SYNAMETRICS TECHNOLOGIES

A division of IndusSoft Technologies Inc.

StressIT User's Guide

SYNAMETRICS TECHNOLOGIES

StressIT User's Guide

27 Sand Hill Ct. Jamesburg NJ
Phone 732.605.7088 • Fax 732.656.9745

Table of Contents

Chapter 1

Introduction

Find the upper limits of your web application

Welcome to the StressIT User Guide. This guide details the features and options of StressIT, a comprehensive tool for load testing Web-based applications.

This guide is intended for engineers and developers who conduct load and scalability testing, during and after the development of their web application.

About StressIT

The need for web application testing is self-evident: for an application in development, you need to know how many users and what transactional processes will prevent your site from performing at peak capacity; and for an application that's already live, you need to ensure that as your mission-critical web application gains in popularity, it will not suffer in performance.

StressIT is a robust stress-testing application that accurately simulates the number of users your site can maintain. It will allow you to chart the breaking point at which your site's performance is no longer acceptable, and will allow you to pinpoint the bottlenecks preventing your site from reaching its maximum performance levels. StressIT allow you to define relevant use-case scenarios for you web site; this customization enables you to test all aspect of your complex multi-tiered web-based application

Advantages of StressIT

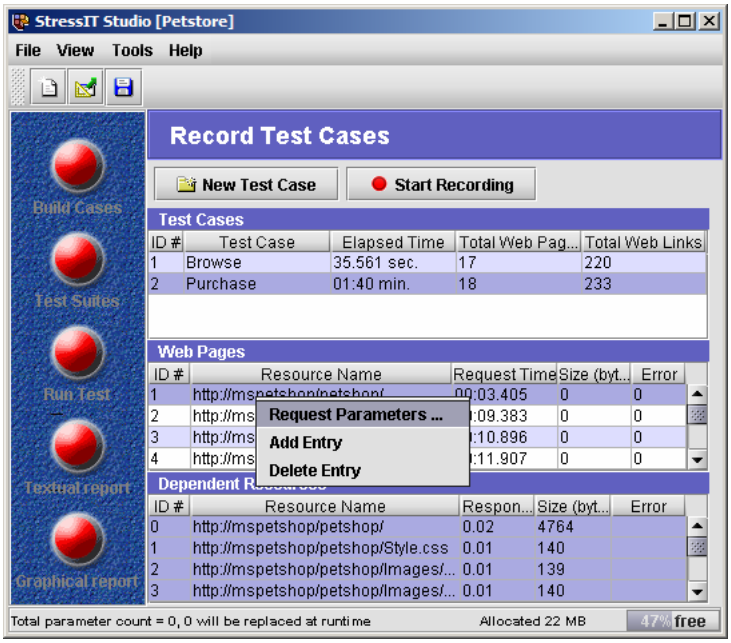
StressIT offers several advantages over other load testing tools available in the market today, including:

- Multi-Platform support
- An intuitive graphical user interface (GUI), which requires no knowledge of programming or scripting
- A single point of control with StressIT's centralized Coordinator
- Data-driven load testing with dynamic data replacement
- Probing the health of the server during test
- Extensive reporting capabilities with exports to any RDBMS including Oracle, DB2, Informix, Sybase, MS SQL Server and others.

Multi-Platform Support

Today's web applications are often deployed in a complex grid of different operating systems. As a result, it is necessary to make sure that testing is simulated from all these operating environments. StressIT is built on a Java-based platform; this provides users with the ability to create virtual users that come from any Java-supported platform, including MS Windows, Apple, Linux, Solaris and other flavors of Unix.

Intuitive GUI



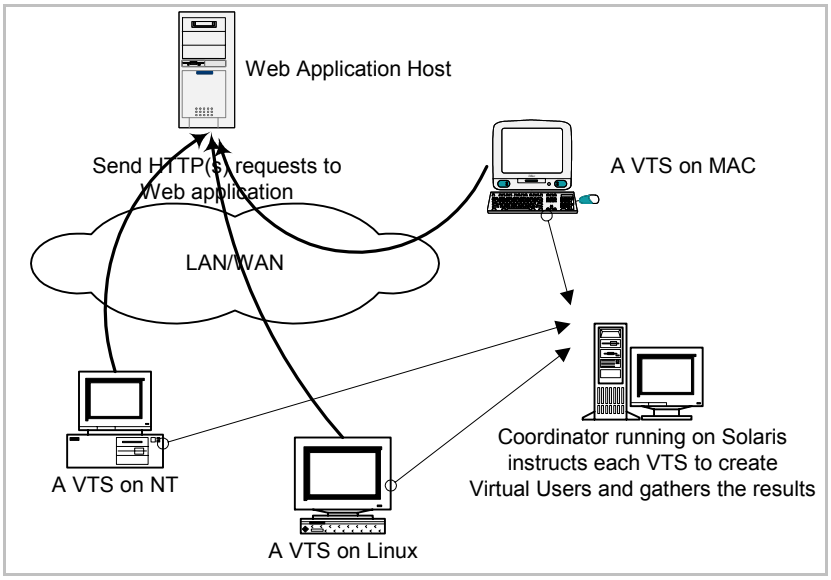
StressIT's intuitive graphical user interface allows users to easily perform recording and playback tasks without dealing with any scripts. This powerful feature allows StressIT to be a useful tool for testers who are not familiar with programming or scripting languages.

With a few simple mouse clicks, any developer or tester can create powerful test

scenarios that can be played back by thousands of virtual users, each accessing a different area of your web site, simulating real world transactions.

Single point of control

Users can be simulated by a single location or from a distributed testing infrastructure located anywhere on a LAN or WAN. In "StressIT-speak" each location is defined as a **Virtual Test Site (VTS)**.



VTS can be located with the same LAN or in a different physical location.

Data-Driven load testing

StressIT supports run-time data replacement for HTTP form and URL parameters. This feature provides the ability to submit requests with different parameters each time a request is submitted. At run-time, the data can be read from predefined Datasets or from previous HTML pages. For example, you can use different usernames, passwords, address, etc for each request. This customized data is automatically read in and passed as parameter values.

Extensive Reporting capabilities

StressIT's extensive reporting capability lets you analyze the test result in a variety of ways. Results of the test are provided as easy-to-read textual and graphical reports. Besides the built-in canned reports, StressIT also allows you to create your own customized reports by exporting the test results to a variety of databases including

Oracle, DB2, Informix, Sybase, MS SQL Server, MS Access and others. These reports can also be created using third party tools such as Crystal Reports.

What can you test with StressIT?

Most applications are usually thoroughly tested by its developers for unit testing, which are usually run one at a time, independent of each other. However, in order to ensure the robustness of the system, an end-to-end test must be run with multiple users hitting the system concurrently. That is where StressIT comes in – it helps you to:

- Chart performance degradation when the number of users are increased
- Exposes logical and functional problems when two or more users try to perform the same task concurrently
- Helps you identify the affects on one module of your application when a different module is modified.

System Requirements

StressIT has two different components that are installed, typically on different machines. These components are the Coordinator and the Virtual Test Site. The requirements of each component are show below.

Coordinator

Operating System:

Any 32-bit Microsoft platform including Windows XP and Windows 2000, Solaris 2.6 or above and any flavor of Linux and Apple Macintosh.

Memory: Minimum 128 MB; 512 MB recommended; 1 GB preferred. The actual memory requirement may vary depending upon the size of your web site. It is always a good idea to run the coordinator on a machine that has plenty of RAM, usually more than 512 MB.

Disk Space: 20 MB

Java Run Time 1.4.1 (JRE is included with the installation file)

VTS

Operating System: Any 32-bit Microsoft platform including Windows XP and Windows 2000, Solaris 2.6 or above, any flavor of Linux and Apple Macintosh.

Memory: Minimum 64 MB; 256 MB recommended; 512 MB preferred. The actual memory requirement may vary depending upon the size of your web site. It is always a

good idea to run the VTS on a machine that has plenty of RAM, usually more than 256 MB.

Disk Space: 20 MB

Java Run Time 1.4.1 (JRE is included with the installation file)

Chapter 2

StressIT Basics

This chapter explains how to get started using StressIT. It explains how to install, and start the program and defines some key terminology that is used in the program. The same terminology is used throughout this manual as well as in the StressIT application.

Download

StressIT is supported on various operating systems. Therefore, it is important that you download the correct file that applies to your operating environment. The following table shows the file names.

File Name	Operating System	Description
StressITVM.zip	Windows	Installer for Windows. This includes the Java runtime version 1.4.1. If you decide to download this file, you do NOT need to download any other file.
StressITNoVM.zip	Windows	Installer for Microsoft Windows. Does not include the Java runtime. You will have to manually download it from Sun Microsystems (http://java.sun.com). You need JRE version 1.4.1 or above.
StressITSolaris.bin	Solaris	Installer for Sun Solaris OS. It does not include the Java runtime. You will have to manually download it from Sun Microsystems (http://java.sun.com). You

		need JRE version 1.4.1 or above.
StressITLinux.bin	Linux	Installer for Linux OS. It does not include the Java runtime. You will have to manually download it from Sun Microsystems (http://java.sun.com). You need JRE version 1.4.1 or above.
StressITMac.zip	Mac OS X	Installer for Apple Macintosh. It does not include the Java runtime. You will have to manually download it from Sun Microsystems (http://java.sun.com). You need JRE version 1.4.1 or above.
StressITOther.zip	Other OS	Installer for other Java-enabled operating systems. It does not include the Java runtime. You will have to manually download it from Sun Microsystems (http://java.sun.com). You need JRE version 1.4.1 or above.

