal from analog to digital; this delay is included in the MOS estimate, as is the "**jitter buffer delay**," the delay introduced by the effects of buffering to reduce inter-arrival delay variations. See *VoIP Score Calculation* on page 10-48 for more information.

Prior to IxChariot 6.0, *packetization delay* was a fixed value - as shown in the last column in Table 10-7. Starting with IxChariot 6.0, packetization delay is calculated as `"Delay between voice datagrams"` + `"look-ahead delay"` + `"frame_size"`. The Default Datagram Size column in Table 10-7 shows the default values for the delay between voice datagrams. You can change this value in the Add a VoIP Endpoint Pair dialog. For example, if you are using code G.723.1 and you change the delay between voice datagrams to 40 ms, the packetization delay will be 47.5.

Packet loss concealment (PLC) is an additional option if you are using the G.711u or G.711a codecs. PLC describes a number of techniques for minimizing or masking the effects of data loss during a VoIP conversation, including replaying the last received packet and estimating the lost packet's content. When PLC is enabled, IxChariot assumes that the quality of your conversation would be improved, but this improvement is only factored into the MOS estimate calculation if any data is lost. At present, PLC is rarely used.

- **G.711u**

    ITU standard for H.323-compliant codecs. Uses the u-law for companding, the most frequently used method in the USA. PLC is an option.

- **G.711a**

    ITU standard for H.323-compliant codecs. Uses the A-law for companding, a popular standard in Europe. PLC is an option.

- **G.726**

    A waveform coder that uses Adaptive Differential Pulse Code Modulation (ADPCM). ADPCM is a variation of pulse code modulation (PCM), which only sends the difference between two adjacent samples, producing a lower bit rate.

- **G.729**

    High-performing codec; offers compression with high quality.

- **G.723.1-MPMLQ**

    Uses the multi-pulse maximum likelihood quantization (MPMLQ) impression algorithm.

- **G.723.1-ACELP**

    Uses the conjugate structure algebraic code excited linear predictive compression (ACELP) algorithm.

- **AMR**

    A patented audio data compression scheme optimized for speech coding. Widely used in GSM and UMTS. Both AMR Narrow Band (NB) and Wide Band (WB) are supported.

Table 10-7 describes the codec types supported by IxChariot, along with associated parameters used to calculate packetization delay, as per ITU Recommendation G.108 p.23.

Table 10-7.   Codec Types

Codec	Data Rate	Default Datagram Size	Frame Size	Jitter Delay	Look Ahead Delay	Theoretical Max MOS	Pre-6.0 Packetization Delay
G.711u	64 kbps	20	1	2 datagrams (40 ms)	0 ms	4.41	1.0 ms
G.711a	64 kbps	20	1	2 datagrams (40 ms)	0 ms	4.40	1.0 ms
G.726	32 kbps	20	10	2 datagrams 40 ms	0.125 ms	4.22	1.25 ms
G.729	8 kbps	20	10	2 datagrams (40 ms)	5.0 ms	4.07	25.0 ms.
G.723.1 MPMLQ	6.3 kbps	30	30	2 datagrams (60 ms)	7.5 ms	3.87	67.5 ms
G.723.1 ACELP	5.3 kbps	30	30	2 datagrams (60 ms)	7.5 ms	3.69	67.5 ms
AMR NB	4.75 kbps[a]	14	20	2 datagrams (40 ms)	5 ms	3.16	N/A
	5.15 kbps	15	20	2 datagrams (40 ms)	5 ms	3.40	N/A
	5.9 kbps	16	20	2 datagrams (40 ms)	5 ms	3.55	N/A
	6.7 kbps	18	20	2 datagrams (40 ms)	5 ms	3.71	N/A
	7.4 kbps	20	20	2 datagrams (40 ms)	5 ms	3.82	N/A
	7.95 kbps	22	20	2 datagrams (40 ms)	5 ms	3.87	N/A
	10.2 kbps	27	20	2 datagrams (40 ms)	5 ms	4.07	N/A
	12.2 kbps	32	20	2 datagrams (40 ms)	0 ms	4.18	N/A
AMR WB	6.6 kbps	18	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	8.85 kbps	24	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	12.65 kbps	33	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A

Table 10-7.   Codec Types (Continued)

Codec	Data Rate	Default Datagram Size	Frame Size	Jitter Delay	Look Ahead Delay	Theoretical Max MOS	Pre-6.0 Packetization Delay
	14.12 kbps	37	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	15.85 kbps	41	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	18.25 kbps	47	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	19.85 kbps	51	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	23.05 kbps	59	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A
	23.85 kbps	61	20	2 datagrams (40 ms)	5 ms	MOS Score not supported	N/A

a.
The network bit rates (as measured by IxChariot) will be higher than the codec bit rates (datagram size will be larger than bitrate*delay between datagrams).This is due to the AMR header being added before the actual AMR voice data. As an example, let's take the AMR NB data rate of 4.75 kbps. Given that the delay between datagrams is 20 ms and the datagram size is 14 bits, the resulting actual bit rate measured by IxChariot is 5.6 kbps.

Silence Suppression

Related Topics
Adding or Editing a VoIP Endpoint Pair on page 10-38
Voice over IP Testing on page 10-34

When you are creating a pair for a voice over IP test, you have the option to use silence suppression in the voice over IP call traffic IxChariot sends on the network. The telecommunications industry is still developing standards for silence suppression. IxChariot's silence suppression option means that no data is sent on the network during periods of call silence (that is, when no one is "talking").

If you activate silence suppression in the Add/Edit VoIP Endpoint Pair dialog box, you should also choose a *voice activity rate*. The activity rate is an indicator of the percentage of time during the call that talking is occurring. In other words, when you set an activity rate, you are determining how the call "looks" on the network—how much actual voice data it contains. By default, IxChariot uses a 50% activity rate, but you can set any rate from 1% to 100%.

Silence suppression will have an effect on test results. If a pair has silence suppression enabled, the throughput and amount of data sent and received will decrease. For example, a G.711u call with an activity rate of 50% should expect to see an average throughput of 32 kbps over the period of the test because the G.711 codec sends at a data rate of 64 kbps. Likewise, the total amount of data sent and received should be 50% less than a call without silence suppression enabled. The effective bandwidth consumption, including protocol headers would be 40 kbps, assuming a 160-byte buffer size and 40 bytes (RTP=12, UDP=8, IP=20) of protocol header overhead.

If the first datagram sent after a period of silence is lost, the delay from the period of silence is factored into the RFC 1889 jitter and maximum delay variation jitter measurements. This could help explain spikes that you may see in jitter results for pairs with silence suppression enabled.

For the AMR codec, inactivity periods are implemented as Silence Descriptor (SID) frames. This requires that Endpoint 1 be version 7.10 SP4 or newer.

> **NOTE**: The generated pattern of traffic and silence periods is absolutely random.

One-Way Delay

Related Topics
Clock synchronization on page 7-6
The One-Way Delay Tab on page 11-27
VoIP Score Calculation on page 10-48

Delay is a key factor in determining voice over IP call quality. IxChariot measures the delay between Endpoint 1 and Endpoint 2 in a single direction (one-way or network delay), as well as end-to-end delay, which includes delay factors such as the codec used, jitter buffers, and fixed delays. See *VoIP Score Calculation* on page 10-48 for more information.

During the Initialization phase of an IxChariot test, E2 tries to get enough clock samples from E1 to determine the round-trip delay time and synchronize the endpoints' high-precision clocks. If E2 does not receive enough clock value replies from E1, clock synchronization times out, in which case "n/a" appears in the results for one-way (network) delay and clock error at the conclusion of the test. You'll notice that your test takes a slightly longer time to initialize the first time you run it due to clock sync.

Throughout the test, the endpoints continue to synchronize their clocks, generating a very small amount of network traffic. And once clock synchronization is initiated between two endpoints, they may continue to exchange clock sync packets periodically for up to 8 hours to maintain clock synchronization, even after completion of the last RTP test involving these endpoints.

Endpoints and Clock Synchronization

Clock synchronization for the purposes of taking one-way delay measurements is available for all endpoints. One-way delay is calculated for RTP datagrams only. When E1 sends RTP datagrams to E2, the datagrams include a timestamp indicating the time they were sent. E2 then calculates one-way network delay by subtracting the datagram's SEND time from the RECEIVE time.

One-Way Delay Results

Results for one-way delay indicate the type of clock used for timing measurements in the final column of results. One-way delay results also include **Estimated Clock Error** and **Maximum Clock Error** to provide estimations of the reliability of one-way network delay measurements.

Asymmetric network links yield a different delay value in each direction between endpoints. Use the Estimated Clock Error results for links known to be symmetric. The Maximum Clock Error should be considered when measuring VoIP latency on asymmetric links, although it is not to be interpreted as an absolute maximum error. On asymmetric links, the actual clock error falls somewhere between the Estimated Clock Error and the Maximum Clock Error.

An Estimated Clock Error of more than 10 ms should be considered too large for reliable one-way delay measurements. The causes for this and more information about one-way delay data, are discussed in *The One-Way Delay Tab* on page 11-27.

> **Note:** If the Endpoint 1 clock is running ahead of the Endpoint 2 clock in a test pair, the one-way delay value will be reported as zero throughout the test.

VoIP Score Calculation

Related Topics
One-Way (Network) Delay on page 11-43
The VoIP Tab on page 11-25
Codec Types on page 10-43
Jitter Buffers on page 10-52
The Lost Data Tab on page 11-31

Treating each endpoint pair as a separate voice over IP "call," IxChariot gives an indication of the relative quality of each call made during a test on your network. Ixia uses a modified version of the ITU G.107 standard E-Model equation to calculate a Mean Opinion Score (MOS) estimate for each endpoint pair.

The *E-Model*, developed by the European Telecommunications Standards Institute (ETSI), has become ITU standard G.107. This algorithm is meant to evaluate the quality of a transmission by factoring in the "mouth-to-ear" characteristics of a speech path. It calculates an *R-value*, which correlates directly with the MOS estimate.

IxChariot modifies the E-model slightly and uses the following factors to calculate the R-value and the MOS estimate:

- **One-Way (Network) Delay**

  Similar to the propagation delay; only the delay factors associated with the network (the "wire") itself are included. IxChariot measures this by synchronizing the endpoints' timers and determining delay in a single direction. Refer to *One-Way Delay* on page 10-47 for more information.

- **End-to-End Delay**

  Latency as measured by adding the following factors:

Table 10-8.   Delay Types

Delay Type	How Calculated
One-way (network) delay	Datagram's RTP timestamp subtracted from time it was received by Endpoint 2. See above.
Packetization delay	Fixed; dependent on codec selected.
Jitter buffer delay	Fixed; dependent on type and size of jitter buffer configured by user
Additional fixed delay	Fixed; user-configured

Like one-way delay, end-to-end delay is in a single direction between the endpoints, but it extends to the VoIP hardware to include all delay factors. One-way delay only measures network delay.

- **Packetization Delay**

  Delay associated with conversion of the voice signal from analog to digital is factored into the score calculation. Varies according to the codec you are using. Refer to *Codec Types* on page 10-43 for more information.

- **Jitter Buffer Delay**

  Delay associated with jitter buffers, which work to reduce variability in datagram inter-arrival times. Refer to *Jitter Buffers* on page 10-52 for more information.

- **Additional Fixed Delay**

  Delay factor from a known, constant source of delay. Necessary for accuracy if your test is emulating hardware associated with a fixed delay factor.

- **Data Loss**

  Total number of datagrams lost. When a datagram is lost, you can lose an entire syllable, and the more datagrams that are lost consecutively, the more the clarity suffers. IxChariot factors in lost data and also includes the amount of consecutive datagram loss that was measured. Refer to *The Lost Data Tab* on page 11-31 for more information. In addition, if you have **packet loss concealment** (PLC) enabled for the G.711 codecs, a delay factor associated with PLC buffering is added. Refer to *Codec Types* on page 10-43 for information about PLC.

- **Jitter Buffer Lost Datagrams**

  Number of datagrams lost due to jitter buffer overruns and underruns. Refer to *Jitter Buffers* on page 10-52 for more information.

A MOS of 5 is excellent; a MOS of 1 is unacceptably bad. The following table (taken from ITU G.107) summarizes the relationship between the MOS and user satisfaction:

Table 10-9. Mean Opinion Scores (MOS)

Mean Opinion Score (lower limit)	User Satisfaction
4.34	Very satisfied
4.03	Satisfied
3.60	Some users dissatisfied
3.10	Many users dissatisfied
2.58	Nearly all users dissatisfied

MOS Score and R-value statistics are not reported for the following AMR codecs:

- AMR NB (if Endpoint 2 is older than 7.10 SP4)
- AMR WB

Understanding Jitter Measurements

Related Topics
Jitter and Delay Variation on page 10-52
The Jitter Tab on page 11-29

IxChariot helps you determine the quality of a multimedia transmission or voice over IP call using two related measurements of jitter. While **RFC 1889 jitter** (calculated for all RTP pairs) shows mean statistical deviance of packet inter-arrival times over a period of time, **delay variation jitter** shows the number of packets (expressed as a percent of the total number of packets sent) that experienced delays of varying durations. The **jitter (delay variation) maximum** reveals when the greatest variation in delay seen for a timing record occurred during the test.

When a datagram is sent, the sender gives it a timestamp. When it is received, the receiver adds another timestamp. These two timestamps are used to calculate the datagram's transit time. If the transit times for datagrams within the same test are

different, the test contains jitter. In a video application, it manifests itself as a flickering image, while in a telephone call, its effect may be similar to the effect of packet loss; some words may be missing or garbled.

The amount of jitter in a test depends on the degree of difference between the datagrams' transit times. If the transit time for all datagrams is the same (no matter how long it took for the datagrams to arrive), the test contains no jitter. If the transit times differ slightly, the test contains some jitter. Jitter values in excess of 50 ms probably indicate poor call quality.

Jitter statistics let you see a short-term measurement of network congestion and can also show the effects of queuing within the network. The jitter value is reset for each timing record, so the jitter statistic for a specific timing record shows the jitter for that timing record only.

Almost all data transfers experience jitter, but it isn't necessarily a problem. Jitter occurs in several patterns. If the delay time for each datagram steadily increases, jitter values increase and the throughput decreases. But it is also possible for jitter to increase while throughput remains constant. In this case, the delay variation fluctuates widely, which could mean poor performance for delay-sensitive applications. Finally, "bursty" jitter—occurring in bursts of datagrams—has the most significant effect on call quality in voice over IP transmissions. If jitter occurs in bursts, it can lead to data loss and a degradation of call clarity. IxChariot measures this "bursty" quality; refer to *The Consecutive Lost Datagrams Tab* on page 11-33 for more information.

Various elements in the network can cause jitter. When troubleshooting jitter, first run a test to benchmark the amount of jitter received when using the TCP/IP stack. Then run tests adding various network elements (such as routers) to determine the element causing jitter.

Another cause of jitter is router queuing algorithms. The combination of the queuing algorithm in the router and the network configuration could cause jitter. To troubleshoot, try running tests using different queuing algorithms on the router.

RFC 1889 Jitter

When calculated according to the specification for RTP, jitter (J) is defined as the mean deviation (smoothed absolute value) of the difference (D) in datagram spacing at the receiver compared to the sender for a pair of datagrams. As shown below, this is equivalent to the difference in the "relative transit time" for the two datagrams; the relative transit time is the difference between a datagram's RTP timestamp and the receiver's clock at the time of arrival, measured in the same units. If Si is the RTP timestamp from datagram I, and Ri is the time of arrival in RTP timestamp units for Datagram I, then for two datagrams (I and j), D may be expressed as follows:

```
D(i,j)=(Rj-Ri)-(Sj-Si)=(Rj-Sj)-(Ri-Si)
```

The inter-arrival jitter is the jitter calculated continuously as each datagram (I) is received from the source. The jitter is calculated according to the formula defined in RFC 1889:

$$J=J+(|D(I-1,I)|-J)/16$$

Jitter is measured in timestamp units and is expressed as an unsigned integer. Whenever the endpoint creates a timing record, the current value of J is sampled.

Jitter and Delay Variation

Related Topics

Understanding Jitter Measurements on page 10-50
The Jitter Tab on page 11-29

When running a streaming script with the RTP protocol, the endpoints calculate the amount of **RFC 1889 jitter** and **delay variation jitter** for each timing record (delay variation jitter is only available with the VoIP Test Module). RFC 1889 jitter is calculated according to the RTP specification; it is the statistical variance of the datagram inter-arrival time expressed as a mean deviation for a single pair. Delay variation jitter, on the other hand, is a more granular distribution, indicating the differences in arrival times among all datagrams for a particular endpoint pair. RFC 1889 jitter is measured for each timing record; delay variation is measured for each datagram, but it is reported per timing record.

If only one datagram is sent in a timing record, the RFC 1889 jitter is zero. It is reset to zero at the beginning of each timing record.

The jitter (delay variation) histogram organizes delay variations into a series of ranges and then shows the number of datagrams that fell into each range, expressed as a percentage of the total number of datagrams sent. A histogram is the only graph type available for jitter (delay variation); no tables are shown. The ranges shown can be configured; see *Result Ranges Tab* on page 7-16. The jitter (delay variation) maximum is graphed differently, per timing record, so that you can see when the greatest jitter occurred during the test. All graphs are available for jitter (delay variation) maximum except the pie graph.

For more information on RFC 1889 jitter and delay variation, see *Understanding Jitter Measurements* on page 10-50.

For explanations of the results you see on the Jitter Tab, see *The Jitter (Delay Variation) Tab* on page 11-30.

Jitter Buffers

Related Topics

Understanding Jitter Measurements on page 10-50
The Jitter Tab on page 11-29
The VoIP Tab on page 11-25

To minimize call disruptions from delay and jitter, VoIP phones and gateways typically have jitter buffers. A jitter buffer can be either frame-based or absolute: a frame-based jitter buffer will hold a given number of voice datagrams, while an

absolute jitter buffer is based on time. For example, a frame-based jitter buffer might hold 2 datagrams, buffering them until a segment of the voice transmission can be reassembled to reduce inter-arrival time variability. An absolute jitter buffer, on the other hand, might be set to 43 ms, and given a typical 20 ms speech frame size, could hold 2 speech frames and allow for an extra 3 ms of variability.

Jitter buffers may also be either static or dynamic. Each buffer implementation has its strengths. But buffering adds delay while smoothing out variability; therefore, adding delay can offset the positive effects of smoothing out jitter. One goal of VoIP network tuning must be to minimize jitter buffer sizes while maintaining call quality.

You can enable or disable jitter buffering for your VoIP tests. When enabled, IxChariot's VoIP support emulates the effects of jitter buffering on your network. The buffers you can configure may be based either on time or on number of datagrams. IxChariot takes the jitter buffer into account when determining the MOS estimate and the number of lost datagrams, and reports Jitter Buffer Lost Datagram statistics in the Test window.

Configure your jitter buffers as you create VoIP pairs, by clicking **Add VoIP Pair** on the Edit menu in the Test window. The supported range of values is 10-1600 ms for an absolute jitter buffer, and 1-8 for a buffer configured in number of voice datagrams. Refer to *Adding or Editing a VoIP Endpoint Pair* on page 10-38 for information on configuring a jitter buffer.

Datagrams that are not contained by the jitter buffer due to excessive delay variation would be lost to the application. *Jitter buffer lost datagrams* include datagrams with delay too great for the jitter buffer you set. Datagrams may be lost due to jitter buffer overruns, or datagrams that had a delay variation greater than the jitter buffer size. Overruns include datagrams that were delayed too long for the jitter buffer you configured; for example, a datagram with a delay of 50 ms would not be contained in a jitter buffer set to 40 ms. Or if 5 datagrams were delayed sequentially, an absolute jitter buffer set to 2 datagrams would not pass three datagrams along to the application. Jitter buffer underruns are also included: datagrams that arrive too quickly while the jitter buffer is still full.

> **Note:** IxChariot's implementation of the jitter buffer does not smooth out jitter. Instead it provides a more accurate MOS estimate by better accounting for datagrams that would have been lost due to underrun or overrun of the jitter buffer (see results for Jitter Buffer Lost Datagrams). In addition, the delay incurred by using a jitter buffer is factored into the MOS. For more information about how IxChariot assesses call quality, see *VoIP Score Calculation* on page 10-48.

Video Stream Testing

Related Topics
Adding or Editing a Video Endpoint Pair on page 10-54
Adding or Editing a Video Multicast Group on page 10-58
Media Delivery Index (MDI) Calculation on page 10-63
The Video Tab on page 11-34

IxChariot provides a Video Endpoint pair for use in testing streaming video traffic and gathering measurements of that traffic. With Video Endpoint pairs, you can generate streaming video for both IP unicast and multicast transmissions. These streams are run over UDP and RTP, and support both IPv4 and IPv6.

Adding or Editing a Video Endpoint Pair

Related Topics
Adding or Editing a Video Multicast Group on page 10-58
Media Delivery Index (MDI) Calculation on page 10-63
The Video Tab on page 11-34
Selecting a QoS Template for Test Application Traffic on page 9-12

To create a streaming video test, select **Add Video Pair** from the Edit menu. IxChariot displays the Add a Video Endpoint Pair dialog, as shown in Figure 10-20. To edit a streaming video test, double-click the row in the test window; In this case, IxChariot displays the current data in the Edit a Video Endpoint Pair dialog.

Figure 10-20.   Adding a Video endpoint pair

The Video Pair parameters are described below: *Basic Parameters* on page 10-55, *Advanced Parameters* on page 10-56, and *Management Parameters* on page 10-57. The dialog opens with the Basic parameters displayed and both the Advanced parameters and Management parameters hidden.

For streamimg video testing, by default, Endpoint 1 executes a test script, receives results from Endpoint 2, and reports them to the Console over the same network segment. Depending on the size of your test or the reliability of your test network, you may want to isolate test setup and results traffic from actual test traffic; doing so can increase the accuracy of results or decrease the likelihood that setup data will be lost, causing the test to fail.

If you want to use an alternate network for setup/results reporting, or a different network protocol or service quality between the Console and Endpoint 1 and between the endpoints, fill in the values for *Use Endpoint 1 address as management address* and *Use Endpoint 2 address as management address*, as described in *Management Parameters* on page 10-57. For example, you might use TCP to connect from the Console to E1, yet run a UDP test between Endpoint 1 and Endpoint 2. But keep in mind that IxChariot only uses connection-oriented protocols for setup communications between the Console and Endpoint 1, and that TCP is required for setup communications between Endpoint 1 and Endpoint 2.

Basic Parameters

Specify the basic configuration parameters, as described below.

*   **Pair Comment (optional)**

    A descriptive word or phrase that lets you easily identify each pair in the Test window.

*   **Endpoint 1 address**

    The network address of a computer playing the role of Endpoint 1 in an IxChariot test. The IxChariot Console contacts Endpoint 1 computers directly, sending them the test setup information you entered. Endpoint 1 acts as the sender of the streaming video test data. Enter a network address that matches the protocol you are using (IPv4 or IPv6).

*   **Endpoint 2 address**

    A computer playing the role of Endpoint 2 in an IxChariot test. Endpoint 1 computers contact the Endpoint 2 computers with test setup information. Endpoint 2 is the receiver of streaming data in a streaming video test.

*   **Network protocol**

    The network protocol used to send the test traffic over the network. The valid choices are: UDP, UDP-IPv6, RTP, and RTP-IPv6.

*   **Service Quality**

    The quality of service (QoS) template to use in tests with these pairs of endpoints. If you have defined QoS templates, select the template that you want to use in this test. Optional field.

*   **Video - Encoding**

    The Video encoding algorithm that the video endpoint pairs will simulate. Valid choices ar: MPEG2 and Custom. Note that the encoding method impacts the size of the UDP/RTP media frames that are generated. For example, MPEG2 media frames are 188 bytes each.

- **Video - RTP Payload Type**

  The RTP Payload type used by the selected encoding.

- **Video - Media Frames per DG**

  The number of media frames per datagram. All signed integers values are valid. The default value for Ethernet is 7.

- **Video - Media Frame Size**

  The media frame size for the selected encoding.

- **Video - Bitrate**

  The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 3 and 15 Mbps. The default is 3.75 Mbps.

  You can set the bitrate units to Mbps (1024 kilobytes), Kbps (1024 bytes), or kbps (1000 bytes).

Advanced Parameters

Click the Advanced button to display or hide the advanced parameters.

- **Timing Record duration**

  The approximate duration of a timing record. This is an indication only, rather than an exact setting. Timing records in IxChariot are based on the volume of data transferred, rather than on timing measurements. If packet loss occurs, the timing record duration will be larger.

- **Number of Timing Records**

  The number of timing records to generate for a test.

  The default value is 50, and the valid range of values is from 1 to 2,147,483,647.

- **Source port number**

  The UDP source port for the test traffic. The default is Auto.

- **Destination port number**

  The UDP destination port for the test traffic. The default is Auto.

- **Initial delay - Value**

  This parameter adds a SLEEP command at the beginning of the application script. It is recommended for emulating the effects of multiple users accessing the network. Select "Constant value" if you want to set an exact value for the delay. With the other options, IxChariot generates values for the delay corresponding to the selected mathematical distribution (Uniform, Normal, Poisson, Exponential). In this case, you can set the lower and upper limits for these distributions. The default is "Constant value."

- **Initial delay - Upper limit**

  This is either the value for the delay if you have chosen for "Constant value"; or the upper limit for the values generated by one of the mathematical distributions. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

- **Initial delay - Lower limit**

  This parameter is visible only if you selected one of the four mathematical distributions. It represents the lower limit for the values generated by the selected distribution. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

Management Parameters

Click the Management button to display or hide the management parameters.

- **Console to Endpoint 1 Management**

  - **Use Endpoint 1 address as management address**

    When this checkbox is selected, the Console will communicate with Endpoint 1 using the network address and protocol specified for Endpoint 1. If the Console needs to send test setup information (such as the address of Endpoint 2) to Endpoint 1 through an alternate network, clear the checkbox and enter the address of that network. You also clear this checkbox if you need to select a different protocol. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1. For RTP and UDP, the default is TCP.

    - If you are changing the network address at which Endpoint 1 contacts Endpoint 2 with setup information (see *Use Endpoint 2 address as management address* on page 10-57), you will probably need to change the address at which the Console contacts Endpoint 1 as well.

    - In tests running streaming scripts, be aware that Endpoint 2 sends results back to Endpoint 1. If the network is unreliable, you should specify a more reliable address for Endpoint 1 here. It will also be used by Endpoint 2 for results flows.

  - **Network Protocol**

    The protocol that the Console will use to contact Endpoint 1 for the purpose of setup.

  - **Service Quality (optional)**

    If a service quality is required by the network protocol, enter or select a value defined on the endpoint computers. Any quality of service templates you've configured are available in the list. The service quality values you enter are remembered in the file `servqual.dat`. Service quality may be selected for both TCP-IPv4 and TCP-IPv6 traffic. Only Ixia and Linux endpoints support IPv6 service quality. Refer to Chapter 9, *Quality of Service Testing* for detailed information.

- **Endpoint 1 to Endpoint 2 Management**

  - **Use Endpoint 2 address as management address**

    The network address and protocol used for test setup and results flows will be the same as those you specified when you created the endpoint pair. If Endpoint 1 needs to send setup information, such as the application script, to Endpoint 2 through an alternate network, clear the checkbox and spec-

ify that network address. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1.

Often, a more reliable network connection can be made between the endpoints at different addresses to ensure that setup and reporting information is received. In tests running streaming scripts, Endpoint 2 sends results back to Endpoint 1. If the network is unreliable, you should specify a more reliable address for Endpoint 2 here to ensure that the endpoints use a reliable network for test setup and reporting.

Adding or Editing a Video Multicast Group

Related Topics
Emulating IP Multicast Applications on page 10-9
Selecting a QoS Template for Test Application Traffic on page 9-12
Adding or Editing a Video Endpoint Pair on page 10-54
Media Delivery Index (MDI) Calculation on page 10-63
The Video Tab on page 11-34

To create a test emulating a streaming video multicast application, first create a video multicast group; select **Add Video Multicast Group** from the Edit menu. IxChariot displays the Add a Video Multicast Group dialog, as shown Figure 10-21. To edit a video multicast group, double-click the row in the test window; In this case, IxChariot displays the current data in the Edit a Video Multicast Group dialog.

Figure 10-21.   Adding a Video Multicast Group

The Video Multicast Group parameters are described below: *Basic Parameters* on page 10-59, *Management Parameters* on page 10-60, and *Video Parameters* on page 10-62. The dialog opens with the Basic parameters displayed and both the Management parameters and Video parameters hidden.

Video multicast group members are the Endpoint 2 computers designated as receivers of streaming video from Endpoint 1 in IxChariot tests. When paired with the sending (Endpoint 1) computer, they are called *video multicast pairs*.

By default, Endpoint 1 executes a test script, collects results, and reports them to the Console over the same network segment. Depending on the size of your test or the reliability of your test network, you may want to isolate test setup and results traffic from actual test traffic; doing so can increase the accuracy of results or decrease the likelihood that setup data will be lost, causing the test to fail.

If you want to use an alternate network for setup/results reporting, or a different network protocol or service quality between the Console and Endpoint 1 and between the endpoints, refer to *Management Parameters* on page 10-60. For example, you might use TCP to connect from the Console to Endpoint 1, yet run a UDP test between Endpoint 1 and Endpoint 2. But keep in mind that IxChariot only uses connection-oriented protocols for setup communications between the Console and Endpoint 1, and that TCP is required for setup communications between Endpoint 1 and Endpoint 2.

Basic Parameters

Specify the basic configuration parameters, as described below.

- **Group Name**

  A name that is unique within the test. If you do not enter a group name in this field, IxChariot creates a group name that is a combination of the IP address and port of the video multicast group.

- **Group Comment**

  A descriptive word or phrase that helps you easily identify each video multi-cast group in the Test window. This field is optional.

- **Endpoint 1 to Multicast Group**

  - **Multicast Address**

    For IPv4, the Class D IP address to use for the IP Multicast test. Valid IP Multicast addresses are 224.0.0.0 through 239.255.255.255. We recom-mend using addresses beginning with 225.0.0.0 or higher because many addresses beginning with 224 are reserved for router usage.

For IPv6, multicast addresses have a prefix of `FF00::/8`. Within the reserved multicast address range of `FF00::` to `FF0F::`, the following addresses are assigned to identify specific functions:

Table 10-10. IPv6 Multicast Address Usage

Range	Usage
FF01::1	All nodes within the node-local scope.
FF02::1	All nodes on the local link.
FF01::2	All routers within the node-local scope.
FF02::2	All routers on the link-local scope.
FF05::2	All routers in the site.
FF02::1:FFXX:XXXX	Solicited-node multicast address, where XX:XXXX represents the last 24 bits of the IPv6 address of the node.

