VersaTest

Cross-Platform Server-Based Testing

VersaTest Classic
Product Introduction Guide

Ascert

taking systems to the edge

For further information:

Ascert, LLC
759 Bridgeway
Sausalito, CA 94965-2842

Telephone: (415) 339-8500
Email: info@ascert.com
Web: http://www.ascert.com

Table of Contents

Chapter One

INTRODUCTION

About this document

This document provides a high-level introduction to VersaTest Classic. It details the members of the product family, and demonstrates their flexibility by showing some of the varied ways they can be used.

We use the term *VersaTest* throughout this document. However with the expansion of the product line in 2008 with the release of the VersaTest Automator and VersaTest Simulator products, the more correct term for the products discussed in this guide should be *VersaTest Classic*. VersaTest Automator and VersaTest Simulator use VersaTest Automation Server (VersaTest/AS) to leverage the power of VersaTest Classic to deliver a standardized, turnkey, integrated, end-to-end automated testing solution for complex environments. Discussion of VersaTest Automator and VersaTest Simulator is not covered any further in this guide.

Intended Audience

This is an introductory document, aimed towards a more non-technical audience. It is suitable for people looking to get a “big-picture” of the product, such as new-users of the product, line managers or other staff looking to evaluate the usefulness of the product in an organization.

Company Background

Ascert was founded in 1992 in San Francisco, California as ***SoftSell Business Systems, Inc***. SoftSell focused exclusively on providing specialized testing software and services to the Tandem NonStop market. With our name change to ***Ascert*** in 2003, came a change in focus. Leveraging the success we enjoyed in the Tandem market, we expanded our product line to offer a full suite of testing products for all server platforms.

We build ‘best-in-class’ software used by many of the world's biggest and most well-known companies. We build automated software testing solutions that help companies measure application performance, reliability and scalability. Our off-the-shelf simulators include solutions for EFT testing, POS testing, ATM testing, IFX testing, EMV / chip card testing, ISO8583 testing and 3270 / 6530 terminal testing. Our custom simulators have been used to test a multitude of environments, including biometric systems and air traffic control workstations.

Our products help testing professionals better manage their procedures and environments through an end to end tool set. Additionally, our professional consulting services augment our customers’ testing teams by helping to implement, teach and customize our software for their unique needs.

VersaTest Background

VersaTest was initially developed in Australia in the early 1990’s under the name ***VPRO***, and was designed solely for the testing of BASE24 applications. BASE24, which is an integrated electronic funds processing and switching application from ACI Worldwide, ran at that time on Tandem NonStop

Computers, and is used extensively for the processing of ATM and POS (Point Of Sale) transactions. Back at that time, Tandem Computers provided the “gold-standard” in server availability, and a suitable testing tool was needed to make sure that the application reliability matched that of the underlying server. VPRO was such a tool.

The computer industry has undergone many changes since then. While Tandem Computers is now part of HP, where the machines are sold under the Integrity NonStop brand, the reliability of commodity servers has increased to the point where they are routinely used for the processing of high-value transactions, that at one time were the sole domain of Tandem Computers.

As the computer industry has evolved over time, so too has the VPRO product. Firstly it was generalized to enable it to be used to test any Tandem-based application, rather than just BASE24; the name of the product was changed to ***VPRO-G*** to symbolize this generalization. The next major evolution came after the product was renamed to **VersaTest**, when VersaTest was generalized still further to enable it to run on any Java-enabled platform, as exists on most commodity servers and operating systems today. As evidenced by its historical roadmap, VersaTest’s evolution can be said to track the evolution of the overall computer industry, allowing the same best-of-breed testing available on the early proprietary systems to be applied to the wider market as it exists today.

VersaTest – The Tagline

VersaTest is a *“message-based testing and simulation environment for mission-critical servers”*. What we mean by that is as follows:

- **Message-based**
  VersaTest is typically used to simulate intelligent devices and applications to enable the testing of other applications to which such entities are attached. These entities include: POS devices; ATMs; biometric readers; host interfaces; credit card processor interfaces; and RADAR displays. Essentially anything that exchanges formatted messages that can be described using VersaTest’s Data Definition Language, across a communications link or middleware supported by VersaTest, can be simulated by VersaTest. By way of contrast, most testing tools available on the market today are geared to testing Web and GUI applications, which typically have an accessible user interface that allows easy manual or “robot-driven” testing.

- **Testing and Simulation Environment**
  VersaTest provides an environment that facilitates the testing of systems and applications by simulating the entities with which these systems interface. To us, *simulating* and *testing* are two different things, which though normally performed together, can actually be separated. In fact we’ll show in this document a real-world example of how VersaTest has been used in an embedded system to simulate a back-end host application during the pilot phase of a new product rollout. Our environment provides an IDE to support the development of new simulators where off-the-shelf simulators are not available. These simulators can be used to perform all manners of testing activities from simple functional testing of single devices at low transaction rates through to performance testing, where VersaTest can simulate many thousands of devices at hundreds of thousands of transactions per second. These testing activities can be carried out and controlled interactively in a standalone manner directly from VersaTest’s own IDE, or embedded within 3rd part test managers, such as HP’s Mercury TestDirector, which is part of HP Quality Center.

- **Mission-critical Servers**
  Going back to our roots, when VersaTest, or VPRO as it was known back then, was initially developed, it goes to the value our customers place on the systems with which we get involved. These systems tend to be server systems core to the business of our customers, where even minor outages represent significant dollars in lost revenue and other unwanted exposure. Who wants to be front page news across the internet when their network is down and they are unable to process customer payments? These customers want the best testing tools available, and we believe that VersaTest falls into that category.

Chapter Two

VERSATEST IN THE APPLICATION LIFE-CYCLE

Introduction

VersaTest is a sophisticated message processing facility specifically designed to simulate and test transactional interfaces and applications.

VersaTest gives the user the ability to rapidly examine and manipulate message traffic in virtually any way, from the passive monitoring of messages flowing across an interface, to the active modification of the messages while in-flight, and then to the complete simulation of one half of the interface. In the latter case, for instance, the user can describe message formats within VersaTest, define how VersaTest is to identify particular message types and delineate which actions are to be performed at specific times or events.

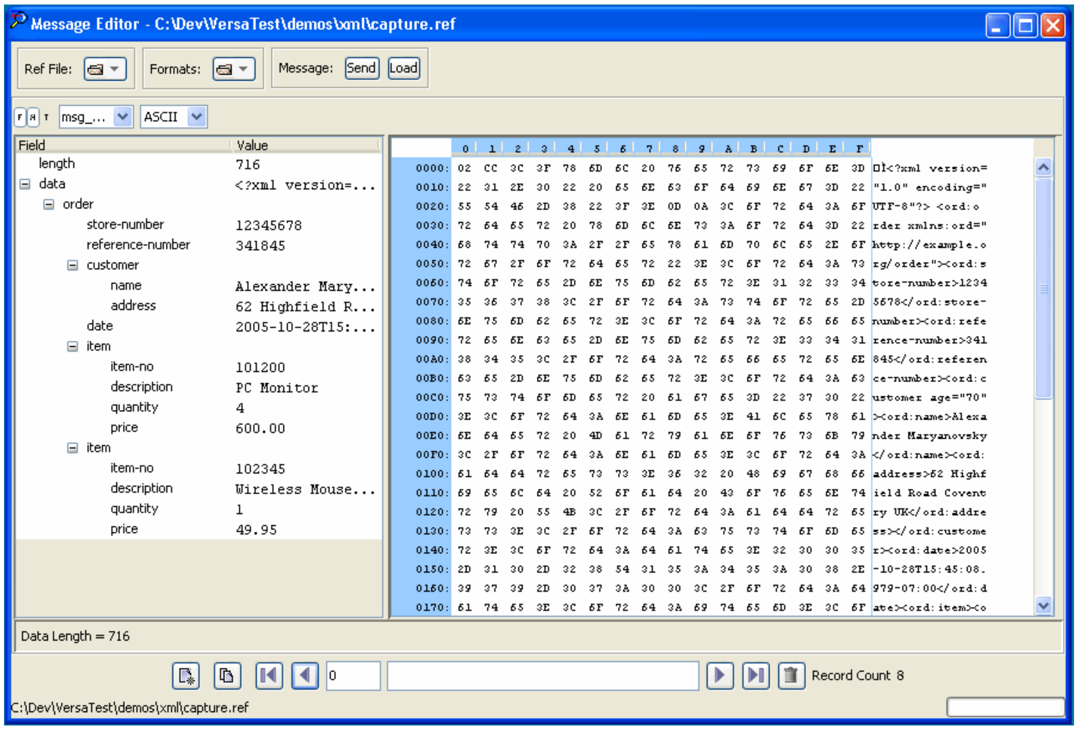
VersaTest has the power and the features to support an application throughout its entire life cycle, beginning with message and process prototyping, development, testing and simulation, and then further, through the stress testing, capacity planning, quality assurance, and monitoring and support stages. This type of tool gives an organization a standardized approach with which to support its applications at all phases of the software development life cycle. Not only does this save considerable time and money in cross-training between departments, it can also reduce the cost and complexity of maintaining multiple tools and procedures.

