DATA SHEET

Ixia Test Composer

Key Features

- Test IDE
- Multi-vendor test authoring
- Open architecture framework
- Comprehensive Ixia application support
- Live interactive capture/replay
- Passive DUT monitoring
- Built-in CSV file analyzer
- Shared test resources
- Full–featured debugger
- Simplified test development process
- Customizable test report generation
- Packet decoder analysis tools

Accelerate Testing for Manual and Regression Testers

Automated testing is far more efficient than manual testing. Even when factoring the time it takes to design, write, and verify an automated test script, the overall time savings – when realized over numerous regression test runs – is significant. Further enhancing this efficiency is Ixia's Test Composer, which greatly accelerates the development of automated test scripts. Using a set of open-source plug-ins, Test Composer seamlessly integrates with most network vendors' equipment to provide a consistent interface to the test script development environment, obviating the need to have developers learn the ins-and-outs of every type of network equipment interface.

Traditional test applications are designed for manual testing. For example, IxNetwork and IxLoad require a lot of interaction, where captures and traces must be analyzed and interpreted interactively at test time. The highly repetitive nature of regression testing, however, can be extremely difficult and impractical in an interactive environment. Human-introduced variations in test execution will interfere with the test and will most certainly create suspicions of any results. Repetitive testing could thus be accomplished far more efficiently in an automated environment.

A good method for automating a test environment consists of capturing test sequences from the manual testing environment and then automating the repetitious steps. A complete automated test can be constructed by collecting and building upon sequences of these captured steps. Such an automated test may well be run tens – if not hundreds – of times for regression and troubleshooting purposes, where human-introduced variations are non-existent. Test engineers are therefore looking to accelerate and qualify their regression tests by creating an automated regression testing environment. Test Composer provides the solution.

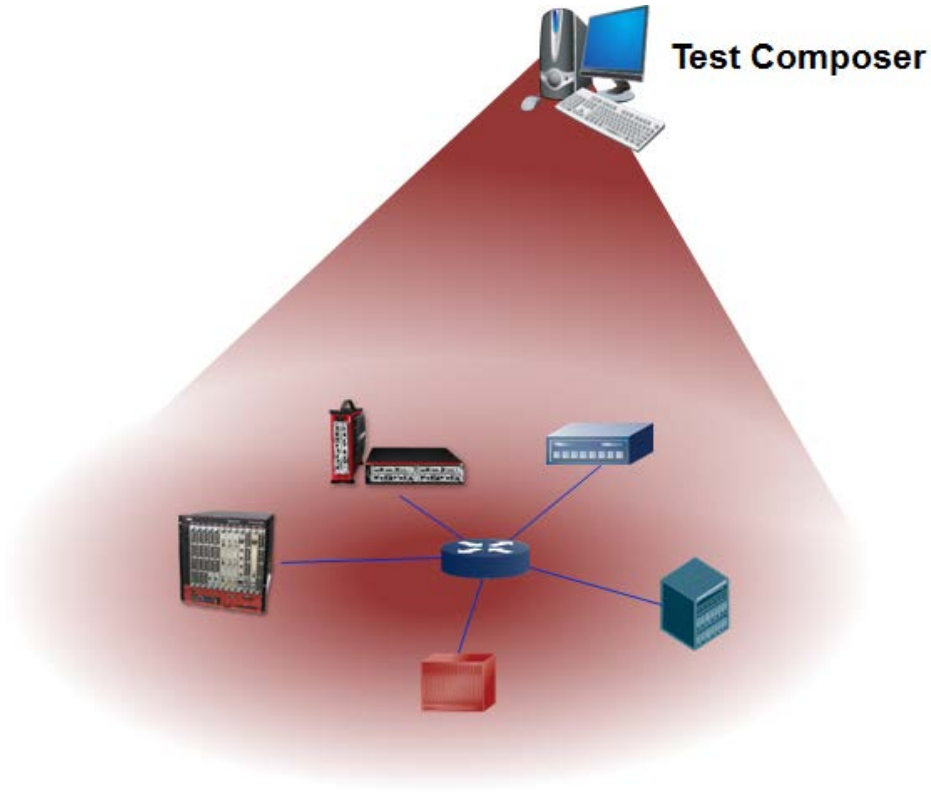

Figure 1 – Sample Figure

Automating the entire test bed speeds up test execution

True network test automation involves more than just automatically launching network traffic generation and collecting the results from the test application. Test automation must also include configuring the device or system under test (DUT/SUT) as well as implementing controlled changes in the lab test bed topology. Ideally, automated tests should run end-to-end without the unpredictability of human intervention. Furthermore, to accurately assess the behavior of the DUT/SUT, tests must be highly repeatable so that multiple test runs can be used to eliminate the effects of spurious variations in the results.