Although IxChariot verifies that the IP Multicast address you enter is within the required range, it does not verify the reserved/unreserved status of the address you enter. See *Emulating IP Multicast Applications* on page 10-9 for more information on multicast addresses.

• **Multicast Port**

The port number that the video multicast group will use. The video multicast port must uniquely identify the multicast group. You can enter values in the range from 1 to 65535. Avoid using well-known port numbers.

• **Endpoint 1 network address**

The "From" address for this video multicast group. Enter a DNS hostname, an IPv4 address, or an IPv6 address. Endpoint 1 acts as the video multicast sender.

• **Endpoint 2 – Multicast Group Members**

Enter (or select from the list) a DNS hostname, an IPv4 address, or an IPv6 address. Then click **Add** to add the endpoint to the video multicast group. To delete a video multicast group member, select the group member in the list and click **Delete**.

• **Network Protocol**

The protocol to use when sending the streaming video multicast data. The valid choices are RTP, RTP-IPv6, UDP, and UDP-IPv6.

• **Service Quality (optional)**

The quality of service (QoS) to use in this streaming video multicast test. If you have defined QoS templates, select one from the list.

Management Parameters

To display the Management parameters, click the MANAGEMENT button. The dialog expands to display the parameters described below. You use these parameters if you want to use an alternate network for setup/results reporting. In this case, you enter a different network address at which the Console will "know"

Endpoint 1 so that you can use separate networks for setup communications between the Console and Endpoint1 and for actual test traffic.

- **Endpoint 1**

  - **Use Endpoint 1 address as management address**

    When this checkbox is selected, the Console will communicate with Endpoint 1 using the network address and protocol specified for Endpoint 1. If the Console needs to send test setup information (such as the address of Endpoint 2) to Endpoint 1 through an alternate network, clear the checkbox and enter the address of that network. You also clear this checkbox if you need to select a different protocol. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1. For RTP and UDP, the default is TCP.

    - If you are changing the network address at which Endpoint 1 contacts Endpoint 2 with setup information (see *Use Endpoint 2 address as management address* on page 10-61), you will probably need to change the address at which the Console contacts Endpoint 1 as well.

    - In tests running streaming scripts, be aware that Endpoint 2 sends results back to Endpoint 1. If the network is unreliable, you should specify a more reliable address for Endpoint 1 here. It will also be used by Endpoint 2 for results flows.

  - **Network Protocol**

    The protocol that the Console will use to contact Endpoint 1 for the purpose of setup.

  - **Service Quality (optional)**

    If a service quality is required by the network protocol, enter or select a value defined on the endpoint computers. Any quality of service templates you've configured are available in the list. The service quality values you enter are remembered in the file `servqual.dat`. Service quality may be selected for both TCP-IPv4 and TCP-IPv6 traffic. Only Ixia and Linux endpoints support IPv6 service quality. Refer to Chapter 9, *Quality of Service Testing* for detailed information.

- **Endpoint 2**

  - **Use Endpoint 2 address as management address**

    The network address and protocol used for test setup and results flows will be the same as those you specified when you created the endpoint pair. If Endpoint 1 needs to send setup information, such as the application script, to Endpoint 2 through an alternate network, clear the checkbox and specify that network address. If you are using a datagram protocol and this box is checked, the Console uses the corresponding connection-oriented protocol for its connection to Endpoint 1.

    Often, a more reliable network connection can be made between the endpoints at different addresses to ensure that setup and reporting information is received. In tests running streaming scripts, Endpoint 2 sends results back to Endpoint 1. If the network is unreliable, you should specify a more

reliable address for Endpoint 2 here to ensure that the endpoints use a reliable network for test setup and reporting.

Video Parameters

To display the Video parameters, click the VIDEO button. The dialog expands to display the video configuration parameters described below.

• **Video - Encoding**

   The Video encoding algorithm that the video multicast pairs will simulate. Valid choices ar: MPEG2 and Custom. Note that the encoding method impacts the size of the UDP/RTP media frames that are generated. For example, MPEG2 media frames are 188 bytes each.

• **Video - RTP Payload Type**

   The RTP Payload type used by the selected encoding.

• **Video - Media Frames per DG**

   The number of media frames per datagram. All signed integers values are valid. The default value for Ethernet is 7.

• **Video - Media Frame Size**

   The media frame size for the selected encoding.

• **Video - Bitrate**

   The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 3 and 15 Mbps. The default is 3.75 Mbps.

   You can set the bitrate units to Mbps (1024 kilobytes), Kbps (1024 bytes), or kbps (1000 bytes).

• **Timing Record duration**

   The approximate duration of a timing record. This is an indication only, rather than an exact setting. Timing records in IxChariot are based on the volume of data transferred, rather than on timing measurements. If packet loss occurs, the timing record duration will be larger.

• **Number of Timing Records**

   The number of timing records to generate for a test.

   The default value is 50, and the valid range of values is from 1 to 2,147,483,647.

• **Source port number**

   The UDP source port for the test traffic. The default is Auto.

• **Initial delay - Value**

   This parameter adds a SLEEP command at the beginning of the application script. It is recommended for emulating the effects of multiple users accessing the network. Select "Constant value" if you want to set an exact value for the delay. With the other options, IxChariot generates values for the delay corresponding to the selected mathematical distribution (Uniform, Normal,

Poisson, Exponential). In this case, you can set the lower and upper limits for these distributions. The default is "Constant value."

- **Initial delay - Upper limit**

  This is either the value for the delay if you have chosen for "Constant value"; or the upper limit for the values generated by one of the mathematical distribution. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

- **Initial delay - Lower limit**

  This parameter is visible only if you selected one of the four mathematical distributions. It represents the lower limit for the values generated by the selected distribution. The valid values are all unsigned integers from 0 to 2147483647 ms. The default is 0.

Media Delivery Index (MDI) Calculation

Related Topics

Adding or Editing a Video Endpoint Pair on page 10-54
Adding or Editing a Video Multicast Group on page 10-58
The Video Tab on page 11-34

The Media Delivery Index (MDI) is a proposed metric, defined in RFC 4445, that can be used as a diagnostic tool or a quality indictor for monitoring the delivery of streaming media on a network. It specifically focuses on the measurement of packet jitter and packet loss in networks carrying streaming media, such as MPEG video, Voice over IP, and other information that is sensitive to arrival time and media loss. IxChariot collects and reports MDI statistics for all Video Pair and Multicast Video Group tests. (Refer to *The Video Tab* on page 11-34 for a description of the statistics reported in the Video tab of the test window.)

The two major factors impacting the quality of a video stream transmitted over an IP network are packet loss and packet jitter. Packet loss can be caused by many factors, including data corruption, insufficient bandwidth, and out-of-order packet delivery. Any packet loss will adversely affect the quality of the delivered video. Packet jitter causes buffer overflows and underflows, either of which will cause unacceptable time distortions in the video streams.

Packet Jitter

Packet jitter is a measure of the variation in arrival rates between individual packets in a media stream. Streaming media requires a consistent and predictable time delay between successive packets as they are received at a destination node. Variations in this interpacket delay causes packet jitter. If packets are delayed by the network, some packets will arrive in bursts with diminished interpacket delays while other packets will arrive with longer interpacket delays. Therefore, the node at the receiving end (the decoder) must buffer the video data to ensure that it can be displayed at its nominal rate. The size of the buffer determines the maximum amount of packet jitter than can be accommodated without experiencing buffer underrun or overrun.

Delay Factor

The MDI consists of two components: the Delay Factor (DF) and the Media Loss Rate (MLR).

The Delay Factor (DF) indicates the maximum difference between the arrival of streaming data and the drain of that data, as measured at the end of each media stream packet. The drain rate refers to the payload media rate. For example, for a typical 3.75 Mbps MPEG video stream, the drain rate is 3.75 Mbps—the rate at which the payload is displayed at a decoding node. The DF is computed at the time that each packet arrives at the IxChariot endpoint, and is recorded for each timing record. The default timing record duration is three seconds.

The DF is computed at the arrival time of each packet at the point of measurement. For IxChariot, the point of measurement is Endpoint 2. It is recorded at set time intervals, typically about one second. For IxChariot, it is measured when each timing record is generated.

If *X* is a virtual buffer where DF is measured,

```
X = |Bytes Received – Bytes Drained|
```

Then, *DF* is calculated as follows:

```
DF = [max(X) – min(X)] / media rate
```

where *media rate* is expressed in bytes per second and *max(X)* and *min(X)* are the maximum and minimum values measured in an interval.

The largest difference is recorded for all intervals in a measurement period. That is, DF is the maximum observed value of the flow rate imbalance over the calculated interval.

Media Loss Rate

The second component in the MDI is the Media Loss Rate (MLR).

The MLR is the count of lost or out-of-order packets, measured over a selected time interval (such as three seconds). That is,

```
MLR = (Packets Expected – Packets Received) / Interval
```

There may be zero or more streaming packets in a single IP packet. For example, it is common to carry seven 188 byte MPEG Transport Stream packets in an IP packet. In such a case, loss of a single IP packet would result in seven lost MPEG Transport Stream packets. Counting out-of-order packets is important because many streaming media applications do not attempt to reorder packets that are received out of order.

Media Delivery Index

The Media Delivery Index (MDI) combines the Delay Factor (DF) and the Media Loss Rate (MLR) values for presentation, and is expressed as:

```
DF:MLR
```

The DF provides a indication of the size of the buffer needed to accommodate the packet jitter in the network. The MLR gives an indication of the extent of media loss as the streaming media traverses the network.

Using IxChariot to Test for MDI

IxChariot streaming video tests can be used in pre-deployment studies for existing networks to verify that a network design can carry video traffic of varying loads at an acceptable level of quality. Testing can also focus on how well individual DUTs accommodate streaming video traffic under varying loads and conditions.

Figure 10-22 shows an example of an IxChariot video test in which the MDI is shown as *0:70.136*. This test indicates the absence of any measurable packet jitter in the network, but also indicates the presence of significant packet loss. This type of result would give a clear indication that video quality impairments are caused by packet loss rather than packet jitter.

Figure 10-22.   MDI Example
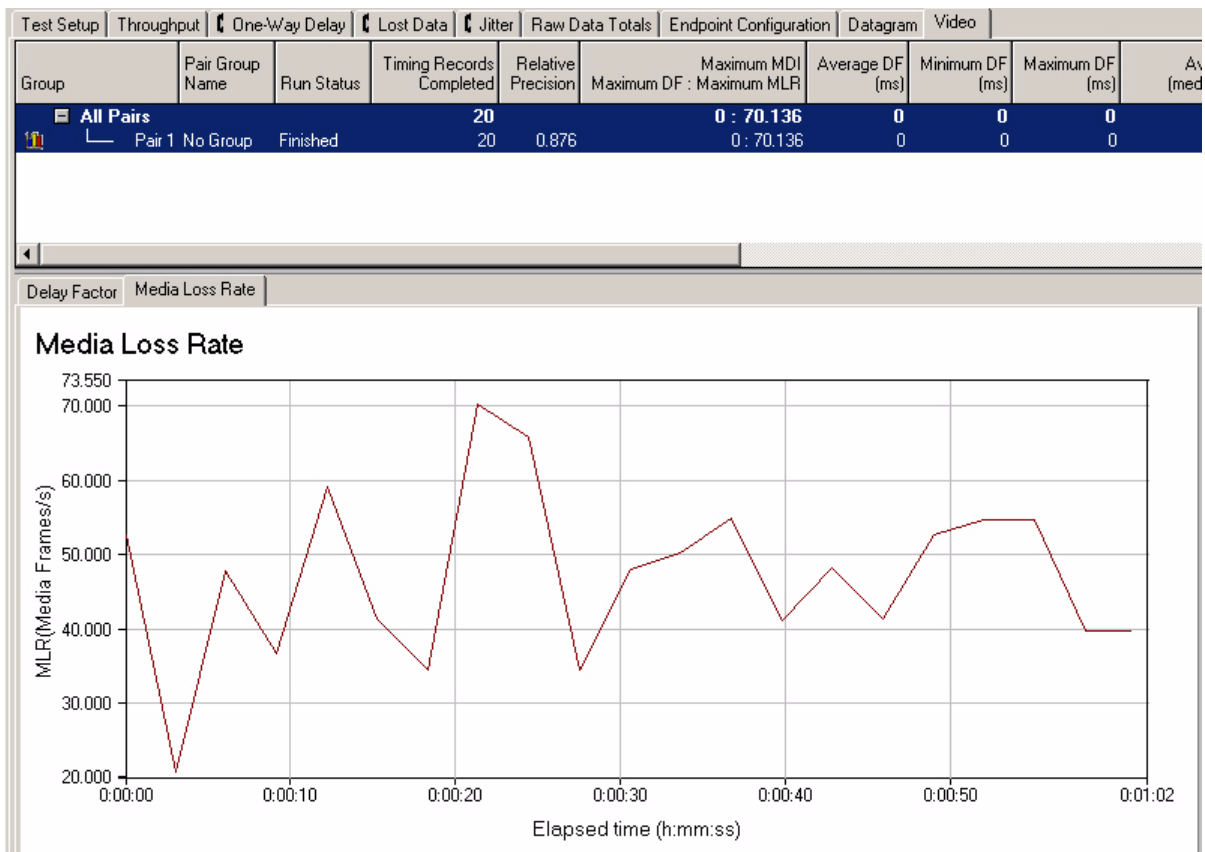
The MDI is especially effective when measurements are taken at intermediate points within a network. Differences in the MDI values at different points in a

network can help identify the sources of problems such as insufficient buffer space allocation, areas of local traffic congestion, and unsuitable QoS parameters. For example, if MDI values jump from very low to very high between successive hops in a path, there is clearly a problem in that network segment.

IPTV Testing

Beginning with release 6.50, IxChariot provides support for testing IPTV implementations, as described in the following topics:

About IPTV

IPTV (Internet Protocol Television) is an architecture for delivering a digital television service using the Internet Protocol over a closed network infrastructure. IPTV refers to both live television (multicasting) as well as stored video (Video on Demand). IxChariot emulates testing of live television only.

An IPTV subscriber requires a set-top box connected to a television to view and switch among the channels to which he or she subscribes. Video content is typically compressed using either an MPEG-2 or an MPEG-4 codec, and is sent in an MPEG transport stream. Live television uses IGMP version 2 for connecting to a multicast stream (television channel) and for changing from one multicast stream to another (television channel switching).

IPTV Emulation in IxChariot Tests

An IxChariot IPTV test includes at least one *receiver group*. Each receiver group contains one or more test pairs that—collectively—emulate the multicast transmission and receipt of one or more IPTV video streams.

An IPTV receiver group represents an individual subscriber to one or more IPTV television channels (such as BBC, ESPN, CNN, and so forth). Endpoint 2 (the subscriber) receives channel video streams that are multicast from Endpoint 1. Each receiver group includes one test pair for each channel that is being multicast.

Each *IPTV channel* object defined in an IxChariot test represents an individual IPTV television channel. An IPTV channel object is associated with a test only if it is added to a receiver group channel list. Each such channel is represented by a single pair in an IxChariot test.

IxChariot Endpoints Supported

Only the following Linux-based Performance Endpoints support IxChariot IPTV testing:

- Linux x86 32-Bit Performance Endpoint (requires Linux kernel 2.6.18)
- Linux PowerPC Performance Endpoint (requires Linux kernel 2.6.18)
- Ixia Load Module Performance Endpoint

IxChariot IPTV Test Configuration

You need to complete three main tasks to configure an IxChariot IPTV test:

1. Set up a test network:

   Your test network requires two connected subnets:

   - Streaming subnet – Endpoint 1 is in this subnet. This endpoint multicasts a video stream for each channel that you define.
   - Receiving subnet – Endpoint 2 is in this subnet. This endpoint receives the video stream for each channel and switches among the channels.

2. Configure IPTV Channels:

   Use the IxChariot IPTV Channels Editor to define each of the channels that you will use in the test. Refer to *Configuring IPTV Channels* on page 10-69 for detailed instructions.

3. Configure IPTV Receiver Groups:

   Create one or more receiver groups to receive the channel video streams. When you create a Receiver Group, IxChariot adds that group to the Test window. Each channel is represented by a pair within the group. Refer to *Creating IPTV Receiver Groups* on page 10-72 for detailed instructions.

Figure 10-23 on page 10-68 illustrates these steps.

In addition to the main tasks described above, you can also manage your IPTV test configurations as described in the following sections:

- *Setting IPTV Defaults* on page 10-77
- *Managing IPTV Channels* on page 10-79

Figure 10-23. IxChariot IPTV Test Configuration Process

Configuring IPTV Channels

To configure a new IPTV channel:
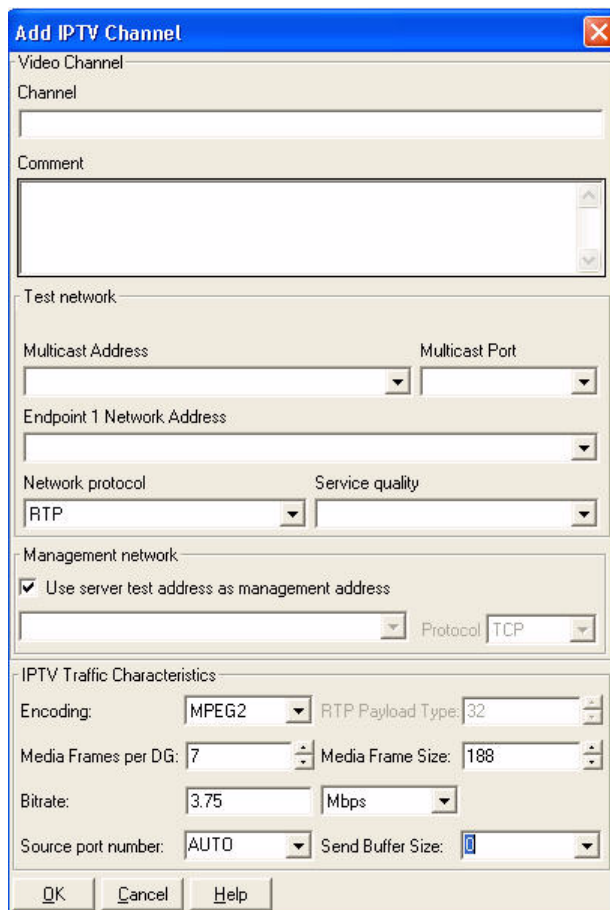
1. Select **Edit** > **IPTV Channels Editor…**

   The IPTV Channels Editor dialog opens.

2. Click **New**.

   The Add IPTV Channel dialog opens.

3. Configure the channel, using the parameters described in *IPTV Channels Parameters* on page 10-70. For example:

4. Click **OK**

IxChariot adds the new channel to the list in the IPTV Channels Editor Dialog.

IPTV Channels Parameters

describes all of the parameters in the IPTV Channels Editor Dialog.

Table 10-11. IPTV Channels Editor Dialog Parameters

Parameter	Description
Video Channel:	
Channel	A unique name for the channel. Typically, this will be a name that a subscriber will use when selecting a channel for viewing (*BBC*, for example). The channel name can be a maximum of 32 bytes in length.
Comment	Optional. A character string containing any desired notes or explanation for this channel. The comment can be a maximum of 64 bytes in length.
Test Network:	
Multicast Address	The multicast address and multicast port to which this channel will send its video stream.
Multicast Port	**Note:** • Each channel must use a unique multicast address. • Only one receiver can listen for the same multicast address on the same machine. Therefore, you can configure multiple receivers on one machine only if each receiver is configured with a unique channel list.
Endpoint 1 Network Address	The network address for the endpoint 1 computer or port. Endpoint 1 acts as the video multicast sender.
Network Protocol	The network protocol to use for this video stream. The options are RTP and UDP.
Service Quality	Optional. The name of the QoS template that will define the service quality for this video stream.
Management Network:	
Use server test address as management address	You can specify the management address in one of two ways: • Either check *Use server test address as management address*; or • specify a management address in the text box.
Protocol	The transport-layer protocol to use for the management network. The choices are TCP, TCP-IPv6, or SPX.
IPTV Traffic Characteristics:	

Table 10-11. IPTV Channels Editor Dialog Parameters (Continued)

Parameter	Description
Encoding	The desired video encoding for this video stream. The choices are MPEG2 or Custom. Note that the encoding method impacts the size of the UDP/ RTP media frames that are generated. For example, MPEG2 media frames are 188 bytes each.
RTP Payload Type	If you select *Custom* encoding, you must specify an RTP Payload type. (If you select *MPEG-2* encoding, this field will be read-only.)
Media Frames per DG	The number of media frames per datagram. All signed integer values are valid. The default value for Ethernet is 7.
Media Frame Size	The media frame size for the selected encoding.
Bitrate	The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 3 and 15 Mbps. The default is 3.75 Mbps.
Bitrate unit of measure	The unit of measure for the bit rate. You can set the bitrate units to Gbps (1024 megabytes), Mbps (1024 kilobytes), Kbps (1024 bytes), or kbps (1000 bits).
Source port number	The source port number for the video stream.
Send Buffer Size	The socket buffer size using for sending the data streams. The default value is determined by the operating system on which the Performance Endpoint is running. **NOTE:** If the SOCKET_SEND_BUFFER_SIZE keyword in the endpoint.ini file specifies the send buffer size, that value overrides the value specified in this dialog.

Creating IPTV Receiver Groups

To configure a new IPTV receiver group:

1. Select **Edit** > **Add Pair** > **Add IPTV Receiver Group…**

The Add IPTV Receiver Group dialog opens. For example:

2. Specify the following Receiver Group name and address information:

Parameter	Description
Receiver	Enter a unique name for the receiver. The receiver name can be a maximum of 32 bytes in length.
Comment	Optionally, enter a comment for this receiver.
Test Network Address	Enter the network address for the endpoint 2 computer or port. Endpoint 2 acts as the receiver of the video multicast.
Management Network Address	You can specify the management address in one of two ways: • Either check *Use server test address as management address*; or • specify a management address in the text box.

3. Enter the Receiver settings:

Parameter	Description
Number of Iterations	The number of times the receiver will switch channels (leave one channel and join another).
Receiver Buffer Size	The socket buffer size used for receiving the data streams. The default value is determined by the operating system on which the Performance Endpoint is running. **NOTE:** If the SOCKET_RECEIVE_BUFFER_SIZE keyword in the endpoint.ini file specifies the receive buffer size, that value overrides the value specified in this dialog.
Switch delay	The number of milliseconds that will elapse between leaving one channel and joining another.

4. Specify the channels that the receiver will switch between. You can do this in one of two ways:

   • To select all the channels that are defined for the test, check *Use all channels defined in test*. IxChariot will add all of the channels to the list.

   • To selectively add channels, select individual channels from the drop-down list, using the **Add** button to add them to the list.

5. Optionally, rearrange the list of channels:

   • Click **Random** to re-order the channels randomly.

   • Click the up and down arrow keys to reposition the channels.

   The final order in which the channels appear in the channel list defines the channel switching order in the test.
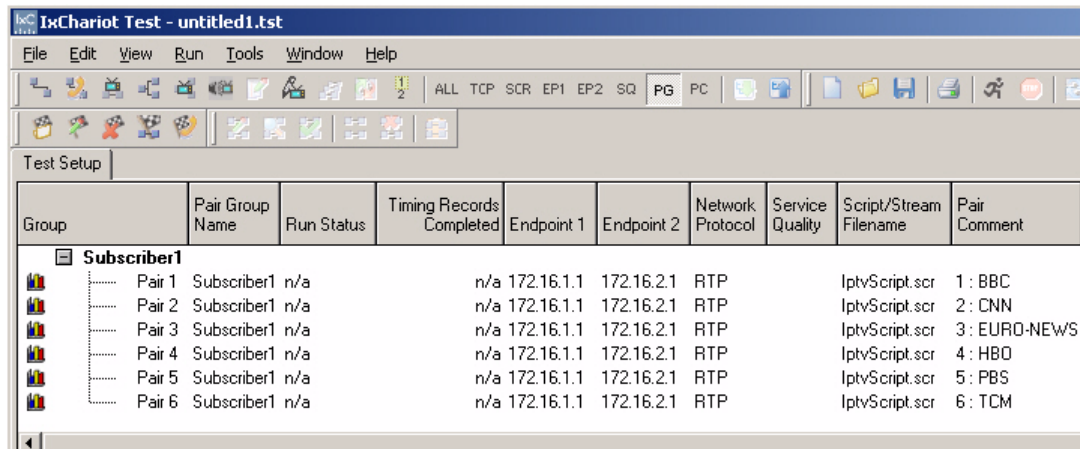
6. Specify the Selected channel control values:

Parameter	Description
Number of TR	The number of timing records that will be generated for each join/leave cycle.
TR Duration	The duration (in seconds) of each timing record.

7. Click **OK** to finalize the Receiver Group specification.

IxChariot opens the group in the Tests window, as shown in Figure 10-24 on page 10-74.

Figure 10-24. IPTV Test With One Receiver Group

Running the IPTV Test

When you complete a Receiver Group specification, IxChariot adds the group to the Tests window, as shown in Figure 10-24. You run the test the same as with any other type of test: select **Run** > **Run**, or click the Run button. The test executes in the following manner:

1. The order in which the pairs execute is specified by the index number shown in the Pair Comment field.

2. The first pair starts running. It simulates the streaming of video from a specific channel (Endpoint 1) to the subscriber (Endpoint 2).

3. The video stream continues for a duration calculated as (*Number of TR* multiplied by *TR Duration*).

4. At that point, the first pair stops and the second pair starts running. This simulates a channel switch. The time delay for the channel switch is specified in the *Switch delay* parameter in the Add IPTV Receiver Group dialog.

5. Each channel in the list executes in order.

6. The entire channel-switching sequence repeats for the number of iterations specified in the *Number of iterations* parameter in the Add IPTV Receiver Group dialog. For example, if you have six pairs and have specified five iterations, each of the six pairs runs five times.

Note: You can configure more than one receiver on the same machine as long as each receiver is configured with a unique channel list. However, if the same channel appears on more than one list, the test will fail with error CHR0493.

Figure 10-25 on page 10-75 shows sample output from an IPTV test.

Figure 10-25. Sample Output from an IPTV Test
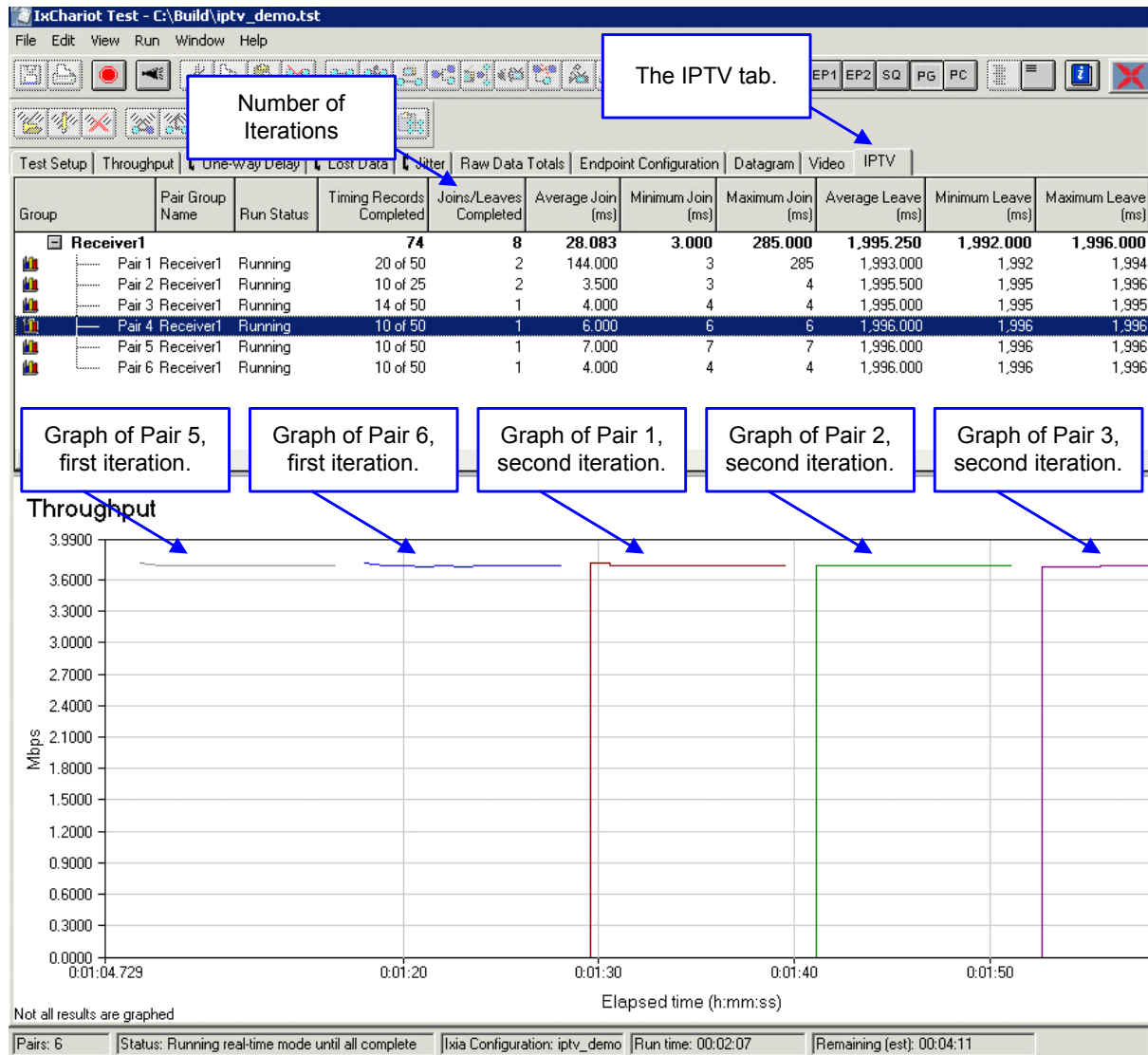
Figure 10-25 shows an IPTV test in progress. In this example, all six pairs have completed their first iteration, and the second iteration has begun (as shown in the Joins/Leaves Completed column). Each pair represents one channel. A channel switch is represented by one pair ending and the next pair beginning.