The remainder of this chapter provides an overview of how VersaTest can be used in various stages of an application’s life cycle.

VersaTest in Development and Unit Testing

VersaTest's versatility and quick set-up is highly valued in the development and testing environment where the requirements of simulators and support programs change frequently.

For example, when developing software for a new interface, VersaTest can be used to simulate the remote side of the interface. VersaTest is able to autonomously generate messages or send messages in response to received messages – or even both at the same time! The user can interactively create messages using VersaTest’s graphical message editor, so enabling ad-hoc / informal testing. These messages can be saved, or recreated programmatically, and combined later in a script to create a more robust / formal simulation and testing environment, to any desired level of complexity.



```
Field                         Value
length                        716
⊟ data                        <?xml version=...
  ⊟ order
      store-number            12345678
      reference-number        341845
    ⊟ customer
        name                  Alexander Mary...
        address               62 Highfield R...
      date                    2005-10-28T15:...
    ⊟ item
        item-no               101200
        description           PC Monitor
        quantity              4
        price                 600.00
    ⊟ item
        item-no               102345
        description           Wireless Mouse...
        quantity              1
        price                 49.95
```

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000: 02 CC 3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D  □Ì<?xml version=
0010: 22 31 2E 30 22 20 65 6E 63 6F 64 69 6E 67 3D 22  "1.0" encoding="
0020: 55 54 46 2D 38 22 3F 3E 0D 0A 3C 6F 72 64 3A 6F  UTF-8"?> <ord:o
0030: 72 64 65 72 20 78 6D 6C 6E 73 3A 6F 72 64 3D 22  rder xmlns:ord="
0040: 68 74 74 70 3A 2F 2F 65 78 61 6D 70 6C 65 2E 6F  http://example.o
0050: 72 67 2F 6F 72 64 65 72 22 3E 3C 6F 72 64 3A 73  rg/order"><ord:s
0060: 74 6F 72 65 2D 6E 75 6D 62 65 72 3E 31 32 33 34  tore-number>1234
0070: 35 36 37 38 3C 2F 6F 72 64 3A 73 74 6F 72 65 2D  5678</ord:store-
0080: 6E 75 6D 62 65 72 3E 3C 6F 72 64 3A 72 65 66 65  number><ord:refe
0090: 72 65 6E 63 65 2D 6E 75 6D 62 65 72 3E 33 34 31  rence-number>341
00A0: 38 34 35 3C 2F 6F 72 64 3A 72 65 66 65 72 65 6E  845</ord:referen
00B0: 63 65 2D 6E 75 6D 62 65 72 3E 3C 6F 72 64 3A 63  ce-number><ord:c
00C0: 75 73 74 6F 6D 65 72 20 61 67 65 3D 22 37 30 22  ustomer age="70"
00D0: 3E 3C 6F 72 64 3A 6E 61 6D 65 3E 41 6C 65 78 61  ><ord:name>Alexa
00E0: 6E 64 65 72 20 4D 61 72 79 61 6E 6F 76 73 6B 79  nder Maryanovsky
00F0: 3C 2F 6F 72 64 3A 6E 61 6D 65 3E 3C 6F 72 64 3A  </ord:name><ord:
0100: 61 64 64 72 65 73 73 3E 36 32 20 48 69 67 68 66  address>62 Highf
0110: 69 65 6C 64 20 52 6F 61 64 20 43 6F 76 65 6E 74  ield Road Covent
0120: 72 79 20 55 4B 3C 2F 6F 72 64 3A 61 64 64 72 65  ry UK</ord:addre
0130: 73 73 3E 3C 2F 6F 72 64 3A 63 75 73 74 6F 6D 65  ss></ord:custome
0140: 72 3E 3C 6F 72 64 3A 64 61 74 65 3E 32 30 30 35  r><ord:date>2005
0150: 2D 31 30 2D 32 38 54 31 35 3A 34 35 3A 30 38 2E  -10-28T15:45:08.
0160: 39 37 39 2D 30 37 3A 30 30 3C 2F 6F 72 64 3A 64  979-07:00</ord:d
0170: 61 74 65 3E 3C 6F 72 64 3A 69 74 65 6D 3E 3C 6F  ate><ord:item><o
```

Data Length = 716

0

C:\Dev\VersaTest\demos\xml\capture.ref

VersaTest Message Editor

The flexibility and high-level nature of VersaTest's scripting language also enables it to be used for prototyping purposes. For example, a telecommunications carrier used VersaTest to perform protocol conversion between X.25 and TCP/IP in order to demonstrate a new service. The script to achieve this conversion was extremely simple and written in a matter of minutes.

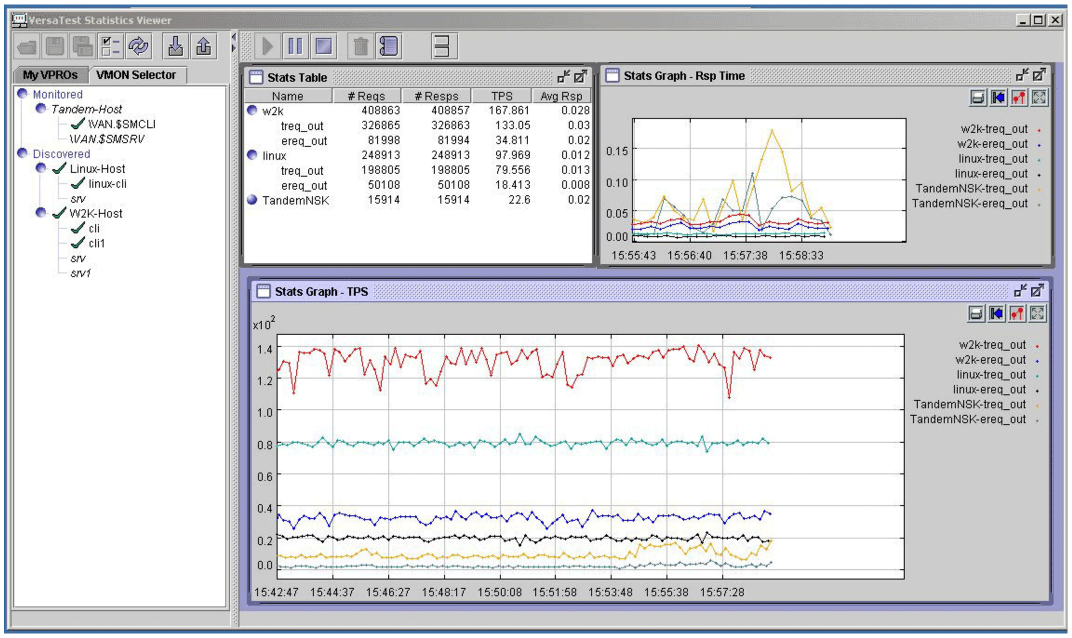
VersaTest in Performance and Stress Testing

Using VersaTest for performance testing can reveal important information about an application and its ability to cope under stress. It offers a variety of features which can help uncover performance problems at the development stage, rather than at the production stage.

One such feature is VersaTest's ability to allow users to write a precise test plan to accurately simulate an entire day's activity including idle periods, linear and exponential ramping, and peak periods. Another feature of VersaTest is that it allows users to interactively take control of a stress test. For instance, if the user notices a problem occurring within the test application but needs to maintain the current load factors in order to discover the cause, he is able to override the test plan and take control, rather than having to repeat the test.

The performance that VersaTest is able to achieve is limited only by available hardware and communications infrastructure. VersaTest can be deployed on multiple platforms in a number of different topologies and tests can be scaled to any desired size.

Performance information gathered during a test is displayed graphically in real-time, and can be saved for future analysis and retrospective 'real-time' replay.



Name	# Reqs	# Resps	TPS	Avg Rsp
w2k	408863	408857	167.861	0.028
treq_out	326865	326863	133.05	0.03
ereq_out	81998	81994	34.811	0.02
linux	248913	248913	97.969	0.012
treq_out	198805	198805	79.556	0.013
ereq_out	50108	50108	18.413	0.008
TandemNSK	15914	15914	22.6	0.02

VersaTest Statistics Viewer

VersaTest in Software Maintenance and Regression Testing

Software maintenance and regression testing are recognized as important steps in an application's life cycle. Typically this involves maintaining and replaying an exhaustive set of transactions created specifically to thoroughly exercise an application.

The next step after such a replay is to verify that the application did, in fact, behave as it should have; this includes comparing the resulting message flow with a previously captured message flow known to be correct.

Until now, this message comparison phase had usually been performed manually, and as a manual procedure, it was very costly, time-consuming and error-prone. With VersaTest, however, there is a facility for automatically comparing these ‘before and after’ message flows, thus making the regression test phase much shorter and more accurate than it had been in the past.

VersaTest contains many additional features which help to simplify this stage of an application's development, all of which work together to provide a more manageable and reliable solution and to save considerable time and resources.

VersaTest for Certification Testing