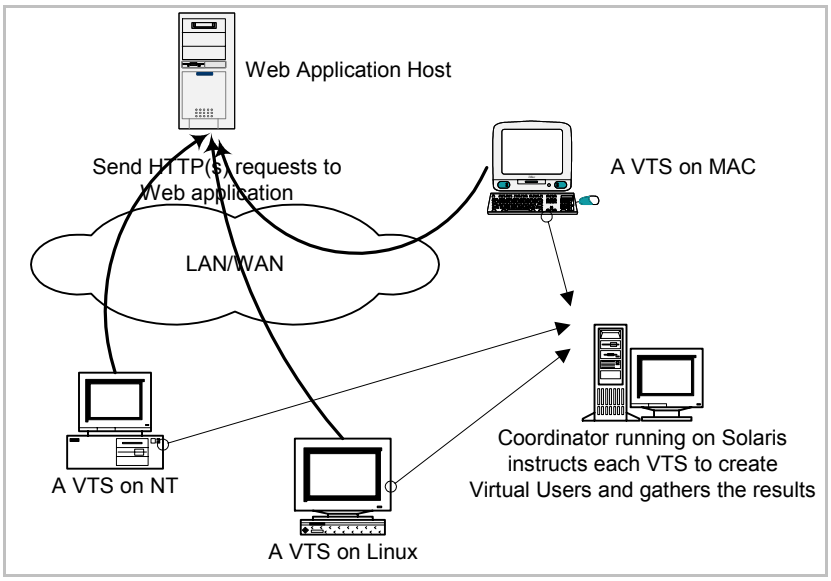
Note: Some of the installers are zipped and therefore, you will need an upzip program to extract the executable

Running the installer

The installation process depends upon what role you want the machine to play where you are installing StressIT. The installation process will prompt you if you wish to install:

- Coordinator and VTS – Select this if this machine will be used as a Coordinator. You can only install **one** coordinator per license purchased
- Virtual Test Site Only – Select this if you will be using this machine as a Virtual Test Site (VTS). There is no limit on how many machines you install the VTS per license.

In a typical setup, you install the Coordinator on one machine and VTS on multiple machines. When a test is performed, the Coordinator will communicate with all the virtual test sites and manage the tasks that are performed by each VTS.



The following section explains the procedure for installing StressIT coordinator and VTS.

Installing StressIT

The StressIT coordinator setup procedure installs both Coordinator and VTS. You should install the coordinator on only one machine for which you have purchased the license.

Steps to install the coordinator are given below.

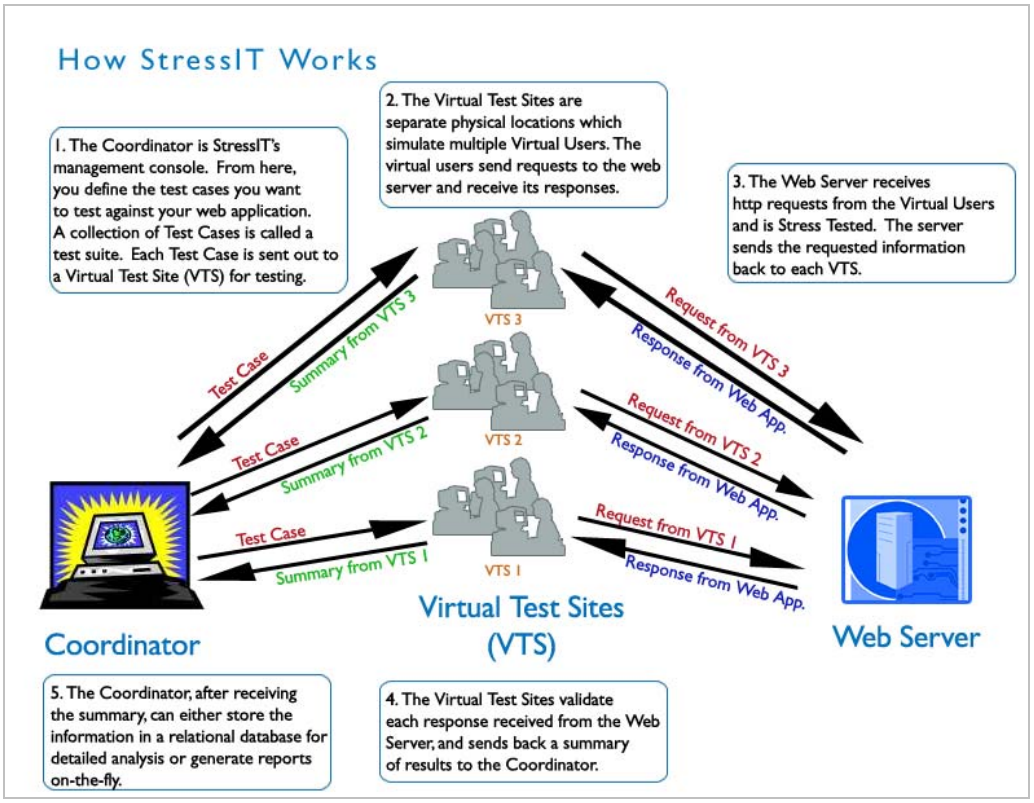
- Download StressIT from Synametrics' web site (http://www.synametrics.com) and run the appropriate install file for your environment.
- Extract the executable file using WinZIP or similar program. You do not have to unzip if the extension of the file is not .zip
- Run the installer program
- Follow the setup instructions to install StressIT Studio

Key Concepts and Terminology

The following section describes the key concepts of StressIT and the terminology that is used in this document as well as the application.

The big picture

The following image shows the interaction between the Coordinator, the VTS and the web server that hosts your web application.



StressIT's three-step process

StressIT provides you with an easy three-step process for testing your web server's performance

- Capturing phase - Capture what you want to test
- Testing phase - Playback your captured instructions with hundreds and thousands of virtual users
- Analysis phase - Collect the resulting data and analyze from StressIT studio or export it to a database for custom reporting

Projects

In StressIT, a project is defined as a logical grouping of test suites, test cases and datasets. Creating and defining a project is the first step in creating load test simulation in StressIT.

Test Case

A Test Case (TC) is a set of transactions, which you define, to test against a web application. One test case often corresponds to a design use case and simulates the likely path your users will follow on your web application. A test case could be as simple as following a link from the home page to the contact us page, or as complex as enacting an e-commerce transaction with the accompanying dynamic interactions that involves session management, database information retrieval and communication with other backend servers.

As the site administrator or developer, it is very important that you define accurate test cases that correspond to a real-world scenario. You want to make sure that the processes you are testing are the ones your users are likely to use. Well-defined and relevant test cases will allow StressIT to provide you with more detailed, accurate information about the stress your web server will be able to handle.

Often a test case has a one-to-one correspondence with a Use Case in your system. For instance, you can define a test case for:

- Lookup catalog – In this case the test case will consist of a number of web requests performed by a user who is browsing through your company's product catalog.

- Purchase product – In this case the test case will consist of a web request required to make a purchase, which might include filling a shopping cart, acquiring necessary information for billing and shipping and credit card processing

Test Suite

A test suite is a collection of test cases. Besides being a collection, it is also treated as a unit of work when the test is run.

Virtual User (VU)

A virtual user simulates a real client/user when a test is run. Each virtual user is given a task that it performs in the course of its lifetime. A typical task includes:

- Preparing an HTTP request that will be sent to the server

- Submitting the request

- Validating the response time and its contents

- Logging that information and sending it to the Coordinator, which will compile the information for analysis

Virtual Test Site (VTS)

StressIT allows you to measure the performance and accuracy of your web application under heavy loads. Therefore, it is crucial that test requests be sent from multiple client machines, or else your client machine will run out of resources and the server will never get a chance to achieve its limit. StressIT defines these client machines as Virtual Test Sites (VTS). Each VTS can be thought of as a physical location where your clients are coming from. Each VTS can have one or more virtual users.

Coordinator

When a test is run against a web server, several VTS machines send HTTP requests simultaneously to the web server, simulating thousands of users. All VTS machines run independently of each other. However, every VTS is connected to a separate process, called a Coordinator. The Coordinator is a program that gathers and manages the results of each request sent from the different VTS.

The Coordinator is assigned the following responsibilities:

- It serves as a management console for the VTS
- Decides which VTS should create a new Virtual User
- Gathers summarized information regarding HTTP requests and errors from all the VTS's

Web Page

A Web Page in StressIT speak corresponds to a URL that a user types into the URL window. For instance http://www.mycompany.com/index.html

Web Page Dependent

A Web Page may have references to other resources such as images, applets, and style sheets. These are called as dependents of a web page.

Datasets

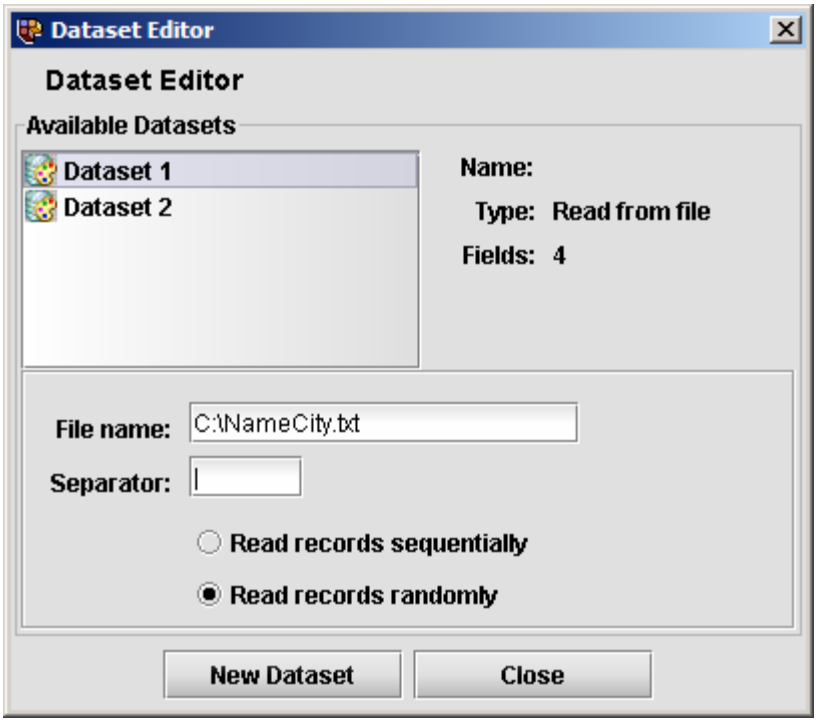
StressIT provides the ability to change HTTP parameter data at runtime, which are read from a dataset. Currently, StressIT supports two kind of datasets:

- Flat file
- Custom

A flat file dataset assumes that you have a plain text file with delimited data. A custom dataset can be used to enter value manually in StressIT

Dataset Editor

To create a new dataset click Tool →Manage Datasets. The following screen will open up:



The screen allows you to modify the parameters for a dataset or create new ones. The dataset shown in the screen refers to a file NameCity.txt that holds a list of cities, state and zip code. For example:

Abilene|TX|79602
Abilene|TX|79605
Abingdon|VA|24210
Acton|MA|01720
Adamstown|MD|21710

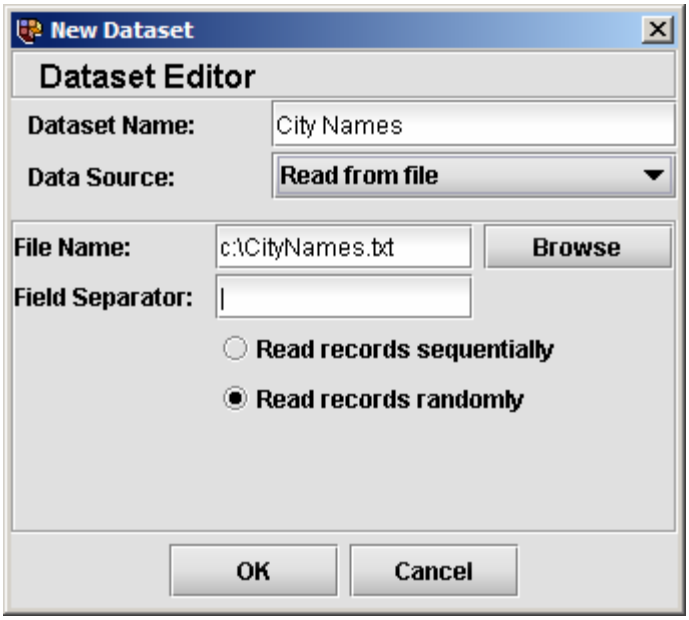
When a dataset is created in this manner, you can use it to provide data for a form that requires these fields. StressIT always assigns fields from one line, which means Abilene will always go with Texas with a zip code of 79602.

If the type of the dataset is "Custom", you see a grid with appropriate rows and columns, which allows you to modify the content and save it back.

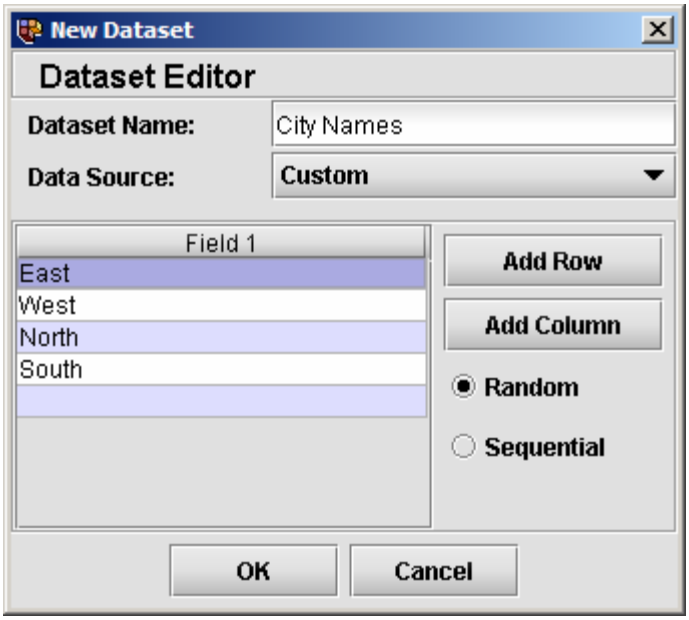
Creating new Datasets

To create a new dataset click the "New Dataset" button in the Dataset Editor. This brings up the following screen.

Dataset that is based on flat files



Custom Dataset – you can add data manually



Dataset Name – This is a friendly name for the dataset. StressIT uses this name throughout the program to refer to a dataset

Data Source – This defines the type of the dataset.

File Name – Name of the file (Available when data source is a flat file)

Field Separator – You can put multiple fields separated by this character (Available when data source is a flat file)

Random Vs Sequential Access

At run time, dataset can be read either sequentially or randomly. In both cases StressIT will load the complete file into memory and then read one line either randomly or sequentially.

When sequential access is selected, every virtual user will read the file sequentially and use the read data to submit HTTP requests. If the file does not have enough data, it will start from the beginning again.

Chapter 3

Phase I - Capturing

Creating Test Cases

As mentioned earlier in the manual, StressIT defines a three-step process to test your web application. This chapter discusses the first phase: Capturing.

Defining a test case

By definition a test case is a set of HTTP request that are sent to the web server. These requests should be logically grouped in such a way that they represent a business use case such as:

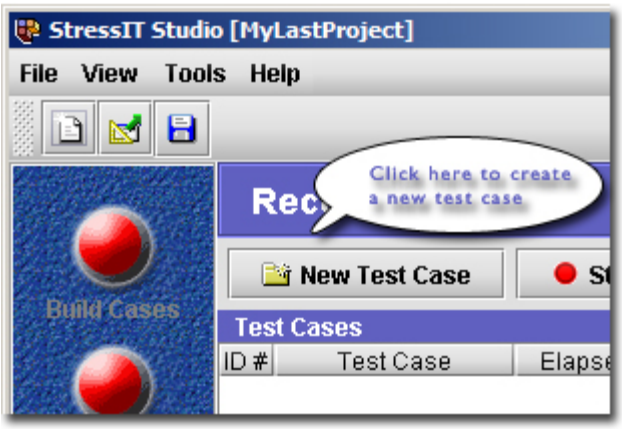
- Catalog Browsing
- Product Purchase
- Administrative tasks

You can create an unlimited number of test case for your web site.

Creating a test case

First, you need to create a test case. Go to the "Record Test Cases" screen by either clicking on the Build cases button or selecting "Record Test Cases" from the view menu. Click the New Test Case button to create a test case. Specify a user friendly name for your test case, which should signify the use case you are trying to test, such as Catalog Browsing.

Once a test case is created you are ready to record individual HTTP requests.



Recording HTTP requests

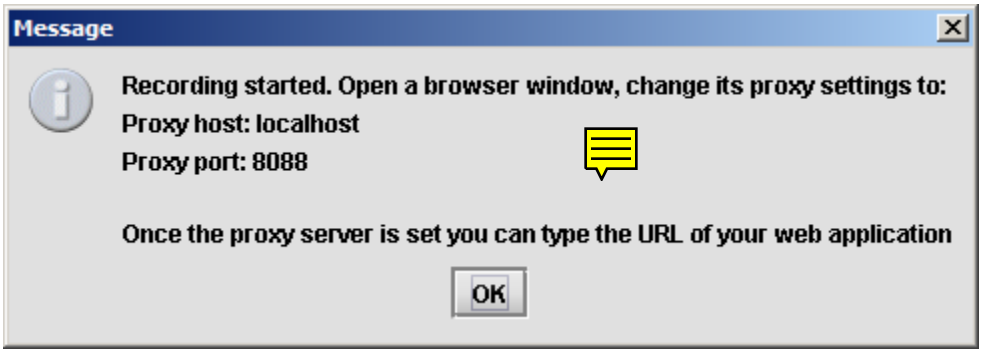
There are two ways to enter HTTP requests in a Test Case:

- Intercepting HTTP communication between a browser and a web server
- Manually entering the URLs

Intercepting HTTP request

This is the easiest way to record HTTP requests that are sent to a web server. Click the "Start Recording" button in the Build Cases panel. Depending on the operating system you are using, StressIT will open up a browser window in which you can browse to your web server application.

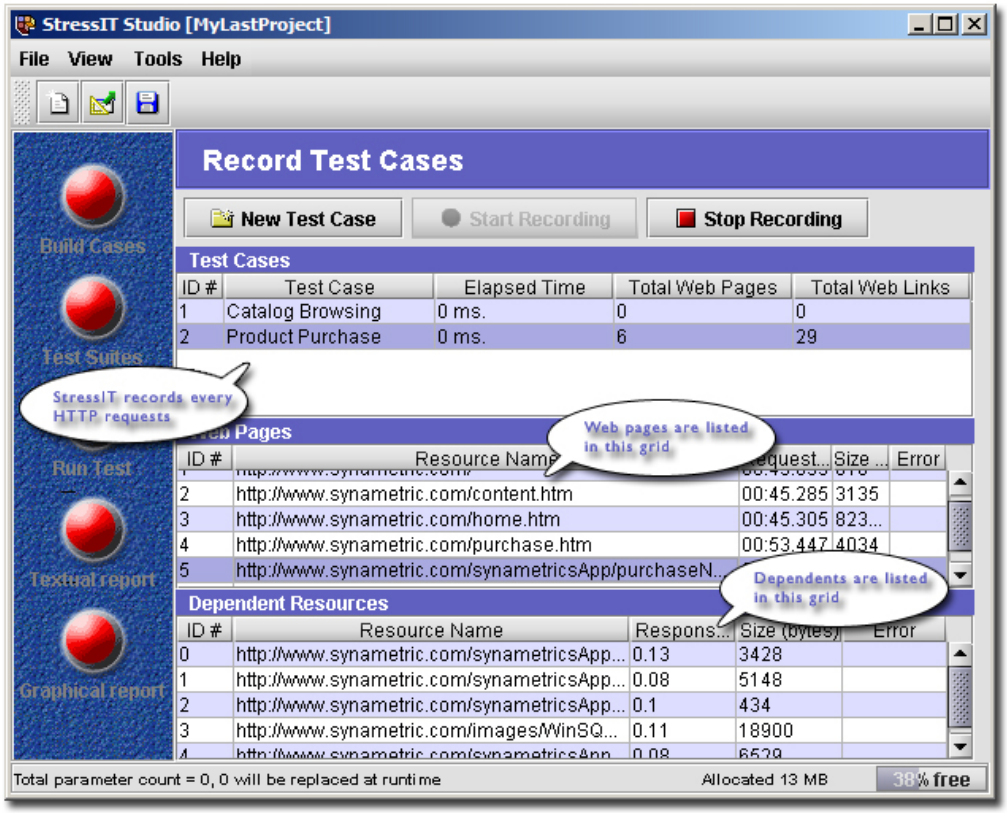
If you are using StressIT on an OS like Solaris, StressIT will not open a new browser window. Instead it will display the following message:



In this case you need to manually open up a browser, change its proxy settings to the specified settings in the message and type the URL.