IPTV Test Statistics   IxChariot IPTV tests help evaluate channel switching performance by providing the follow report data:

- IGMP Join Latency: The time between a request to join a multicast group and the receipt of the first byte of data from that multicast group.

- IGMP Leave Latency: The time between a request to leave a multicast group and the receipt of the last byte of data from that multicast group

Join and leave latencies are particularly important, as they directly impact video quality, as illustrated in Figure 10-26. As this figure shows, the cumulative effect of join and leave latencies with each succeeding channel switch can result in excessive bandwidth use which results in a degradation of image and sound quality.

Figure 10-26. Impact of Join and Leave Latencies
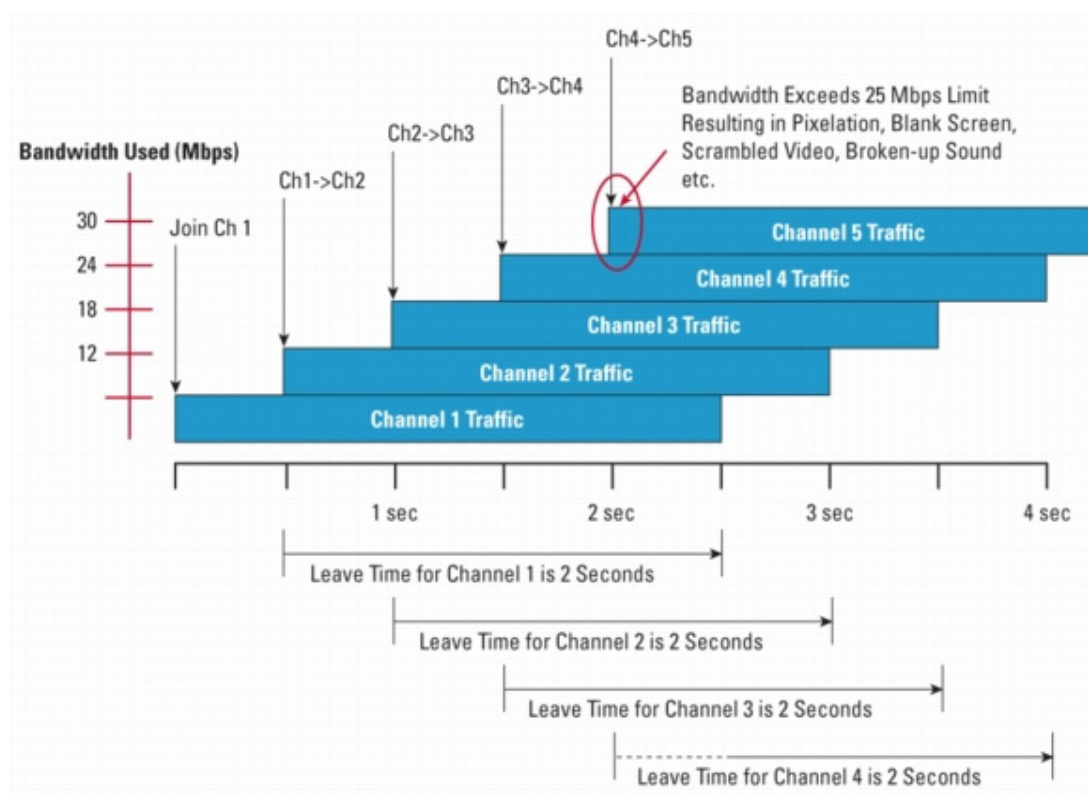
Join and Leave Statistics in the Main Test Window

As shown in Figure 10-25 on page 10-75, the IPTV tab in the test window displays the minimum, maximum, and average join and leave latencies, as well as the number of joins and leaves completed, for each pair. These statistics are described in Table 10-12 on page 10-77.

Table 10-12. Join and Leave Statistics in the Main Test Window

Column	Description
Joins/Leaves Completed	Real-time statistic that shows how many join/leave iterations were completed
Average Join	Real-time statistic that shows the average of the multicast joins until that moment
Minimum Join	Real-time statistic that shows the minimum multicast join delay until that moment
Maximum Join	Real-time statistic that shows the maximum multicast join delay until that moment
Average Leave	Real-time statistic that shows the average of the multicast leaves until that moment
Minimum Leave	Real-time statistic that shows the minimum multicast leave delay until that moment
Maximum Leave	Real-time statistic that shows the maximum multicast leave delay until that moment.

Join and Leave Statistics in the Timing Records Window

The IPTV tab in the Timing Records dialog displays the join and leave latency for each iteration of the selected pair. Note that the number of entries in this table does not equal the number of timing records. This is because the join and leave latency statistics are calculated for each join/leave iteration, and multiple timing records will have been generated for each such iteration.

Setting IPTV
Defaults

You can set the default values for the IPTV channels and IPTV receiver in the IPTV Defaults tab of the Change User Settings window. As with other user settings, the IPTV defaults are loaded from the registry when IxChariot is started and are saved when IxChariot exits.

Table 10-13 describes the parameters for which you can set default values:

Table 10-13. IPTV Defaults in the Change User Settings Window

Parameter	Description
IPTV Traffic Characteristics:	
Encoding	The desired video encoding for this video stream. The choices are MPEG2 or Custom.
RTP Payload Type	If you select *Custom* encoding, you must specify an RTP Payload type. (If you select *MPEG-2* encoding, this field will be read-only.)
Media Frames per DG	The number of media frames per datagram. All signed integer values are valid. The default value for Ethernet is 7.
Media Frame Size	The media frame size for the selected encoding.

Table 10-13. IPTV Defaults in the Change User Settings Window (Continued)

Parameter	Description
Bitrate	The nominal data rate of the video stream. All signed integer values are valid. Typical values for MPEG2 are between 3 and 15 Mbps. The default is 3.75 Mbps.
Bitrate unit of measure	The unit of measure for the bit rate. You can set the bitrate units to Gbps (1024 megabytes), Mbps (1024 kilobytes), Kbps (1024 bytes), or kbps (1000 bits).
Source port number	The source port number for the video stream.
Send Buffer Size	The socket buffer size used for sending the data streams. The default value is determined by the operating system on which the Performance Endpoint is running. **NOTE:** If the SOCKET_SEND_BUFFER_SIZE keyword in the endpoint.ini file specifies the send buffer size, that value overrides the value specified in this dialog.
Test Network:	
Network Protocol	The network protocol to use for this video stream. The options are RTP and UDP.
Service Quality	Optional. The name of the QoS template that will define the service quality for this video stream.
IPTV Receiver Settings – Channel Control:	
Number of TR	The number of timing records that will be generated for each join/leave cycle.
TR duration (sec)	The duration of each timing record (in seconds).
IPTV Receiver Settings – Receiver Settings:	
Number of iterations	The number of times the receiver will switch channels (leave one channel and join another).
Receive Buffer Size	The socket buffer size used for receiving the data streams. The default value is determined by the operating system on which the Performance Endpoint is running. **NOTE:** If the SOCKET_RECEIVE_BUFFER_SIZE keyword in the endpoint.ini file specifies the receive buffer size, that value overrides the value specified in this dialog.
Switch delay (ms)	The number of milliseconds that will elapse between leaving one channel and joining another.

Managing IPTV
Channels

You use the IxChariot IPTV Channels Editor to create and maintain a collection of channel definitions. Rather than define a set of channels for each test, you can define one or more sets of channels, export them to files, then import them into each new test that your create.

Creating New Channels

Refer to *Configuring IPTV Channels* on page 10-69 for instructions for creating new channels.

Deleting a Channel

To delete a channel:

1. Select **Edit** > **IPTV Channels Editor** to open the IPTV Channels Editor Dialog.

2. Select the channel from the Existing Channels List.

3. Click **Delete**.

If the selected channel is not referenced by any IPTV Receiver Groups, IxChariot deletes it without prompting for confirmation. However, if the selected channel is referenced by one or more IPTV Receiver Groups, IxChariot prompts you to confirm the action.

Editing a Channel

To modify the parameters of a channel:

1. Select **Edit** > **IPTV Channels Editor** to open the IPTV Channels Editor Dialog.

2. Select the channel from the Existing Channels List.

3. Click **Edit**.

   IxChariot opens the channel definition in the Edit IPTV Channel Dialog.

4. Modify the parameters, as required.

   Refer to *IPTV Channels Parameters* on page 10-70 for description of each of the parameters.

5. Click **OK** to save your changes and exit from the Edit IPTV Channel Dialog.

Note that you can also edit a channel by double-clicking its name in the Add/Edit IPTV Receiver Group dialog.

Replicating a Channel Definition

You can create a new channel definition by replicating an existing channel:

1. Select **Edit** > **IPTV Channels Editor** to open the IPTV Channels Editor Dialog.

2. Select the channel that you wish to replicate from the Existing Channels list.

3. Click **Replicate**.

   IxChariot opens the channel definition in the Edit IPTV Channel Dialog.

4. Assign the channel a unique name.

5. Assign the channel a unique multicast address / multicast port combination.

6. Modify any other parameters, as required.

   Refer to *IPTV Channels Parameters* on page 10-70 for description of each of the parameters.

7. Click **OK** to save your changes and exit from the Edit IPTV Channel Dialog.

IxChariot adds the new channel definition to the Existing Channels list.

Exporting Channel Definitions

You can output the definition of one or more channels to a file. Once saved, you can then import the saved definitions into other tests. To export channel definitions:

1. Select **Edit** > **IPTV Channels Editor** to open the IPTV Channels Editor Dialog.

2. Select one or more channels from the Existing Channels List.

3. Click **Export**.

   IxChariot opens the export IPTV Channels dialog.

4. Enter a name for the file.

5. Optionally, select a different destination folder.

   The default folder is IxChariot\Application Groups.

6. Click **Save**.

IxChariot saves the file using the file type of chn.

Importing Channel Definitions

You can import channel definitions from a previously-exported chn file into an IxChariot test. To import channel definitions:

1. Select **Edit** > **IPTV Channels Editor** to open the IPTV Channels Editor Dialog.

2. Click **Import**.

   IxChariot opens the default IPTV Channels export folder (IxChariot\Application Groups).

3. Optionally, select a different folder.

4. Select the desired chn file.

5. Click **Open**.

6. If necessary, resolve any duplicate channel names, multicast addresses, and multicast ports from the import file:

   a: A duplicate channel name appears in red. You cannot import the channel until you modify the name to make it unique.

   b: A duplicate multicast address / multicast port combination appears in red. You cannot import the channel until you modify the address or port such that the multicast address / multicast port combination is unique.

   c: Click **Skip** if you wish to skip over a channel. In this case, that channel will not be imported.

   d: Click **Skip All** if you wish to skip over all the channels. In this case, no channels will be imported.

Collecting TCP Statistics

Related Topics

Miscellaneous Run Options on page 7-11
The TCP Statistics Tab on page 11-36

If you are running a test on an Ixia chassis, you can instruct IxChariot to collect a set of TCP statistics for the endpoints executing the test. To do so, select "Collect TCP statistics" in the Run Options tab of the Run Options dialog.

TCP Messages Included in the Statistics

TCP statistics are collected for TCP packets that include SYN, FIN, ACK, and RST messages, as well as for TCP connections, TCP retransmissions, and time-outs.

Note that IxChariot collects ACK-for-FIN statistics, but these are pure ACKs for the FINs that one side had sent. If an RST is sent as a response to a FIN, IxChariot counts it as an RST received and not as an ACK-for-FIN. Figure 10-27 shows a partial packet trace on an IxChariot TPC Statistics test. Packet 7 in the trace is an [RST, ACK] message and, as such, IxChariot counts it as an RST rather than as an ACK-for-FIN message. If the TCP header has both the ACK and RST control bits set, IxChariot counts it as an RST message.

Figure 10-27.   Packet Trace on a TCP Statistics Test

How TCP Statistics are Measured

TCP statistics are measured by the Performance Endpoints on a per timing record basis, and are collected "in between" timing records. That is, the TCP statistics are collected between START_TIMER and END_TIMER statements (as with all other timing record values). When a timing record is cut (as defined in an IxChariot script), the values for each of the TCP statistics are recorded and forwarded to the IxChariot Console. The gathered data are processed, stored, and made available for display. The processing includes the calculation of minimum, maximum, and average values for TCP retransmissions.

Long connection scripts are more effective than short connection scripts in collecting TCP statistics. Also note that a SLEEP command inserted before the Close enhances the ability to capture the final TCP statistics before closing the

socket. Without this added SLEEP command, the final timing record may be cut before the final TCP statistics reach the port, in which case these final statistics will be lost.

Be aware that gathering TCP statistics can put significant load on the Endpoint hardware, especially if generating timing records at a high rate. For example, a short connection script cutting a new timing record every 20 ms and gathering 20 TCP statistics on 500 pairs requires 1,000,000 extra system calls. This can significantly reduce maximum performance.

TCP Statistics Requirements and Limitations

TCP statistics collection requires IxOS 3.80 or higher and is available only on load modules with a Power-PC 405 or Power-PC 750 processor: TXS8, TXS4, STXS4, SFPS4, ALM-T8, ELM-ST2, TXS2, LM10GE700F1B-P, LM622MR, OLM1000STXS24.

Note also that complete IPv6 TCP statistics are only available in IxOS 4.10 and above. In IxOS 4.0 (and earlier), the following IPv6 TCP statistics are not reported:

• SynReceived

• FinReceived

• FinAckReceived

• ResetReceived

The TCP statistics that are not reported are marked "N/A" in the TCP statistics tab.

Tips for Testing

Related Topics

Response Time Testing on page 10-85
Throughput Testing on page 10-85
Designing IxChariot Performance Tests on page 10-84
How Long Should a Performance Test Run? on page 10-86
Using IxChariot for Stress Testing on page 10-87
Reducing the Number of Timing Records on page 10-88

Even though it is quite easy to begin setting up and running tests with IxChariot, it is an extremely powerful tool, with a wide range of features and settings. In most cases, IxChariot provides ways for you to change defaults or configure test parameters. Each setting has an impact on the results you see, so it is important to establish goals and standards ahead of time. Consider the following questions: How well do you want each component of your network hardware to perform under *realistic* conditions? How well should it perform both before and after a new application is rolled out or a new priority scheme is introduced? For each type of network transaction, or for each application, is your highest priority maximum throughput, or minimal response time?

The following topics provide some performance and stress testing tips, with observations derived from our years of experience using IxChariot to test a vari-

ety of network equipment. We outline some techniques we think you'll find help-ful for making configuration decisions, running performance and stress tests, reducing the number of timing records generated, and testing on unreliable net-works.

Using UNLIMITED as the Data Rate

The send_data_rate variable in IxChariot scripts provides data rates ranging from 28.8 kbps to 155.52 Mbps. In addition to these specific rates, you can choose UNLIMITED as the data rate. When you choose UNLIMITED, the endpoints send data as fast as possible. Following are some suggestions for deciding whether or not to set the data rate to UNLIMITED for test pairs:

• When your test includes multiple pairs, avoid using UNLIMITED as the data rate. It is very possible for one pair to consume all available bandwidth, which will cause the other pairs in your test to time-out.

• The UNLIMITED data rate can be appropriate for stress testing. But as noted above, if your test comprises multiple pairs, the UNLIMITED data rate may cause the test to fail.

• In general, try to select a send_data_rate value that matches your network configuration. For example, if your test network is configured with T3 line rates, select 44.736 Mbps as your send_data_rate, rather than UNLIMITED.

Designing IxChariot Performance Tests

Related Topics
Response Time Testing on page 10-85
Throughput Testing on page 10-85
Performance Testing on page 7-2

Building a performance test with IxChariot is simple. This section offers sugges-tions for creating effective, repeatable performance tests. The **Performance Testing** checkbox, which you can access from the Run Options dialog box, can automatically change the default settings for reporting and result collection to help your network log higher performance and ensure that your results are as accurate as possible. See *Performance Testing* on page 7-2 for more information.

In general, always choose the following options for your performance tests:

• Report timings using batch. The alternative to batch, real-time reporting, causes timing records to flow across the same network you are measuring. This can really disrupt what's being measured, potentially changing your results by several hundred percent.

• Run for a fixed duration, or run until any pair completes. These Run Options cause all the pairs to end at the same time. Otherwise, your results become skewed as some pairs keep running (thus getting more bandwidth), while oth-ers have completed.

• Don't use an endpoint in the same computer as the Console. You don't want the endpoint and Console to be competing for CPU cycles or for access to the protocol stacks. If the Console computer must be an endpoint, use RUNTST.exe to run the test from a command line, rather than the Console.

• Don't poll the endpoints. Polling requires extra data flows, slightly perturbing what's being measured. Only poll if you suspect something's gone wrong, in

which case you should probably start the test over anyway. See *Polling the Endpoints* on page 7-5 for more information.

- Don't validate data upon receipt or examine CPU utilization. Data validation and collection of CPU utilization consume extra CPU cycles at the endpoints.

- Don't run other software on the endpoint computers, and disable screen savers. See "Virus scanners" in *Factors Affecting Results* on page 11-52 for more information.

Response Time Testing

Related Topics

Designing IxChariot Performance Tests on page 10-84
Throughput Testing on page 10-85
Performance Testing on page 7-2

Consider these recommendations when attempting to measure minimum response time:

- Use the script called `Response_time` when testing for the lowest possible response times.

  This script sends a 100-byte file and waits for an acknowledgment. It simulates a typical, time-sensitive network transaction.

  It Is a good idea to experiment with this script's `transactions_per_record` variable because you might find that the timing records you generate are too short to give meaningful results. Increasing the number of transactions increases the length of timing records.

- If your network supports full duplex, use symmetrical endpoint pairs when simulating multiple pairs.

  Add pairs two at a time. For example, set up Computer A as Endpoint 1 in Pair 1, and Computer A as Endpoint 2 in Pair 2.

Throughput Testing

Related Topics

Designing IxChariot Performance Tests on page 10-84
Response Time Testing on page 10-85
Performance Testing on page 7-2

Consider these recommendations when attempting to measure maximum throughput:

- Use the script called `Throughput` when testing for maximum throughput on typical networks. This script sends 100,000 bytes from Endpoint 1 to Endpoint 2, then waits for an acknowledgment. It thus simulates the core file transfer transaction performed by many applications. Experiment with its `file_size` variable. We've seen that a good size is 1 MB (10 times `Throughput`'s default of 100,000 bytes), but 10 MB is worth experimenting with.

- Use the script called `High_Performance_Throughput` (in the `Benchmarks` script directory) when testing for maximum throughput on high-performance networks that support speeds of 100 Mbps and greater. This script, designed for use with TCP, sends 10,000,000 bytes from Endpoint 1 to Endpoint 2, then waits for an acknowledgment. It instructs endpoints running on Microsoft Windows 2000, Windows Server 2003, and Windows XP to use overlapped I/O to achieve the highest possible throughput.

  To achieve the most accurate results, do not edit the send_data_rate parameter: leave it set to UNLIMITED. (However, refer to *Using UNLIMITED as the Data Rate* on page 10-84 for a description of potential problems resulting from the use of the UNLIMITED data rate.)

- Use the script called `Ultra_High_Performance_Throughput` when testing for maximum throughput on high-performance networks that support speeds of 100 Mbps and higher. This script, designed for use with TCP, sends 2,147,483,647 bytes from Endpoint 1 to Endpoint 2, then waits for an acknowledgment. It instructs endpoints running on Microsoft Windows NT, Windows 2000, Windows XP, and Windows Vista to use overlapped I/O to achieve the highest possible throughput.

  In contrast to the High-Performance Throughput script, this script sets send and receive buffers for the connection (a conn_send_buffer for each endpoint and a conn_rcv_buffer for each endpoint). These buffers specify the desired TCP window size for the connection, and they support the RFC 1323-specified TCP window scale option. All four buffers are set to 512 MB (536870912 bytes). In addition, the send_buffer_size and receive_buffer_size variables are set to 1 MB (1048576 bytes).

  For a detailed description of this topic, refer to the *Modifying the TCP Window Size* on page 5-52.

  To achieve the most accurate results, do not edit the send_data_rate parameter: leave it set to UNLIMITED. (However, refer to *Using UNLIMITED as the Data Rate* on page 10-84 for a description of potential problems resulting from the use of the UNLIMITED data rate.)

- Use symmetrical endpoint pairs when simulating multiple pairs.

- Add pairs two at a time. For example, set up Computer A as Endpoint 1 in Pair 1, and Computer A as Endpoint 2 in Pair 2.

- Use endpoints running on Windows.

- If using the `FTPget` or `FTPput` scripts, don't set the number of repetitions greater than 1.

How Long Should a Performance Test Run?

Related Topics
Designing IxChariot Performance Tests on page 10-84
Relative Precision on page 11-47
Understanding Timing on page 11-1

Designing a good test can require a bit of experimenting. The ideal IxChariot test has the following qualities:

- It runs long enough so that the relative precision shown on the test results is small. See *Relative Precision* on page 11-47 to understand the quality of your test results.

- Timing record durations should be long enough to avoid timer errors. See *Understanding Timing* on page 11-1 for a discussion of the timer resolution and its effect on test accuracy. Avoid any timing records shorter than 10 ms.

- Timing records are taken frequently enough to show fluctuations in performance during the test.

- The ideal test doesn't contain too many timing records. Tests with more than 10,000 timing records use up considerable memory and disk space and make the IxChariot GUI slow and cumbersome. See *Reducing the Number of Timing Records* on page 10-88 for information on reducing the number of timing records and avoiding short timing records.

Timing records that are too short often trigger warnings, which appear next to any affected pairs after a test run has completed. They indicate that your results contain inaccuracies, or that not all results could be graphed or included in the totals. See *Finished Test Warnings* on page 12-6 for more information on the types of warnings you may see, what they mean, and how to avoid them.

A high relative precision value means either that you are running in a network whose performance is fluctuating, or that your test is too short. Assuming you don't want to change the network you are testing, you need to make the test run for a longer duration. Most performance tests should last between two and five minutes.

You control the duration of a test by varying the number of endpoint pairs, the number of timing records to generate, the number of transactions per timing record, and the amount of data sent in each transaction. For the sake of simplicity, let's assume that the number of endpoint pairs and the amount of data are fixed.

If you have one timing record for each transaction, you can see the performance fluctuations that occur from transaction to transaction. A large test running small transactions, like the `Creditl` script, could generate hundreds of thousands of timing records. This would require a tremendous amount of memory at the Console and take a lot of disk space. On the other hand, if you only generated one timing record for an entire test, the results would be easy to work with, but you would not be able to see any variation in performance from transaction to transaction. The trick is to find a balance.

Using IxChariot for Stress Testing

Building stress tests with IxChariot is simple, but it is worthwhile to proceed with caution to avoid bogging down your network and making it unavailable for a while. This section offers suggestions to help you design better stress testing. These suggestions generally advise settings that are the opposite of those used for real performance testing—they generate lots of network traffic, really stressing your hardware and software.

- Run for a fixed duration.

Decide how long you want to stress your network and endpoint computers. However, do some experimenting—you don't want to return thousands of timing records to the Console that you aren't going to use anyway.

Set a low value for the `number_of_timing_records` script variable, and a high value for the `transactions_per_record` variable. Multiply your `transactions_per_record` by 10 times the number of hours your test runs; for example, increase it by 10 times for 1 hour, 100 times for 10 hours, and so on. Note, however, that the `number_of_timing_records` variable is ignored when running for a fixed duration. Be sure to experiment with high `transactions_per_record` settings to avoid flooding the Console with too many timing records.

- Report timings using real time.

    Real-time reporting causes timing records to flow across the network as they're generated, increasing the amount of network traffic.

- Regularly poll the endpoints.

    Polling generates extra data flows, in addition to data sent in the scripts and timing records.

- Validate data upon receipt.

    If you validate all the data that transferred among endpoints under stress conditions, you may detect problems with network hardware or software.

- Use random `SLEEP` times.

    Use the uniform distribution of `SLEEP` times to emulate many users, pausing slightly between transactions. Choose a range of about 0 to 2 seconds (2000 ms).

- Set your `send_data_type` to `NOCOMPRESS`.

    This is the toughest data to compress—you'll keep network components that do compression busy trying to find patterns in this data.

- Set your send_data_rate to `UNLIMITED`.

    If your test uses a single pair, set the send_data_rate to UNLIMITED. However, when your test includes multiple pairs, using UNLIMITED as the data rate can cause one pair to consume all available bandwidth, in which case the other pairs in your test will time-out.

Reducing the Number of Timing Records

Related Topics

Because endpoints take their measurements at the tops of the protocol stacks, many operating-system and computer factors may affect the results. It Is important to avoid short timing records, which don't provide enough data for accurate results. Correspondingly, you'll want to avoid having too many timing records in your test results. IxChariot becomes sluggish when the number of returned timing records is above 10,000. Adjusting the numbers will take some experimenting; the `transactions_per_record` variable you use should be different when you are testing 56k modems from when you are testing a Gigabit Ethernet. A couple of absolute rules apply:

- Avoid any timing records shorter than 10 ms.

- Avoid generating more than 10,000 total timing records for a test.

When you run a test for a fixed duration, an endpoint ignores the
`number_of_timing_records` script variable. If the test uses a non-streaming
script and contains no `SLEEP` times, or uses a streaming script and uses
`UNLIMITED` for the `send_data_rate`, an endpoint completes as many transac-
tions during that time as it can. With a streaming script, the number of transac-
tions that Endpoint 1 runs is based on the `send_data_rate`. Therefore, running
a test for a particularly long duration hugely increases the number of timing
records the endpoints generate.

For example, you may run a test with one endpoint pair that generates 100 timing
records in 20 seconds on your network. If you run the same test with the fixed
duration set to one hour, IxChariot generates approximately 18,000 timing
records. And additional pairs increase this number exponentially.

The sections below provide tips for avoiding excessive timing records in testing
with non-streaming scripts and with streaming scripts.

Using Non-Streaming Scripts

Related Topics
Reducing the Number of Timing Records on page 10-88
Using Streaming Scripts on page 10-90

When you are running a test for a fixed duration, we recommend increasing the
scripts' `transactions_per_record` values so the test's total number of tim-
ing records is less than 10,000.

Here's a way to reduce the number of timing records by a tenth:

1. Click **Add Pair** on the Edit menu to create a new pair.

2. Click **Select script**. Choose a non-streaming script and click **Edit this script**.

3. Double-click the `transactions_per_record` variable in the bottom half
   of the Script Editor.

4. Increase by ten times the Current Value variable for
   `transactions_per_record`. For example, if the number of
   `transactions_per_record` is 50, change the number to 500. Click **OK** to
   save your changes.

5. On the Run menu, click **Set run options**. Click **Run for a fixed duration.** In
   the **Duration** field, type 1 minute.

6. Run the test and view the results. Using the Raw Data Totals tab, look at the
   Number of Records. This is the total number of timing records generated in
   one minute. We recommend about 50 to 100 records per pair for good statisti-
   cal significance. If the results contain more than a total of 10,000 timing
   records, go back and further increase the `transactions_per_record`.
   Otherwise, continue to the next step.

7. Increase the script's `transactions_per_record` variable to match the duration of your test.

Here's the formula used to determine the number of transactions per record:

```
transactions_per_record = (number of minutes to run) *
(transactions_per_record in the one- minute test)
```

Example:

In the steps above, you entered 500 for the script's `transactions_per_record` variable and ran the test for one minute. The results should have been about 300 timing records per pair (100 per 20 seconds = 300 per minute). Three hundred timing records per pair is fine if you are not running hundreds of concurrent endpoint pairs.

Now you want to run the test for a weekend (48 hours). Consider the math:

- `transactions_per_record` = (60 minutes per hour) x (48 hours) x (500 transactions per record)
- `transactions_per_record` = (2,880 minutes) x (500 transactions per record)
- `transactions_per_record` = 1,440,000

Open the script and change the `transactions_per_record` to 1440000. Then change the **Duration** to 48 hours in the Set Run Options dialog box.

Using Streaming Scripts

Related Topics
Using Non-Streaming Scripts on page 10-89
Reducing the Number of Timing Records on page 10-88

Streaming scripts do not use the `transactions_per_record` variable; a timing record is generated at the end of each `SEND` command. To reduce the number of timing records, increase the `file_size` variable on the `SEND` command. The `file_size` variable controls the amount of data per transaction.

If you specify a `send_data_rate` other than `UNLIMITED`, you can use the following equations to estimate the number of timing records you will receive. The basic network equation is:

```
time of timing record = # of bits / bits per second
```

Thus, for the using script variables, the equation is:

```
time of timing record = (file_size * 8) / send_data_rate
```

For example, if you are using the `Realaud` script with the script default parameters, you can expect each timing record to take about 1.39 seconds: (14,040 bytes * 8)/80,736 bps.

Use the following formula to determine the value to use for the `file_size` variable:

```
file_size = time of timing record / (# of timing
records*(send_data_rate * 8))
```

If you are running a script at full speed, that is, with the `send_data_rate` set to `UNLIMITED`, here's a way to reduce the number of timing records by a tenth:

1. Highlight a pair and click **Edit Pair** on the Edit menu.

2. Click **Select script**. Choose a streaming script and click **Edit this script**.

3. Double-click the `file_size` variable in the bottom half of the Script Editor.

4. Increase by ten times the **Current Value** for the `file_size`. For example, if the number of `file_size` is 3000 bytes, change the number to 30000.

5. On the Run menu, click **Set run options**. Click **Run for a fixed duration.** In the **Duration** field, type 1 minute.

Run the test and view the results. Using the **Raw Data Totals** tab, look at the Number of Records. This is the total number of timing records generated in one minute. We recommend about 50 to 100 records per pair for good statistical significance. If the results contain more than a total of 10000 timing records, go back and further increase the `file_size` variable.

Eliminating DNS Latency from Test Results

Related Topics

Designing IxChariot Performance Tests on page 10-84
Factors Affecting Results on page 11-52

When running tests where you use the hostname to define Endpoint 1 and Endpoint 2 and the `CONNECT` command is inside the test script's timing loop (as in short connections), IxChariot must "translate" the hostname to an IP address. If IxChariot cannot do this during test setup, response time results include the time required for the endpoints to "translate" the hostname.

This gives you an inaccurate response time number for the test. And sometimes, IxChariot is unable to "translate" both host names to IP addresses.

In some tests, the amount of DNS latency required to "translate" host names can be significant. For example, on Windows NT, the following search can contribute to the time required to resolve a hostname:

1. The endpoint inspects its own hostname.

2. The `HOSTS` file is inspected on the local computer.

3. The DNS Server is queried.

4. The NetBIOS name cache on the local computer is queried.

5. The WINS server is queried.

6. B-Node broadcasts are made from the local computer.

7. The `LMHOSTS` file is consulted on the local computer.

To eliminate DNS latency from your response-time results, use numerical IP addresses instead of host names.