We live in a world that is getting progressively more interconnected, but in more and more diverse ways. In order to manage the process of connecting to our partners we must often certify that our systems meet a required set of specifications. The testing required to achieve this is in some ways similar to that used for regression testing, but comparison of a previous test run is inappropriate because of the non-deterministic nature of the testing. So instead of comparing one set of transactions against another set, more formality and feedback is given to the process. Formality meaning that a successful test run must be audited and retained for inspection. The feedback given during the testing process should indicate more than that a transaction has failed, but why it has failed; this requires more intelligence to be built into test scripts to enable more granular tests to be performed on individual fields of the messages.

The VersaTest Automation Server (VTAS), which is an add-on to VersaTest, provides this capability by separating out the simulation of a device from the test cases that use the simulation. This enables structured testing of message flows that provide detailed feedback on whether or not the flow meets its specifications.

VersaTest as a Technical Support Aid

VersaTest’s ability to manipulate messages in real-time also allows it to be used as an intelligent line monitor, actively passing messages between destinations while optionally saving a copy to disk for later examination or replay. For example, if a problem is noticed in a live application, the exact sequence of messages can be captured in that live application and then replayed in the development environment, thus enabling faster problem resolution.

VersaTest's ability to remain dormant and then become active at a particular time or event simplifies the matter of capturing the sequence of events causing a problem.

VersaTest also offers many powerful message display and formatting options. One such feature is VersaTest’s ability to identify messages 'in flight' and apply a DDL format (Data Definition Language). The result is that technical support staff can instantly see the messages converted for display in a readable form, labeled with field names and offsets. This saves valuable time, allowing them to concentrate on the problem at hand.

Additionally, the user is also able to apply these formatting options later when examining a file of captured messages using the built-in graphical message file editor.

Finally, it is possible to import into VersaTest, messages obtained from other sources. For example, BASE24 audit files or files from a communications line trace can be imported into VersaTest either for analysis and reporting, or in order to recreate a message stream to replay to a system under test.

Chapter Three

VERSATEST ENVIRONMENTS

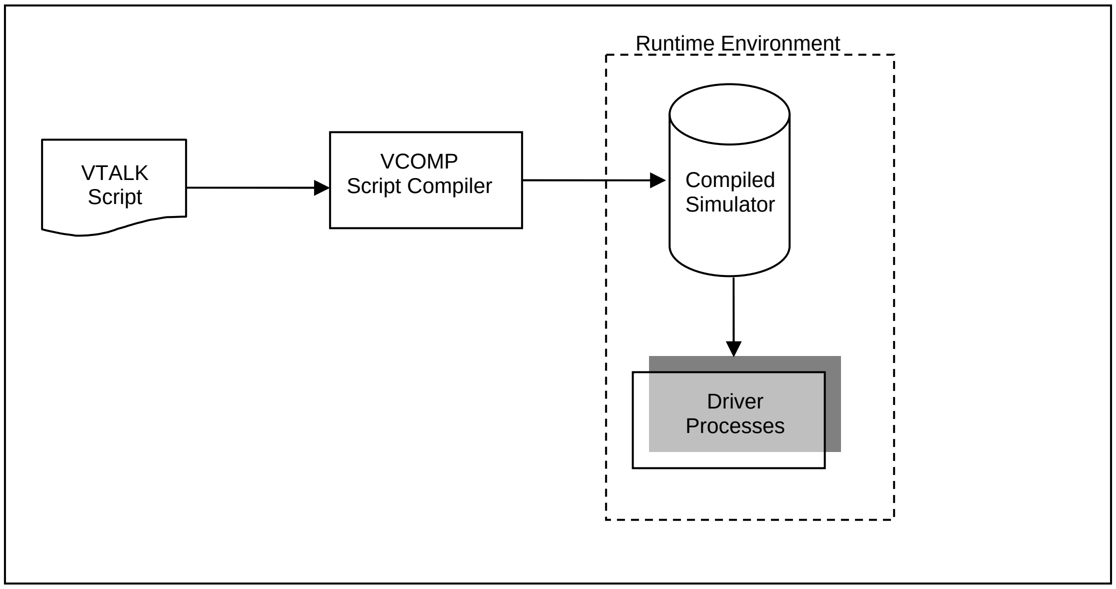
Introduction

VersaTest has evolved from a closed, point-solution for the testing of a single high-value application on an extremely high-end proprietary computing platform, to a generalized, distributed testing and simulation environment running on commodity platforms that is capable of being integrated into third party enterprise test management products.

This evolution has resulted in a great deal of flexibility in the utilization of VersaTest. This chapter describes the major components of VersaTest and shows various environments in which VersaTest could be used, so as to help the reader understand the scope of solutions possible when deploying VersaTest. These are meant to be illustrative, rather than definitive; there are many combinations possible for installation of VersaTest beyond what are shown here.

VersaTest Simulators

The normal use of VersaTest is to perform some testing or simulation activities. In order to do this, a VersaTest Simulator must first be created. The simulator contains the instructions required for VersaTest's runtime environment to process messages to and from the system under test – for example "when I receive a balance enquiry, I return a balance response". These instructions are written in an optimized scripting language called VTALK, and then compiled for use by driver processes in VersaTest's runtime environment, thereby performing the *simulation*.



The VTALK scripting language is open and accessible to all licensed VersaTest users. It was developed by Ascert to simplify the construction of simulators that contain otherwise complex technologies and techniques. For example, VTALK contains language constructs that provide easy standardized access to commonly-found technologies including the following:

- **Message formats**
  XML; ASN.1; ISO8583
- **Communication protocols**
  TCP/IP; X.25; MQ Series; XPNET; UDP; SNA; OSI/TS; TLAM; Bisync
- **Security**
  DES; 3DES; RSA; DUKPT; EMV; LRC; MAC (X9.19 & ISO9797-1); MOD10/LUHN

Users with knowledge of other scripting or programming languages should have no difficulty learning the VTALK language and creating and/or maintaining simulators using VersaTest's Integrated Development Environment (IDE).

Ascert also maintains a library of pre-built simulators which require little or no changes prior to running.

VersaTest Script Library

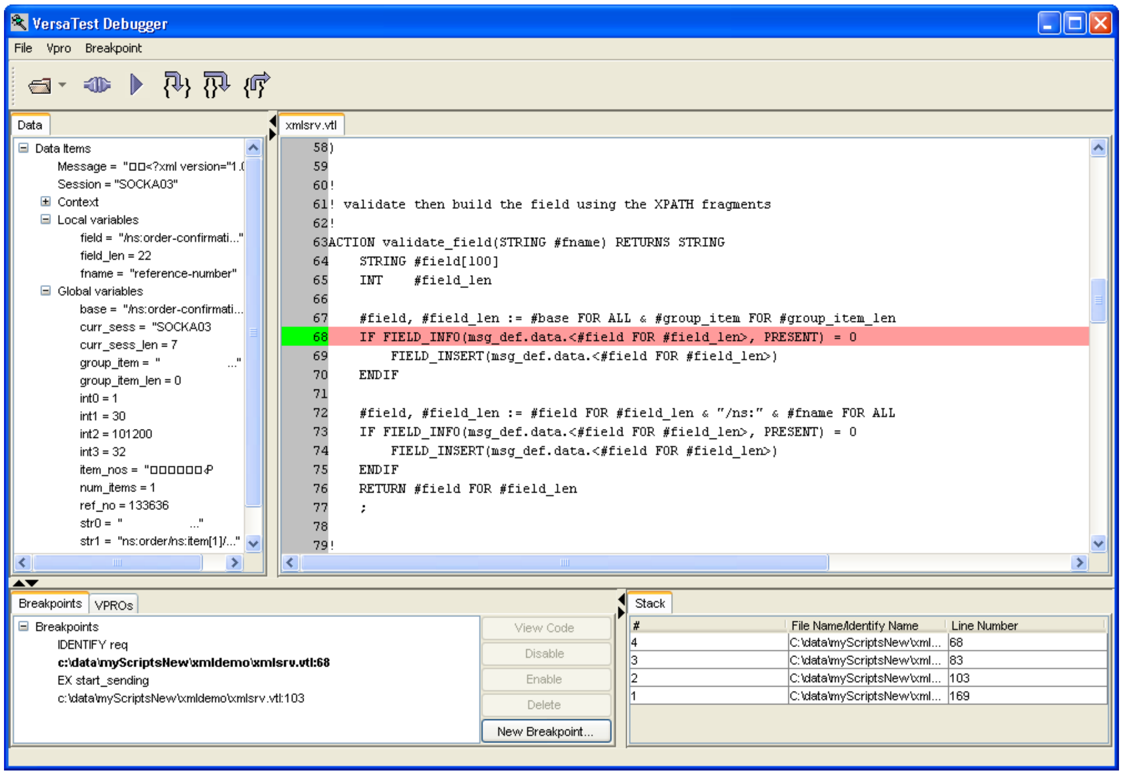
Ascert maintains a library of simulators for commonly-found devices and interchanges. Subscribing to this library provides customers more of a turnkey testing experience, to avoid having to maintain their own simulators and learn the semantics of VTALK.

All the simulators in the library use a standardized operational approach, are delivered pre-compiled, and separate out the device-specific simulation characteristics from the more generalized test-centric functions. This allows the customer to create their own high-level test plans using a common data-driven approach across all of their simulators. Ascert also provides its own automation layer, VersaTest Automation Server (VTAS), to control these common simulators, enabling, for example, the test plans to be created in HP's Mercury TestDirector. This will be discussed in more detail, later in this chapter.

VersaTest Integrated Development Environment

VersaTest provides the ability for users to create their own simulators, or customize template simulator scripts provided by Ascert and other third party suppliers. This ability makes customers self-reliant, and avoids any vendor lock-in when maintaining their simulators.

VersaTest's VWIN provides an IDE to support the development of new simulators. A description of the IDE is beyond the scope of this document, but it includes modern features such as color-coded syntax highlighting and a debugger that provides local and remote debugging.