How does interception work? StressIT Studio has a built-in proxy server for HTTP, which allows it to capture all the communication between a web browser and a web server

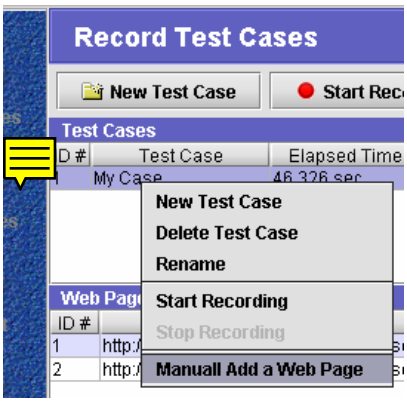
As you type different URLs in the browser, StressIT studio creates appropriate Web Page and Web Page dependent objects that are embedded into a test case.



Tip: The status bar in StressIT shows if a Web Page has any form or URL parameters. Click a web page to see the status bar getting updated

When you capture a web page using the record feature, besides capturing the URL of the resource, StressIT will also record if there are any query parameters, HTTP header values, user id and password for authentication and other necessary information.

Capture web pages manually



An alternative method to capture the URL is using the Manual URL entry dialog box, which is invoked by selecting "Manually Add a Web Page" from the pop-up menu, which appears when you right click within either the Test Cases or Web Page section of the page.

Adding Web Pages to the test case in this way has the following advantages over intercepting the HTTP requests:

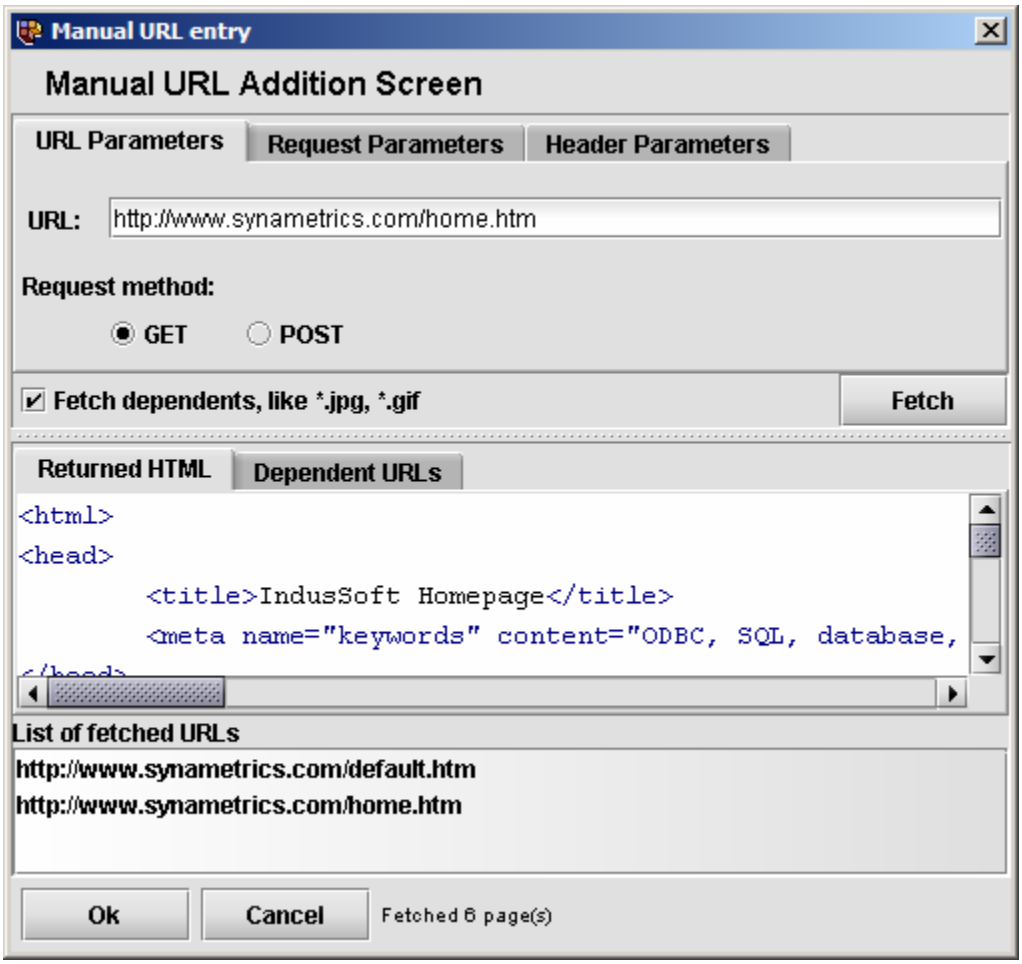
- It allows you to optionally ignore the

dependent pages. For instance, if you decide to test only the dynamic content and do not want to capture the requests for images, style sheets or any other static content, manually entering the web pages allows you to do this.

- You can add additional parameters for the HTTP query.
- You are able to change the HTTP method (POST vs GET)
- Use HTTPS instead of HTTP

Tip: Each time to hit Enter or click the "Fetch" button, a URL gets added to the test case. Hence, you can enter as many web page as you like without closing the window

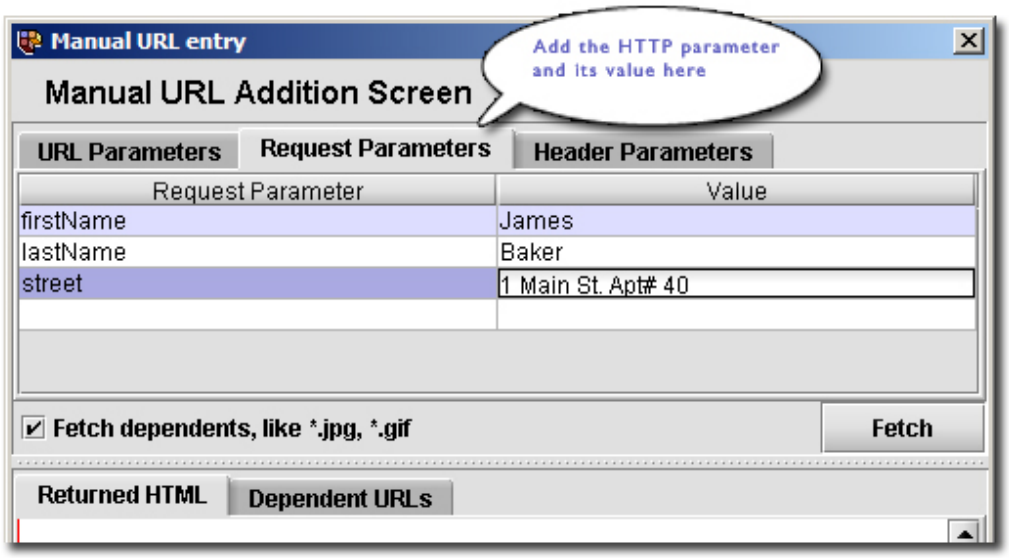
The following screen shows how to capture web pages manually.



Type the desired URL in the field and either hit the Enter key or click the Fetch button. At this point StressIT will send an HTTP request to the web server. When the server returns the HTML, StressIT parses the HTML and can, optionally, ask for the dependent resources such as images.

Adding Request Parameters

Often you need to submit parameters to the URL, also referred to as the HTTP Query String. These parameters can be added by added entries in the grid which is available on the "Request Parameter" tab. StressIT will append these parameters to the URL before submitting the request, provided the HTTP method is GET. In case the HTTP method is POST, StressIT will append these parameters to the content part of the request. Refer to the HTTP Reference in the appendix for differences between GET and POST.



Note that in the above image you can create as many parameters as you like and StressIT will appropriately encode the URL with the correct string. For example, the resultant URL for the above image will be:

> http://www.xyx.com/guest.jsp?firstName=James&lastName=Baker&street=1+Main+St.+Apt%23+40

Similarly, you can add parameters to the HTTP header.

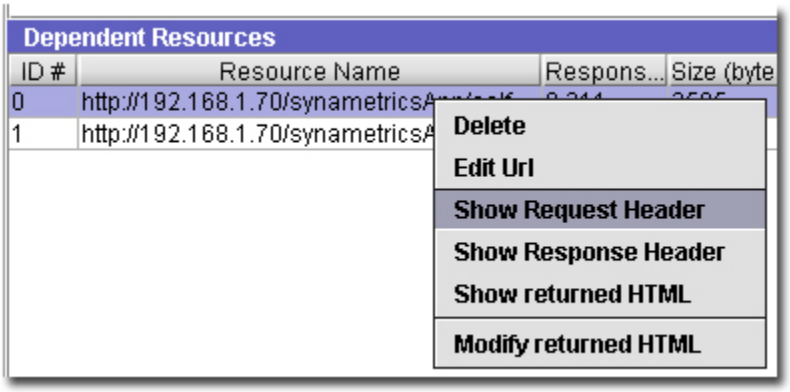
Viewing HTTP Request/Response Headers

Every request that is sent to the web server is associated with an HTTP header, which can be viewed by selecting "Show Request Header" or "Show Response Header" from the pop-up menu. This pop-up menu is invoked by clicking the right mouse button while a resource name is selected in the Dependent Resources grid.

HTTP headers have a series of name-value pair that describe a request to the web server. Since HTTP is a session-less protocol, many web applications put session information in the HTTP header. In the following sections you will learn how to modify the contents of the header when testing is performed.

Viewing HTML sent by the server

Similarly, you can view the HTML that was sent by the web server by selecting "Show Returned HTML" from the pop-up menu. You can optionally modify the returned HTML.



Resource ID

Every web page dependent in StressIT is identified by a resource ID. This resource id is used throughout StressIT to identify a request to the web server.