TCP Testing with the Close_Type Parameter

In extensive performance testing with IxChariot, we've found that the type of close used for TCP connections affects test results. When short-connection TCP transactions are quickly repeated back-to-back, their transaction rate can significantly decrease over time, depending on how the connections are closed. The `Credits` script is a good example of such a series of transactions.

TCP offers two types of close: normal and abortive. Abortive close (or abortive disconnect) gives the most consistent performance for repeated short connections. This means that when one side closes the TCP connection, the other side will respond with a `RST`. In a normal close (or normal disconnect), when one side closes the TCP connection, the other side responds with a `FIN`.

Not all TCP/IP stacks support abortive close; it depends on your operating system. If your OS does not support it, internal control blocks can build up in the TCP/IP stack, particularly when you are running short-connection scripts, which generate a large number of closes per second. If these internal control blocks start to affect your endpoints' ability to process these connections, you can see slow-downs and a general performance degradation.

By default, IxChariot scripts use an abortive close, or Reset. By closing the connection immediately, the `RST` flag avoids a possible loss of resources that can occur when two computers are tied up in `TIME_WAIT` state. However, the Script Editor lets you choose a normal close for a test by editing the `close_type` parameter that appears in the script just after the `DISCONNECT` command. Just double-click the `close_type` parameter and choose **Normal** from the list to change it.

This option is not supported on MVS, Solaris 2.4, or Linux operating systems.

Emulating Multiple Users

Related Topics
Using IxChariot for Stress Testing on page 10-87

A single IxChariot pair represents the traffic generated by many users of a given application. The default values in IxChariot application scripts drive this traffic as fast as possible, creating more traffic than any one end user can generate. By design, IxChariot generates enough traffic to get accurate measurements.

For example, in the `Filesnds` script, a client computer transfers a 100-kB file and receives an acknowledgment. With the default script variables, the file transfer repeats 100 times, each transfer happening immediately after the conclusion of the last. On a 10Mbps Ethernet, this takes about 10 seconds. The most active user does not perform 100 file transfers every 10 seconds. So to realistically emulate a single user, you should slow things down.

The `transaction_delay` variable in all IxChariot application scripts sets the time between repetitions of the scripted transaction. The default setting tells the endpoints to immediately repeat the scripted transaction until the script completes. By contrast, a typical end user might transfer a file to the file server about once every 20 minutes. Therefore, there is an average `transaction_delay` of 20 min. for one user. The `transaction_delay` variable accepts values in milliseconds (ms); 20 min. is 1200 seconds and 1200000 ms (do not use separators when entering values in the Script Editor). A delay between transactions of

1200000 slows IxChariot down so that it generates the typical file-transfer traffic of a single user.

To represent multiple users, use similar calculations. If we have generated the traffic level of one typical user, then 10 times that traffic would represent 10 users. Therefore, we *reduce* the `transaction_delay` by a factor of 10, to 120 000 ms (every 2 minutes), to represent 10 users' traffic. Reduce the `transaction_delay` by a factor of 100, to 12000 ms (every 12 seconds), to represent 100 people.

The mathematical limit of the reasoning process we have outlined breaks down when the delay time is not significantly larger than the response time of a given script. To find out the limit in your network, run a quick test with `transaction_delay` set to 0. Look at the resulting response time. This value is the lower limit to consider because if the transaction delay is less than the response time, the next transaction should commence before the previous one completes, which is impossible. You can represent more users on a faster data link technology. Similarly, if the script contains a shorter transaction but still has the same amount of time between transactions, you also have a lower response time and can represent more users. On the other hand, larger file sizes (1MB or so) reduce the number of users you can represent.

The mathematical limit of this model is not the same as the practical limit, however. You should consider what you are really trying to discover about your network. Keep in mind some additional factors:

- Calls to the system clocks work differently on different operating systems. Thus OS/2, Windows 3.1 and NetWare should not be used for delay times below 1 second.

- When using our benchmarking class of scripts, always use the short connection scripts rather than the long connection scripts. The short connection scripts include the overhead of establishing and taking down connections. The traffic these scripts represent is also bursty, with long periods of no traffic, and then brief periods of full activity. This is how traffic in the real world works.

To further refine your emulation of multiple users, you can also randomize the `SLEEP` times in the scripts. Rather than leaving our `transaction_delays` at a set time, we can randomize them over a range of values. Refer to "Setting `SLEEP` Times" in the *Application Scripts* guide for more information. To set `SLEEP` times, choose a distribution from the Current Value list, and set an upper and lower limit. For example, 900000 for 15 minutes, 1500000 for 25 minutes, etc.

The model we have developed here uses a single connection at a time. It can emulate thousands of users over a period of time, but it cannot emulate more than one user accessing network resources at *exactly* the same time. Therefore, there's never any contention for bandwidth. The solution is to divide the multiple users among several endpoint pairs, preferably among several endpoint computers. To take contention for network bandwidth into account, first decide how many connections must happen simultaneously. The number of pairs should match the number of simultaneous connections, with no delay between transactions. As a

general rule, 1000 users are better represented by several pairs spread over several computers than by a single pair with a transaction delay.

Wireless Network Testing

IxChariot is an excellent tool for testing wireless networks, but you may want to do some extra configuration to ensure that your tests complete successfully. The IxChariot Console and the endpoints send three different types of data over the network: test setup information, actual test traffic, which resembles application traffic, and test results. IxChariot tests might not initialize or complete properly on a wireless network because data might be lost during test setup or result reporting.

Therefore, you might want to set up an alternate network just for test setup and results communications between the Console and the endpoints. IxChariot lets you configure a separate setup address and/or protocol for situations in which you want to keep setup and reporting flows separate from actual test traffic, or for wireless testing, if you want to prevent the loss of setup and results data.

Use computers that have two or more NIC cards as endpoints in your wireless testing. Set up an alternate network connection between the additional NIC cards. After you create an endpoint pair, click **Edit Pair Setup** on the Edit menu to specify new addresses:

• where the IxChariot Console can send test setup information to Endpoint 1

• where Endpoint 1 can contact Endpoint 2 for setup and results collection.

All other test traffic can then flow over the wireless network. Refer to *Cloning Hardware Performance Pairs* on page 5-25 for more information.

> **Note**: If you want to isolate setup traffic from test traffic, make sure there's no route between the two networks you use. Some operating systems (Windows, Solaris) are able to recognize a choice of two paths to the same destination and perform implicit routing, thus possibly choosing the wrong network interface.

Testing with a SOCKS Proxy Server

In cases where the IxChariot Console doesn't have direct connectivity to the Endpoint 1 computers, the Console can be instructed to instead go through a server running SOCKS 5. A SOCKS server, which is an IP-based proxy server that can run TCP and UDP applications, runs on the Ixia Hardware Chassis. In a typical setup, the SOCKS server is able to contact all the Ixia cards running endpoints that have been designated as E1. Or else the SOCKS server acts as a proxy for endpoints on an isolated network that the IxChariot Console can't reach. The Console must connect directly to the proxy server for all communications to

those endpoints. The endpoints acting as E1 will then send results back to the Console via the proxy server. An illustration is shown below:

Figure 10-28.   SOCKS Proxy Usage
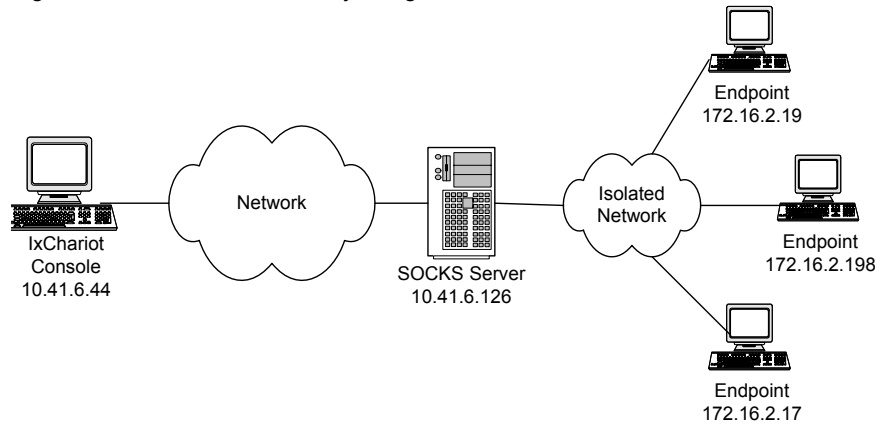
The Console knows to contact the SOCKS server anytime it sees an E1 address in the range 172.16.2.1 to 172.16.2.254.

To use a SOCKS proxy server in IxChariot tests, you can instruct the Console to contact the SOCKS server for any communications with the endpoints within a certain IP address range by taking the steps outlined below.

> **Note:** When Ixia ports are used as endpoints, Stack Manager performs all necessary setup for using a SOCKS proxy server.

Here's what you need to do to manually set up SOCKS server support:

First, create a file in .txt format to specify the IP address of the SOCKS server. Use the following format:

```
SOCKD@="ip addr of socks server" "ip subnet serviced" mask
```

For example, let's say you're instructing the IxChariot Console to look for a SOCKS server with IP address 10.41.6.126 each time it encounters an Endpoint 1 address in the range 172.16.2.1 to 172.16.2.254, with a subnet mask of 255.255.255.0. The figure above illustrates this scenario. Your text file would contain the following line:

SOCKD@=10.41.6.126   172.16.2.0 255.255.255.0

Save the file in a folder that's accessible to the Console--in one of the IxChariot program folders, for example. Then set the registry as follows:

1. Click **Start > Run** and enter `regedit` at the prompt.

2. Navigate to `HKEY_LOCAL_MACHINE` and click the `SOFTWARE` key.

3. Right-click and select **New > Key**. Add a key called "`IXIA Communications`".

4. Right-click the new `IXIA Communications` key and follow the same procedure to add a sub-key called "**socks**".

5. Right-click the new socks key and select **New > String**. Supply a string in the following format:

```
config  "pathname to text file defining the SOCKS server"
```

For example, enter:

```
config  c:\program_files\Ixia\IxChariot\sockserver.txt
```

Once you've edited the registry and provided a .txt file with the full network path to the SOCKS server, you can set up IxChariot tests normally; however, when you define the endpoint pairs, you'll supply an IP address from within the SOCKS server's designated range as either the Endpoint address or as the "Use Endpoint 1 address as management address" address in the Edit Pair Setup dialog box (see *Cloning Hardware Performance Pairs* on page 5-25 for more information).

The steps above instruct the IxChariot Console that anytime it sees an IP address of 172.16.2.x as the "From Endpoint" address or "Console Knows E1" address, it must send the request to the SOCKS server. The SOCKS server then forwards the ensuing traffic on to the location the proxy has for Endpoint 1.

Using Aliases

Related Topics

IPX/SPX Aliases on page 5-3
Add IPX/SPX Aliases on page 5-3

The TCP/IP protocols provide a mechanism for defining aliases or nicknames. You can create your own alias directory for the IPX/SPX protocols.

For example, build an IxChariot test using the network addresses of "TEST1" and "TEST2." The Console resolves these nicknames before starting a test. Using this scheme, you can create tests that don't include any real LU names, IP addresses, or IPX addresses—just aliases. By changing the alias, you change which systems will run the test.

- **TCP/IP**

  You can define nicknames for IP addresses by defining them in the file named HOSTS, in your ETC directory. The following example illustrates creating an alias for TEST1 and TEST2:

  ```
  44.44.44.60 TEST1
  44.44.44.62 TEST2
  ```

- **IPX/SPX**

  IxChariot stores aliases for IPX addresses in SPXDIR.dat in the directory where you installed IxChariot. See *IPX/SPX Aliases* on page 5-3 for information on adding entries and making changes to this file.

  ```
  44.44.44.101 00000002:006097c3f512
  44.44.44.103 00000002:00a0247b58de
  44.44.44.104 34242342:000000000001
  ```

Automating Tests to Form a Simple Monitor

Although IxChariot is not a network monitor, you can still create an IxChariot test that tracks network performance. From the IxChariot Console, create a test with endpoint pairs that have the scripts and network addresses that you want to measure. For example, set the `number_of_timing_records` variable to 1000, and the `delay_between_transactions` variable to 600000 in each script. Because the delay is in milliseconds, the value 600000 causes a delay of 600 seconds (10 minutes). Set the Run Options to show results in real time. When you start the test, you will see the results updated every ten minutes. If you stop the test, you can view the endpoint details to find how your network performance changes during the time of your test.

Benchmark Testing Tools

The independent testing laboratory CMP has developed two tools to help you perform specific tasks with IxChariot, namely, comparing the performance of different types of network interface card (NIC), and evaluating virtual private network (VPN) configurations. Both tools work in conjunction with a previously-installed version of IxChariot, and both are available free on the World Wide Web. To download the CMPmetrics for NICs or CMPmetrics for VPNs Benchmark product, go to www.techweb.com/cmpmetrics.

11

Understanding Results

Related Topics

Custom Printing and Export Options on page 5-64
Test Window Tabs on page 11-14
Printed/Exported Results on page 11-39
Calculations on page 11-14

The following topics discuss the results generated when you run a successful test. They provide explanations intended to help you interpret your results and include technical information about how IxChariot generates timing measurements.

• See *Custom Printing and Export Options* on page 5-64 for information on printing and exporting data.

• The topics listed under *Test Window Tabs* on page 11-14 describe and explain the information shown on the results tabs in the Test window.

• The topics listed under *Printed/Exported Results* on page 11-39 describe and explain the information you see in printed or exported test results.

Understanding Timing

Related Topics

Timing Records and Graphs on page 11-6
How Long Should a Performance Test Run? on page 10-86
How Inactive Time Affects Aggregate Values on page 11-4

Three timing values show up throughout the results. Following are detailed explanations of IxChariot's terms *elapsed time, measured time*, and *inactive time*.

• **Elapsed Time**

When referring to an entire test, the Elapsed Time is the duration from when the pairs started running until they stopped. This value appears at the top of the printed and exported results, as well as in the status bar at the bottom of a Test window.

When referring to a single timing record, the Elapsed Time is the time at the endpoint when the timing record was cut. Time 0 is when all the pairs com-

---

pleted initialization and the test's Run Status went to Running. This value appears in the Timing Records Details when you click **Show Timing Records** on the View menu.

- **Measured Time**

    The sum of the times in all the timing records returned for an endpoint pair. This may be less than the amount of time the script was actually executing (that is, the elapsed time), depending on how much activity or SLEEPs the script performed outside of its START_TIMER and END_TIMER commands.

    This value appears in the Throughput[1], Transaction Rate, and Response Time results.

    When referring to a single timing record, the Measured Time is the time measured between the START_TIMER and END_TIMER commands in a script. This value appears in the Timing Records Details.

- **Inactive Time**

    The Inactive Time is the time spent outside the START_TIMER and END_TIMER commands in a script. This is time when an endpoint isn't doing work that's being measured. If this inactive time is more than 25 ms between timing records, it is shown; otherwise, inactive time is shown as blank for times below this threshold of 25 ms. The inactive time for each timing record is shown in the Timing Records Details. See *Timing Records and Graphs* on page 11-6 for more information about inactive time.

    The TCP Statistics measures are unique in that these values are collected during inactive times. For information about the TCP Statistics, refer to *The TCP Statistics Tab* on page 11-36.

    The clock timers used when timing scripts are generally accurate to 1 ms, although this accuracy depends on the endpoint operating system. For most tests, this is more than sufficient. If the transactions in a test are too short, this timing accuracy can cause problems.

    If the **measured time** of timing records drops, the resolution of 1 ms becomes a higher percentage of the actual measured time. For example, if the measured time is 5 ms, there is a + or -10% potential for error, since the actual time is somewhere between 4.5 and 5.5 ms.

    For repeatability and confidence, use

    - long timing record durations, or

    - many timing records.

    See *How Long Should a Performance Test Run?* on page 10-86 for more information.

Start, End, and Run Times

Related Topics
Test Window Status Bar on page 3-14
Understanding Timing on page 11-1
How Inactive Time Affects Aggregate Values on page 11-4

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

Understanding the Run Status on page 11-7

The Run Time shown in the Status Bar is equal to the Test End Time minus the Test Start Time. Depending on how you've configured your Run Options, the times shown in the Status Bar might change slightly between similar tests. That's because IxChariot calculates the End Time differently, based on how a test completes.

- For tests that complete successfully:
  - If the Run Option is "Run until any pair ends," the End Time is calculated as the Start Time plus the smallest Elapsed Time for any pair, rounded up to the next whole second.
  - If the Run Option is anything else, the End Time is calculated as the Start Time plus the largest Elapsed Time of any pair, rounded up to the next whole second.
- For tests that complete with errors:
  - If the Run Option is "Run until any pair ends," the End Time is calculated as the Start Time plus the Elapsed Time for the pair that caused the test to stop, rounded up to the next whole second.
  - If the Run Option is anything else, the End Time is the Start Time plus the largest Elapsed Time of any pair, rounded up to the next whole second, just as it is if the test completes successfully.

Examining Timing Records

Related Topics
Understanding Timing on page 11-1
Understanding Results on page 11-1

While a test is running or after a test has been run and has results, select an endpoint pair in the Test window, then click **Show Timing Records** on the View menu to see the individual timing records for that pair.

In the Timing Records dialog box, each timing record is assigned a number in the order in which the record was generated. At the top left, the total number of timing records is shown.

The timing records contain slightly different information for tests that use non-streaming scripts and tests that use streaming scripts. Both contain the Elapsed, Measured, and Inactive Time values in seconds. See *Understanding Timing* on page 11-1 for definitions of each. For each record, information about the data gathered during that record is shown in labeled columns. The columns of data correspond to the tabs showing in the Test window for your particular test.

For non-streaming scripts, timing records include data on throughput[1] and response time, as well as bytes sent and received. For streaming scripts, because delivery is not guaranteed, results include the number of datagrams sent, received, lost, and out-of-order, as well as jitter statistics if the RTP protocol was

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

used in the test. Information about how each performance metric is derived may be found in *Understanding Results* on page 11-1.

Hardware Performance Endpoints and VoIP Hardware Performance Endpoints provide additional real-time statistics, including bytes received and sequence errors.

> **Note:** The Throughput values displayed for both types of Hardware Performance Endpoints is the data bit rate. That is, the rate at which the data portion of packets are sent, ignoring any packet gaps and preambles.

• Refresh

Updates the contents of this dialog box while a test is running. Newly-received timing records are added to the bottom of the list.

• Show Latest

Refreshes the contents and scrolls to the last timing record.

How Inactive Time Affects Aggregate Values

Related Topics
Understanding Timing on page 11-1
Understanding Results on page 11-1

Tests with non-zero inactive time can produce inaccurate and misleading values in the aggregate fields because of the way IxChariot calculates timing records and aggregate values.

To understand why this is the case, let's consider a test with two pairs running over a 10 Mbps Ethernet. If we ran each pair alone, we could probably get 8 Mbps. Assume that we construct a test with our two pairs, such that each pair spends half of its time sleeping outside of the timing record loop and half of its time sending data. Due to competition for the Ethernet, we'll find that one pair often sends while the other pair sleeps, exchanging roles frequently. Since the SLEEP time is not counted in the throughput calculations, both pairs report that they achieved about 8 Mbps, resulting in an aggregate throughput of 16 Mbps on a 10 Mbps Ethernet! The aggregate occurs across the sum of the *measured* times, not the *elapsed* time. See *Understanding Timing* on page 11-1 for more information.

As a result of scenarios like these, IxChariot calculates throughput for a group of endpoints by taking the inactive time into account. Thus for a group of endpoints using a non-streaming script, total bytes sent and received by all pairs is divided by the total elapsed time (which is the elapsed time of the longest-running pair in the test). For groups using a streaming script, the total bytes sent and received by all Endpoint 2s is divided by the total elapsed time. For a multicast group, the highest total bytes received by an Endpoint 2 is divided by the elapsed time of the longest-running pair.

You can group endpoints in your results by specifying a method of sorting—by network protocol or by script, for example. See *Sorting and Grouping Pairs* on page 5-26 for more information.

Watch for spots on the line graph where a pair's throughput drops to zero, or examine the Inactive Time column in the Timing Record Details. To avoid skewed results, only use non-zero SLEEPs inside the timing record loop (that is, within the START_TIMER and END_TIMER commands).

How Streaming Loss Is Calculated

Related Topics

Understanding Timing on page 11-1
Examining Timing Records on page 11-3

The results from streaming scripts include the number of datagrams sent, received, and lost. Streaming loss is calculated as follows:

- Datagrams per timing record = N.

- Each datagram contains x bytes.

- E1 sends datagrams to E2.

- E2 counts the number of datagrams received from E1.

- When E2 receives N datagrams, it writes the timing record.

- When the timing record is written:

  E1 has sent $M*x$ bytes.

  E2 has received $N*x$ bytes.

- Data loss occurs if the number of bytes received ($N*x$) is less than the number of bytes sent ($M*x$).

Following is an simple example of the streaming loss calculation:

N = 4	(datagrams per timing record)
M = 5	(datagrams sent by E1 when E2 writes the timing record)
x = 8	(number of bytes per datagram)
M*x = 40	(bytes that were sent by E1 when the timing record is written)
N*x = 32	(bytes that were received by E2 when the timing record is written)
data loss = 8	(bytes sent minus bytes received)

Timing Records and Graphs

Before you can thoroughly analyze the results from IxChariot tests, it is helpful to understand how timing records contribute to the data and the graphs you see of that data. Timing records contain measurements of all the transactions that take place between a pair of endpoints after the `START_TIMER` command and before the `END_TIMER` command in an application script.

You may have noticed that a throughput line graph sometimes looks like a staircase, or it drops to zero. This behavior is the graphical representation of a pair with a non-trivial amount of inactive time. To illustrate, let's walk through an example.

The endpoint follows a script (such as `Credit1`) and measures how long it takes to perform the commands in the script. For a script with default settings, the endpoint runs the script as fast as possible; it is sending and receiving data over the network almost all the time.

Change the `transaction_delay` script variable to about 1000 ms, and the endpoint doesn't use the network for that period of time. For those 1000 ms, when the endpoint is inactive, throughput is zero.

Even for scripts with `transaction_delay` set to 0, it is possible for the endpoint to be inactive (i.e., not performing the script). Typically, this can occur on a computer with many endpoint threads active or other applications running concurrently. Since an endpoint doesn't "own" the computer, it has to share it with the other applications. Consequently, it can take a measurable amount of time for the endpoint just to finish one timing record and start the next.

You can see the amount of inactive time by viewing the Timing Record Details. If the endpoint is inactive for more than 25 ms between timing records, the graph drops to zero to show this inactivity. This will appear to you as a "staircase effect."

Pair Reinitialization and Graphs

Related Topics

Error Handling Tab on page 7-14

When you configure a test to allow pair reinitialization during test execution, your graphs will reflect any inactive periods resulting from the reinitialization of a failed pair. Figure 11-1 shows a partial throughput graph for a pair that failed and was reinitialized during test execution.

Figure 11-1.   Throughput Graph for a Reinitialialized Pair
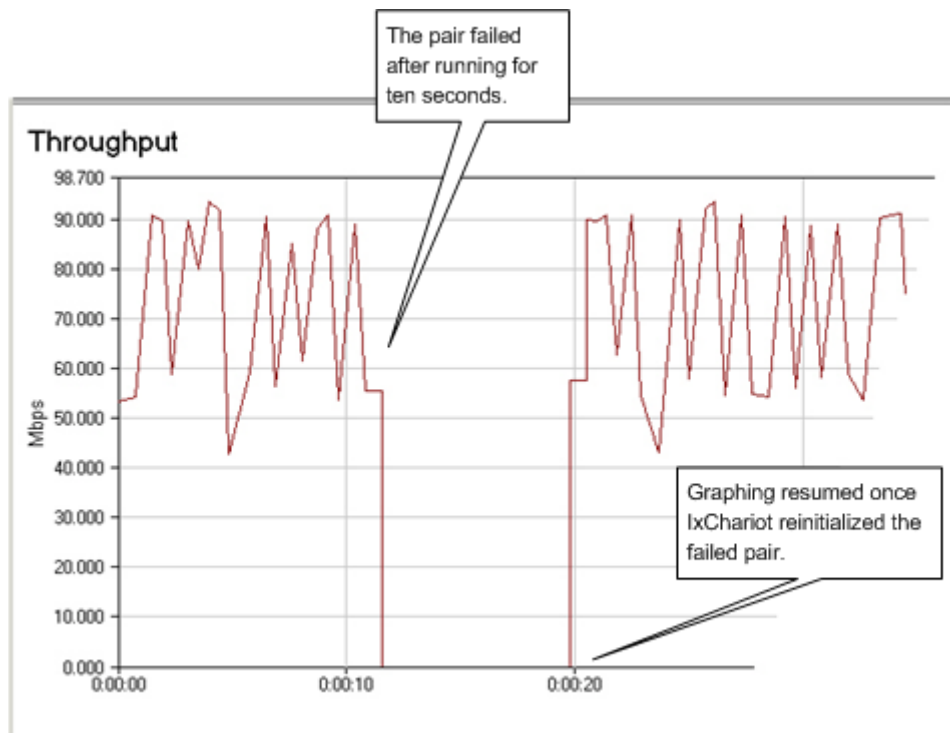
Understanding the Run Status

Related Topics

Test Window Status Bar on page 3-14
Stopping a Running Test on page 5-31

As you run a test, your endpoint pairs go through a series of stages. Their status at each stage is reported on the Status bar at the bottom of the Test window. Each run status is described below:

- **Resolving names**

    The Console is determining the actual network addresses if domain names or aliases were supplied in the test setup. See *Eliminating DNS Latency from Test Results* on page 10-91 for more information.

- **Initializing**

    The Console is contacting each of the Endpoint 1 computers, and sending each of them its script. Each Endpoint 1 then splits the script in half, and forwards the proper half to Endpoint 2. Three steps occur in the initializing process:

- The Console contacts Endpoint 1 and/or Endpoint 1 contacts Endpoint 2.

- Both Endpoint 1 and Endpoint 2 have been successfully contacted.

- The script commands are sent to Endpoint 1, then forwarded to Endpoint 2.

- **Initialized**

  An individual endpoint pair reaches this stage when it has finished "Initializing," and has reported back to the Console. When all endpoint pairs reach this stage, the Console issues the calls to start executing all the scripts.

- **n/a**

  The test either has not started running or has completed running, but does not have enough information to return data.

- **Running**

  The scripts are running between the endpoints in each endpoint pair.

  If you've set up the test to "Run until any script completes," IxChariot shows the estimated time remaining in the status bar.

- **Polling**

  The Console can poll endpoints on a scheduled basis; you can also manually poll by clicking the Poll icon on the toolbar. When you poll the endpoints during a running test, the Console sends a message to each of the Endpoint 1 computers in the test. An endpoint pair is in Polling status when it is returning the number of timing records generated so far for this test.

- **Requested stop**

  The test is over; the Console has sent a request to each endpoint pair to stop the script now executing. An endpoint pair has the "Requested stop" status while the Console waits to hear back from Endpoint 1 in each pair that it is now stopping.

- **Stopping**

  This stage can be reached in three conditions:

  - An endpoint pair has completed its script, and you have previously chosen to end the run when the first endpoint pair completes its script or runs for a fixed duration. Therefore, the Console is stopping all the remaining endpoint pairs.

  - You've clicked the **Stop** button while a test is running in the Test window. The **Stop a running test** popup appears to show you the status of the command. Alternatively, you've closed the Test window while a test is running. The Console is stopping all the running endpoint pairs.

  - An error occurred on one of the pairs, so the Console is stopping the other active pairs.

  Running tests are not stopped in the middle of a transaction; endpoints only stop after an END_TIMER command. Stopping can take a long time if you are running a test with large SEND sizes (say, you are simulating a file transfer of more than 10 million bytes over a LAN). See "Program Control Commands" in the *Application Scripts* guide for more information about the END_TIMER and SEND commands.

Stopping can also take between 20 and 50 seconds when running pairs using SPX on Windows NT or Windows 2000/2003, doing loopback (both endpoints have the same address). If the endpoint is on a `RECEIVE` call, the protocol stack can pause for almost a minute before returning.

- **Finished**

  The run has completed. If the test ran long enough to generate results, they are shown.

- **Error detected**

  At least one of the endpoints has detected an error, which it has reported to the Console. Depending on your Run Options settings, the Console may now be trying to stop the other running pairs. Subsequent timing records received by the console will be discarded.

- **Abandoned**

  You stopped the running test, and then you clicked the Abandon Run button. An endpoint pair was running, but the Console has abandoned it without waiting for the remainder of its timing records. The endpoints may still be executing their scripts and attempting to send timing records back to the Console, which is now discarding them. If you abandon endpoints that you think were very busy, it is best to wait about 2 minutes before starting another performance test.

Graph Configuration

Related Topics

Graph Configuration: Axis Details on page 11-10

A graph is always shown at the bottom of the Test window when results are available (unless you have disabled graphing or the test is using more than 5,000 pairs). Click **Graph Configuration** on the View menu to choose what type of graph is shown.

Choose among line-, bar-, or pie-type graphs. Histograms are shown as bar graphs. The type of graph you select applies to all tabbed views of your results.

- **Graph Content: Pairs or Groups**

  Determines how the results are aggregated in your graph. The graph you've chosen shows either the pairs you've marked (giving you more detail) or the groups you've marked (giving you a way to combine many pairs).

- **View Options**

  Shows or hides the graph legend or a background grid. The legend interprets the colors and patterns used for each pair or group. The grid lines help you visualize your data.

- **Axis Details**

  Determines how much data is included in the graph and how it appears. For more information about the Axis Details dialog box, see *Graph Configuration: Axis Details* on page 11-10.

Click **Apply** in the Graph Configuration dialog box to see the immediate effect of your choices. If you like what you see, click **OK**. If you want to reset your

changes and try again, click **Undo**. Click **Cancel** to close the dialog box and cancel the changes you've applied.

Graph Configuration: Axis Details

Related Topics

Graph Configuration on page 11-9

Click **Graph Configuration** on the View menu to open the Graph Configuration dialog box. Choose the type of graph you want to see, including line graph, pie graph, four types of bar graph, and two types of histogram. Then click **Axis Details** to choose the data shown in the line graph, bar graph, or histogram of your results.

In the Axis Details dialog box, you define the time frame shown on the graph or histogram during a running test or after the test has run, as well as the range for the line graphs.