```
Data Items
    Message = "□□<?xml version="1.(
    Session = "SOCKA03"
  Context
  Local variables
      field = "/ns:order-confirmati..."
      field_len = 22
      fname = "reference-number"
  Global variables
      base = "/ns:order-confirmati...
      curr_sess = "SOCKA03
      curr_sess_len = 7
      group_item = "                ..."
      group_item_len = 0
      int0 = 1
      int1 = 30
      int2 = 101200
      int3 = 32
      item_nos = "□□□□□□₽
      num_items = 1
      ref_no = 133636
      str0 = "                ..."
      str1 = "ns:order/ns:item[1]/..."
```

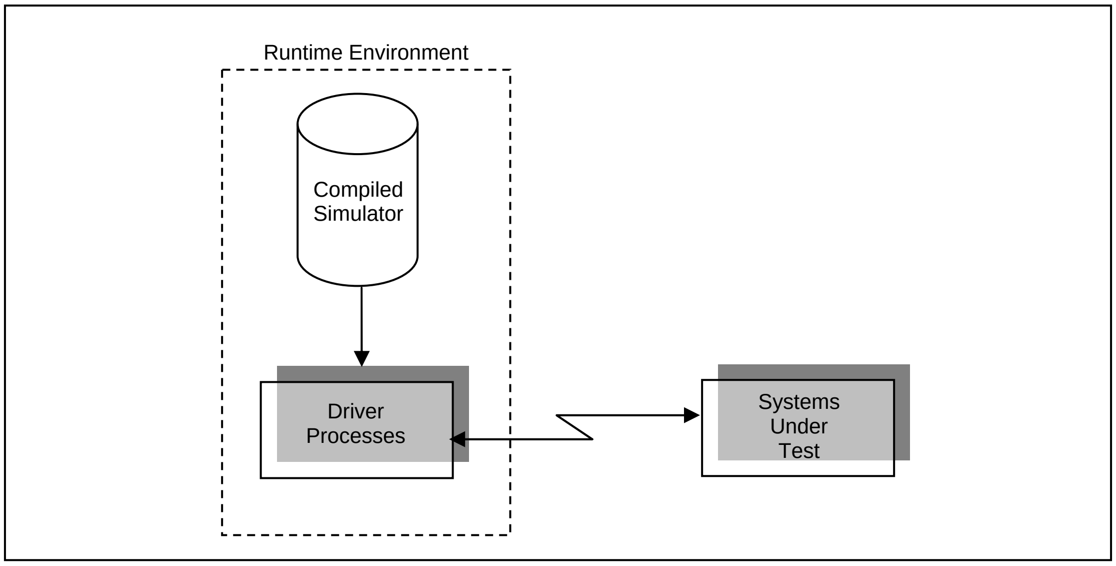
```
58)
59
60!
61! validate then build the field using the XPATH fragments
62!
63ACTION validate_field(STRING #fname) RETURNS STRING
64    STRING #field[100]
65    INT    #field_len
66
67    #field, #field_len := #base FOR ALL & #group_item FOR #group_item_len
68    IF FIELD_INFO(msg_def.data.<#field FOR #field_len>, PRESENT) = 0
69        FIELD_INSERT(msg_def.data.<#field FOR #field_len>)
70    ENDIF
71
72    #field, #field_len := #field FOR #field_len & "/ns:" & #fname FOR ALL
73    IF FIELD_INFO(msg_def.data.<#field FOR #field_len>, PRESENT) = 0
74        FIELD_INSERT(msg_def.data.<#field FOR #field_len>)
75    ENDIF
76    RETURN #field FOR #field_len
77    ;
78
79!
```