TIP: Resource ID is used through out StressIT to identify a URL.

Resource ID comprises of three parts

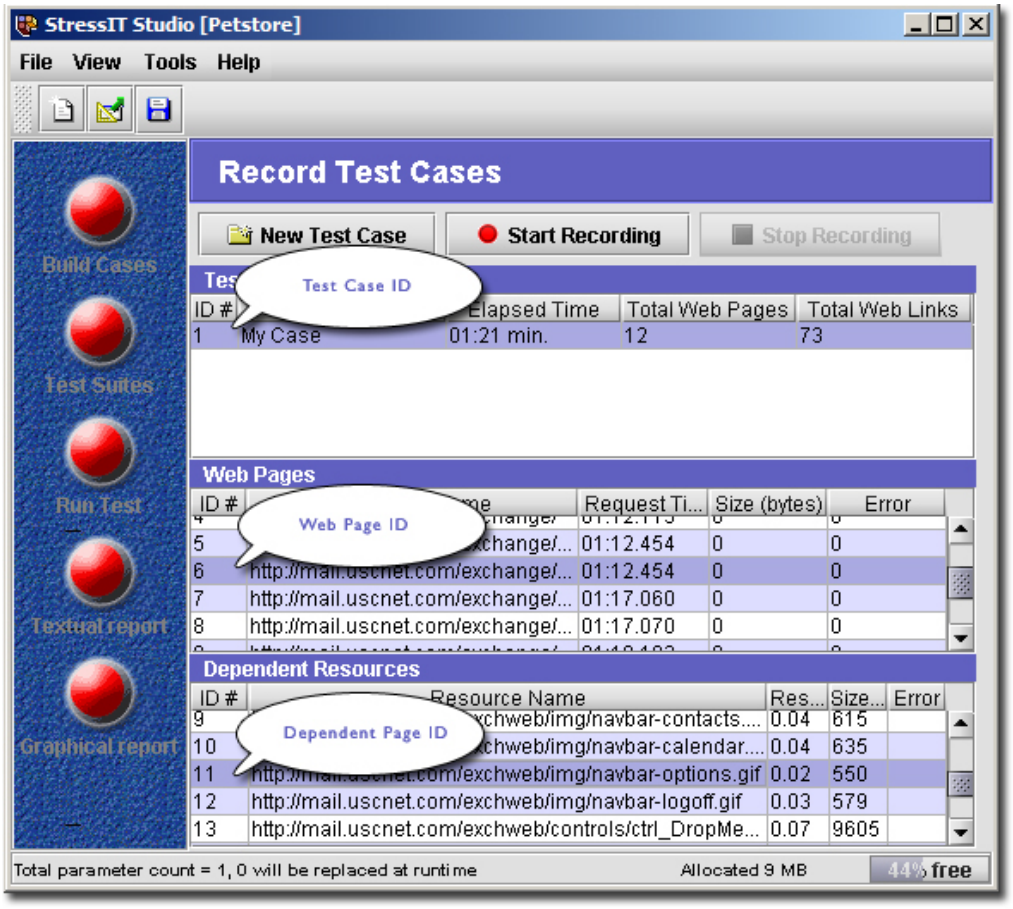
- Test case number – starts with a 1
- Web page number – starts with a 1
- Web page dependent number – starts with a 0

The first columns in all three grids (Test Cases, Web Pages, and Dependent Resources) correspond to an id. A resource ID is generated by simply concatenating these three number separated by a dash character.

For instance:

1-3-23 signifies a resource that is in Test Case 1, Web Page 3, and a dependent resource of 23.

Notice that a web page dependent, unlike test case and web page, starts with a 0. The dependent on the 0 position corresponds to the main URL. All the remaining entries are dependents such as images, style sheets, class files for Java applets, etc..



In the above figure the Resource ID for the selected URL is **1-6-11**

Chapter 4

Modifying Test Cases

This chapter shows some advance features of StressIT

Modifying Web Page Parameters

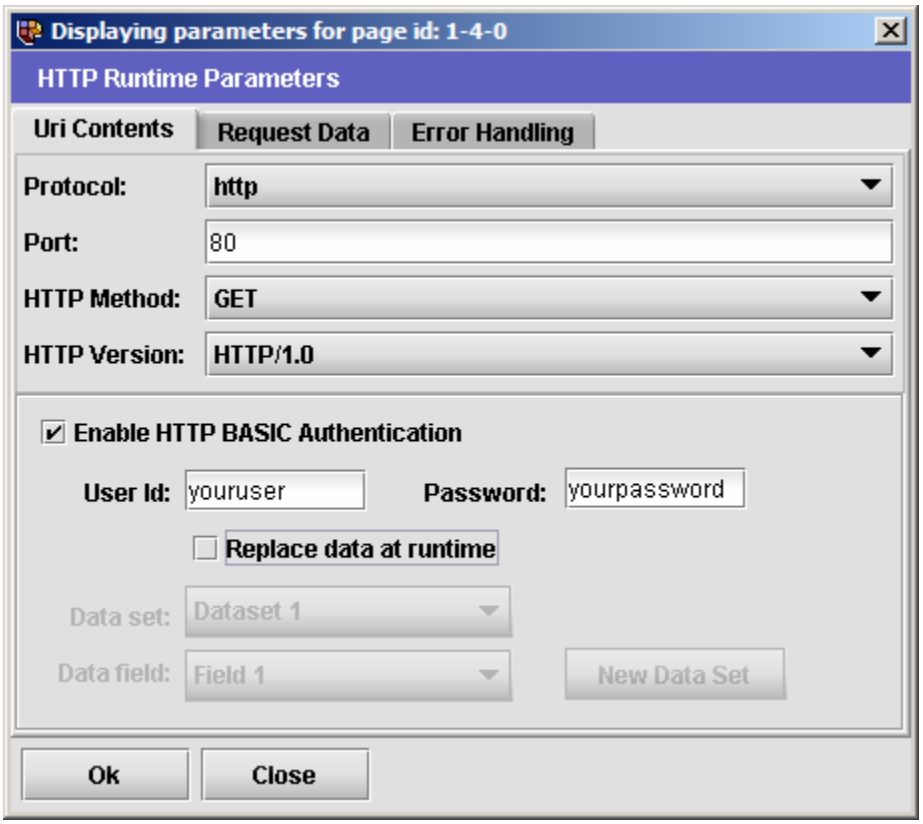
There are two types of web applications: static and dynamic. Static web pages are those that refer to a static file, which usually resides on the web server's hard disk. In other words, the content of a static file does not change. Dynamic pages on the other hand do not point to a file; instead they invoke a program running on a web server or an application server, which then generates the HTML content dynamically.

In the case of a dynamic web page, you need to send different parameters to the web server on every request. These parameters can be used for:

- User Authentication
- Form variables
- URL parameters
- Session management (When cookies are not used)

The following sections will explain how to change the HTTP request parameter, also called a request query, in StressIT.

Before you change any runtime parameter, you need to invoke the HTTP Runtime Parameter dialog box, which is done by double clicking a web page from the web page grid or selecting "Modify HTTP Request Parameters" from the pop-up menu, which brings up the following screen



The title of this screen specifies the Resource ID for which you are editing the parameter for.

This screen allow you to change the following entries

- Protocol – Switch between HTTP or HTTPS
- Port – TCP/IP port on which the server is listening. Default is port 80
- HTTP Method – This can be either POST or GET. Refer to the appendix for a description of HTTP Methods
- HTTP Version – This is the version of HTTP. Supported versions are HTTP 1.0 and HTTP 1.1

User Authentication

There are several ways to authenticate users in a web application. The two most common mechanisms to challenge a user for his/her credential are:

- Using HTTP BASIC authentication

- Using an HTML form

HTTP BASIC Authentication

StressIT supports both mechanisms of authentication. If a web page requires BASIC authentication, StressIT recognizes this and enables the check box for BASIC authentication. Here you have a choice of changing the value or reading the values at runtime from a Dataset. Datasets are defined in the previous section of the manual. Datasets are a data repository containing text values which can are passed to an HTTP request.

Tip: If you use runtime data replacement for authentication, both user id and password must be defined in one field separated by a comma.

If you wish to change the data at runtime, check the "Replace data at runtime" box and select the appropriate Dataset values.

Data replacement is necessary during testing so that every user can login using a different login ID password.

When you define a new Dataset for authentication, both User id and password must be in the same field, separated by a comma:

Here is a sample of an actual dataset file that can be used for HTTP BASIC authentication

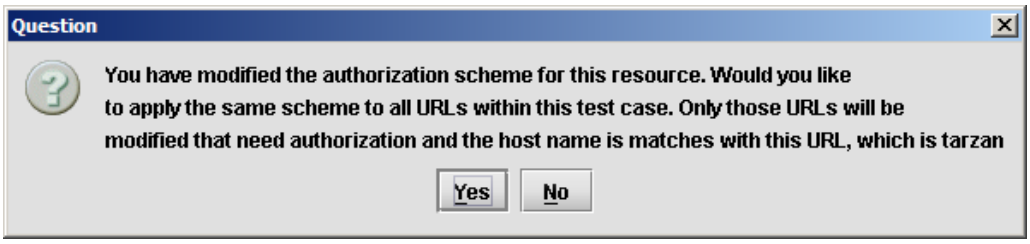
```
guest,guest123
public,open
scott,tiger
```

Notice that there is only one column in the file and each line represents a set of user id and password

Modifying user id and password for a test case

In the case of HTTP BASIC authentication, a client has to send the user id and password with every HTTP request. Therefore, you need to modify each and every web page and its dependent in order for the authentication scheme to work. Since this is a potentially tedious task, StressIT allows you to change the authentication scheme for the whole test case at once.

When you change the authentication scheme for any Web Page, StressIT asks you the following question and if you say "Yes", it will modify all web pages within your test case that require authentication and where the host name is same.