The Histogram of Timing Records and Histogram of Averages represent the frequency distribution of timing records and of data for Throughput[1], Transaction Rates, Response Times, Lost Data, VoIP data, or RFC 1889 Jitter. (Jitter data is only available for tests that used the RTP protocol.) The histograms plot data across the horizontal axis and the percent of total samples along the vertical axis. Modify the range of the horizontal axis to see the information you want to see. See *Comparing Histograms* on page 11-12 for more information about the data you see.

The bar graph depicts Throughput, Transaction Rate, Response Time, Lost Data, VoIP data, or Jitter for each pair/group in a test. (Jitter data is only available for tests that used the RTP protocol.) Each pair/group appears in the bar graph across the horizontal axis, and the data appears along the vertical axis. By default, all results are shown. Modify the range of the vertical axis to see the information you want.

If you are using 640 x 480 screen resolution (VGA mode), IxChariot only shows the bar graphs for 89 pairs in a test.

- **Horizontal axis**

  Graphs the elapsed time of the test. You can specify the time range shown.

- **Vertical axis**

  Graphs the data for the tab selected in the Test window. You can specify the results range shown. The results range defaults to contain minimum and maximum ends, which cover the total range of data. Decrease or increase the range of the vertical axis by changing the minimum and/or maximum range ends.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

- **While running**

  The time frame of activity shown to you while a test is running. Results must be reported in real time for these parameters to apply. By default, the last 60 seconds of a test run are shown.

- **After running**

  The time frame of activity shown after the test has run and the results have been reported. By default, the total length of time that the test ran is shown. Increase or decrease the elapsed time shown on the line graph by specifying a time frame.

- **Minimum Auto**

  The minimum end of the elapsed time frame shown in the graph. Clear this checkbox to set your own minimum. Enter a value for the hours, minutes, and/or seconds to indicate the time at which you wish the graph to begin showing results. For example, if you type a value of 1, the elapsed time in the line graph begins at one minute: no test results before one minute are shown. Decimal values may be entered in the Sec field to show milliseconds.

- **Maximum Auto**

  The maximum end of the elapsed time frame shown in the graph. Clear this checkbox to set your own maximum. Enter a value to determine the end of the elapsed time range shown. For example, if you type a value of 3, the elapsed time in the line graph ends at 3 minutes. No test results after three minutes are shown. Decimal values may be entered in the Sec field to show milliseconds.

- **Number of Divisions**

  Histogram only. Determines the frequency divisions shown across the horizontal axis. By default, 20 divisions are shown. Increase the number of divisions to show more detail in the histogram; decrease the number of divisions to show more summary information in the histogram.

Click **Apply** to apply the changes to the graph and get a preview. Click **OK** to return directly to the Test window or **Previous** to return to the Graph Configuration dialog box. Clicking **Undo** lets you cancel your changes after you've applied them and returns you to the Test window. Click **Cancel** to return to the Graph Configuration dialog box without making any changes.

Graph Configuration: Axis Details for Bar Graphs

Related Topics

Graph Configuration: Axis Details for Histograms

Related Topics

Graph Configuration: Axis Details for Line Graphs

Related Topics
Graph Configuration: Axis Details on page 11-10

Comparing Histograms

When you compare the results of tests using IxChariot's Comparison window, you can change the graph configuration to create different types of graphs comparing test data. However, when you choose to compare histograms, the Comparison window makes a slight adjustment. To avoid placing histogram bars over each other and make the data easier to read, the Comparison window shows a line graph of histogram results; the data shown on each axis is the same as that shown in the histogram.

Here's an example of a comparison of throughput[1] histograms from two IxChariot tests:

Figure 11-2.   Comparing Histograms
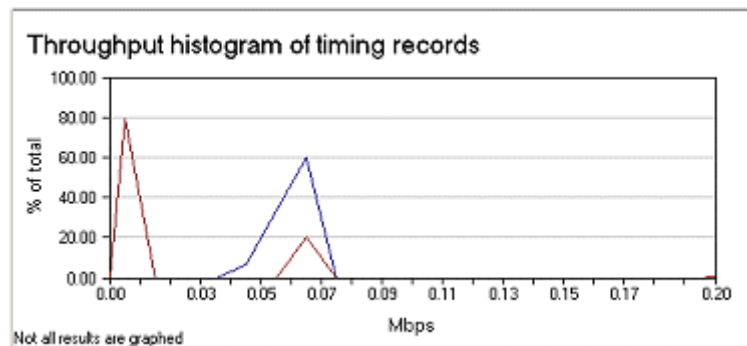
Each test's results are shown as a separate line, but the y axis indicates the percentage of the total number of pairs in the test with those throughput results, as in the histogram. Refer to *Comparing Test Results* on page 5-43 for more information about comparing tests.

Availability of Results for Graphing

Related Topics
How to End a Test Run on page 7-3
Graph Configuration: Axis Details on page 11-10

Before IxChariot receives enough results from the endpoints to make a complete graph, you see the message "No results available to graph" in the lower part of the Test window. Often only a few seconds pass before a graph can be drawn. If your Run Options specify that the test runs until all pairs end and your timing

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

records are being reported in batch mode, results will not be graphed until all pairs report results. See *The Jitter (Delay Variation) Tab* on page 11-30 for more information.

However, a completed test where no graph appears may show the message "Results are not available for graphing." This message indicates that the test results could not be graphed because no timing records were returned. An error may have occurred, or if you stopped the test, no pairs may have had time to complete their scripts.

If you change the Graph Configuration so that the minimum and maximum values on the vertical axis are too high or too low, the message "Not all results are graphed" appears just below the graph, at the bottom of the Test window (Figure 11-3).

Figure 11-3. Graph configuration with min/max to high or low
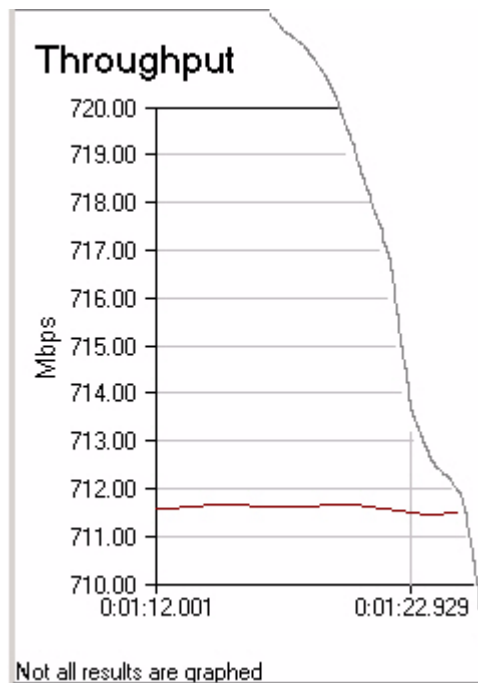
If this occurs, you can change the graph by clicking **Graph Configuration** on the View menu, then clicking **Axis Details**. See *Graph Configuration: Axis Details* on page 11-10 for more information.

Test Window Tabs

Related Topics

Once you've run a test that shows some results, a series of tabs appears next to the **Test Setup** tab in the Test window. Click a tab to bring it to the front. Each tab changes the way results are shown in the graph and focuses your attention on a particular aspect of your test results. The data columns are slightly different for each tab.

Help is available for each tab, including explanations of the columns of data shown. Consult *Understanding Results* on page 11-1 for more information.

Calculations

Related Topics

The Test Totals are divided into three major sections that correspond to the three basic types of measurements taken in an IxChariot test. Following are explanations of how IxChariot calculates throughput, transaction rate, and response time.

- **Throughput**[1]

  The throughput is calculated for pairs (non-streaming scripts) with the following equation:

  ```
  (Bytes_Sent + Bytes_Received_By_Endpoint_1) /
  (Throughput_Units) / Measured_Time
  ```

  Average throughput[2] of a **group** is calculated this way for a non-streaming script:

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.
2. Really, the average of the total goodput.

```
bytes_sent + bytes_received (total for all pairs) /
(Throughput_Units) / elapsed time
```

If you are running a streaming script, the following equation is used for Pair throughput:

```
bytes received (by E2) / (Throughput_Units) /
Measured time
```

Average throughput for a **group** of pairs running a streaming script is calculated this way:

```
bytes_received_by_endpoint_2 (total for all pairs) /
(Throughput_Units) / elapsed_time
```

Average throughput for a **multicast group** is calculated as

```
max total bytes received by any multicast E2 /
(Throughput_Units) / elapsed time
```

In all calculations, *elapsed time* is the elapsed time of the longest-running pair in the test. The *measured time* is the sum, in seconds, of all the timing record durations returned for the endpoint pair.

Here's how each of the script variables is defined:

- `Bytes_Sent`: the number of bytes sent by Endpoint 1 of a pair.

- `Bytes_Received`: the number of bytes received by the endpoint of a pair.

- `Throughput_Units`: the current throughput units value, in bytes per second. For example, if the throughput units are KBps, 1024 is the `Throughput_Units` value. In this example, the throughput units number is shown in the column heading as Mbps, which is 125,000 bytes per second (that is, 1,000,000 bits divided by 8 bits per byte). See *Raw Data Totals* on page 11-44.

- **Transaction Rate**

  The calculations are shown in transactions per second. This rate is calculated as follows:

  ```
  Transaction_Count / Measured_Time
  ```

- **Response Time**

  The response time is the inverse of the transaction rate. The calculations are shown in seconds per transaction. This value is calculated as follows:

  ```
  Measured_Time / Transaction_Count
  ```

- **Lost Data**

  The lost data is the difference between the number of bytes sent by Endpoint 1 and the number of bytes Endpoint 2 actually received. Lost data is only calculated when a pair is running a streaming script (for example, in a VoIP or IP Multicast test). Only payload data is included in the calculations.

The Test Setup Tab

The **Test Setup** tab is shown when you create a new test. The columns on the right reflect the values you entered when you created each pair:

- Group

- Pair group name (if any)

- Pair run status

- Number of timing records completed

- Endpoint 1 and Endpoint 2 addresses

- Network protocol

- Service quality, if any was used

- Application script or stream filename

- Pair comment (if any was assigned)

- How the Console communicates with Endpoint 1—network address, protocol, and service quality

- How Endpoint 1 communicates with Endpoint 2—network address

- UDP Compliant with RFC768 (refer to *UDP Configuration* on page 5-6).

To edit an existing pair, double-click it to open the Edit Pair dialog box. If the pair is a member of a multicast group, the Edit a Multicast Group dialog box is shown. For more information about each type of information listed on this tab, see *Adding or Editing an Endpoint Pair* on page 5-19.

The Endpoint Configuration Tab

Related Topics

Endpoint Configuration Details Dialog Box on page 11-17

The **Endpoint Configuration** tab is shown while a test is running or after a test has been run and has results. It shows details about the software run at the endpoints in your test. The first four columns in each row show information about Endpoint 1; information about Endpoint 2 is shown in the next four columns. To view detailed information about endpoint configuration, highlight one pair and click **Show Endpoint Configuration** on the View menu.

Here's what each column of data shows:

- **IxChariot Version**

  The version number for the endpoint.

- **Build Level**

  The internal build number used by Ixia, helpful when contacting us for service and support.

- **Product Type**

  Available in retail and beta types.

- **Operating System**

  The name of the operating system on which the endpoint is running.

View more detailed information about each endpoint by highlighting a pair and clicking **Show Endpoint Details** on the View menu.

Endpoint Configuration Details Dialog Box

Related Topics

The Endpoint Configuration Tab on page 11-16

This dialog box shows detailed network information about each of the endpoints in the pair you've highlighted.

- **Version**

  Version of Performance Endpoint software installed on each endpoint computer.

- **Build Level**

  Build level of Performance Endpoint software installed on each endpoint computer.

- **Product Type**

  Type of IxChariot Console, such as Retail, Evaluation, etc., used to configure and run the test.

- **Operating System**

  Operating system used by each endpoint computer.

- **CPU Utilization**

  Whether the endpoint operating systems support the collection of CPU utilization data.

- **OS Version (Major), OS Build Number, and CSD Version**

  Version and build level of the operating system, plus the service pack level, installed on each endpoint computer.

- **Memory**

  Amount of RAM at each endpoint computer, shown in KB.

- **IPX, SPX, TCP, UDP, and RTP Default Send Size**

  Default buffer size used by the SEND command for each protocol. The endpoints return their default SEND sizes to the Console.

- **WinSock API information**

  Version of the WinSock API actually used for the test. At runtime, IxChariot queries the WinSock stack for a version of the WinSock API.

The Throughput Tab

Related Topics

Calculations on page 11-14
Throughput Units Tab on page 6-30
Confidence Intervals on page 11-49

The **Throughput**[1] tab is shown while a test is running or after a test has run and has results. For more information about how results are calculated, see *Calculations* on page 11-14.

• **Group/Pair**

The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

• **95% Confidence Interval**

Estimated range of values with a 95% probability of statistical accuracy. Shows "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Confidence Intervals* on page 11-49 for information about the calculation of confidence intervals.

• **Average**

Average throughput[1] of all timing records received from a pair. In the Group row (shown in bold), the throughput is calculated as follows: the total number of bytes sent and received by the E1s in the group is divided by the elapsed time of the longest-running pair in the group. Note that any idle intervals within the timing records of one or more pairs are included in the elapsed time; this means that the average of the group may be lower than the minimum throughput of any particular pair.

• **Minimum**

The minimum throughput of all timing records received from a pair; for a group, the minimum value received from all pairs in the group.

• **Maximum**

The maximum throughput of all timing records received from a pair; for a group, the maximum value received from all pairs in the group.

• **Measured Time**

Sum of the timing record durations for each pair. To see the individual timing records for an endpoint pair, select its row, then click **Timing records** on the View menu.

• **Relative Precision**

The consistency among a pair's timing records. An increase in the Relative Precision correlates directly with increased variability among timing records. Displays "n/a" while a test is running; shows values when a test ends (and there are at least 2 timing records).

Your throughput results can be shown in different units of measurement. To change the unit of measure in which your throughput results are shown, click **Throughput Units** on the View menu. See *Raw Data Totals* on page 11-44 for more information.If you view the data using the bar graph with max/avg/min, the display shows (for each pair) the maximum value at the top of the upper bar segment, the average value at the top of the middle bar segment, and the minimum value at the top of the lower bar segment.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.
1. Really, the average of the total goodput.

The Response Time Tab

Related Topics

Calculations on page 11-14
Confidence Intervals on page 11-49

The **Response Time** tab is shown while a test is running or after a test has run and has results. The response time is the inverse of the transaction rate; it is the time, in seconds, needed for one transaction. For more information about how the results are calculated, see *Calculations* on page 11-14.

• **Group/Pair**

The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

• **95% Confidence Interval**

Estimated range of values with a 95% probability of statistical accuracy. Shows "n/a" while a test is running; shows values when a test ends (and there are at least 2 timing records). See *Confidence Intervals* on page 11-49 for information about the calculation of confidence intervals.

• **Average**

Average response time for each pair; for a group, the average response times of all pairs in the group.

• **Minimum**

The minimum response time of all timing records received from a pair; for a group, the minimum value received from all pairs in the group.

• **Maximum**

The maximum response time of all timing records received from a pair; for a group, the maximum value received from all pairs in the group.

• **Measured Time**

Sum of the timing record durations for each pair. To see the individual timing records for an endpoint pair, select its row, then click **Show timing records** on the View menu.

• **Relative Precision**

The consistency among a pair's timing records. An increase in the Relative Precision correlates directly with increased variability among timing records. Displays "n/a" while a test is running; shows values when a test ends (and there are at least 2 timing records).

If you choose to view this data using the bar graph with max/avg/min, the display shows (for each pair) the maximum value at the top of the upper bar segment, the average value at the top of the middle bar segment, and the minimum value at the top of the lower bar segment.

The Transaction Rate Tab

Related Topics

Calculations on page 11-14
Confidence Intervals on page 11-49

The **Transaction Rate** tab is shown while a test is running or after a test has run and has generated results. The transaction rate is the number of script transactions that are executed per second. For more information about how the results are calculated, see *Calculations* on page 11-14.

- **Group/Pair**

  The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

- **95% Confidence Interval**

  Estimated range of values with a 95% probability of statistical accuracy. Shows "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Confidence Intervals* on page 11-49 for more information.

- **Average**

  Average transaction rate for each pair. In the Group row (shown in bold), the average transaction rate of each pair or of each group is added to form an aggregate average.

- **Minimum**

  The minimum transaction rate of all timing records received from a pair; for a group, the minimum value received from all pairs in the group.

- **Maximum**

  The maximum transaction rate of all timing records received from a pair; for a group, the minimum value received from all pairs in the group.

- **Measured Time**

  Sum of the timing record durations for each pair. To see the individual timing records for an endpoint pair, select its row, then click Timing records on the View menu.

- **Relative Precision**

  The consistency among a pair's timing records. An increase in the Relative Precision correlates directly with increased variability among timing records. Shows "n/a" while a test is running; shows values when a test ends (and there are at least 2 timing records).

If you view this data using the bar graph with max/avg/min, the display shows (for each pair) the maximum value at the top of the upper bar segment, the average value at the top of the middle bar segment, and the minimum value at the top of the lower bar segment.

The 802.11 Tab

Related Topics

The Raw Data Totals Tab on page 11-22

The **802.11** tab is shown while a test is running or after a test has run and has gen-erated results. Only one of the endpoints measures the RSSI value. For streaming pairs, measurements are done at endpoint 2. For other pairs, measurements are done at endpoint 1.

- **Group/Pair**

  The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

- **Run Status**

  Indicates whether the endpoint pair ran its script to completion.

- **Timing Records Completed**

  The number of timing records this test generated. Gives an indication of the reliability of the results.

- **Measured Time**

  Total time recorded by all of the timing records for this endpoint pair. This may differ greatly from the time during which the script was actually execut-ing (that is, the elapsed time, which is shown at the top of the results), depending on how much activity or how many SLEEPs the script demanded outside of its START_TIMER and END_TIMER commands. This value is shown in seconds.

- **Average E1 (dBm)**

  The average RSSI for endpoint 1, taking into account all timing records.

- **Minimum E1 (dBm)**

  The minimum RSSI for endpoint 1, taking into account all timing records.

- **Maximum E1 (dBm)**

  The maximum RSSI for endpoint 1, taking into account all timing records.

- **Number Unique Access Points (E1)**

  The number of access points associated with endpoint 1 during the test.

- **Average E2 (dBm)**

  The average RSSI for endpoint 2, taking into account all timing records.

- **Minimum E2 (dBm)**

  The minimum RSSI for endpoint 2, taking into account all timing records.

- **Maximum E2 (dBm)**

  The maximum RSSI for endpoint 2, taking into account all timing records.

- **Number Unique Access Points E1**

  The number of access points associated with endpoint 2 during the test.

> **NOTE**: This feature's implementation on Linux differs from that on Windows in that it is based on wireless extensions for Linux; as such, it is functional only on systems which have wireless extensions installed, namely the wireless.h header (which exposes SIOCGIWSTATS and SIOCGIWAP ioctl calls), and corresponding iwconfig application.

The Raw Data Totals Tab

Related Topics

Miscellaneous Run Options on page 7-11

The **Raw Data Totals** tab is shown while a test is running or after a test has been run and has results. You can the data that goes into calculating the results, including the number of timing records and transactions for each pair, as well as the number of bytes sent and received by Endpoint 1.

• **Group/Pair**

  The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

• **Group/Pair**

  Endpoint pair number of a row. This number corresponds to the pair number indicated in the previous subsections.

• **Number of Records**

  Number of timing records generated during the test by this endpoint pair. To see the individual timing records for an endpoint pair, select its row in the Test window, then click **Show timing records** on the View menu.

• **Transaction Count**

  Number of transactions performed by an endpoint pair. Sometimes the transaction count equals the number of records. If these two fields differ, it is because the script uses an `INCREMENT_TRANSACTION` command within a `LOOP` to emulate multiple transactions per timing record.

• **Bytes Sent by E1**

  Bytes of data sent by Endpoint 1 in a pair. If the pair is a member of a multicast group, the bytes sent by E1 total includes only one pair's sent value.

• **Bytes Received by E1**

  Bytes of data received by Endpoint 1 in a pair.

• **Percent CPU Utilization of E1**

  Percentage of the CPU utilization for Endpoint 1 for the duration of the test. This column is only shown if the Collect Endpoint CPU Utilization checkbox on the Run Options dialog box is selected. The column shows Not supported if the endpoint does not support CPU utilization. See "Miscellaneous Run Options" for information on how IxChariot calculates the CPU utilization percentage.

- **Percent CPU Utilization of E2**

   Percentage of the CPU utilization for Endpoint 2 for the duration of the test. This column is only shown if the Collect Endpoint CPU Utilization checkbox on the Run Options dialog box is selected. The column shows Not supported if the endpoint does not support CPU utilization. See "Collect endpoint CPU utilization" in *Throughput, Transaction Rate, and Response Time* on page 11-41 for information on how IxChariot calculates the CPU utilization percentage.

- **Measured Time**

   Total time recorded by all of the timing records for this endpoint pair. This may differ greatly from the time during which the script was actually executing (that is, the elapsed time, which is shown at the top of the results), depending on how much activity or how many SLEEPs the script demanded outside of its START_TIMER and END_TIMER commands. This value is shown in seconds.

- **Relative Precision**

   Statistical value indicating the consistency among the timing records for a pair. This is the same number shown in the summaries for the throughput, transaction rate, and response time. See *Relative Precision* on page 11-47 for more information.

- **BSSID E1**

   For 802.11 tests, this indicates the Base Service Station ID that was in use by Endpoint 1. Note that E1 values are only valid for non-streaming scripts.

- **RSSI E1**

   For 802.11 tests, this indicates the Receive Signal Strength Indicator for Endpoint 1, as a dBm value between -10 and -100. -10 is a strong signal and -100 is a weak signal. Note that E1 values are only valid for non-streaming scripts.

- **BSSID E2**

   For 802.11 tests, this indicates the Base Service Station ID that was in use by Endpoint 2. Note that E2 values are only valid for streaming or VoIP scripts.

- **RSSI E2**

   For 802.11 tests, this indicates the Receive Signal Strength Indicator for Endpoint 2, as a dBm value between -10 and -100. -10 is a strong signal and -100 is a weak signal. Note that E2 values are only valid for streaming or VoIP scripts.

The Datagram Tab

Related Topics

Datagram Run Options on page 7-18
Hints for Interpreting Data Shown on the Datagram Tab on page 11-25

The **Datagram** tab is shown while a test is running or after a test containing datagram pairs has been run and has results. From this tab, you can see details on the handling of datagrams by the endpoints. Datagram support is only available for tests using the IPX, RTP, or UDP protocols; for pairs using other protocols, "n/a" is shown.

- **Group/Pair**

  The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

- **Total DGs Sent by E1**

  Number of datagrams sent by Endpoint 1. This column is shown for tests using either a streaming or a non-streaming script. For non-streaming scripts, includes datagrams that were retransmitted.

- **Duplicate DGs Sent by E1**

  Number of datagrams Endpoint 1 had to retransmit because it didn't receive an acknowledgment from Endpoint 2 before the Retransmission Timeout period expired. The number of duplicates sent is the number of duplicates received plus the number of datagrams lost. Only applicable for tests using a non-streaming script.

- **Total DGs Received by E1**

  Number of datagrams, both original and retransmitted, received by Endpoint 1. In an ideal network setting, where no datagrams are lost or delayed, the number shown in this column will be the same as the number shown in the **Total DGs Sent by E2** column. Both of these columns are applicable only for tests using non-streaming scripts.

- **Duplicate DGs Received by E1**

  Number of datagrams with the same sequence number that were received by Endpoint 2.

- **DGs Lost, E1 to E2**

  Datagrams sent from Endpoint 1 to Endpoint 2 that were not received by Endpoint 2. The values shown here are an approximation: some of these "lost" datagrams sent by E1 may have been delayed in the network and would have been received by E2 if given enough time. Once a script completes, however, there's no longer any need for E2 to wait to receive those datagrams. This column is only applicable for non-streaming scripts.

- **Total DGs Sent by E2**

  Total number of datagrams sent by Endpoint 2. This column is applicable for tests using non-streaming scripts. For non-streaming scripts, this includes datagrams that were retransmitted.

- **Duplicate DGs Sent by E2**

  Datagrams Endpoint 2 had to retransmit because it didn't receive an acknowledgment from Endpoint 1 before the Retransmission Timeout period expired. The number of duplicates sent is the number of duplicates received plus the number of datagrams lost. Only applicable for tests using non-streaming scripts.

- **Total DGs Received by E2**

  Datagrams received by Endpoint 2. Shown for tests using either a streaming or non-streaming script.

- **Duplicate DGs Received by E2**

    Datagrams with the same sequence number that were received by Endpoint 2. This column is applicable for streaming or non-streaming tests. The values shown here are an approximation: some of these "lost" datagrams sent by E2 may have been delayed in the network and would have been received by E1, given enough time. Once a script completes, however, there's no longer any need for E1 to wait to receive those datagrams.

- **Datagrams Out of Order**

    Datagrams that were received out of sequence. Only applicable for tests using streaming scripts. Datagrams reported as out of order may also be reported as lost data in the previous timing record. That is, in the previous timing record a datagram was delayed such that it was not received before the end of the record; that datagram would be considered lost in that timing record. When it is received in the subsequent timing record it will be perceived as out of order.

Hints for Interpreting Data Shown on the Datagram Tab

Related Topics
Datagram Run Options on page 7-18

- If the number shown in the Duplicate Datagrams Received by E1 column is large when considered as a percentage of Total Datagrams Received by E1 column, consider setting the Retransmission Timeout higher, to prevent Endpoint 2 from retransmitting too often. However, if datagrams are being lost, changing this setting will only increase the number of duplicate datagrams. See *Setting Retransmission Values* on page 10-5 for a full discussion.

- If you are using a non-streaming script and the number shown in the **Datagrams Lost, E1 to E2** column is too large, either the Window Size parameter in the Datagram Run Options is too large, or the network is being used by too many applications at the same time.

- If the test is using a streaming script and the results show a large number of duplicates at Endpoint 2, there is probably a problem, such as a loop, in the network configuration. Look at the configuration of network components to find the problem. A router is a good place to start.

- If you are using a streaming script and the results show a high number of lost datagrams in the test, change the data rate to a slower data rate. The sender may be sending the datagrams faster than the receiver can receive the data.

The VoIP Tab

Related Topics
Jitter Buffers on page 10-52
Voice over IP Testing on page 10-34
VoIP Score Calculation on page 10-48
Understanding Jitter Measurements on page 10-50

For completed tests containing voice over IP endpoint pairs, the VoIP tab is shown. The data provided for voice over IP pairs indicates the quality of the calls made during the test; the MOS (Mean Opinion Score), for example, is an ITU standard derived from actual listeners' subjective judgments about call quality.

Other measurements are used in calculating the MOS. Refer to *VoIP Score Calculation* on page 10-48 for more information.

• **Group/Pair**

Shows the group name and each pair within the group.

• **All Pairs**

Totals for all pairs in the test are shown in this row.

• **Run Status**

Indicates whether the endpoint pair ran its script to completion.

• **Timing Records Completed**

The number of timing records this test generated. Gives an indication of the reliability of the results.

• **MOS Average**

Average Mean Opinion Score for a pair or group in the test.

• **MOS Minimum**

The lowest (worst) MOS for an individual pair or group in the test.

• **MOS Maximum**

The highest (best) MOS for an individual pair or group in the test.

• **R-Value Average**

Average R-value for an individual pair or group in the test. Rounded to two decimal places. The R-value is used to calculate the MOS estimate. See *VoIP Score Calculation* on page 10-48 for more information.

• **One-Way Delay (Network) Average**

Latency as the difference between the time a datagram was sent by E1 and the time it was received by E2. Delay is expressed as an average for all datagrams sent in a single timing record. ITU standard G.114 recommends that the maximum one-way delay should be 150 ms with 50 ms of processing time to achieve adequate voice quality.

• **End-to-End Delay Average**

Latency as the sum of the following delay factors: packetization delay, one-way (network) delay, jitter buffer delay, and additional fixed delay (if any). Delay is expressed as an average for all datagrams sent in a single timing record.

• **RFC 1889 Jitter Average**

Variation in datagram inter-arrival times, expressed as an estimate of mean statistical deviance per timing record (per RFC 1889, the RTP specification). Jitter values calculated for each timing record are averaged for each pair or for each group. Refer to *Understanding Jitter Measurements* on page 10-50 for more information.

• **Percent Bytes Lost, E1 to E2**

Data sent by Endpoint 1 that never reached Endpoint 2, expressed as a percentage of the total amount of bytes sent. For most VoIP users, 2-5% is acceptable.

- **Maximum Consecutive Lost Datagrams**

   The highest number of consecutive datagrams lost for a particular endpoint pair. A high number of consecutive lost datagrams indicates poor call quality.

- **Jitter Buffer Lost Datagrams**

   Datagrams lost due to jitter buffer overruns, or the number of datagrams that had a delay variation greater than the jitter buffer size, and jitter buffer under-runs, datagrams that arrived too quickly while the jitter buffer was still full. Refer to *Jitter Buffers* on page 10-52 for more information.

The One-Way Delay Tab

Related Topics

One-Way (Network) Delay on page 11-43
Voice over IP Testing on page 10-34
One-Way Delay on page 10-47

Endpoints can calculate delay statistics in a single direction for VoIP pairs and for pairs using the RTP protocol. Delay is calculated for each timing record in a test. These measurements are useful in testing time-sensitive applications because they can help to pinpoint sources of delay. One-way or network delay excludes sources of delay apart from the "wire" itself, while end-to-end delay includes all sources of delay, such as the codec used, jitter buffers, and fixed delays. See *One-Way Delay* on page 10-47 for more information.

Not all endpoint operating systems support the clock synchronization that is necessary for calculating one-way delay. Currently, this feature is available on all operating systems except for AIX. Operating systems that do not support one-way delay show "n/a" in the results. "n/a" is also shown if E2 could not get a clock value from E1, or if not enough clock samples were generated to get reliable measurements.

> **Note:** If the Endpoint 1 clock is running ahead of the Endpoint 2 clock in a test pair, the one-way delay value will be reported as zero throughout the test.

- **Group/Pair**

   The name of a group of pairs—based on how you've grouped endpoint pairs in the Test window—or its pair number (based on the order in which you added pairs to the test).

- **All Pairs**

   Totals for all pairs in the test.