```
Breakpoints
    IDENTIFY req
    c:\data\myScriptsNew\xmldemo\xmlsrv.vtl:68
    EX start_sending
    c:\data\myScriptsNew\xmldemo\xmlsrv.vtl:103
```

#	File Name/Identify Name	Line Number
4	C:\data\myScriptsNew\xml...	68
3	C:\data\myScriptsNew\xml...	83
2	C:\data\myScriptsNew\xml...	103
1	C:\data\myScriptsNew\xml...	169

Standalone Testing & Simulation Environment

Once a simulator module has been created or downloaded, then it can be deployed for use within a VersaTest installation. The simplest form that an installation can take is as a single stand-alone, self-contained simulator.

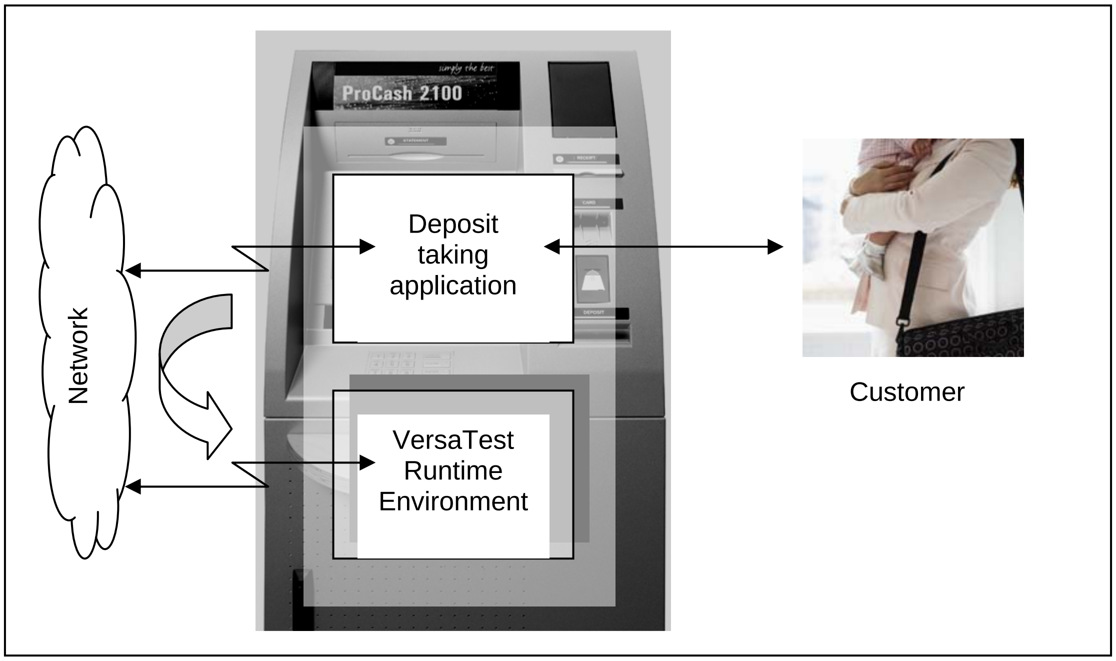


In this scenario, there is no operator control of the environment. The environment is started, perhaps automatically when a computer is booted, and runs indefinitely from that point on. This could be characterized as “set and forget” operation.

The only VersaTest components required in this scenario are the core infrastructure components, of which the most important is VPRO, whose message processing capabilities perform the actual simulation. Taken together these core infrastructure components provide VersaTest with its runtime environment.

This type of installation would normally be used when VersaTest is simulating some type of responder, such as a payment authorization system. An institution may use such a stand-alone simulator as a substitute for a real internal or external system, knowing that the simulator is always available and less resource intensive. They may also provide such a stand-alone system to their partners to enable their partners to test their own systems prior to being allowed to connect to a real application.

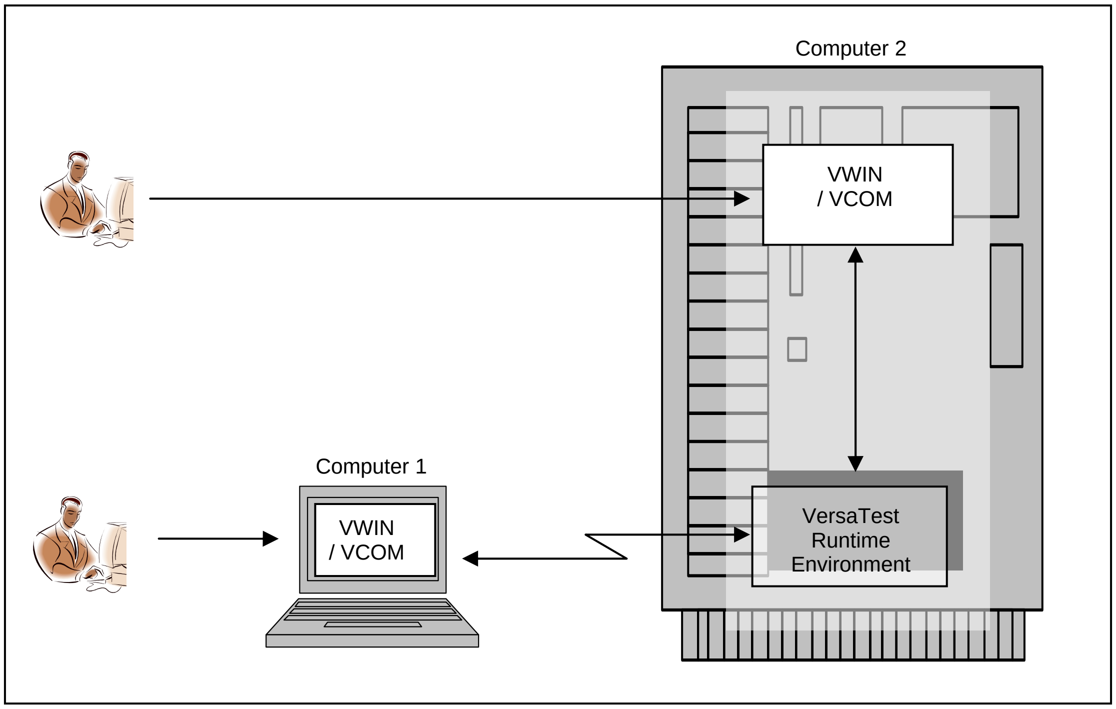
Because VersaTest can be run on any Java-enabled platform, it is also possible to use VersaTest in this same way on embedded devices. This approach was taken by a European bank who wanted to pilot a new service within their electronic deposit machines. However the bank did not want to incur the cost of their software provider writing new code for the back-end host and the risk of promoting that code to production. The solution devised by the bank and its ATM vendor was to run VersaTest on the electronic deposit machine on the machine's own OS. VersaTest then simulated the back-end host to the deposit taking service, also running on the machine.



This pilot was so successful, that it was decided to use VersaTest for the full production roll-out, with the solution now being used at several European banks.

Interactive Testing & Simulation Environment

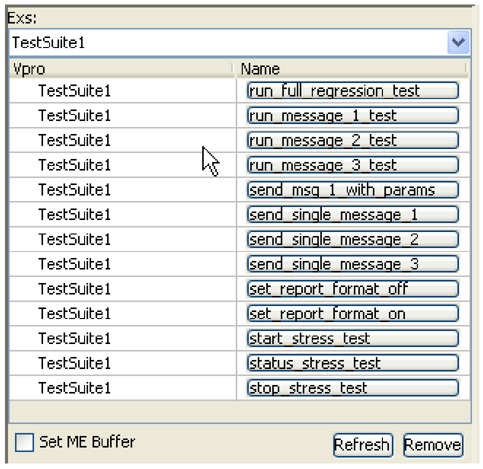
Standalone environments discussed in the previous section run, by definition, completely unattended. Often though, some level of operator control is needed over the environment. With VersaTest this is achieved using either our graphical client interface (VWIN) or our command-line interface (VCOM). These can either be run on the same system as the runtime environment, or from a remote system using a TCP/IP connection.



The same VPRO can be simultaneously monitored and controlled by multiple operators – changes made by one operator are visible to all operators.

The type of operator control exerted over the environment will vary considerably based on the purpose of the script and the manner in which it was developed, since the script-writer is able to create functions that are published to the user interface. Examples of what these '*EX*' functions could do are as follows:

- Start a stress test that automatically generates transactions at a given rate.
- Alter the rate of transaction generation for the previous stress test.
- Change the percentage of transactions to be authorized in a responder script.
- Reset / display the running value of transactions processed.
- Initiate the processing of a set of test cases defined in an external spreadsheet; this is known as *data-driven testing*, and is one method of abstracting the simulation from the test cases that drive the simulation.
- Take the data entered by the user in VWIN's graphical message editor, and use it to populate a message to be sent to the system under test.