Form-based Authentication

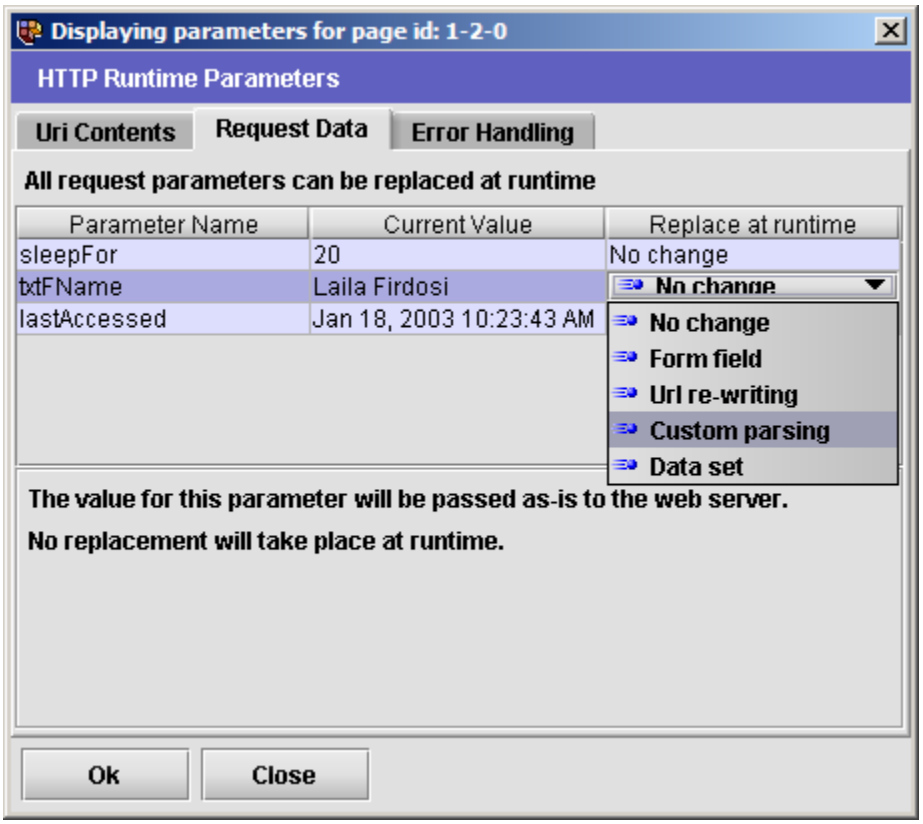
Another method for authentication is form based authentication, where a dynamic page on the web server captures the user credentials. Based on the values passed, the user is authorized for various levels of service.

Refer to the section on how to modify Form parameters if you wish to change values for this type.

Modifying HTTP query parameters

As mentioned earlier in the manual, StressIT allows you to change the HTTP query parameters that are sent to the server. These parameters can be part of a URL, in which case the method type is GET, or they can be part of an HTML form, in which case the method can be GET or POST.

Although the underlying mechanism for GET and POST is quite different, StressIT allows you to change the parameters for both methods in similar manner.



The second tab (Request Data) on the HTTP Runtime Parameters screen allows you to change the values for the parameters. The third column contains a selection box that allows you to pick different options, which define how and where will StressIT decides to assign value parameters at runtime. These options are defined below

No Change

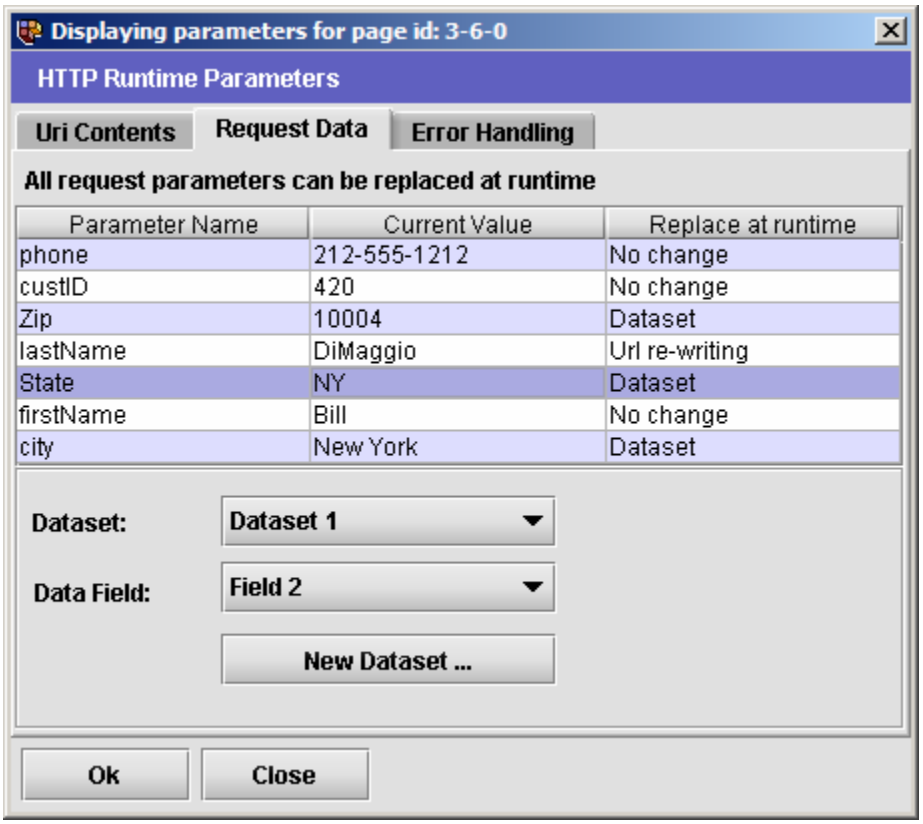
This is the default value of the parameter and it means that the value specified for "Current Value" will be passed as-is to the server. You can change the value of "Current Value" by simply typing the new value in the grid.

Form Field, Url Re-write, Custom Parsing

These options are used for session management and therefore are discussed in the next section.

Dataset

At runtime StressIT can read values from a dataset, which were defined previously in the manual.



The dataset combo box allows you to select a dataset from available datasets. If none is available you can click the "New Dataset" button to add one. Please refer to the Dataset section for details.

The Data Field refers to a field within the dataset.

EXAMPLE:

In the above figure, three fields are configured for dataset. This means that during the testing phase, StressIT will read the values for City, State and Zip from a dataset. Since there is a relationship between City, State and Zip – that is a city should belong to an appropriate state with a correct zip code, they all should be configured to come from the same dataset with different fields. The table below summarizes the values.

Table 1

Field Name	Dataset Name	Field Number
City	Dataset 1	Field 1
State	Dataset 1	Field 2

Zip	Dataset 1	Field 3

Assume that the dataset is configured to read a file and its format is:

Abilene|TX|79602
Abilene|TX|79605
Abingdon|VA|24210
Acton|MA|01720
Adamstown|MD|21710

This means that at runtime, StressIT will pick one line at a time, parse it into three different fields and assign values to city, state and zip respectively.

Session Management

Since HTTP is a connection-less protocol, web applications cannot rely on TCP/IP to maintain relationships between multiple requests. However, there are multiple ways for a web application to maintain relationships between different requests. The notion of maintaining relationship between multiple requests is also known as session management or state management. The mechanisms used for session management are:

- Cookies
- Hidden form fields
- URL Rewriting

Cookies

Many application use cookies to store state information, which are stored in the HTTP Header. In this case a web application sets a cookie, which is stored by the client and returned back to the server in subsequent requests.

StressIT automatically recognizes a cookie and sends it back to the server in subsequent calls. You do not have to do anything special in StressIT to enable support for cookies.

Hidden form fields

Another mechanism for maintaining state information from one page to another is hidden form fields. If your server uses this mechanism to store state information, you need to tell StressIT which fields you need to transfer from one page to another. Refer to the example below for details on how StressIT supports this option.

URL Rewriting

Similar to Hidden form fields, this option requires that values must be passed from one HTML page to another. Refer to the example below for details on how StressIT supports this option.

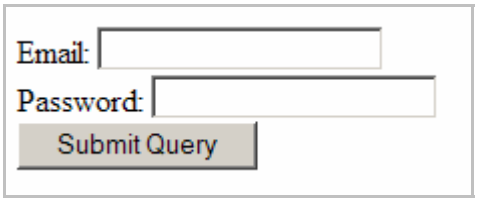
EXAMPLE

In order to explain hidden form fields and URL rewriting let's take an example. In this example, a user wants to change the subscription delivery option of a magazine. There are two choices: either the user can get the magazine through postal mail or in electronic format through email. There are three web pages involved in this process:

- Login page – User logs in using their email address and password
- Selection page – User selects whether they want the magazine through email or postal mail
- Confirmation page – The server saves the new information and displays a confirmation page.

The HTML for all three web pages is given below

Screen image for LoginPage.htm



HTML code for LoginPage.htm

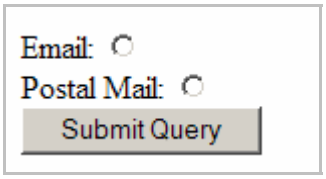
Resource ID: 4-1-0

```
<html>
<body>

<form action="changeOption.cgi">
Email: <input type="text" name="email">
Password: <input type="text" name="password">
<input type="hidden" name="sessionID" value="J33497365485451">
<input type="submit">
</form>

</body>
</html>
```

Notice that in the above code, the client will submit call changeOption.cgi, which is a program running on the server. This program is responsible for authenticating the user .If he/she is authorized, the next screen is displayed which prompts the user to change the delivery option. This form includes a sessionID field, which will be used for all subsequent calls to the server

The second screen prompts the user for a choice.

Screen image for changeOption.cgi



HTML code for changeOption.cgi

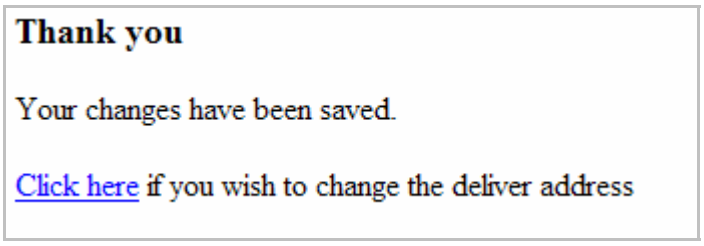
Resource ID: 4-2-0

```
<html>
<body>

<form action="saveChoice.cgi">
Email: <input type="radio" name="deliveryOption" value="1"> <br>
Postal Mail: <input type="radio" name="deliveryOption" value="2">
<input type="hidden" name="sessionID" value="J33497365485451">
<input type="submit">
</form>

</body>
</html>
```

Let's analyze the HTML for the second page. It has three form fields: a set of radio buttons, one submit button and one hidden field that holds a sessionID. This session id will connect this request with the user who was authenticated and authorized in the first step.