- **Run Status**

   Indicates whether the endpoint pair ran its script to completion.

- **Timing Records Completed**

   The number of timing records this test generated. Gives an indication of the reliability of the results.

- **One-Way Delay Average (ms)**

   The average delay of all timing records received from a pair; for a group, the average of all values received from all pairs in the group.

- **One-Way Delay Minimum (ms)**

  The minimum delay of all timing records received from a pair; for a group, the minimum value received from all pairs in the group.

- **One-Way Delay Maximum (ms)**

  The maximum delay of all timing records received from a pair; for a group, the maximum value received from all pairs in the group.

- **Estimated Clock Error (ms)**

  An estimate of the maximum discrepancy between the synchronized high-precision clocks on E1 and E2 that are used to measure one-way delay. The estimated clock error can be added to and subtracted from one-way delay measurements to yield an upper and lower bound for the actual one-way delays.

  As explained in Table 11-1, the value that is displayed in this column depends upon the software release level of both the IxChariot console and the Performance Endpoints.

Table 11-1.   Value Displayed for Estimated and Maximum Clock Error

IxChariot Console	Performance Endpoints	Value Displayed for Estimated and Maximum Clock Error
IxChariot 6.70 or higher	Both E1 and E2 are running pre-6.70 software,	Values are shown from the beginning of test run and remain unchanged once the run finishes. Both estimated error and maximum error can have values below 0.5 ms.
Any software version	E1 is running 6.70 (or higher) software. E2 is running pre-6.70 software.	Values are shown at the beginning of test run and remain unchanged once the run finishes. Both estimated error and maximum error can have values below 0.5 ms.
IxChariot 6.70 or higher	E1 is running pre-6.70 software. E2 is running 6.70 (or higher) software.	The value is shown as "n/a" while the test is running. Computed values are displayed only after the run finishes. Estimated and maximum error have to be at least 0.5 ms. (The 0.5 millisecond value accounts for the mathematical rounding that occurs during the clock-synchronization process and the calculation of one-way delay.)

The estimated clock error can be unexpectedly large for any of the following reasons:

- During clock synchronization, network conditions changed such that data flowed between the endpoints significantly more rapidly in one direction than in the other. Accurate clock synchronization requires a symmetric connection between endpoints, where data is sent equally fast in both directions.

- One or both of the endpoints was busy with other processing during clock synchronization.

- **Maximum Clock Error (ms)**

    The largest possible discrepancy between the synchronized high-precision clocks on E1 and E2, which are used to measure one-way delay. This metric allows for the worst-case asymmetric connection, where data travels much faster in one direction than in the other. (See "Estimated Clock Error," above.)

    The maximum clock error is directly proportional to the time it takes to send a datagram from E1 to E2 and back to E1. On asymmetric network links, the actual clock error falls somewhere between the Estimated Clock Error and the Maximum Clock Error. The more asymmetric the link, the closer the actual error falls to the Maximum Clock Error.

    As with the estimated clock error, and as explained in Table 11-1 on page 11-28, the value that is displayed in this column depends upon the software release level of both the IxChariot console and the Performance Endpoints.

- **Clock Used**

    Indicates the clock source used to measure the delay. The possible values are Endpoint, Hardware Timestamp, or External Synchronization.

The Jitter Tab

Related Topics

Understanding Jitter Measurements on page 10-50
Jitter and Delay Variation on page 10-52

The **Jitter** tab is shown after a test using RTP has been run and has results. Jitter support is only provided for a test using the RTP protocol; if you are running the VoIP Test Module, you'll see two different types of jitter measurement. While **RFC 1889 Jitter** results show mean statistical deviance of packet inter-arrival times over the elapsed time of the test, **Jitter (Delay Variation)** results show the number of datagrams (expressed as a percent of the total number of datagrams) that experienced delay variations of various durations. For more information, see *Jitter and Delay Variation* on page 10-52.

- **Group/Pair**

    The name of a group of pairs—based on how you've grouped endpoint pairs in the Test window—or its pair number (based on the order in which you added pairs to the test).

- **All Pairs**

    Totals for all pairs in the test are shown in this row.

- **RFC 1889 Jitter Average**

    Average jitter statistic in milliseconds of all timing records for a pair; for a group, the average jitter of all pairs in the group that had jitter. Calculated according to RFC 1889. Refer to *Jitter and Delay Variation* on page 10-52 for more information.

- **RFC 1889 Jitter Minimum**

    Lowest jitter statistic in milliseconds of all timing records for a pair or group.

- **RFC 1889 Jitter Maximum**

    Highest jitter statistic in milliseconds of all timing records for a pair or group.

- **Jitter (Delay Variation) Maximum**

    Highest difference between datagram inter-arrival times, in milliseconds, measured by a pair or group.

To see the individual timing records for an endpoint pair, select its row in the Test window, then click **Show timing records** on the View menu.

Because jitter statistics are not part of a whole, you cannot view a pie graph for jitter data. To determine whether the jitter has exceeded your thresholds, use the histogram graph, or click the Jitter (Delay Variation) tabs at the top of the line graph. You can easily see the ranges of the jitter values and determine if the amount of jitter in the test exceeds the thresholds.

The Jitter (Delay Variation) Tab

Related Topics
The Jitter (Delay Variation) Maximum Tab on page 11-30
The Jitter Tab on page 11-29
Understanding Jitter Measurements on page 10-50
Jitter and Delay Variation on page 10-52
Result Ranges Tab on page 6-29

For VoIP endpoint pairs, IxChariot calculates jitter two ways. **RFC 1889 jitter** is an average, the statistical variance of the datagram inter-arrival time expressed as a mean deviation for a single pair. **Jitter (delay variation)** (VoIP Test Module only) indicates the differences in arrival times among all datagrams for a particular RTP pair. Delay variation jitter measurements, in milliseconds, are placed in ranges so that you can compare them to such benchmarks as the size of the jitter buffer and the standards for call quality established for the hardware you are using.

The Jitter (Delay Variation) histogram organizes delay variations into a series of millisecond ranges and then shows the number of datagrams that fell into each range, expressed as a percentage of the total number of datagrams sent. A histogram is the only graph type available for delay variation. The ranges shown can be configured; see *One-Way (Network) Delay* on page 11-43. Refer to *The Jitter Tab* on page 11-29 for more information about the data shown.

For general information about jitter, see *Understanding Jitter Measurements* on page 10-50.

The Jitter (Delay Variation) Maximum Tab

Related Topics
The Jitter Tab on page 11-29
Understanding Jitter Measurements on page 10-50
Jitter and Delay Variation on page 10-52

IxChariot calculates jitter and delay variation for RTP streaming pairs. **RFC 1889 jitter** is the statistical variance of the datagram inter-arrival time expressed as a mean deviation for each pair. **Delay variation jitter** indicates the differences in arrival times among all datagrams for a particular endpoint pair. The **Jitter (Delay Variation) Maximum** tab graphs variations between the arrival times of all datagrams sent during each timing record. A jitter (delay variation) maximum that exceeds the jitter buffer you are using indicates poor call quality.

The Jitter (Delay Variation) Maximum line graph of timing records shows when the greatest variability in datagram arrival times occurred during the test and indicates which pairs were most affected. You can choose several different graph types for maximum delay variation; however, the pie graph is not available because jitter and delay variation are not calculated as parts of a whole. Refer to *The Jitter Tab* on page 11-29 for more information.

For general information about jitter and delay variation, see *Understanding Jitter Measurements* on page 10-50.

The Lost Data Tab

Related Topics
Result Ranges Tab on page 7-16
The Consecutive Lost Datagrams Tab on page 11-33
The Maximum Consecutive Lost Datagrams Tab on page 11-33

The **Lost Data** tab is shown for a test containing pairs running streaming scripts. This tab shows high-level information about streaming results. The Lost Data graph offers three tabbed views, one for lost data statistics, a histogram showing statistics on Consecutive Lost Datagrams, and a graph showing when the Maximum Consecutive Lost Datagrams occurred during the elapsed time of the test. This second view is particularly useful for voice over IP testing; a high number of consecutive lost datagrams indicates poor call quality.

For additional information about individual datagrams, check the **Datagram** tab.

- **Group/Pair**

  The group of pairs—based on how you've grouped endpoint pairs in the Test window—or the pair number (based on the order in which you added pairs to the test).

- **Bytes Sent by E1**

  Total bytes sent in the test. For a streaming test, E1 sends data to E2, with no acknowledgments. For a multicast group, you may see the same value for all the pairs because the E1 is sending the data once for all of the E2s. The value may be different if E2 fails during the test or loses a large amount of data. For a multicast group, the total is not the aggregate total (because the data was sent only once).

- **Bytes Received by E2**

  Total bytes received by E2 in the test. Only applicable to pairs running a streaming script.

- **Bytes Lost from E1 to E2**

  Bytes of data lost. Calculated as the difference between the first two columns of data. Only applicable to pairs running a streaming script.

- **Percent Bytes Lost E1 to E2**

  Bytes lost, shown as a percentage of the total number of bytes sent. Only applicable to pairs running a streaming script. The aggregate value for this percentage uses an aggregate value of Bytes Sent by E1.

- **E1 Throughput**[1]

  Throughput achieved by E1. Only applicable to pairs running a streaming script. Derived by dividing the total number of bytes sent by the total time. If no data was lost by the pair, E1 throughput is the same as the throughput for the pair: check the **Average Throughput**[2] column on the **Throughput** tab for this information.

- **Maximum Consecutive Lost Datagrams**

  The highest number of consecutive datagrams lost for a particular streaming pair.

- **Measured Time**

  Sum of the timing record durations for each pair. To see the individual timing records for an endpoint pair, select its row, then click **Timing records** on the View menu (or right-click a row).

- **Relative Precision**

  The consistency among a pair's timing records. An increase in the Relative Precision correlates directly with increased variability among timing records. Displays "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Relative Precision* on page 11-47 for more information.

The Lost Data graph for this tab shows the percentage of lost bytes for the selected pairs/groups over the elapsed time. This graph can show you at what time during the test the data was lost. If there was no data lost for the selected pairs/groups, this graph is shown as empty. If you view this data in a pie graph, the graph shows cumulative totals. A single group or pair shows the same as multiple groups/pairs.

Within streaming scripts an out of sequence datagram will first be considered as lost. If the datagram is received within the current timing record, its category will be changed to that of an out of sequence datagram. If, however, the datagram is received in the next timing record it will be considered as lost in the current timing record and out of sequence in the subsequent timing record.

IxChariot does not re-order out-of-order datagrams. These datagrams are, in fact, lost to the application. For example, assume that you run a test in which endpoint 1 sends 400 datagrams, endpoint 2 receives 388 of those datagrams, 12

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.
2. Really, the average of the total goodput.

datagrams are lost, and 122 datagrams are received out-of-order. In this case, the Lost Data tab will show the "Percent Bytes Lost E1 to E2" as 33.5% (134 out of 400). The "Bytes Lost" column in the Lost Data tab includes both lost packets and out-of-order packets.

The Consecutive Lost Datagrams Tab

Related Topics

Result Ranges Tab on page 7-16
The Lost Data Tab on page 11-31
The Maximum Consecutive Lost Datagrams Tab on page 11-33
VoIP Test Module Features on page 10-35

View statistics on Consecutive Lost Datagrams by clicking the Lost Data tab in the Test window; then click the **Consecutive Lost Datagrams tab** just above the graph. IxChariot graphs statistics on consecutive lost datagrams by placing results into a series of ranges. When Endpoint 1 has completed sending the data, it tells Endpoint 2 how much data was sent so the lost data totals can be calculated.

The ranges of results may be configured so that you can accurately gauge the quality of a call on your network, based on equipment standards. Refer to *One-Way (Network) Delay* on page 11-43 for more information about configuring lost datagram ranges.

On the Consecutive Lost Datagrams tab, the only available graph is a histogram, which shows the **frequency of datagram loss**. Consecutive lost datagrams are placed into ranges based on the number lost consecutively. The histogram shows how often losses within a certain range occurred. To see exactly which timing records had data loss, select an endpoint pair and click **Show Timing Records** on the View menu. The last columns of the **Lost Data** tab in the Timing Records dialog box show consecutive datagram loss grouped by the configured result ranges; scroll over to see them.

For more information about when the most datagrams were lost during the test, click the **Maximum Consecutive Lost Datagrams** tab.

The Maximum Consecutive Lost Datagrams Tab

Related Topics

Result Ranges Tab on page 7-16
The Lost Data Tab on page 11-31
The Consecutive Lost Datagrams Tab on page 11-33

 Determine when the greatest data loss occurred during a multimedia or voice over IP test by clicking the Lost Data tab in the Test window; you'll find the **Maximum Consecutive Lost Datagrams** tab just above the graph. To calculate lost data, Endpoint 1 tells Endpoint 2 how much data was sent in each timing record. E2 compares the amount sent to the amount it actually received.

The Maximum Consecutive Lost Datagrams tab graphs the **number of data-grams lost** on the y axis and the **time the loss occurred** on the x axis.

To see exactly which timing records had consecutive data loss, select an endpoint pair and click **Show Timing Records** on the View menu. The **Lost Data** tab in the Timing Records dialog box includes a Maximum Consecutive Lost DGs column.

For more information about datagram loss, refer to *The Lost Data Tab* on page 11-31 or *The Consecutive Lost Datagrams Tab* on page 11-33.

The Video Tab

Related Topics
Adding or Editing a Video Endpoint Pair on page 10-54
Adding or Editing a Video Multicast Group on page 10-58
Media Delivery Index (MDI) Calculation on page 10-63

For completed tests containing video endpoint pairs or video multicast groups, the Video tab is shown. The data provided for video pairs indicates the quality of the video transmission that was made during the test and provides a set of statistics related the Media Delivery Index (MDI). Refer to *Media Delivery Index (MDI) Calculation* on page 10-63 for more information.

- **Group/Pair**

  Shows the group name and each pair within the group.

- **All Pairs**

  Totals for all pairs in the test are shown in this row.

- **Run Status**

  Indicates whether the endpoint pair ran its script to completion.

- **Timing Records Completed**

  The number of timing records this test generated. Gives an indication of the reliability of the results.

- **Relative Precision**

  The consistency among a pair's timing records. An increase in the Relative Precision correlates directly with increased variability among timing records. Displays "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Relative Precision* on page 11-47 for more information.

- **Maximum MDI**

  The maximum Media Delivery Index (MDI), expressed as:

  ```
  Maximum-DF : Maximum-MLR
  ```

  Refer to *Media Delivery Index (MDI) Calculation* on page 10-63 for more information

- **Average DF**

  The average Delay Factor (DF).

- **Minimum DF**

  The minimum Delay Factor (DF).

- **Maximum DF**

  The maximum Delay Factor (DF).

- **Average MLR**

  The average Media Loss Rate (MLR). MLR shows the number of media packets lost per second.

- **Minimum MLR**

  The minimum Media Loss Rate (MLR).

- **Maximum MLR**

  The maximum Media Loss Rate (MLR).

- **95% Confidence Interval (DF)**

  Estimated range of DF values with a 95% probability of statistical accuracy. Shows "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Confidence Intervals* on page 11-49 for information about the calculation of confidence intervals.

- **95% Confidence Interval (MLR)**

  Estimated range of MLR values with a 95% probability of statistical accuracy. Shows "n/a" while a test is running; shows values when a test ends (if there are at least 2 timing records). See *Confidence Intervals* on page 11-49 for information about the calculation of confidence intervals.

The IPTV Tab

For tests containing IPTV endpoint pairs, the IPTV tab is shown. IxChariot IPTV tests help you evaluate channel switching performance by providing channel switching join and leave latency report data:

- **Group**

  Shows the name of the IPTV receiver group and each pair within the group.

- **Pair Group Name**

  Shows the name of the IPTV receiver group associated with each pair.

- **Run Status**

  Indicates whether the endpoint pair ran its script to completion.

- **Timing Records Completed**

  The number of timing records this test generated. Gives an indication of the reliability of the results.

- **Joins/Leaves Completed**

  Real-time statistic that shows how many join/leave iterations were completed

- **Average Join**

  Real-time statistic that shows the average of the multicast joins until that moment

- **Minimum Join**

  Real-time statistic that shows the minimum multicast join delay until that moment

- **Maximum Join**

  Real-time statistic that shows the maximum multicast join delay until that moment

- **Average Leave**

  Real-time statistic that shows the average of the multicast leaves until that moment

- **Minimum Leave**

  Real-time statistic that shows the minimum multicast leave delay until that moment

- **Maximum Leave**

  Real-time statistic that shows the maximum multicast leave delay until that moment.

The TCP Statistics Tab

Related Topics

Miscellaneous Run Options on page 7-11
Tips for Running Large-Scale Tests on page 8-12
Collecting TCP Statistics on page 10-82

If you are using an Ixia chassis for your endpoints and you enable *Collect TCP Statistics* (either globally or for a specific test), IxChariot collects a set of TCP statistics and presents them in the TCP Statistics tab. This data provides a statistical view of TCP control activities, such as connection establishment and retransmission of lost packets. TCP statistics are available only for tests that are using TCP as the transport protocol.

The following list describes the data elements that are reported in the TCP Statistics tab:

- **Group/Pair**

  Shows the group name and each pair within the group.

- **Run Status**

  Indicates whether the endpoint pair ran its script to completion.

- **Timing Records Completed**

  The number of timing records this test generated. Gives an indication of the reliability of the results.

- **Syn Sent**

  Total number of SYNC packets transmitted by Endpoint 1.

- **Syn Received**

  Total number of packets received by Endpoint1 that have the SYNC flag set. Note that this statistic includes both SYN and SYN/ACK packets.

- **Syn Failed**

  Total number of connection attempts for which Endpoint 1 reset the connection before synchronization was established.

- **Connections Established**

  Total number of TCP connections that were established by Endpoint 1.

- **Fin Sent**

  Total number of FIN packets transmitted by Endpoint 1.

- **Fin Received**

  Total number of FIN packets received by Endpoint 1.

- **Fin/Ack Sent**

  Total number of ACKnowledge packets transmitted by Endpoint 1 in response to a FIN sent by Endpoint 2.

- **Fin/Ack Received**

  Total number of ACKnowledge packets received by Endpoint 1 from Endpoint 2 in response to a FIN sent by Endpoint 1.

- **Reset Sent**

  Total number of RESET packets transmitted by Endpoint 1.

- **Reset Received**

  Total number of RESET packets received by Endpoint 1.

- **TCP Retransmissions**

  Total number of retransmission of any TCP segment.

- **Maximum TCP Retransmissions**

  The highest number of retransmissions among the TCP segments included in the timing records.

- **Minimum TCP Retransmissions**

  The lowest number of retransmissions among the TCP segments included in the timing records.

- **Average TCP Retransmissions**

  The average number of retransmission is calculated at the pair, group, and test levels. At the pair level, it is the number of TCP transmissions divided by the number of timing records. At the group level, it is the average of all the pairs in the group. At the test level, it is the average of all the groups in the test.

- **TCP Timeouts**

  Total number of TCP timeouts that occurred on Endpoint 1. A TCP connection times out if there has not been an expected response from the peer for a defined amount of time.

Following is a description of the graphs that are available for TCP Statistics:

- **Line graph of timing records for Pairs**

  A line graph of timing records for pairs shows the average rate of a TCP statistic per timing record. For example: 10 retransmissions in a 5 second timing

record results in a graph showing 2 retransmissions per second for the duration of the timing record.

- **Line graph of timing records for Groups**

  This graph is the aggregate of values of the individual pairs belonging to a group.

- **Bar graphs of averages, minimums and maximums**

  This graph is available only for TCP retransmissions. The graph shows the maximum, minimum, and average retransmissions per timing record.

- **Pie graph**

  The pie graph shows the distribution of a particular TCP statistic (such as SYN Sent) across pairs or groups.

- **Histogram of timing records**

  This graph is available only for TCP retransmissions. It shows the distribution of a particular TCP statistic across timing records.

- **Histogram of averages**

  This graph is available only for TCP retransmissions. It shows the percentage of pairs that have a certain number of average TCP retransmissions.

Printed/Exported Results

Related Topics

Custom Printing and Export Options on page 5-64
Test Window Tabs on page 11-14
Reading Your Test Results on page 11-39

The results sections of the exported or printed output correspond to the tabbed sections of the Test window that appear when you have results. They show the summary of results for Throughput[1], Transaction Rate, and Response Time.

If you are exporting pairs that are part of a multicast group, it may appear that the totals do not include an aggregate of all the pairs. That's because for multicast, some of the data should only be counted once. For example, if 5 pairs of a multicast group have throughput of 5 Mbps, the actual effect on the network is not 25 Mbps; instead, it is only 5 Mbps. This is true because although the data is sent only once, 5 different endpoints received the data and reported the throughput.

> **NOTE**: Multicast pairs may be counted more than once in group totals if they appear in different groups based on the grouping you have chosen, for example, Group by Endpoint 2. In these cases, the multicast information may be counted once for each of the group totals, but it will only be counted once for the test Total.

Reading Your Test Results

Related Topics

Throughput, Transaction Rate, and Response Time on page 11-41
Endpoint Configuration on page 11-44
Summary, Run Options, and Test Setup Section on page 11-40

Printed or exported results contain three major sections, which are summarized in detail below:

- The *Summary, Run Options, and Test Setup Section* provides an overview of how the test was set up and when it was run.

- The Results sections, beginning with *Throughput, Transaction Rate, and Response Time* on page 11-41, show totals, including the average, minimum, and maximum values for throughput, transaction rate, response time, and streaming results. The datagram totals (if applicable), plus endpoint configuration and raw data totals, are also shown.

- The *Endpoint Configuration* on page 11-44 section gives detailed information about each endpoint pair and shows all the timing records for that pair.

> **NOTE**: When printing PDF reports, make sure to select **Fit to Printable Area/Shrink to Printable Area** from the printing options so that all report information is printed.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

Summary, Run Options, and Test Setup Section

Related Topics
Start, End, and Run Times on page 11-2
Understanding Timing on page 11-1
Datagram Run Options on page 7-18

The first section of the printed or exported results of each test is broken down into four subsections:

- **Summary**: lists basic information about the test, including its filename, the version of IxChariot used, and how the test completed;

- **Run Options**: reprises the settings you selected from the Run Options dialog box, such as the Reporting Type (that is, batch or real-time). If your test used a datagram protocol, the datagram settings are also listed here;

- **Test Setup (Endpoint 1 to Endpoint 2)**: gives information about test settings, including the network addresses for the endpoints of each pair;

- **Test Setup (Console to Endpoint 1)**: gives information about the connection between the Console and E1 for each pair.

- **Test Setup (Endpoint 1 to Endpoint 2)**: gives information about the connection between the endpoints for test setup flows and result reporting.

In each subsection, "**n/a**" appears when a field is not present or does not apply.

Here is an example of the first section of the printed or exported results, for a test with four endpoint pairs:

Table 11-2.   Summary -- C:\Program Files\Ixia\IxChariot\Tests\T1Switch.tst

Console version	5.40
Console Build Level	11
Console Product Type	Retail
Filename	C:\Program Files\Ixia\IxChariot\Tests\T1Switch.tst
Run Start Time	Thursday, March 01, 2002, 10:05:08 AM
Run End Time	Thursday, March 01, 2002 10:07:39 AM
Elapsed Time	00:02:31
How the test ended	Ran to completion
Number of pairs	3

This subsection shows information about when the test was run (if at all), and how long the run took.

Run start time and **Run end time** mark the date and time when the test was started at the Console and when the test completed, excluding the time spent formatting results. The **Elapsed time** is the difference, in seconds, between the start and end times. See *Start, End, and Run Times* on page 11-2 for more information. If the test had any errors, "Ended with errors" is indicated next to "**How the test ended**."

In the throughput[1], transaction rate, and response time results for each pair, IxChariot also shows the **Measured time**. The measured time is the sum of the times in all the timing records returned for that endpoint pair. Thus, the *measured* time for any endpoint pair is always less than the total *elapsed* time for a test. See *Understanding Timing* on page 11-1 for more information.

The following subsection shows the Run Options used for this test. If datagram protocols were used for any of the pairs, the Datagram Options are also shown. See *Lost Data* on page 11-42 for detailed information on each of these values.

Table 11-3.   Test Setup (Endpoint 1 to Endpoint 2)

Group/Pair	Endpoint 1 Knows Endpoint 2 (Setup)	Endpoint 2 Setup Protocol
All Pairs		
Pair 1	10.10.9.123	TCP
Pair 2	10.10.9.125	TCP
Pair 3	10.10.9.125	TCP

The Test Setup subsections show the test setup for each pair. Setup details between the endpoints and between the Console and Endpoint 1 are given in different tables. See *Adding or Editing an Endpoint Pair* on page 5-19 for details on each of these fields. For the Console to Endpoint 1 connection, test setup details include the group or endpoint pair number, the network address at which the Console communicates with Endpoint 1, the Console protocol, the service quality (if any), and the descriptive comment for this pair (if one was entered).

Throughput, Transaction Rate, and Response Time

Related Topics

Calculations on page 11-14
The Throughput Tab on page 11-17
The Response Time Tab on page 11-19
The Transaction Rate Tab on page 11-20

Three subsections of the exported or printed output show results for Throughput[2], Transaction Rate, and Response Time. At the Console, these are shown on separate tabs. In printed or exported results, each subsection is structured the same way. The first column lists groups and pairs. **Group** rows are given names based on how you've grouped or sorted endpoints in the Test window. **Pairs** are numbered based on the order in which you added them.

The next three columns show the **Average**, **Minimum**, and **Maximum** timing record calculations for each pair. For the **Totals** rows (shown in bold), aggregate values are shown. **Group** rows (shown in bold) calculate overall results for the group; for more information about how results are calculated, refer to *Calculations* on page 11-14.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.
2. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

> **Note:** If a pair is a member of a multicast group, the throughput total includes only one pair's throughput value (whichever pair had the highest).

The next column shows the 95% Confidence Interval for each pair. The last two columns are the same for each subsection: Measured Time and Relative Precision. All the values and their calculations are discussed below in detail.

Some result types, such as the Measured Time and the Relative Precision, aren't totaled for a group. See *The Throughput Tab* on page 11-17, *The Response Time Tab* on page 11-19, or *The Transaction Rate Tab* on page 11-20 for explanations of each type of data in these results.

Lost Data

Related Topics
The Maximum Consecutive Lost Datagrams Tab on page 11-33
The Consecutive Lost Datagrams Tab on page 11-33
The Lost Data Tab on page 11-31

When a pair is running a streaming script (for example, in a VoIP or IP Multicast test), Endpoint 2 keeps track of lost data, data that was discarded and not received by the receiving endpoint. From the header information in the RTP and UDP packets, Endpoint 2 determines how much data was sent and calculates lost data totals by subtracting the amount of data it actually received. Only payload data is included in the calculations.

For more information, see *The Lost Data Tab* on page 11-31.

Consecutive Lost Datagrams

Related Topics
The Maximum Consecutive Lost Datagrams Tab on page 11-33
The Consecutive Lost Datagrams Tab on page 11-33
The Lost Data Tab on page 11-31

A histogram is the only graph available for the data on the **Consecutive Lost Datagrams** tab.

The y axis graphs the **number of consecutive datagrams lost**, expressed as a **percentage** of the total number of datagrams sent. The x axis places the lost datagrams into ranges, based on the **number of datagrams lost consecutively**. These ranges may be configured; refer to *One-Way (Network) Delay* on page 11-43 for more information.

Results for Maximum Consecutive Lost Datagrams are also available; refer to *The Maximum Consecutive Lost Datagrams Tab* on page 11-33 for more information.

Maximum Consecutive Lost Datagrams

Related Topics
The Maximum Consecutive Lost Datagrams Tab on page 11-33
The Consecutive Lost Datagrams Tab on page 11-33
The Lost Data Tab on page 11-31

A histogram is the only graph available for the data on the **Consecutive Lost Datagrams** tab.

The Maximum Consecutive Lost Datagrams tab graphs the **number of datagrams lost** on the y axis and the **time the loss occurred** on the x axis.

To calculate lost data, Endpoint 1 tells Endpoint 2 how much data was sent in each timing record. E2 compares the amount sent to the amount it actually received. For more information about datagram loss, refer to *The Lost Data Tab* on page 11-31.

Results for Consecutive Lost Datagrams are also available; refer to *The Maximum Consecutive Lost Datagrams Tab* on page 11-33 for more information.

VoIP

Related Topics
The VoIP Tab on page 11-25
VoIP Score Calculation on page 10-48

If you are running the IxChariot VoIP Test Module, check the check boxes grouped under "VoIP Results" in the Custom Export dialog box to see voice over IP results in your exported or printed reports.

Refer to *VoIP Score Calculation* on page 10-48 for more information about the MOS estimate and other calculations. For explanations of each type of data shown in reports, see *The VoIP Tab* on page 11-25.

One-Way (Network) Delay

Related Topics
The One-Way Delay Tab on page 11-27
One-Way Delay on page 10-47
VoIP Score Calculation on page 10-48

One-way or network delay is calculated for RTP datagrams if you are running the IxChariot VoIP Test Module. When E1 sends RTP datagrams to E2, the datagrams include a timestamp indicating the time they were sent. E2 then calculates one-way delay by subtracting the datagram's send time from the receive time. One-way network delay measurements can be taken on the following endpoint operating systems:

- Windows

- Linux

- Solaris.

Unsupported endpoint operating systems show "**n/a**" in the results columns.

For explanations of each type of data shown in reports, refer to *The One-Way Delay Tab* on page 11-27.

Endpoint Configuration

Related Topics

The Endpoint Configuration Tab on page 11-16

Following summaries of the numeric results of a test is a description of how the endpoints were configured when the test was run, including their operating systems and the version of the endpoint software they were running. At the IxChariot Console, Endpoint Configuration is shown in a separately-tabbed area of the Test window. For more information about what's shown, see *The Endpoint Configuration Tab* on page 11-16.

Raw Data Totals

Related Topics

The Raw Data Totals Tab on page 11-22

The Raw Data Totals show you the data that is used in calculating results. You can see the number of timing records and transactions for each pair, as well as the number of bytes sent and received by Endpoint 1.

For more information about what's shown in the Raw Data Totals, refer to *The Raw Data Totals Tab* on page 11-22.

802.11

Related Topics

The 802.11 Tab on page 11-21

The Timing Records section shows you the data that is used in calculating results. You can see the individual RSSI and BSSID values for each timing record.

For more information about what's shown in the Timing Records section, refer to *The 802.11 Tab* on page 11-21.