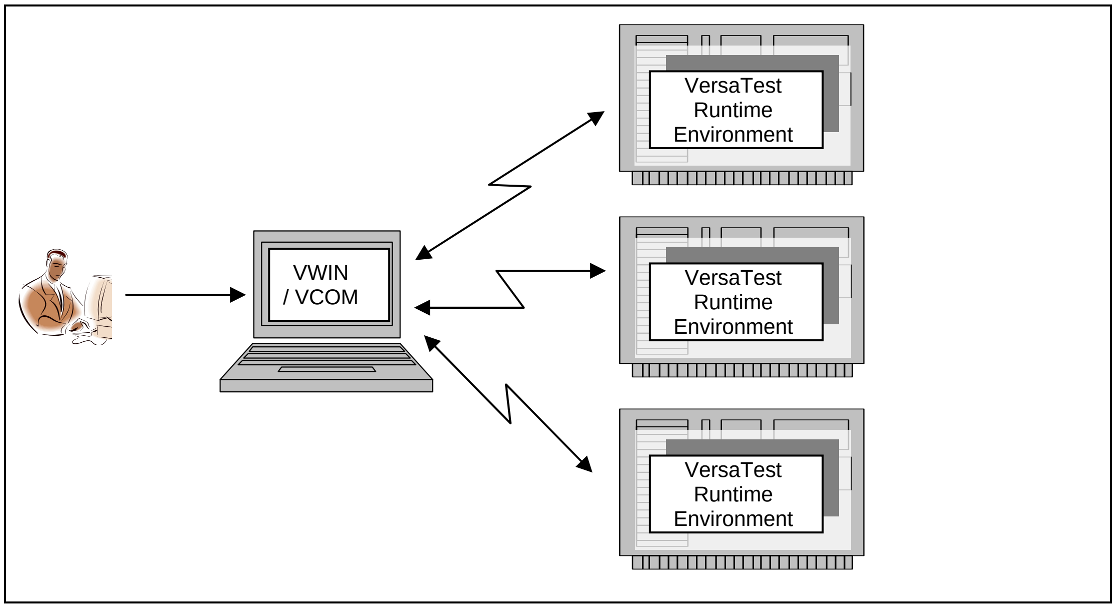
Vpro	Name
TestSuite1	run_full_regression_test
TestSuite1	run_message_1_test
TestSuite1	run_message_2_test
TestSuite1	run_message_3_test
TestSuite1	send_msg_1_with_params
TestSuite1	send_single_message_1
TestSuite1	send_single_message_2
TestSuite1	send_single_message_3
TestSuite1	set_report_format_off
TestSuite1	set_report_format_on
TestSuite1	start_stress_test
TestSuite1	status_stress_test
TestSuite1	stop_stress_test

Selecting EX functions from VWIN

Distributed Test Environment

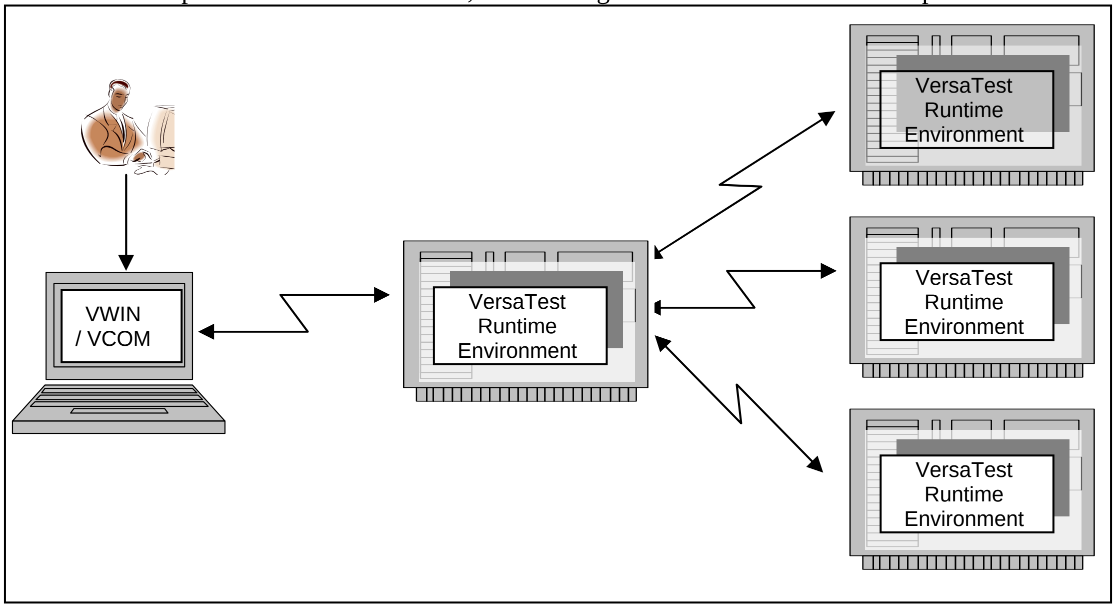
The previous section showed how a simulator could be singly controlled. It is also possible for larger tests to be created in arbitrarily complex manners.

The simplest way of doing this requires no changes to the script, and is achieved by simply replicating the VPRO processes, perhaps onto completely separate platforms.


VersaTest Runtime Environment
VWIN / VCOM
VersaTest Runtime Environment
VersaTest Runtime Environment

VWIN can simultaneously control multiple VPROs, so any of the commands mentioned in the previous section could be sent to any or all VPROs desired by the operator. In this manner, all of the VPRO processes can be instructed to do the same thing, though they are all effectively operating independently.

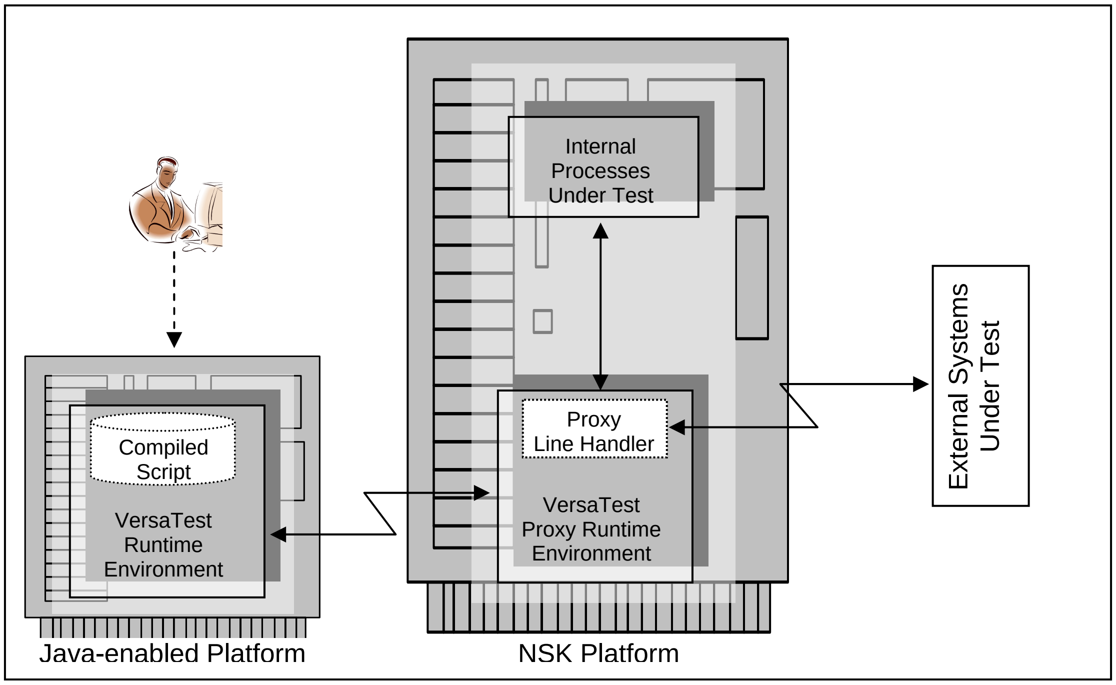
When tighter coupling is required between the simulator processes, then it is possible to create *master-slave* relationships between the simulators, so enabling one VPRO to control multiple other VPROs.


VersaTest Runtime Environment
VWIN / VCOM
VersaTest Runtime Environment
VersaTest Runtime Environment
VersaTest Runtime Environment

Now the operator controls the master VPROs, which are then responsible for delegating actions to the slave VPROs. This requires explicit scripting to achieve, but it does allow for the exchange of information and coordination between the individual simulators.

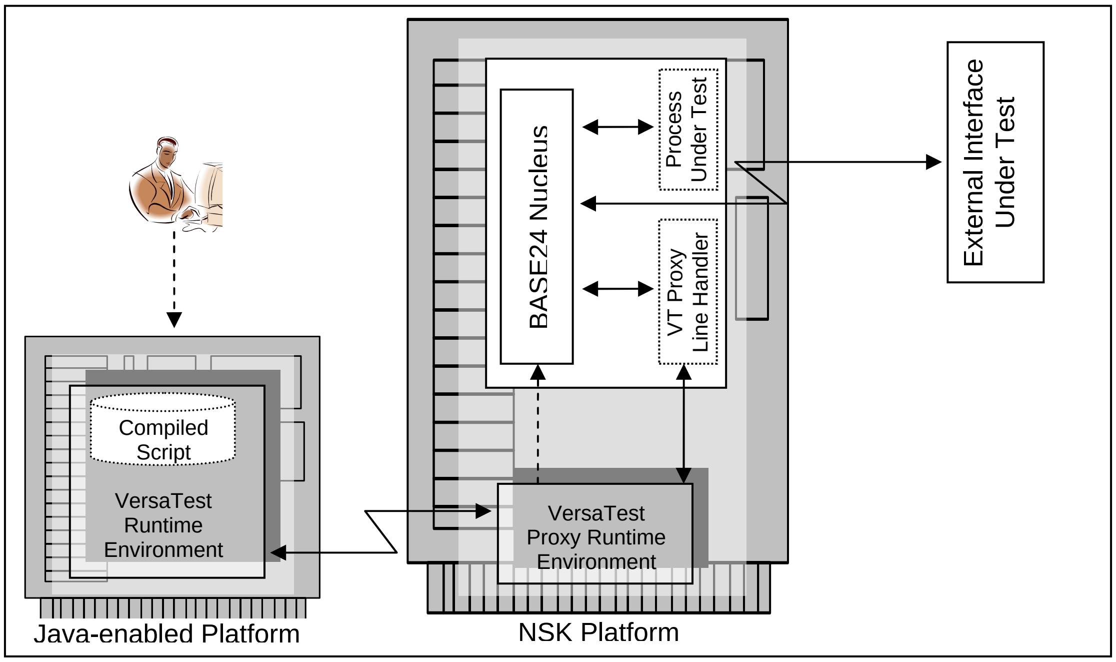
Proxy Test Environment – Cross-platform Testing

Separate VersaTest runtime environments exist for HP NonStop Servers, and Java-enabled platforms. The runtime environment for Java-enabled platforms is the more sophisticated of the two, with a richer scripting language, including support for XML, among other things, and an Integrated Development Environment. Scripts developed for NonStop Servers will also run independently on the Java-enabled platforms, as long as the Java-enabled platform supports the required communications protocol. If the Java-enabled platform does not support these *legacy* protocols, or the Java-enabled platform does not have direct connectivity to the system under test (e.g. it's behind a firewall), then the VersaTest Proxy Line Handler (PLH) can be used.


Internal Processes Under Test
Proxy Line Handler
VersaTest Proxy Runtime Environment
External Systems Under Test
Compiled Script
VersaTest Runtime Environment
Java-enabled Platform
NSK Platform

The result is that the VersaTest scripts are run on the Java-enabled platforms, with full support for the IDE and all other enhanced features, but the simulator script uses the legacy communications facilities of the NonStop Server. Currently the PLH supports $RECEIVE, PATHSEND, XPNET, TCP/IP, X.25 and Websphere MQ, with other protocols being provided as demand necessitates.

The implementation of the PLH also improves the VersaTest user experience in other ways, because it can automatically reconfigure the NonStop Server to match the communication / environmental requirements of the script. For example, if the script is to use XPNET for its communications, then VersaTest can automatically configure itself as a new satellite process within a BASE24 environment.



This enables direct communication to the BASE24 nucleus from test scripts running on completely different platforms.

Automated Testing & Simulation Environment

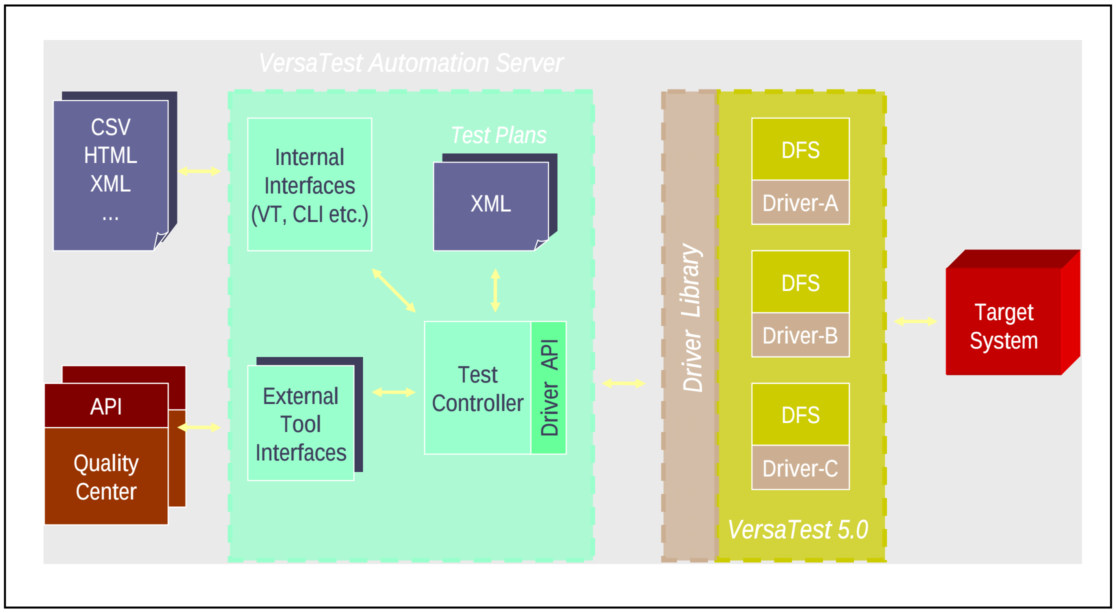
Isn't VersaTest an automated testing and simulation tool? Can you ever have too much automation?!