The third screen displays a confirmation message and offers a link that will take the user to another screen that allows him/her to change the delivery address.

Screen image for saveChoice.cgi



HTML code for saveChoice.cgi

Resource ID: 4-3-0

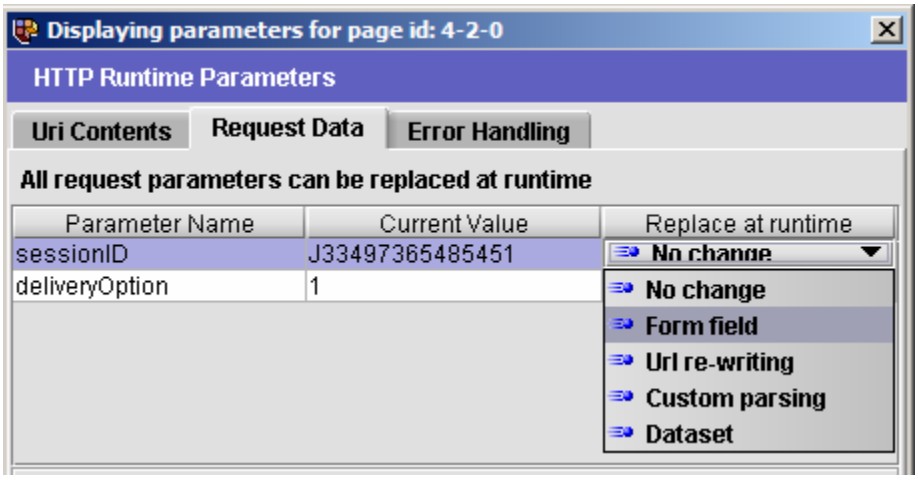
```
<html>
<body>
<h3>Thank you</h3>
Your changes have been saved. <p>
<a href="changeAddress.cgi?sessionID=J33497365485451">Click here</a> if you wish to
change the deliver address
</body>
</html>
```

Notice that the third page does not have a form, but a URL link that will take the user to the next step. Besides holding the resource (changeAddress.cgi), the href attribute of the anchor tag also specifies a parameter which contains the session information.

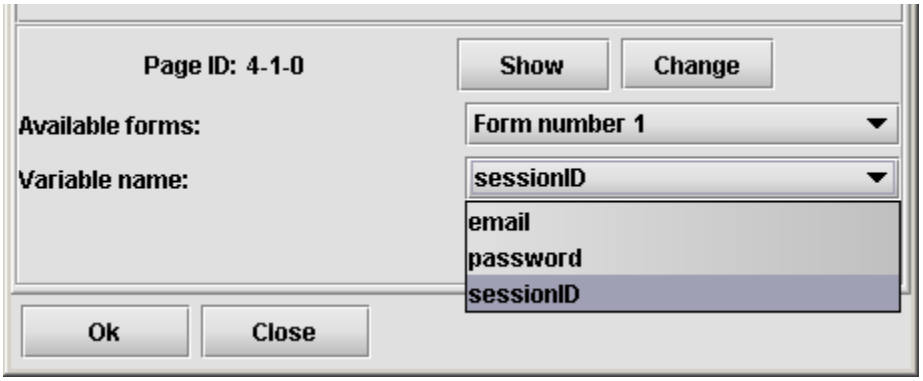
Extracting hidden form fields values

Now let's see how to extract the session id from the HTML returned by the first page, whose resource id is 4-1-0, and pass that value to the next request where resource ID is 4-2-0. Bring the HTTP Runtime parameter dialog box by double-clicking the second web page (resource id = 4-2-0)

First, select "Form field" from the "Replace at runtime" combo box.



StressIT will parse the HTML returned by resource id 4-1-0 and it figures out that there is one form with three fields and fills the combo box appropriately.

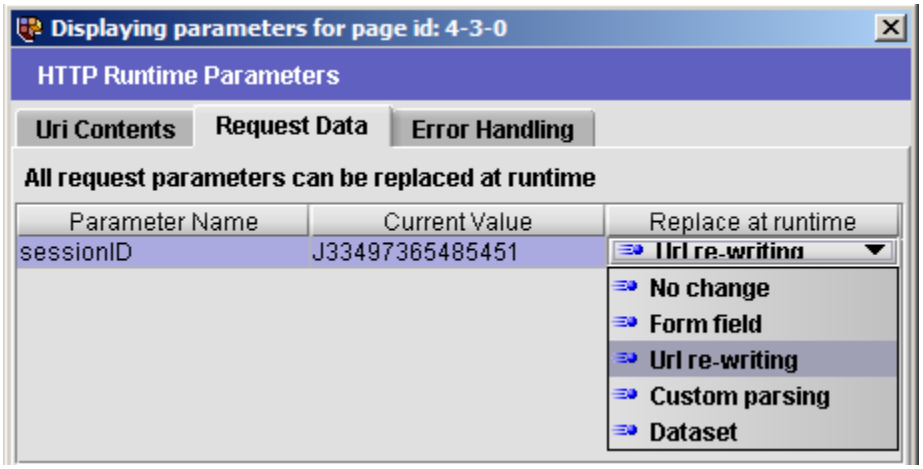


Therefore, all you need to do is select "Form number 1" in the available form field and "sessionID" in the variable name field. During runtime, StressIT will handle the parsing automatically.

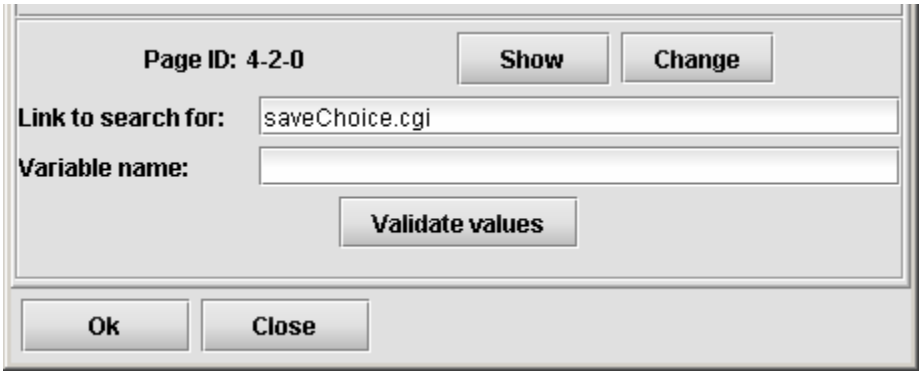
Extracting session information from a URL

The third page, which displays the confirmation, also contains a link to a fourth page (resource ID 4-4-0) that the user can optionally click to change the delivery address. Therefore at runtime StressIT must extract the sessionID from the third page and pass it to the fourth one.

Again, bring up the "HTTP Runtime Parameters" dialog box by double clicking the fourth entry in the web page grid where resource ID is 4-4-0, click the "Request Data" tab and select "Url re-writing" from the combo box.



The bottom part of this screen displays a link and asks you to fill a parameter name.



Type sessionID in the Variable name field and click the "Validate values" button to see if your entries are correct.

Finally, click ok to save your values.

Custom parsing

Although StressIT will able to parse most of the web pages sent by the server, it is possible that in certain circumstances it won't be able to parse these values automatically. One example is parsing Javascripts. In this case, you need to choose custom parsing and specify the exact statement that you want StressIT to search for and pass it to the next request.

For simplicity's sake, we will use the above example to do custom parsing and show to how to extract the same information using this option.

Custom parsing uses Regular Expressions (RegEx for short) to parse the previous HTML page. This syntax for regular expression is similar to Perl If you are familiar with Perl, using RegEx in StressIT should be easy. A small tutorial for Regular Expression can be found in the Appendex. For a more precise description of the behavior of regular expression constructs, please see Mastering Regular Expressions, Jeffrey E. F. Friedl, O'Reilly and Associates, 1997.

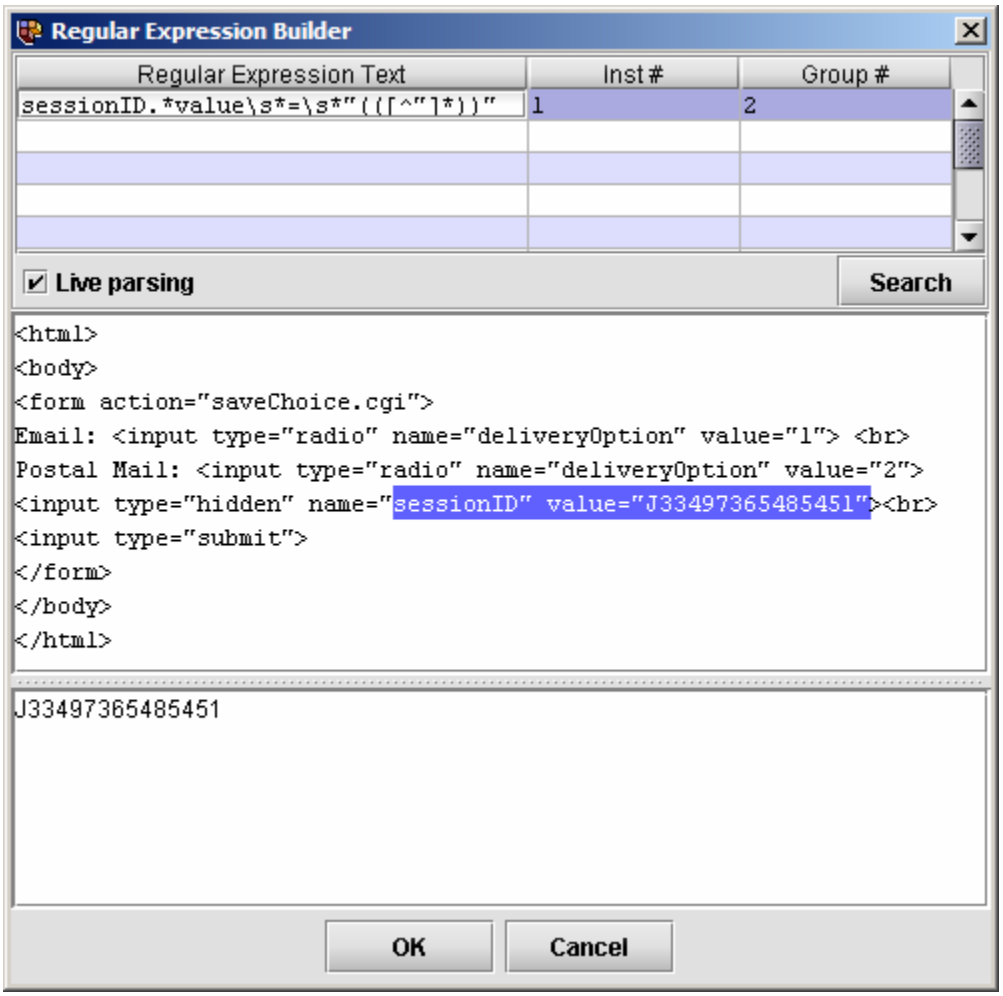
Select "Custom Parsing" from the Replace at Runtime combo box, which will change the bottom part of the dialog box. Notice that you cannot type anything in the "Regular Expression Text". This forces the user to click the "Invoke Regex Builder" button which brings up the Regular Expression Builder in StressIT.