Further complicating the network testing environment is the fact that network elements are themselves very complicated devices. The intricacies of each network element must be thoroughly understood by the test engineer to create an acceptable test environment. This inevitably requires a great deal of time, as the test engineer must familiarize him/herself with the equipment being tested as well as the proper environment in which the test must run. Obviously, as network elements become more complex, so do the tools needed to test them.

For the quality assurance (QA) manager, finding enough test engineers with the right set of skills can be extremely difficult. QA managers must often choose between someone who understands the network elements and their underlying technology, and someone who is proficient at scripting the test gear. QA managers thus appreciate technologies that simplify the testing process so that their engineers can focus on designing effective test cases rather than wrestling with the complexities of the testing environment.

By reducing the time to automate new testing scenarios, companies can reduce the overall time-to-test, increase the number of automated tests produced, and ultimately get their product to market faster with higher quality.

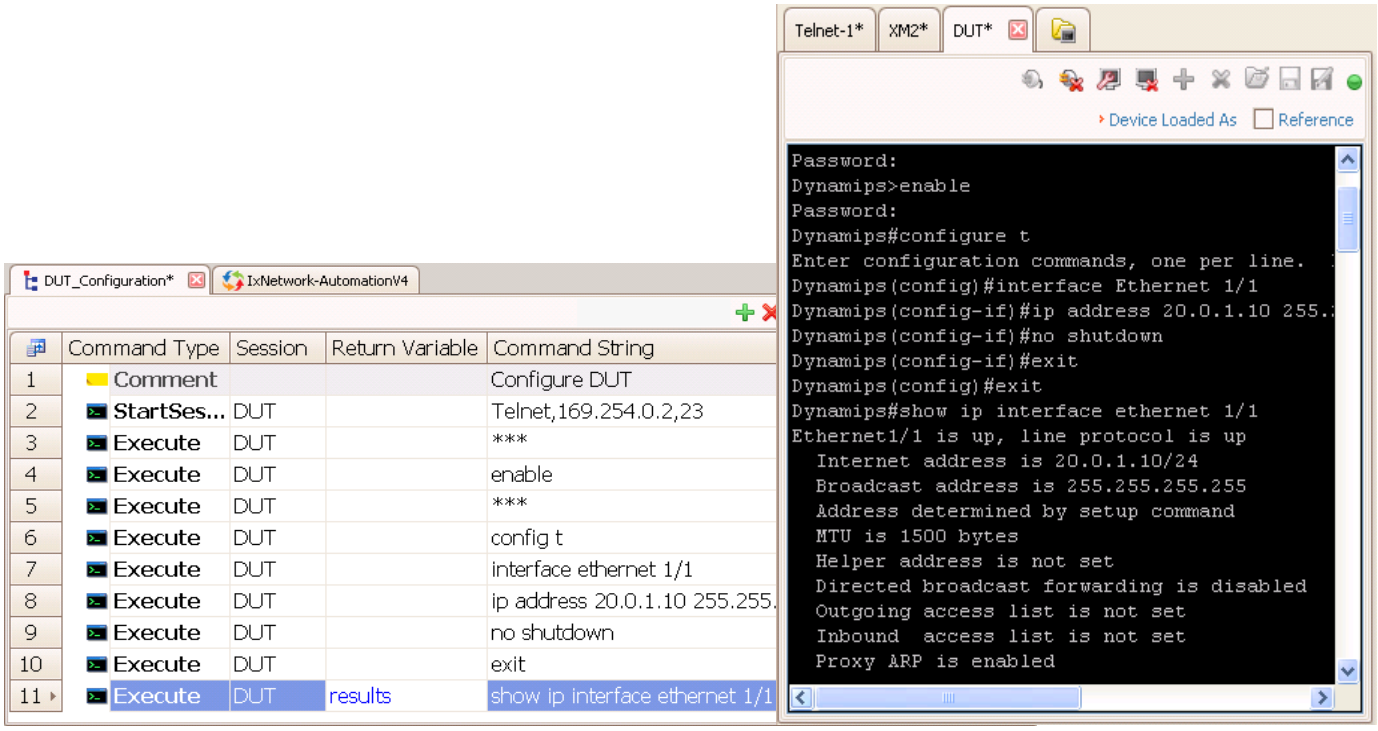

Figure 2. Capturing Live Sessions for Creating Automated Scripts

Multi-Session Capture Simplifies Test Development

Test Composer is a development environment that greatly reduces the time to develop automated tests while unifying the configuration and results collection of various test tools and DUTs. Test Composer allows the test engineer to create automated tests using multiple devices and test equipment simultaneously. Test Composer works by capturing commands and responses while test engineers interact with live sessions connected to the DUTs. Test Composer then uses the captured information to help the test engineer construct automated test scripts. These scripts can then be modified as necessary using a GUI-based editor that reflects the actual contents of the test configuration. Test engineers do not need to know the script command syntax; the syntax is constructed for them by Test Composer.