The environments described thus far have shown VersaTest being used in either a stand-alone manner or an interactive manner. In either case, VersaTest can automatically process messages and/or test data, however the automation performed is specified within VersaTest's script. It is also possible to externally control the operation of VersaTest through a separate automation layer.

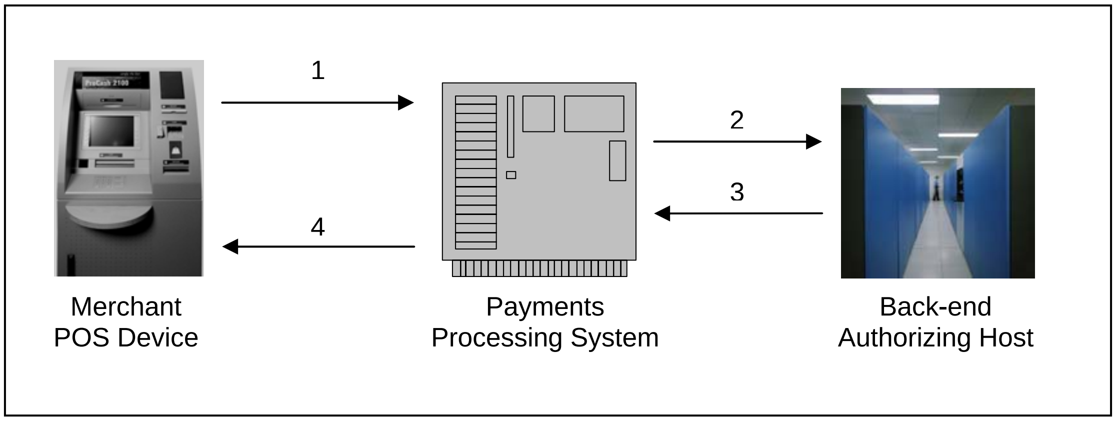
Ascert's test control layer is called *VersaTest Automation Server* (*VTAS*). It provides many benefits over the interactive control of VersaTest:

- Separates out the high-level test cases from the low-level device simulation, improving ease of test case maintenance.
- Expands the scope of a test – test cases can span multiple devices. For example, a test could be created that injects a message to the front-end of an application, and verifies that a matching message is received across an interface on the back-end.
- Enables the test cases to be embedded in third party Test Managers, such as HP's TestDirector, which is part of the Quality Center Suite. This hides the VersaTest simulator within familiar enterprise-wide testing products, thereby reducing the learning curve when including VersaTest in a test environment.
- Enables multiple testing tools to be used as part of a single integrated test.
- Closes the gap between test specification and test execution

VTAS uses drivers from VersaTest's driver library, which provide a standardized approach to test automation. The VTAS environment is as follows:



VTAS provides a simplified, high-level declarative language for specifying the contents of test cases. These closely mirror the language and constructs used to formally specify the operation of an interface. Take for example the following diagram showing how a payments processing system may authorize a merchant request via a back-end host.

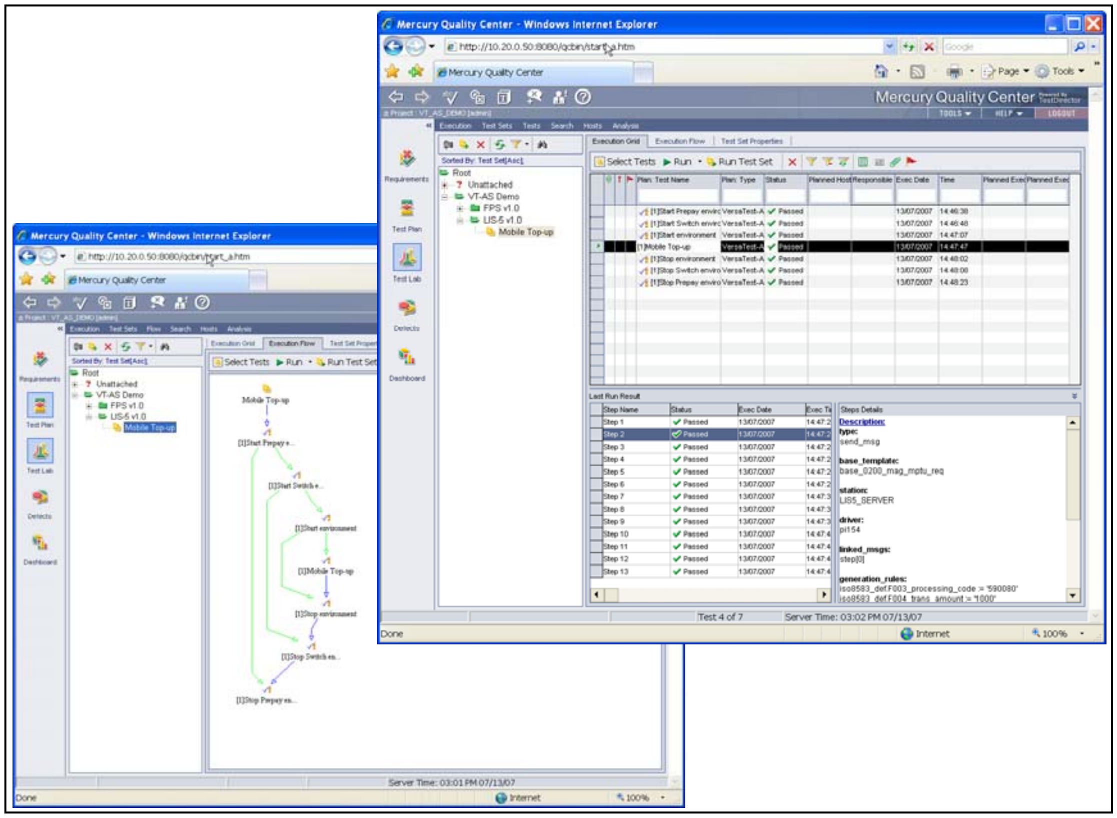


The steps in VTAS to test using two simulators would be something like the following:

1. Send message to payments processor
2. a) Expect message at back-end from payments processor
   b) Validate message field values against values sent in Step 1
3. Build and send response to payments processor
4. a) Expect message from payments processor
   b) Validate message field values against those sent in Step 3

The actual validation can be as complex as desired, and would normally tightly follow the message processing specifications, enabling a high-level of feedback for messages that fail validation.

These test specifications can then be stored in spreadsheets, as happens in many testing departments, or embedded in the above-mentioned third party test managers. Some example screenshots of HP's Mercury TestDirector being used with VTAS follow:



Plan: Test Name	Plan: Type	Status	Planned Host	Responsible	Exec Date	Time	Planned Exec	Planned Exec
[1]Start Prepay envirc	VersaTest-A	✓ Passed			13/07/2007	14:46:38		
[1]Start Switch envirc	VersaTest-A	✓ Passed			13/07/2007	14:46:48		
[1]Start environment	VersaTest-A	✓ Passed			13/07/2007	14:47:07		
[1]Mobile Top-up	VersaTest-A	✓ Passed			13/07/2007	14:47:47		
[1]Stop environment	VersaTest-A	✓ Passed			13/07/2007	14:48:02		
[1]Stop Switch enviro	VersaTest-A	✓ Passed			13/07/2007	14:48:08		
[1]Stop Prepay enviro	VersaTest-A	✓ Passed			13/07/2007	14:48:23		

Last Run Result

Step Name	Status	Exec Date	Exec Ti
Step 1	✓ Passed	13/07/2007	14:47:2
Step 2	✓ Passed	13/07/2007	14:47:2
Step 3	✓ Passed	13/07/2007	14:47:2
Step 4	✓ Passed	13/07/2007	14:47:2
Step 5	✓ Passed	13/07/2007	14:47:2
Step 6	✓ Passed	13/07/2007	14:47:2
Step 7	✓ Passed	13/07/2007	14:47:3
Step 8	✓ Passed	13/07/2007	14:47:3
Step 9	✓ Passed	13/07/2007	14:47:3
Step 10	✓ Passed	13/07/2007	14:47:4
Step 11	✓ Passed	13/07/2007	14:47:4
Step 12	✓ Passed	13/07/2007	14:47:4
Step 13	✓ Passed	13/07/2007	14:47:4

Steps Details

Description:
type:
send_msg

base_template:
base_0200_mag_mptu_req

station:
LIS5_SERVER

driver:
pi154

linked_msgs:
step[0]

generation_rules:
iso8583_def.F003_processing_code := '590080'
iso8583_def.F004_trans_amount := '1000'

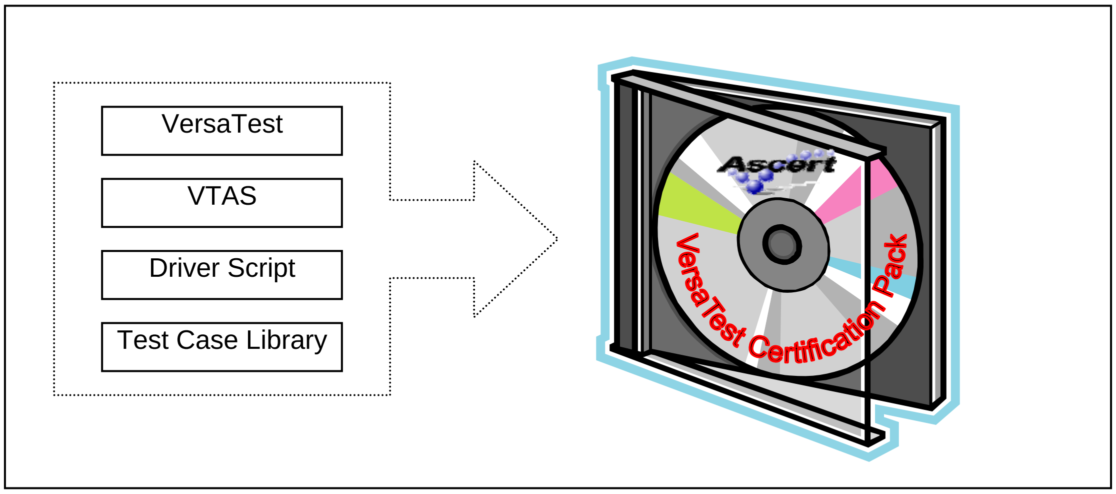
Test 4 of 7 Server Time: 03:02 PM 07/13/07

Server Time: 03:01 PM 07/13/07

Turnkey Testing & Simulation Environment