TCP Statistics

Related Topics

The TCP Statistics Tab on page 11-36
Custom Printing and Export Options on page 5-64

For text export, the TCP Statistics section of the report lists the TCP statistics calculated for each pair that participated in the test, as well as the totals for all pairs. For CSV export, the TCP Statistics section lists the TCP statistics calculated for each pair that participated in the test, as well as a listing of each timing record that was generated during the test. If you are producing HTML reports or printed reports, you can customize the report output, as described in *Custom Printing and Export Options*. You can choose whether or not to include the TCP statistics results (the TCP statistics calculated for each pair); and you can also choose whether or not to include graphs for each of the statistics generated during the test. TCP statistics are available only on Ixia ports used as endpoints.

Endpoint Pair Configuration

Related Topics

Endpoint Pair Timing Records on page 11-46
Endpoint Pair Variables on page 11-47

In the Endpoint Pair Configuration section of an IxChariot report, information pertaining to each endpoint pair is shown, including its configuration, its script, its script variables, and the individual timing records for a particular test.

See the following topics for more information:

- *Endpoint Pair Script* on page 11-46

- *Endpoint Pair Timing Records* on page 11-46

- *Endpoint Pair Variables* on page 11-47

The details for each configured endpoint pair are shown in separate sections. The first part of each Endpoint Pair section shows a number of configuration parameters and information about what happened during a test. These are as follows:

- **Endpoint 1**

  The Endpoint 1 address used by this endpoint pair.

- **Endpoint 2**

  The Endpoint 2 address used by this endpoint pair.

- **Network Protocol**

  The protocol used between the two endpoints.

- **Service Quality**

  The service quality used between the two endpoints; this field shows "n/a" if service quality is not supported by the selected protocol.

- **Script Name**

  The filename of the script used between this pair of endpoints.

- **Pair Comment**

  The endpoint pair comment (an optional field).

- **Console Knows Endpoint 1**

  Address used by the Console to contact E1 with test setup information.

- **Console Protocol**

  Network protocol used between the Console and E1. Sometimes a connection-oriented protocol is used for communications between the Console and E1 so that tests will not pause indefinitely; see *Cloning Hardware Performance Pairs* on page 5-25 for more information.

- **Console Service Quality**

  The service quality used between the Console and E1; this field shows "n/a" if service quality is not supported by this protocol.

- **Endpoint 1 Knows Endpoint 2**

  Address used by E1 to contact E2 with test setup information.

- **Endpoint 2 Setup Protocol**

  Network protocol used between the endpoints for test setup communications. TCP is the only protocol currently supported.

Endpoint Pair Script

This part of the results shows the script that was used, including any variables you set. Here's an example:

Script

```
Filercvl.scr, version 4.x -- File Receive, Long Connection

 Endpoint 1                        Endpoint 2
 ------------------------------- --------------------------
SLEEP
   initial_delay=0
CONNECT_INITIATE                  CONNECT_ACCEPT
  source_port=AUTO                  destination_port=AUTO
  send_buffer=DEFAULT               send_buffer=DEFAULT
  receive_buffer=DEFAULT            receive_buffer=DEFAULT
LOOP                              LOOP
  number_of_timing_records=100     number_of_timing_records=100
  START_TIMER
  LOOP                              LOOP
    transactions_per_record=1        transactions_per_record=1
    SEND                             RECEIVE
     file_size=100000                 file_size=100000
     send_buffer_size=DEFAULT         receive_buffer_size=DEFAULT
     send_datatype=NOCOMPRESS
     send_data_rate=UNLIMITED
    CONFIRM_REQUEST                  CONFIRM_ACKNOWLEDGE
    INCREMENT_TRANSACTION
  END_LOOP                            END_LOOP
  END_TIMER
  SLEEP
    transaction_delay=0
 END_LOOP                           END_LOOP
DISCONNECT                        DISCONNECT
```

Endpoint Pair Timing Records

Related Topics
Test Window Tabs

The next section of the results shows all of the timing records generated by this endpoint pair during the run. These are the raw results used to create the statistics for this endpoint pair that are shown in the Results section.

For an explanation of each type of data shown in the Timing Records tables, see the topics grouped under *Test Window Tabs*

Endpoint Pair Variables

This part of the results shows a summary of the variable settings in the script used by this test. Here's an example, taken from a voice over IP test:

Table 11-4.　　　Endpoint Pair Variables

Variable Name	Value	Description
`initial_delay`	2000	Pause before the first transaction
`number_of_timing_ records`	50	How many timing records to generate
`file_size`	240000	How many bytes in the transferred file
`send_buffer_size`	160	How many bytes of data in each `SEND`
`receive_buffer_si ze`	Default value	How many bytes of data in each `RECEIVE`
`send_datatype`	`NOCOMPRES S`	What type of data to send
`send_data_rate`	`UNLIMITED`	How fast to send data
`destination_port`	`AUTO`	What port to use for Endpoint 2
`close_type`	Reset	How connections are terminated
`source_port`	1030	What port to use for Endpoint 1

`SEND` and `RECEIVE` buffers can be set to the value "`DEFAULT`." This setting tells an endpoint to use buffers that are the default size for the network protocol being used. `DEFAULT` lets you use the default buffer size for each protocol, without having to modify the script to handle protocol differences. The default value differs depending on the protocol and platform being used. IxChariot uses the most common value for each particular environment.

Endpoint Configuration Details

Related Topics
Endpoint Configuration Details Dialog Box on page 11-17
The Endpoint Configuration Tab on page 11-16

This part of the results shows extensive details about the endpoint programs, their operating systems and protocol stacks. This information differs among operating systems and endpoint versions.

For an explanation of each type of configuration data shown, refer to *Endpoint Configuration Details* on page 11-47.

Relative Precision

Related Topics
How Long Should a Performance Test Run? on page 10-86
Confidence Intervals on page 11-49

The confidence interval is a well-known statistical measurement for the reliability of the calculated average. Unfortunately, it does not provide a good mechanism to compare tests using different scripts that do different things; you cannot easily compare the reliability of two tests by looking at the confidence interval of a file transfer script and the confidence interval of an inquiry script.

The Relative Precision is a gauge of how reliable the results are for a particular endpoint pair. Regardless of what type of script was run, you can compare relative precision values.

The relative precision is obtained by calculating the 95% confidence interval of the Measured Time for each timing record, and dividing it by the average Measured Time. This number is then converted to a percentage by multiplying it by 100. The lower the Relative Precision value, the more reliable the result.

A "good" Relative Precision value is 10.00 or less. On an empty LAN, you can get Relative Precision values of less than 1.00 on many tests. See *How Long Should a Performance Test Run?* on page 10-86 for further discussion of how to set up tests to obtain reliable results. See *Confidence Intervals* on page 11-49 for information about the calculation of confidence intervals.

IxChariot maintains its internal numerical values with more significant digits than those shown in the results. If you were to calculate values like Relative Precision from the other numbers shown in the results, your calculation may differ slightly from the numbers displayed by IxChariot.

Confidence
Intervals

Related Topics

Here is a formal definition of a confidence interval, using statistical terms:

> *A confidence interval is an estimated range of values with a given high probability of covering the true population value.*

The term "probability" in this definition points out the fact that IxChariot is doing a sampling of a real (finite) set of measurements. If it could sample all of the possible measurements of a network (with infinite time and resources), it could be 100% sure that the calculated average is the correct value. Since IxChariot always generates a smaller-than-infinite set of measurements, some doubt as to whether it has really calculated an average that approximates the "real" average will always linger.

To state the definition another way, there is a 95% chance that the actual average lies between the lower and upper bound indicated by the 95% Confidence Interval.

Confidence Interval Calculation

Here is how IxChariot calculates the 95% Confidence Interval:

1. IxChariot first calculates the standard deviation of the timing records for throughput, transaction rate, and response time.

2. It then calculates the standard error, which is the standard deviation divided by the square root of the number of timing records.

3. Next, IxChariot uses a statistical table to look up a "t" value, using the number of timing records minus one.

4. The confidence interval is the "t" value times the standard error.

5. IxChariot rounds the confidence interval to three decimal places and displays it as a range in the "95% Confidence Interval" column of the test results (for throughput, transaction rate, and response time).

walks through a specific example to demonstrate these calculations.

Confidence Interval Example

This example uses a small throughput test to demonstrate how the confidence interval is calculated. Figure 11-4 displays both the results of the test and the timing records from the test.

Figure 11-4.   Timing Records Used for Confidence Interval
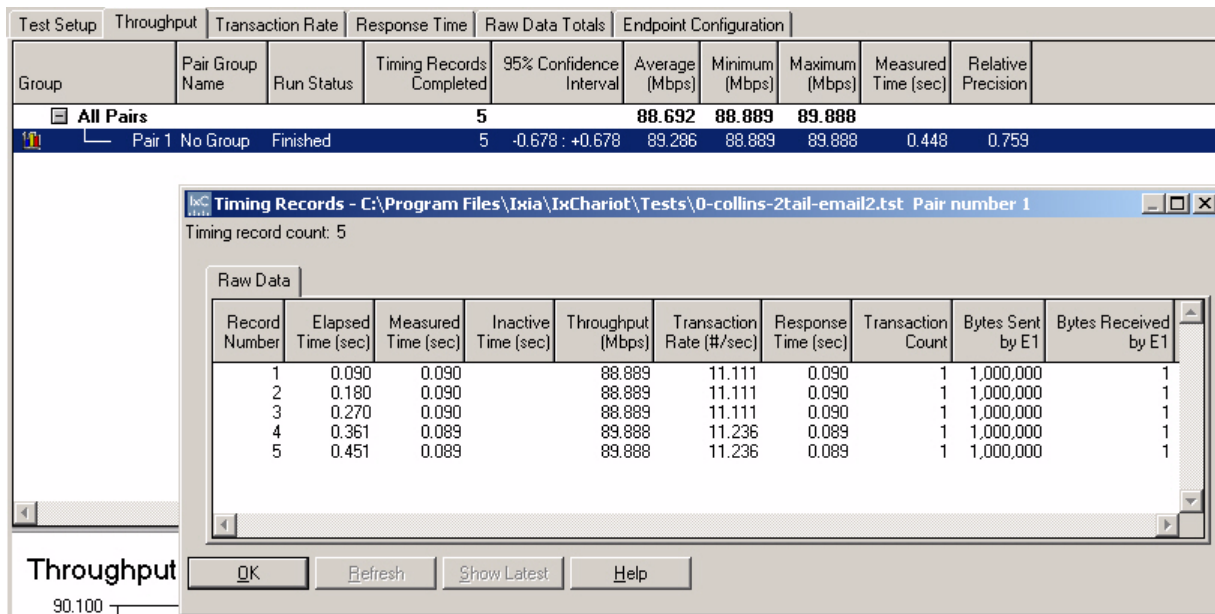
Table 11-5 lists the timing records for throughput, transaction rate, and response time, followed by the confidence interval calculations. The confidence interval calculations were produced using a spreadsheet program.

Table 11-5.   Example Confidence Interval Calculations

	Throughput (Mbps)	Transaction Rate (#/sec)	Response Time (sec)
	88.889	11.111	0.09
	88.889	11.111	0.09
	88.889	11.111	0.09
	89.888	11.236	0.089
	89.888	11.236	0.089
StdDev:	0.547174835	0.06846532	0.000547723
StdErr:	0.244704025	0.030618622	0.000244949
t-value:	2.776	2.776	2.776
Interval:	0.679298374	0.084997294	0.000679978
Rounded:	0.679	0.085	0.001

Here is an explanation of the calculations for the Throughput column in Table 11-5:

1. The standard deviation for Throughput (and the other columns as well) was calculated using a STDEV spreadsheet function. The input to the function is the list of values from the five timing records (88.889, 88.889, 88.889, 89.888, 89.888).

2. The standard error is calculated as the standard deviation (0.547174835) divided by the square root of the number of timing records (the square root of 5 is 2.236068).

3. The t-value is 2.776. (Student-t tables are found in most statistics textbooks.)

4. The Confidence Interval is the t-value multiplied by the standard error (2.776 * 0.244704025).

5. The Confidence Interval (0.679298374) is rounded to three decimal places.

In this specific example, our Confidence Interval calculations for Transaction Rate and Response Time are identical to those in the IxChariot test, while our calculation for Throughput differs by +0.001. This difference is most likely the result of differences in the degree of precision used in the spreadsheet calculations versus those used in the IxChariot calculations.

Negative Numbers in a Confidence Interval

You may sometimes see a negative number in the lower bound of a 95% Confidence Interval. The statistical calculations being used assume an unbounded normal distribution, which could contain negative samples. We know you cannot get negative numbers in real life (communications never go faster than the speed of light). Thus, when you get a negative number on the left side of your confidence interval, you should have very low confidence in the results of the test.

See *How Long Should a Performance Test Run?* on page 10-86 for further discussion of how to set up tests to obtain reliable results.

Factors Affecting Results

Related Topics

Designing IxChariot Performance Tests on page 10-84
Understanding Results on page 11-1

The endpoint programs exhibit identical behavior on identical computers, whether they're playing the role of Endpoint 1 or Endpoint 2. For example, you have an endpoint pair, executing a script from Address A to Address B. Given that your network performs the same in different directions, the results you obtain should be statistically identical to those obtained by executing the same script from Address B to Address A—if the two computers are identical, and the results are returned to the Console in batch.

To increase the consistency of your results, re-use the original test that you saved. Recreating a test manually increases the likelihood that there will be slight differences from the original test that could produce different results. Be sure that you do not modify the Run Options for the test. Changing these options will generate inconsistent results.

The endpoints use your actual equipment and make the same API calls as your network applications. Therefore, the endpoints are very sensitive to a wide variety of differences in the hardware and software they use, just as your real applications are. These differences can significantly affect the results you see. Executing a script from A to B can give much different results than executing the same script from B to A, if the two computers differ. This can be disconcerting when executing performance benchmarks, unless you carefully control all aspects of the hardware and software you are using.

Following are some of the factors we've found in our testing that can affect the consistency of your results. We've listed them in the order of significance that we've observed.

- **CPU Speeds**

  Aside from `SLEEPs`, scripts execute on endpoints as fast as their CPU and network hardware allow, unless limited by the `send_data_rate` variable.

- **Endpoint Operating Systems**

  Operating systems vary significantly in the ways they handle network calls. For example, you can use the same Intel-based computer, yet see differences in the speed with which they execute TCP calls on Windows 3.1 or Win32 operating systems. Don't substitute different computers and operating systems without expecting to see differences. (The Windows 3.1 endpoint, without true multitasking, performs poorly in the role of Endpoint 1.)

- **Protocol Stacks**

  Network protocol software always changes its performance behavior from release to release. Every different protocol stack varies from those by other manufacturers.

  For example, the `NNTP` script encounters a part of TCP's delayed acknowledgment (`ACK`) algorithm. At the end of the `NNTP` script is a loop, where Endpoint 1 sends 25 bytes, and Endpoint 2 receives 25 bytes and sends 1,500 bytes back to Endpoint 1, which receives the 1,500 bytes. If the TCP/IP stack on Endpoint 1 has implemented the delayed `ACK` algorithm, it typically waits

(50-200ms) before sending back an `ACK`. In the `NNTP` script, the `SEND` of 1,500 bytes is broken up into 2 frames (one of 1,460 bytes and one of 40 bytes). The 1,460 byte frame is sent, and the TCP/IP stack on Endpoint 1, using the delayed `ACK` algorithm, may wait up to 200ms before sending the `ACK`. After the `ACK` is sent, the 40-byte buffer is sent by Endpoint 2. This time, Endpoint 1's receipt of 1,500 bytes is satisfied, and Endpoint 1 sends the 25 bytes with the piggybacked `ACK`. Because Endpoint 1 is waiting the 200ms, each timing record takes about 20 seconds.

We have discovered in our testing that protocol stacks on Linux, NetWare, and IBM's MVS operating systems have implemented the delayed `ACK` algorithm. On other operating systems without this algorithm, the script runs in less than 2 seconds because the delays are reduced.

- **Changing Software Versions**

  Make particular note of software versions, BIOS levels, CSDs, fix packs, and service packs used on the endpoints. All software we know has its behavior altered with each change you make. Some software is being continually improved by its developers, trying to get ever better performance. Other software grows in features, which add more paths through the software. Sometimes, additional internal checking is added to make the software more robust, which slows its overall performance.

  To get consistent results between a pair of computers, every piece of relevant software must be at exactly the same version and fix level. This includes the BIOS software, the operating system software, the network device drivers, the network protocol stacks, and the IxChariot software itself.

- **Virus Scanners**

  Sometimes virus-scanning software can use up a large percentage of CPU time while the endpoints are running. If you have endpoint auditing enabled, virus scanning programs on the endpoints will check the entire audit file, slowing the test down and consuming extra resources. You probably should disable virus scanning before running a test.

- **Network Configuration**

  Modifying the settings on the routers and switches affects the connected computer's performance. Changing the physical configuration of the network can also produce inconsistent results.

- **Number of Pairs per Endpoint**

  If you try to run too many pairs on a single endpoint, they will contend with each other during the initializing phase of a test. Although our tests are designed to ensure that all pairs start simultaneously, contention for CPU time can cause some pairs to start more slowly. This could potentially show up in test results as higher-than-possible aggregate throughput values, such as 120 Mbps on a 100-Mbps network. If you see such values, try reducing the number of pairs running at a single endpoint.

- **Setting the TCP Receive Window**

  Changing the value of the TCP Receive Window parameter at an endpoint can affect the results you see when testing for maximum throughput. The receive window specifies the number of bytes a sender can transmit without receiving an acknowledgment. Note that this is a TCP/IP stack configuration

parameter. Many TCP/IP stacks ship with a default value of 8 KB. Changing to a larger value changes performance, increasing throughput on some stacks and reducing it on others. We recommend experimenting with the values you use for this parameter.

For example, in Windows XP, the TcpWindowSize Registry parameter determines the maximum TCP receive window size of the computer. To set the TcpWindowSize Registry value in Windows XP, start the Registry Editor (`REGEDT32.EXE`), then navigate to the following location:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
Tcpip\Parameters
```

Modify the Registry value named TcpWindowSize (or create the Registry value if necessary), and set it according to the following value data:

- Value Name: TcpWindowSize

- Key: Tcpip\Parameters

- Value Type: REG_DWORD - Number of bytes

- Valid Range: 0 - 0xFFFF

The recommended size for optimal TCP throughput depends on the network itself. For high speed and high latency networks, Ixia recommends a registry setting of 512K bytes. For highest efficiency, the receive window must be an even multiple of the TCP Maximum Segment Size (MSS). Although the maximum size for TcpWindowSize is 64KB, most operating systems support TCP Large Window Extensions (RFC1323). RFC1323 provides a window scale option (WSCALE), which permits TCP to support window sizes larger than 64KB. Most systems automatically request WSCALE when the socket buffer is larger than 64KB or when the other end of the TCP connection requests it.

Correspondingly, in the application script, change the `send_buffer_size` and `receive_buffer_size` to one-half the Registry value of TcpWindowSize.

- **Other Active Programs or Network Activity**

Other programs that are active alongside the endpoint programs compete with the endpoints for the system CPU, affecting performance. This is especially noticeable with DOS and Windows applications.

Unless you are using dedicated wiring between endpoint programs, you may be competing for network bandwidth with programs running on other computers. Competing traffic affects test results. We've also noticed that a network adapter can keep busy handling excessive broadcast or multicast traffic in a network.

- **Console and Endpoint Together in One Computer**

The IxChariot Console and endpoint can reside together in the same computer. However, the endpoint program is competing with the Console for resources. We recommend not using the endpoint at the Console computer when doing serious performance measurements.

- **User-Created Data Files in Tests**

There is no imposed limit on the size of `Userxx.cmp` files on Ixia Linux endpoints, other than the availability of system resources. We have successfully created and used `Userxx.cmp` files ranging from 1MB to 10MB. Because an

endpoint loads the entire .cmp file into memory when the test is started (to avoid disk I/O while the test is running), it is possible that a .cmp file could exceed available memory.

If your `Userxx.cmp` file is too large to fit in memory, you'll receive error message **CHR0270**. Try increasing the limit placed by UNIX systems upon shared memory segments, as described below.

• On IBM AIX, Compaq Tru64 UNIX (Digital UNIX), and Sun Solaris, the limit is the largest amount of memory a process can allocate, which is determined by the amount of virtual memory.

• On Linux, there is no way to configure a larger shared memory segment. The limit depends on the implementation of Linux.

• On HP-UX, use the HP-UX `SAM` facility:

   • As a root user, start `SAM` by entering `sam`

   • Open the Kernel Configuration menu

   • Open the Configurable Parameters menu

   Increase the `shmmax` parameter as necessary.

• **Protocol Configuration and Tuning**

   Changing a network windowing parameter or buffer size in any network hardware or software will vary the results you see.

• **Screen Savers**

   Screen savers consume CPU resources mightily when they kick in. Make sure they aren't active on endpoint computers while taking serious measurements.

• **Sleep Time Variations**

   The `SLEEP` command on many platforms, or operating systems, has a granularity of approximately 10 milliseconds (ms). This means that even though you can specify a `SLEEP` distribution of 1 to 10 ms, for example, the actual sleep times may cluster around 10 ms. Similarly, if the distribution is from 11 to 20 ms, the actual times may cluster around 20.

   Some platforms, such as AIX and HP-UX, are better at sleeping for the requested period of time. Do some experimenting to get a more precise sense for how the endpoints are interpreting the `SLEEP` commands in your scripts.

• **Foreground vs. Background Endpoint Programs**

   Different operating systems behave differently in how they allocate resources to programs that run in the foreground, in the background, as an icon, or as a service or detached. Often, this behavior is tunable.

• **Multiple Endpoint Pairs on One Computer**

   Running a large number of endpoint pairs on a single computer can create interesting results. We have seen two different situations of which you should be aware.

   First, in TCP/IP, we have seen that the protocol stacks allow scripts that create large data flows, such as file transfers, to dominate access to the protocol stack. This means that a disproportionate percentage of the data traffic comes from the larger transactions. If you are trying to get a good mix of small and

large transactions, we recommend using one endpoint system for each type of transaction.

Second, even though multitasking operating systems are supposed to be "fair" about giving different processes equal amounts of time, you will always see variations. For example, when running 20 connections on one system, we normally see about a 20% difference in performance between the highest and lowest. If you are testing an intermediate system, such as a switch or router, this does not make much difference, because you are mostly concerned with the aggregate throughput of all the pairs.

To stress-test a piece of equipment with few endpoint computers, you might create "virtual" network addresses for the computers you have. For more information about virtual addressing, see *Configuring Virtual Addresses on Endpoint Computers* on page 8-15.

- **Available RAM and Disk Swapping**

   The amount of RAM in a computer affects program performance in many ways. Performance degrades significantly whenever swapping to disk occurs (that is, there is not enough physical RAM).

12 *Troubleshooting*

Related Topics

If you encounter problems while running IxChariot tests, they're probably related to failed connections, to the communications software in your network, or to the way you've configured your tests. IxChariot's messages offer information to help you understand why errors occur and guidance to help you avoid them in the future.

Figure 12-1.   IxChariot Error Message
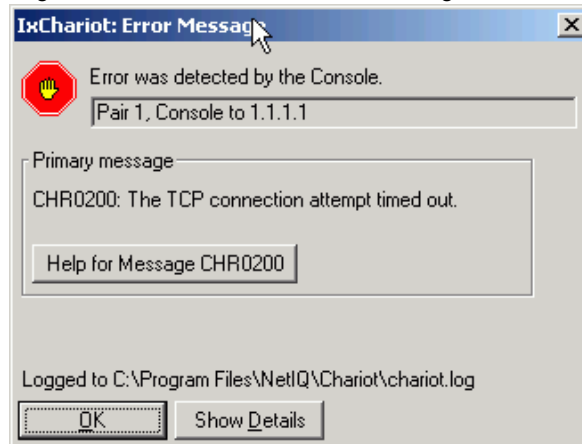
The following topics will help you find the information necessary to solve problems you encounter.

IxChariot logs every error that occurs during testing and provides an interface to help you analyze errors. See *The Error Log Viewer* on page 12-8 for more information.

Troubleshooting Guidelines

Related Topics

Troubleshooting on page 12-1
Determining Which Computer Detected the Error on page 12-2

Our Support Team is always happy to assist you with any problems you encounter. We recommend you try the following steps before calling for assistance, as you can usually locate efficient solutions in the existing documentation for many common problems:

1. Check the Technical Support Web site. This site provides the following:

    •   Frequently-asked questions (and answers)

    •   Links to the latest fixes and third-party software updates

    •   Downloads of the latest documentation, including product user guides and specification sheets

    •   Tuning tips

    •   Application Script Library

    To reach the Technical Support Web site, point your browser to
    www.ixiacom.com/support/chariot/.

2. Review the "*Troubleshooting*" topics. They provide solutions to many common problems as well as information about viewing the error logs and getting the latest product updates and fixes.

3. Review the README file. This file contains updated information that does not appear in this version of the manual. It Is a good idea to print this file and keep a copy nearby.

Determining Which Computer Detected the Error

Related Topics

The Error Log Viewer on page 12-8

The first step in troubleshooting is usually determining which computer detected the error. All **Operator Actions** described by the message help should be taken at the computer that detected the error, unless otherwise specified.

•   For most errors involving setup and file manipulation, the Console is the computer that detects the error.

•   For errors that occur while running a test, the error could have been detected on a particular endpoint pair by the Console, by Endpoint 1, or by Endpoint 2. The program that detects the error reports to the Console, which shows the error and logs it. The first line of the Console error message tells which endpoint computer detected the error.

If for some reason you cannot see the error at the Console, examine the error log at the endpoints involved in the problem. (See *The Error Log Viewer* on page 12-8 for more information.) A formatted error log entry should contain a line that resembles the following:

```
Error was detected by Endpoint 1.
```

If the error was detected by Endpoint 1 or Endpoint 2, check the test setup at the Console to determine the actual network address of the computer where the error was detected.

Although one computer may detect an error, the solution may actually lie elsewhere. For example, if Endpoint 1 detects an error indicating that a network connection could not be established, it may be because of a configuration error in the middle of the network at Endpoint 2, or between Endpoint 1 and Endpoint 2.

If You Find a Problem

Related Topics

Problem Sleuth on page 12-4

We need your help in collecting information to pinpoint any problems you might encounter.

1. Write down what you were doing when the problem occurred.

2. Most important is the sequence of steps you took, the keys you clicked or menu selections you made, and the files you were working with.

3. Collect relevant IxChariot test and results files.

4. Capture the test file you were using when the problem occurred (a binary file created at the Console, with extension `.tst`) and save the file.

5. Collect IxChariot debugging files. Capture any debugging files generated by our product:

   • `assert.err` (an ASCII file)

   • `Chariot.log`, `runtst.log`, `clonetst.log`, `chrapi.log`, `endpoint.log` (binary files)

   • Protection fault information (ASCII files)

   • For Windows NT, file `drwtsn32.log` (an ASCII file).

   GanyPS, a utility that ships with IxChariot, gathers relevant data for you so that you may FTP it back to the Ixia Technical Support team.

   To run `ganyps.exe`, enter the following at a command prompt in the directory where you've installed IxChariot:

   ganyps filename

   See *Problem Sleuth* on page 12-4 for more information about GanyPS.

6. Collect network configuration files.

   Capture the binary configuration files for the network protocols you are using. For TCP/IP, copy the files found in the following directory:

   `d:\path\system32\drivers\etc`

   where *d:* and *path* are the drive and path where you installed Windows 2000/2003/XP.

7. Take network trace files.

   To recreate a problem, we may ask you to activate tracing for the network protocols you are using. If you recreate the problem with tracing active, we'll need the resulting trace files and the .tst file with the correlated IP addresses seen in the trace.

8. Capture operating system configuration files.

IxChariot writes entries in the Windows Registry. If you are familiar with using the REGEDT32 command, you may consider capturing a text listing of the Registry.

Problem Sleuth

Related Topics
If You Find a Problem on page 12-3

In many cases, our support personnel will ask you to gather certain files from your machine that are needed to determine the sort of problem you are experiencing. We've included a tool that will automate that procedure for you.

Gany Problem Sleuth (GanyPS) collects pertinent data for problem determination, which you may FTP to the people on the support team who can find a quick solution.

GanyPS builds a list of files and zips them within a directory created during product installation.

To run GanyPS, enter the following at a command prompt in the directory where you've installed IxChariot:

```
ganyps filename
```

The usage is as follows:

filename: When you contact the support team, you should receive an incident number, expressed as the filename to enter here

Reading Error Messages

Related Topics

The Error Log Viewer on page 12-8

IxChariot errors are reported at the Console in the Error Message dialog box. To re-open an error message dialog box highlight the affected pair in the Test window and click **Show Error Message** on the View menu.

The top line identifies which computer detected the error. The next line gives the pair number and the network names of any affected endpoints.

- **Primary Message**

  The text of the error message. Matches the primary text of the message in the *Message Reference*.

- **Message Help**

  Explains why the error probably occurred and offers advice for avoiding it in the future.

- **Secondary error information**

  Provides technical information for further isolation of the problem. Does not appear in every message. For SPX and TCP problems, shows the port number and call number,

- **Show Details**

  Provides advanced technical information about the problem. For example, it shows the return code number for failed communications calls. For SPX and TCP problems, shows the port number and call number, among other values.

  The date and time of the error are always listed first, followed by where the error was detected. The text of the primary and optional secondary messages follows, including their error numbers.

  The details that are logged next vary, depending on the type of error and the network protocol being used. Network personnel and communications programmers might find these details helpful in debugging failures. Some of the details are useful only to the Ixia Technical Support team.

  You can copy text in this dialog box to the clipboard: highlight it and click **Ctrl+C**. Text that has been copied to the clipboard can be pasted into any compatible application (**Ctrl+V**).

- **Logged To**

  The path and name for this error log file. All communications errors reported to the Console, or that the Console detects, are written to the Console's error log.

See *The Error Log Viewer* on page 12-8 for more information about error logs for IxChariot and the endpoints.

See the *Message Reference* for a numerical listing of all IxChariot error messages.

Show Error Message

Related Topics

Reading Error Messages on page 12-5
The Error Log Viewer on page 12-8

If there's an error associated with a selected endpoint pair, click this item on the View menu to open the Error Message dialog box. **Show Error Message** is grayed out until a test encounters an error. It remains grayed until you highlight the pair or pairs that are marked "Error: CHR*n*" in the Test window. The error message number (*n*) that corresponds to the error in the *Message Reference* is also shown next to the pair.

The IxChariot Console invalidates data from a pair that ends with an error. "N/a" appears in the columns labeled **95% Confidence Interval** and **Relative Precision**.

For more information about error messages, see*Reading Error Messages* on page 12-5. Error messages apply to tests or pairs that did not complete successfully. For tests that did complete their run but have associated warnings to indicate that the data may have been compromised by test setup conditions, click **Show Warning Message(s)** on the View menu.

Finished Test Warnings

Related Topics

Warnings Tab on page 6-35

If a test is configured with Run Options that may make its results inaccurate or non-significant for one or more endpoint pairs, the Warning Messages dialog box pops up when the test completes.

The IxChariot Console does not invalidate the results you receive from a test that finishes with these warning messages. However, a warning appears next to the affected endpoint pair where it is shown in the Test window. To re-open the warning dialog box, highlight the pair and click **Show Warning Message(s)** on the View menu.

Warnings **CHR0336**, **CHR0337**, and **CHR0338** can be disabled on the **Warnings Tab** in the Change User Settings notebook. Warnings **CHR0335** and **CHR0358** cannot be disabled. See *Warnings Tab* on page 6-35 for more information.

- **Show Details**

  Provides advanced technical information about the problem. For example, it shows the return code number for failed communications calls.

- **Message Help**

  Explains why the error probably occurred and offers advice for avoiding it in the future. Takes you to the appropriate place in the *Message Reference*.

  The following Finished Test Warnings address common issues with test configuration that may render test results inaccurate or non-significant.

- **CHR0335:** A timing record was received with a measured time of 0 milliseconds.

  A script command was performed in less than one millisecond, an insignificant amount of time. The test cannot be graphed, or the graph of your test results does not include all data. When a timing record with a 0 time value is received, the value is rounded to 1 for calculations.

  If the groups or tests are being shown in a line graph, the timing records with 0 measured times are excluded because they do not contribute to the group or test. For other types of graphs that show maximum, minimum, and average values, the group or test itself is not graphed.

  Edit the script you used for this test so that it takes more time to complete. See *How Long Should a Performance Test Run?* on page 10-86 for a complete discussion.

- **CHR0336:** A timing record was received with a measured time between 1 and 20 milliseconds.

  The clock timers used in timing scripts are generally accurate to within 1 millisecond (ms). For most tests, this is more than sufficient, but if the transactions in a test are too short, problems may arise. For example, the lower the measured time of the timing records, the higher the percentage of the actual measured time that 1 ms represents. If the measured time is 5 ms, there is a + or -10% potential for error, since the actual time is somewhere between 4.5 and 5.5 ms. See *Understanding Timing* on page 11-1 for more information.

  - For more information about the preferred length of timing records, see *How Long Should a Performance Test Run?* on page 10-86.

  - For details on editing script variables such as `transactions_per_record`, see *Troubleshooting Guidelines* on page 12-2.

- **CHR0337:** More than 500 timing records per pair were generated in Batch Reporting Mode.

  Excessive timing records create extra traffic on the line and interfere with performance results. For details on limiting the number of timing records, see *Reducing the Number of Timing Records* on page 10-88.

- **CHR0338:** The test has run for less than 1 second while collecting endpoint CPU utilization.

  Meaningful data on CPU utilization at the endpoints cannot be collected unless the test runs for a longer period of time.

  For more suggestions on test run times, see *How Long Should a Performance Test Run?* on page 10-86.

- **CHR0358:** A substantial inactive time value was received for a timing record.

  Your graphs might look a little different from your results, simply because some of the time the endpoints spent processing the application script or voice over IP transmission took place outside the `START_TIMER` and `END_TIMER` commands (therefore, outside the timing records). Your test data is still valid, but your test did have some "inactive time," which is graphed as

"0," so the graphs may appear slightly different from the results. See *Understanding Timing* on page 11-1 for more information.

Show Warning Messages

Related Topics
Finished Test Warnings on page 12-6
Warnings Tab on page 6-35

If a test completes but a pair or pairs is marked "**Finished: Warnings**" in the Test window, it is likely that your test wasn't configured properly. A warning may appear because a given test is configured in such a way that its results may be inaccurate or non-significant for one or more endpoint pairs. If the results are compromised, the Warning Messages dialog box pops up at the end of a test run.

The IxChariot Console does not invalidate the results you receive from a test that finishes with these warning messages. However, IxChariot places a warning, visible in all tab views, next to the affected endpoint pair where it is listed in the Test window. A pair that shows warnings is labeled "Finished with Warning(s)" in the Test window after a test run completes.

To view information about the warning, highlight the relevant endpoint pair. Then click **Show Warning Message(s)** from the View menu.

See *Finished Test Warnings* on page 12-6 for more information about warnings.

The Error Log Viewer

Related Topics
Basic Tasks in the Error Log Viewer on page 12-9
Error Log Viewer Menus on page 12-12

Similarly, whenever one of the endpoint programs encounters a problem it cannot report to the Console, it logs that problem to an error log file at the endpoint. If you have access to the endpoint computer's Endpoint directory, you can export the endpoint error log files and read them with the Error Log Viewer. Whenever one of the console programs encounters a problem, it logs error information into an error log file at the Console. Use the Error Log Viewer to read the error information. To open the Error Log Viewer, select **View Error Logs** from the Tools menu.

IxChariot writes the Console error log file to the following default directory: C:\Documents and Settings\User\My Documents\IxChariot. You can change the location of this error log by entering a new location in the **Where to write console error logs** field in the **Directories** tab of the Change User Settings window. For Windows endpoints, the endpoints' error logs are written to the directory where the endpoints are installed. For the location of the endpoints' error logs for other types of endpoints, please see the corresponding endpoint chapter in the IxChariot Performance Endpoints guide.

• The Console writes to the file `Chariot.log`

- `RUNTST` writes to the file `runtst.log`

- `CLONETST` writes to the file `clonetst.log`

- C and Tcl API writes to the file `chrapi.log`

- Endpoints write to the file `endpoint.log`

The Error Log Viewer shows the record number of the entry, the date and time, the source of the error and a brief description of the error.

You can select the criteria for the entries that you want to view. See *Filtering Log Entries* on page 12-11 for more information.

If you need to view an error log on an endpoint computer, use the `FMTLOG` program to format the binary error log. See *Functional Limitations and Known Problems* on page 12-16 for more information.

To maintain excellent performance from IxChariot, you should keep the log files fairly small. As soon as the `Chariot.log` or `runtst.log` files exceed 5 MB, IxChariot warns you with a popup message on startup. IxChariot will never stop logging errors, no matter how large your log files become, but you should delete large log files to keep them from affecting performance. The Error Log Viewer does not include support for deleting individual log entries.

Basic Tasks in the Error Log Viewer

Related Topics
Searching an Error Log on page 12-10
Filtering Log Entries on page 12-11
Viewing Error Details on page 12-11
Error Log Viewer Menus on page 12-12

- **Open Logs**

  To open an error log file, click **Open** on the File menu. From the Open a Log File dialog box, select the file you want to open and click **Open**. The error log is shown in the Error Log Viewer.

- **Save Log Information to a File**

  Save the information shown in the Error Log Viewer to a file by clicking **Save Filtered Log As** on the File menu. If entries are filtered, the Error Log Viewer saves only the entries that meet the current filtering criteria. Enter a filename in the dialog box and click **Save**.

- **Return to IxChariot**

  Click **Exit** on the File menu to exit the Error Log Viewer and return to the IxChariot GUI.

- **Filter Entries**

  Using the View menu in the Error Log Viewer, you can view all entries shown in the Error Log Viewer, or you can view only the entries that meet specified criteria. To view all entries, click **All Entries** on the View menu. To specify the criteria for the entries you want to view in the Error Log, click **Filter Entries** on the View menu. See *Filtering Log Entries* on page 12-11 for more information.

•   **Sort Entries by Record Number**

The **Ascending** and **Descending** menu items on the Sort by Record Number submenu let you view the entries shown in the Error Log Viewer in either ascending or descending order.

•   **View the Most Recent Errors**

To update the Error Log Viewer, click **Refresh**. The Error Log Viewer reloads the error log that you are viewing. Any newly-written entries are now shown in the Error Log Viewer.

•   **Navigate Quickly Among Entries**

On the Options menu, **Wrap Log** lets you move from the last entry shown in the Error Log Viewer to the first entry by clicking **Next** in the Log Viewer Details dialog box. Wrapping also lets you go from the first entry to the last entry by clicking **Previous**. A check mark indicates that Wrap Log is enabled by default. To disable this option, remove the check mark.

•   **Save Your Error Log Viewer Settings**

You can save the following settings to be used the next time you access the Error Log Viewer by clicking **Save Settings on Exit** on the Options menu:

   •   Filtering

   •   Sorting

   •   Wrapping entries

   •   Font

The Error Log Viewer saves the most recent settings for these items when you exit.

•   **Change the Font**

Select **Font** to change the fonts used in the Error Log Viewer.

Searching an Error Log

Search the entries shown in the Error Log Viewer to find a specific word or phrase. Click **Find** on the View menu to access the Find dialog box.

•   **Search for**

Enter or select the word or phrase you want to find. The search is not case-sensitive.

•   **Up**

Begins the search from your current cursor position and proceeds to the top of the error log.

•   **Down**

Begins the search from your current cursor position and proceeds to the bottom of the error log.

•   **Wrap at end**

Begins the search from your current cursor position, but searches the entire error log and returns to the current position.

- **Find**

  Begins the search. The first occurrence of the word or phrase in the Search for field is highlighted.

- **Find Next**

  Finds the next occurrence of the word or phrase you are searching for.

Viewing Error Details

To get more information about an entry in the Error Log Viewer, highlight the entry and click **Detail** on the View menu. The Error Log Viewer Details dialog box for the selected entry shows detailed information about the selected entry.

- **Record**

  Shows the order in which errors were detected.

- **Help for message**

  Provides more information about the error, including the steps you can take to avoid it.

- **Next**

  Shows details about the next entry in the error log. When you click **Wrap log** on the Options menu, the Error Log Viewer wraps around from the end back to the beginning. For example, if you selected the last entry in the Error Log Viewer, you can view the first entry by clicking **Next**.

- **Previous**

  Shows details about the previous entry in the Error Log Viewer.

Filtering Log Entries

Click **Filter** on the View menu to determine which entries in the error log are shown in the Error Log Viewer.

- **View From fields**

  Filter out records with certain date/time combinations. If you want the Error Log Viewer to show the first entry generated, click **First**. To begin with an entry generated at a certain time, click **View From**. In the **Date** fields, select the date of the first entry to show. In the **Time** fields, select the time of the first entry to show. Filter selection is based on a 24-hour clock (that is, for 11 PM select 23), but the Log Viewer actually displays the errors with "AM" and "PM" designations.

- **View Through fields**

  If you want the last entry shown in the Error Log Viewer to be the last entry generated, click **Last**. To view only the entries generated before a specific date and time, click **Entries on**. In the **Date** fields, select the date of the last entry to show. In the **Time** fields, select the time of the last entry to show.

- **Only show records with**

  If you want the Error Log Viewer to show only entries containing a specific error message, check **Primary Message**. In the **CHR** field, enter the message number that you want to view.

- **Detector**

  To view entries in the error log that were generated or detected by a specific component of an IxChariot test, check the check boxes next to the sources whose errors you want displayed in the Error Log Viewer. In the endpoint error log, the term *Master* refers to the main endpoint program; if an error occurs during test initialization, endpoints cannot yet be distinguished by the terms Endpoint 1 and Endpoint 2.

Error Log Viewer Menus

The Error Log Viewer includes the following menus:

Table 12-1.   Error Log View Menus

Menus	Tasks
File	**Open** an error log **Save** an error log **Exit** the Error Log Viewer
View	**Filter** the entries shown in a log **Sort** the entries shown in a log **Find** a keyword or phrase View **Detailed** Error Information **Refresh** the entries shown in the Error Log Viewer
Options	**Wrap** entries in a log **Save** log **settings** upon exit Change the **font** used in the Error Log Viewer
Help	Access **help** for the Error Log Viewer

Shortcut Keys for the Error Log Viewer

You can use the following keys and key combinations in Error Log Viewer, instead of using the mouse.

Table 12-2.   Shortcut Keys for the Error Log Viewer

Key or Key Combination	Command Invoked
F1	Get help for the Error Log Viewer.
F2	View an index of all the available help topics.
F3	Exit the Error Log Viewer.
F9	Show the keys and key combinations available in a window.
F11	Open the About Error Log Viewer dialog box, which shows your version and build level, and lets you get product support information.

Table 12-2.   Shortcut Keys for the Error Log Viewer (Continued)

Key or Key Combination	Command Invoked
Ctrl+S	Save an error log file.
Ctrl+F	Search for a word in the Error Log Viewer.
Ctrl+G	Find the next occurrence of the last word you searched for in the Error Log Viewer.
Ctrl+O	Open an error log file.
Alt+F4	Close any window or dialog box. When used to close a dialog box, it has the same effect as clicking the **Esc** key or clicking **Cancel** with the mouse.

In addition to these keys, the **Alt** key can be used in combination with any under-scored letter to invoke a menu function. The menu function must be visible and not shown in gray. For example, clicking **Alt**+**F** shows the File menu.

Common Problems

Following are descriptions of problems you may encounter. We've compiled this section after a great deal of testing and years of contact with customers.

Assertion Failures

IxChariot does a lot of internal checking on itself. You may see the symptoms of this checking with an "Assertion failed" message. If you see this in a message at the Console, it asks whether you want to exit. The best choice is "Yes"—choosing "No" probably results in a protection fault or yet another assertion failure. If you choose No and you are able to continue (that is, the bug was minor), we recommend saving your test files as soon as possible.

If you encounter an assertion failure, please write down the sequence of things you were doing when it occurred. By default, IxChariot captures details related to the problem in an ASCII text file named `assert.err` in C:\Documents and Settings\User\My Documents\IxChariot. Save a copy of the `assert.err` file, and send it back to us via email at *support@ixiacom.com*.

Damaged Files

Binary files can be damaged (that is, truncated) if you copy them using wildcards at a command prompt. The problem, which stems from a limitation in DOS, occurs when the hex character `X'1A'` is encountered. For example, the following command is the *wrong* way to copy a binary script file for the database-update transactions:

```
copy db*.scr temp.scr
```

Doing a `DIR` for the file `Dbases.scr` and the file `Temp.scr` should show that the `COPY` command has truncated the file. The following command is also the *wrong* way to copy binary files:

```
copy db*.scr temp.scr /b
```

This creates a truncated file that is even a byte smaller than the previous copy. Here's the *right* way to copy binary files—don't use wildcards!

```
copy dbases.scr temp.scr
```

High-Precision Timer (CHR0359 error)

If you attempt to execute a VoIP or streaming pair test and you receive a CHR0359 error message (An error was detected in the high precision timer), you may be able to resolve the issue by following the instructions given in the Microsoft Knowledge Base article entitled "Programs that use the QueryPerformanceCounter function may perform poorly in Windows Server 2000, in Windows Server 2003, and in Windows XP". You can either search for the article by name or reference it at this address: http://support.microsoft.com/kb/895980.

Insufficient Resources

If you receive an Insufficient Resource error while running IxChariot, your computer does not have access to the amount of memory required to successfully run IxChariot. You should close other applications that you currently have running and then restart IxChariot.

Insufficient Threads

The IxChariot Console creates one or more threads for each endpoint pair when running a test. This is in addition to the threads created by the underlying network software (as well as those used by other concurrently-running applications).

In our testing, we did not exhaust threads in our default settings for Windows NT or Windows 2000/2003 until we reached about 7000 threads. We don't believe you'll encounter out-of-threads problems, but please let us know if you do.

Locale Could Not Be Determined

The locale file tells IxChariot the language of the version of IxChariot that you are using. Based on the language, IxChariot determines what time and date format, the comma separation, and the type of money symbol to use.

The locale file is located in the directory where IxChariot is installed. The format of the filename is `lang[xxx].lcl,` where `xxx` denotes the code for the language used in this version of IxChariot. For example, if you are using the English language version of IxChariot, the name of the locale file is `langenu.lcl`. The settings pre-defined in the locale file override any settings you select on the Regional Settings in the Control Panel.

If the locale file is not located in the directory where IxChariot is located, IxChariot will not run, and you will receive an error message. In such a case, the file might not be in the directory because of an install error or because it has been deleted. You'll need to reinstall IxChariot so that the locale file will be reinstalled. IxChariot should now run correctly.

Network Congestion

Network congestion may result in a number of different failures:

• Incomplete sessions

• Lack of test results

In addition to reducing the amount of traffic, it may be helpful to change the TCP connection `close_type` from *Normal* to *Abortive* (using the Script Editor).

Refer to the *IxChariot Scripts Development and Editing Guide* for detailed information about TCP connection types.

Protection Faults and Traps

Protection faults or traps are the operating system's way of telling you when a program is trying to use memory that it doesn't own. They can occur in an IxChariot program, in any library routines called by IxChariot, or in the operating system itself. The default way that they're handled is with a message box. This message box shows program instruction values in hex, which aren't helpful to you as a user.

- **If you are using Windows NT, Windows 2000/2003, or Windows XP:**

  Windows NT, Windows 2000/2003, and Windows XP write an entry to a file named `drwtsn32.log` when they encounter a trap or protection fault. This file is written to the directory where you installed Windows. Its default location is `c:\Winnt`. It contains information that is immensely helpful to us in finding and fixing bugs. When you contact the Technical Support team, they may suggest that you send the Dr. Watson file to us via email.

Streaming Testing

For streaming scripts, don't set your file size or buffer size too low when sending at a high data rate. Small file or buffer sizes cause Endpoint 1 to generate too many timing records. Large sizes avoid returning an aggregate throughput[1] value that's greater than the network's capacity (which can only occur with non-zero `SLEEP` times).

The Packet Blaster scripts may appear to emulate devices which generate large volumes of background traffic, but they don't. Application scripts require reliable delivery of data (which implies the use of acknowledgments), whereas frame throwers do not. Streaming scripts would therefore be more appropriate. See the *Application Scripts* guide for information on setting the delivery rate for data.

Throughput Spikes in Streaming Tests

In cases where many pairs are running streaming scripts, at times a receiver may not be able to receive each datagram as fast as it comes in, due to thread switching or timing mechanisms. If this happens, datagrams may be queued in the TCP/IP stack up to a certain limit. On Win32 operating systems, this limit is 8 KB, and on most Unix OSs, the limit is 64 KB. When the receiver is able to receive these datagrams, they are received very quickly, which may appear as a spike in the throughput results. The Measured Time will also be very low in these cases, in comparison with what is expected.

If you see similar results fairly frequently, you may need to run fewer pairs on the Endpoint 2 computers. Or you may try reducing the `send_data_rate` of the sending computer.

---

1. IxChariot measures the throughput associated with packet payload, ignoring headers. This is referred to as Goodput in RFC 2647.

Windows
Application Errors
During Large Tests

During tests with many pairs running simultaneous scripts on Windows end-points, extra stress on the operating system can cause it to log application errors to the Windows Event Viewer. These errors are only logged if the endpoints are also collecting data on CPU utilization. In Windows 2000/2003, the event looks something like the following:

```
The timeout waiting for the performance data collection
function 'PerfOS' in the 'C:\WINNT\system32\perfos.dll'
Library to finish has expired.
```

When the endpoint operating system is under stress, it may miss one or more of its attempts to collect CPU samples. The endpoint queries the CPU collection thread every .5 seconds; therefore, even if you see this error or a similar one, the endpoint has typically managed to collect enough samples for a reasonably accurate CPU utilization measurement, particularly if only a few application errors are logged within a short time frame.

If many errors are being logged frequently, check to see if antivirus software is active during the test. You should always disable antivirus software during a stressful network test.

Functional Limitations and Known Problems

The latest functional limitations in this version of IxChariot are described in the README file. A README file is installed at the Console and at the endpoints.

Check our Web site regularly for information about the latest bugs and fixes; the IxChariot FAQ page is a good place to start.

Known Issue with Windows Server 2003 SP1

An IxChariot test may end with an erroneous CHR245 error under the following circumstances:

*   Endpoint 1 is on the same machine as the IxChariot Console.

*   Endpoint 2 is running the Windows Performance Endpoint version 6.30 or above (non blocking command manager), installed on a Windows 2003 Server running Service Pack 1 (SP1).

*   The protocol is RTP.

*   The test has a relatively large number of pairs (over 200).

This problem occurs because Microsoft Windows Server 2003 Service Pack 1 (SP1) implements a security feature that reduces the size of the queue for concurrent TCP/IP connections to the server. This security feature is intended to help prevent denial of service attacks. Under heavy load conditions, the TCP/IP protocol in Windows Server 2003 SP1 may incorrectly identify valid TCP/IP connections as a denial of service attack. This behavior could cause the test to end with an erroneous CHR245 error.

You can avoid this problem by disabling the security feature in Microsoft Windows Server 2003 Service Pack 1. Refer to Microsoft Article ID 899599 (http://support.microsoft.com/default.aspx?scid=kb;en-us;899599) for additional information and instructions.

Operating System Limitations: Streaming Tests with Solaris or MVS Endpoints

You might see some operating-system problems with endpoints running on MVS or Solaris platforms. The problem appears with streaming tests, either at a very fast rate or with many pairs using small datagram buffer sizes. With these tests, the endpoint may cause the operating system to lock up.

We have seen lock-up problems with Solaris (version 2.6 and later) when running certain kinds of streaming tests. In particular, we ran a 35-pair IxChariot test in which each pair used an application script that specifies small buffers (40 bytes each) at 64 kbps. Running this test to a Sun Ultra 5 computer (as the Endpoint 2) caused Solaris to completely lock up; the computer did not respond to network, keyboard, or mouse input.

The cause is that the Endpoint 2 computer was overwhelmed with thousands of small datagrams, which the TCP/IP network stack could not process quickly enough. Either the RAM (in our case, the computer had 64 MB of RAM) or CPU power needs to be increased to handle the load.

Known Problems in IBM's Communications Server for Windows NT

- MVS reports **CHR0129** during a connection

  Occasionally when running throughput-intensive tests from CS/NT to MVS, MVS will report "**CHR0129** An established APPC connection failed during processing" with sense data of `0x20010000`.

- CS/NT version 5.00 returns sense data of `0x00000000`

  CS/NT version 5.00 returns sense data of `0x00000000` when a connection fails due to reaching the session limit for the mode specified.

  Solution: If sense data `0x00000000` is received, consider whether this may be your problem.

- CS/NT version 5.01 returns unexpected sense data.

  In CS/NT version 5.01, all sense data is essentially unusable.

Known Problems in Microsoft's SNA Server

If you are using Microsoft's SNA Server, we strongly recommend version 4.0 or higher.

At a minimum, SNA Server version 2.11 with Service Pack 2 is essential if you hope to do much extended testing. With SNA Server 2.11, there are still numer-

ous problems. Many of these can potentially lock up the SNA Server clients, requiring you to reboot them.

We've also found vast differences between the performance and stability of the different LAN protocols that SNA Server uses to connect clients to itself. The worst performance and least stable tests involve running multiple pairs from client to client. Better performance and stability occur running from SNA Server client to SNA Server, and the best performance occurs running from SNA Server to other genuine APPC computers, such as Communications Server for NT or another SNA Server.

Here are the major items we've seen, grouped by LAN protocol.

- **APPC**

    For Win32 computers using SNA Client, stopping and starting the SnaBase service while the IxChariot GUI is open can cause IxChariot to crash after running tests with APPC.

- **TCP/IP**

    Overall, having SNA Server clients connect over TCP/IP gives the best performance and stability. However, running more than about 3 pairs per endpoint often causes hangs during the test. These are usually reported with IxChariot message **CHR0257**. Refer to the Help for this message for potential solutions.

- **Named Pipes**

    This is somewhat stable, though performance is approximately half that of TCP/IP. In addition, running more than a few tests at the same endpoint will often return with the SNA Server client reporting "Out of Memory"—even on computers with substantial amounts of RAM.

- **IPX/SPX**

    Running even one pair using a script with good throughput characteristics runs well for a short period of time (about 20 or 30 seconds), but then performance will decrease several orders of magnitude.

Thus, the problems we've encountered generally center around exceeding the low capacity of the SNA Server.

Error #625 may occasionally occur while booting if the endpoint is using SNA Server: "`The SnaBase service must be started before any Service which uses SnaBase.`" The only work around is to start the endpoint service manually.

Known Problems in Microsoft's TCP/IP for Windows NT/ 2000

Microsoft's TCP/IP stack is sensitive to extended barrages of short connections. Performance will remain consistent for a while, but you can expect to see delays of 3 or 6 seconds periodically as TCP/IP recovers its internal control blocks.

Getting the Latest Fixes and Service Updates

We recommend working with the very latest upgrades for the underlying operating system and communications software. Here are the best sources we've found for the software used by the Console and endpoints. Additional information is provided in the individual endpoint topics in the *Performance Endpoints* guide.

Updates for IBM's SNA Software for Windows NT or Windows 2000/2003

For information on IBM's Personal Communications (PCOMM) family of software, access their Web site at www.software.ibm.com/network/pcomm/support/

For information on IBM's Communications Server for Windows NT, access their Web site at www.software.ibm.com/network/commserver/support/fixes/fixes_csnt.html (fixes).

Updates for Microsoft SNA Server

Microsoft posts code and driver updates directly to the following Web site: www.microsoft.com/sna/default.asp

We strongly recommend that users of Microsoft SNA Server version 2.11 apply Service Pack 2 to the SNA Server. Updates are available from the Web site mentioned above. See the README file associated with this Service Pack for more information.

Also, the file 211sp2ic.exe (at the above downloads site) is recommended for Windows NT computers running the SNA Server client version 2.11 software.

Updates for Microsoft Windows

Microsoft posts code and driver updates directly to the following Web site: www.microsoft.com/windows/downloads

Updates for Novell Client Software

Novell posts code and driver updates to the following Web site: http://support.novell.com/

A   *IxChariot File Types*

IxChariot and the endpoints use and create several different types of files. The IxChariot test file, with the extension `.tst`, and the application script file (`.scr`) are the basic IxChariot file types. To ensure the integrity of the data in a test file, IxChariot protects your files and ensures that while a test file is open, other programs cannot write to it. When script files are opened, they are read directly into the test being constructed or modified. A test file contains a separate script for each endpoint pair, allowing full flexibility in the choice of script variables. Scripts are stored in a compact, binary format, so script files and test files (without results) are rarely large.

Test and script files are protected from damage by a checksum. This makes modifying them with a binary or hexadecimal editor impractical.

IxChariot uses the following naming conventions for file extensions, listed alphabetically:

Table A-1.   File naming conventions

File Extension	File Description
.AUD	An ASCII audit file found at the endpoints. The file endpoint.aud contains a record for each time a test is started and stopped. The records are in comma-delimited format, enabling easy input into spreadsheet programs. For more information, see the SECURITY_AUDITING and AUDIT_FILENAME keywords for the endpoint.ini file described in your Performance Endpoints guide.
.CHM	Help files. This format can only be read by Internet Explorer 4.0 and higher.
.CHN	An IPTV channel definition file.
.CSV	An ASCII comma-separated values file, generated at the Console. This contains the results of a run, in a format suitable for loading into a spreadsheet program such as *Excel* or *Lotus 1-2-3*. See *Export Options for .CSV Files* on page 5-63 for information on exporting the `.csv` file format from IxChariot.

Table A-1.   File naming conventions (Continued)

File Extension	File Description
.DAT	An ASCII data file, kept in the directory where the IxChariot Console is started. `Deregister.dat`—contains the 3 fields that are saved when IxChariot is de-registered: registration number, previous license code, and de-registration key `Endpoint.dat`—contains endpoint network addresses entered at the Console `Servqual.dat`—contains RTP/TCP/UDP Quality of Service (QoS) templates `SPXDIR.dat`—contains IPX addresses and their aliases `MCG.dat`—contains IP Multicast groups
.ERR	An ASCII error log file generated internally by any of the IxChariot programs. If the file `assert.err` is generated, a program defect may affect the operation of IxChariot. Keep a copy of the file and be prepared to send it to the support staff if requested.
.GIF	A binary graphics file. If you export in `HTML` format and choose to export graphs of your results, each graph is saved in a separate file. Files in the `GIF` format are suitable for loading into many word processors, graphics applications, and Web browsers.
.IAG	A binary file containing the definition of an Ixia application group. An application group is a collection of pairs (one or more) that will be used in a synchronized pair test. The application group identifies that participating pairs and defines the synchronization events for the test. .IAG files are supported in IxChariot 6.20 and higher.
.HTM	An ASCII file with `HTML` tags. This is the default file extension for exporting tests and results for use as Internet Web pages. You can view a formatted page with any Web browser that supports tables. Embedded graphs are saved as separate gif files.
.INI	An ASCII initialization file found at the endpoints. The file `endpoint.ini` determines the capabilities of the endpoint and what consoles can connect to it.
.LCL	A binary locale file that determines the time and date format, the comma separation format, and the type of money symbol to use based on the language of the version of IxChariot. This file must be in the directory where IxChariot is installed for IxChariot to run.
.LOG	A binary log file generated by the Console, `RUNTST`, `CLONETST`, or any endpoint. See *The Error Log Viewer* on page 12-8 for more information.
.PDF	PDF files. They can be read using Adobe Reader.

Table A-1. File naming conventions (Continued)

File Extension	File Description
.SCR	A binary script file, containing the network calls and their script variables. This file is protected from damage with a CRC checksum, so it should not be modified, even with a binary editor. IxChariot 6.10 can write script files in version 6.10, 6.0, 5.40, 5.20, 5.10, 5.0, 4.3, 4.2, or 4.1 format.
.SDF	Stream Definition File – a TCL-based script that generates stateless Ixia hardware streams for TCP, UDP, and ICMP.
.STR	An Ixia specific stream file. This type of file is used exclusively with hardware performance pairs and VoIP hardware performance pairs.
.TST	A binary test file, containing endpoint pair definitions and their associated scripts, and, optionally, the results of one run. This file is protected from damage with a CRC checksum, so it should not be modified, even with a hex editor. 6.0 can write test files in version 6.0, 5.40, 5.20, 5.10, 5.0, 4.3 formats and can read files from older versions. 6.0 test files cannot be read by older versions of IxChariot.
.TXT	An ASCII text listing file. This is the default file extension when exporting tests and results to a text file.
.XML	A file saved by the IxChariot Visual Test Designer in Extensible Markup Language. Includes data about the test Diagram. Can be viewed in a Web browser.

Index