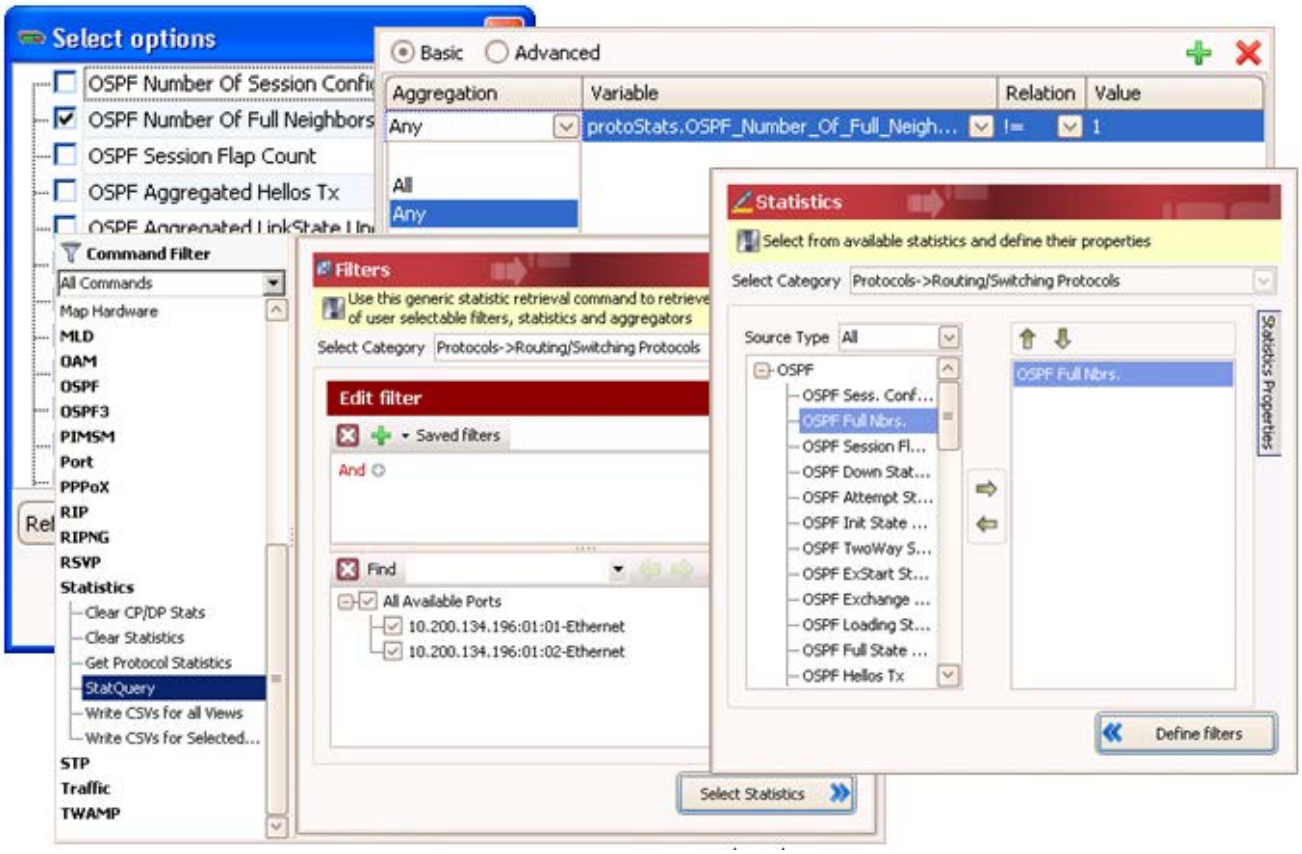

Figure 3. Command Editor

GUI-Driven Command Creation

Test Composer has an extensible command interface that allows test engineers to customize the set of available commands used to develop tests. Ixia plug-ins are guaranteed to work in Test Composer in conjunction with the other vendor plug-ins. By using various multi-vendor plug-ins, test engineers can coordinate the commands across tools and DUTs in a highly-efficient way. Test teams no longer need to invent the glue to bring these test tools together.

Test Composer has the ability to passively monitor the system log messages produced by the DUTs and take proactive steps to flag events or take corrective action as they happen. This instantaneous response to system log events during a test makes it easier to analyze and diagnose failed tests. It also makes it possible to either recover from a failure event and continue the test cycle, or to truncate the test and avoid wasting valuable test time.

Once the test has been written, Test Composer has a debugger that allows for quick verification of the validity and repeatability of the test.

Key Features

- Test IDE – provides a working environment that simplifies the test development process.
- Multi-vendor test authoring – provides a unified scripting environment that coordinates and synchronizes test steps. Test Composer tracks the commands executed in the different test tools and DUTs and replays them as a script in the desired order.
- Open architecture framework – an open architecture framework that can accept command plug-ins developed by other vendors. Plug-ins can be easily added to the framework to extend and enhance the test engineer's ability to control and script interactions with other test vendors' tools.
- Comprehensive Ixia application support – complete coverage of the Ixia portfolio of products including IxNetwork, IxLoad, IxAutomate, IxChariot, and many more. Test Composer has the broadest list of supported applications and the deepest level of command support of any product in the industry.
- Live interactive capture/replay – create a test by simply entering commands in one or more live sessions connected directly to the DUT and/or test tool, then craft the captured sequence into an automated test or procedure that can be used as part of other automated tests.
- Passive DUT monitoring – configure live sessions to passively monitor session logs and consoles for various messages, such as errors or warnings, and then take proactive steps to flag the errors and optionally recover from them. Supported sessions include SNMP and SYSLOG.