A common problem with users new to automated testing tools is the setting of unrealistic expectations. New users tend to envision that a testing tool can be brought in and immediately used in a new environment, forgetting that almost all environments have some unique aspects. This hopeful approach can be summarized as *testing by ESP!!* The reality is that automated testing is not simple.

There are, however, situations where a nearly turnkey solution can be provided. In these instances,, common interfaces are to be simulated, which is what often happens with certification systems. In this situation, VersaTest and VTAS can be delivered along with a test certification pack, containing all the scripts required to certify the operation of devices attached to the simulator. Everything is delivered in a single installation file.



Chapter Four

WHERE DO I GO FROM HERE?

Introduction

There is extensive documentation available for VersaTest. Where you go from here will depend on how you intend to use VersaTest.

If you want to…

- **…obtain the documentation for VersaTest**
  - The documentation is installed with every copy of VersaTest and is also available from the Downloads area of our website at http://www.ascert.com/support. Note that the Downloads area of our website is protected and some downloads require registration – if you don't see a downloadable file that you know exists, it probably means you're not logged on properly.
- **…install a copy of VersaTest**
  - If not administered through your own corporate software then you can obtain a copy of VersaTest through the Downloads area of our website at http://www.ascert.com/support. Note that the Downloads area of our website is protected and this download requires registration – if you don't see a downloadable file that you know exists, it probably means you're not logged on properly.

    There are multiple files that can be used for installation of VersaTest, which depend on the platform on which it is to be installed. Each download also has the installation instructions for the file attached, as well as the release notes for that version.

    Note that a full VersaTest installation requires a license key provided by an Ascert staff member.
- **…learn how to operate VersaTest using already-existing simulators**
  - If you have access to an installed VersaTest environment, then your next step should be to perform the tutorials installed with VersaTest. These are documented in the manual entitled *VersaTest Tutorials*.
  - Following that, for a more in-depth understanding of the operation of VersaTest, read the manual entitled *VersaTest User's Guide*.
- **…learn the features of an existing simulator**
  - All Ascert simulators come with documentation that describes their operation and any simulator-specific features and functions available to the operator. Refer to that documentation for more information.
- **…learn how to build new simulators from scratch, or modify existing ones**
  - Read the manual entitled *VersaTest Scripting Guide*, which provides both a reference to the VTALK scripting language and a *Getting Started* tutorial that includes our

version of the ubiquitous “hello, world” example.
Note that when it comes time to test your new simulator, you will need familiarity with the operation of VersaTest, as detailed above.

- **…find a reference for all the start-up options for VersaTest programs and utilities**
    - Read the manual entitled *VersaTest Command Line Reference*. This documents all the command-line settings for both the NonStop and Multi-Platform environments
- **…learn more about how to use VersaTest from HP Quality Center**
    - This documentation is in progress – check our website for any updates.
- **…find answers to a question we’ve not answered here**
    - The Support area of our website (http://www.ascert.com/support) contains a knowledgebase that is frequently updated with new information. Note that the information you see in the knowledgebase will differ depending on whether or not you are a registered user of our website.
    - If you can’t find the answer in our knowledgebase, feel free to post a message in our Support forums or send an email to our Technical Support group.