- Built-in CSV file analyzer – easily read and analyze CSV files as part of an automated test. Many test tools produce output in CSV formats. The CSV Analyzer can read these files, extract relevant data, and aggregate the data for analysis. Then use the data to automatically determine success or failure of the test.
- Shared test resources – create a variety of shared resources like procedures and device profiles that allow the test team to leverage their work among the different test cases and make those test cases more portable.
- Full–featured debugger – a debugging environment that shortens the time between creating and debugging a test. The debugger has a full set of features, including breakpoints, execution progress indicators, variable watches, and global and session logging streams for diagnosing failures within the test.
- Simplified test development process – a unique combination of capturing live interactions with the devices and test tools plus the full flexibility of a script editor and debugger.
- Customizable test report generation – a customizable test report generator automatically generates the test report directly from the data collected by the test, with built-in tables, bar charts, line graphs, captured logs, and command responses. It consolidates all the data needed to analyze the results of a test into a single report, complete with a table of contents. The report content and format can be tailored dynamically in real time.
- Packet decoder analysis tools – allows the test to read in any pcap file and analyze the contents of the captured packets using GUI-driven filtering and analysis.

Product Ordering Information

931-3201

Test Composer DUT Configuration Bundle – designed for individual developer's testing requirements for automating DUT configurations

931-3202

Test Composer Test Authoring Bundle – designed for individual developer's testing requirements complete with the full suite of plug-ins

931-3203

Test Composer Test Authoring Bundle – designed for individual developer's testing requirements including the suite of plug-ins for DUT configuration and L2-3 network testing

931-3204

Test Composer Test Authoring Bundle – designed for individual developer's testing requirements including the suite of plug-ins for DUT configuration and L4-7 application testing

This material is for informational purposes only and subject to change without notice. It describes Ixia's present plans to develop and make available to its customers certain products, features, and functionality. Ixia is only obligated to provide those deliverables specifically included in a written agreement between Ixia and the customer.