Regular Expression Builder

You specify the Regular Expression in the grid

This text box shows the HTML that is returned as a result of the previous request.

This text box shows the text that is extracted using the regular expression that you specified in the grid.



There are three parts of this screen. The first is where you specify the regular expression text, the second displays the HTML string that will be searched and the last is the result of the search.

The grid where you type the regular expression has three columns:

- Regular Expression Text – This is where you type the regular expression.
- Inst # - This is the instance number of the text found. For example, if there is more than one form you can specify the instance number of the form.
- Group # - This refers to a group number within a regular expression. Grouping is done by enclosing the expression by an open and close parenthesis. This is a very important feature that helps isolate the text that you want to extract.

If you specify more than one line in the grid, StressIT will take the output of the previous line as the input of this line and parse that.

Check the live parsing option to parse the HTML as you type the regular expression. We recommend that you leave this option unchecked when dealing with large HTML text to keep the window responsive. When this option is off, you need to click the "Search" button in order to parse the HTML.

Some examples of Regular Expressions are show below.

Expression:

```
<\s*input\s*type\s*=\s*["*]\s*hidden\s*([^>]*)>
```

Description – Returns one instance of <input> tag that contains a hidden field.

Expression:

```
value\s*=\s*"([[^"]*))
```

Description – Returns the value part of any field. By using this in conjunction with the first example you can extract a value for hidden field.

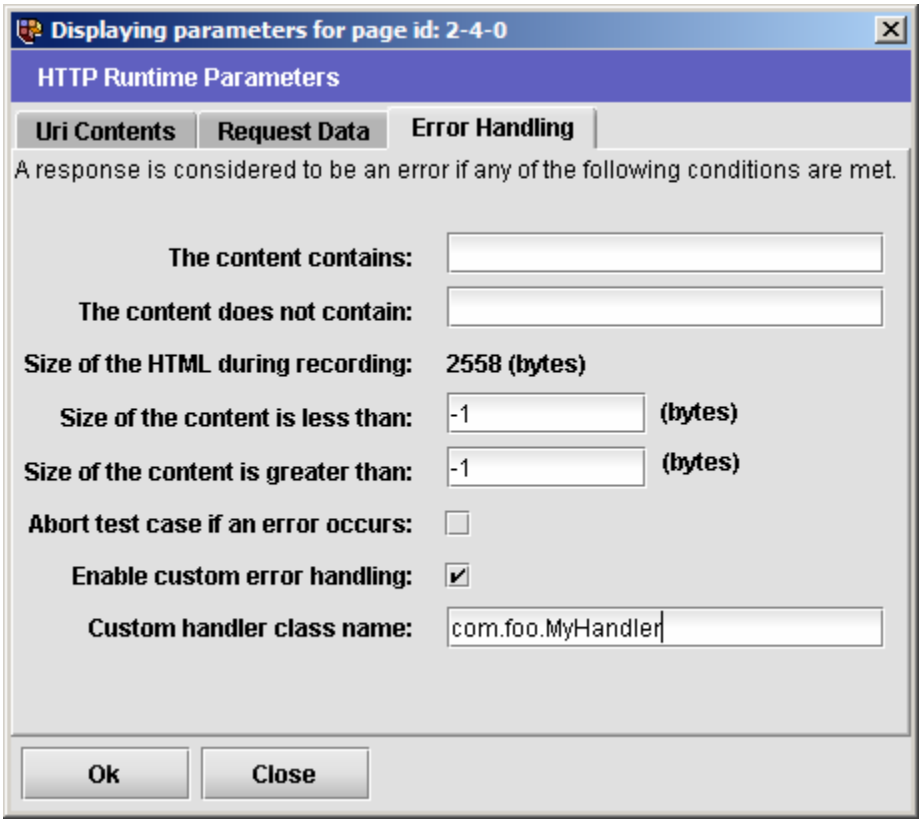
Content Validation and Error Handling

StressIT allows you to specify different criteria for validating the output that is returned by the server. Based on these criteria, StressIT determines if the result should be treated as an error. There are three kinds of errors: Logical Error, Server Error and Network Error. In this chapter we briefly discuss Logical Errors. A more detailed description of errors is available in the following chapters.

A Logical Error is a type of error that fails to meet a certain criteria that is specified during the Capturing phase of StressIT. These criteria are based on three factors:

- Size of the returned HTML
- Content of the returned HTML
- Custom – Invokes an external program that decided if the returned HTML is an error

The following screen displays the third tab of "HTTP Runtime Parameters", which is used to specify the content validation parameters.



All the fields on this screen are explained below.

- **The content contains** – You can type any regular expression in this field. If at runtime the result of this regular expression is found in the content, it will be treated as error. For example, if you using search for the word "Error" in the HTML

- **The content does not contain** – This field also expects a valid regular expression and if the result of this expression is NOT found in the response HTML, it will be treated as an error. For example, if you expect the word "Thank you" in the result, StressIT will flag that as an error if it is not found.

- **Size of the content is less than** – StressIT considers the result to be an error if the size of the HTML is less than what is specified in this field. Specify a -1 to ignore this criterion.

- **Size of the content is greater than** – StressIT considers the result to be an error if the size of the HTML is greater than what is specified in this field. Specify a -1 to ignore this criterion

- **Abort test case if an error occurs** – If this box is checked. StressIT aborts all the remaining web pages in the test case. This useful if the error occurs on a page that is crucial to the program logic. For example, if user authentication page raises an error, it does not make sense to proceed any further.

- **Enable custom error handling** – Check this field if you wish to create a custom error handling mechanism. Please refer to the next section for further details

- **Custom error handler class name** – Name of the class that will handle the error. Please refer to the next section for further details

Custom Error Handling

It is quite possible that in certain complex applications the built-in error handling mechanism of StressIT may not work for you. Consider a scenario when you wish to validate the contents of the returned HTML against entries in a database to determine if the server returned a valid result. In this case you need to create a Custom Error Handler.

A Custom Error Handler is a module that extends StressIT's built-in capabilities. This module must be written in Java and must implement a certain interface. If you are not familiar with the Java programming language, please consult with Synametrics Professional Services to find a solution.

How does it work

A custom error handler is a simple Java class that implements the following interface.

com.synametrics.custom.ICustomErrorHandler

This interface contains only one method: handleError, which is called by StressIT for responses that has custom error handling enabled. The parameter of this method are described in the table below

Table 2

Parameter	Data type	Description
url	String	The url of the current request
queryString	String	This string holds the queryString, which was sent to the server and contains the GET or POST parameters in the following format:

		firstName=Bill&lastName=Clinton
returnedHtml	String	Holds the HTML script that was returned by the server. You can manually parse this string and determine if it should be considered to be an error.
resourceID	String	Resource ID of the URL.
responseCode	Int	This is a 3 digit HTTP response code. If this number is greater then 299, StressIT will treat it as a ServerError. Therefore, you are not required to handle this error. However, you may want still want to perform some customized processing, such as logging the error to a database.

Returned value

The returned value of this method must either be NULL, which means that the custom error handler did not find any errors in the returned HTML or a valid CustomError object. This object holds a description and a flag that specifies if this error should be treated as a fatal error for the test case. Please refer to the Java doc documentation for this object in the Appendix section

EXAMPLE

The following example is a sample Java class that acts as a custom handler.

```
package com.foo.custom;

public class SampleCustomHandler implements ICustomErrorHandler {
    public CustomError handleError(String url, String queryString,
                                   String returnedHtml,
                                   String resourceID,
                                   int responseCode) {

        //You can write the custom code here to determine if the
        //returnedHtml string should be treated as error.
        //This sample simply searches for the word "ERROR"
        //in the returnedHtml. If it is found the handler treats
        //it as an error.
        if(responseCode > 299 || //If the repsonse code is > then 299, StressIT
                                 //treat this as a server error. Therefore, no
                                 //need to parse the result.
           returnedHtml == null ||
           returnedHtml.length() == 0){
            return null;
        }
        if(returnedHtml.toLowerCase().indexOf("error") > 0){
```

```
            CustomError anError =
                new CustomError("The word error was found in the HTML");
            return anError;
        }
        return null; //If this is not an error, the method should return null
    }

}
```

Let's examine the above code. Notice that there is only one method, which returns null if the http status code is greater then 299, the html string is blank or there is no error found.

For example's sake, this method searches for the word "error" in the text. A real world handler will most likely do much more complex parsing. If the returned HTML is considered to be an error, the method creates a new CustomError object and returns it.

Caution:

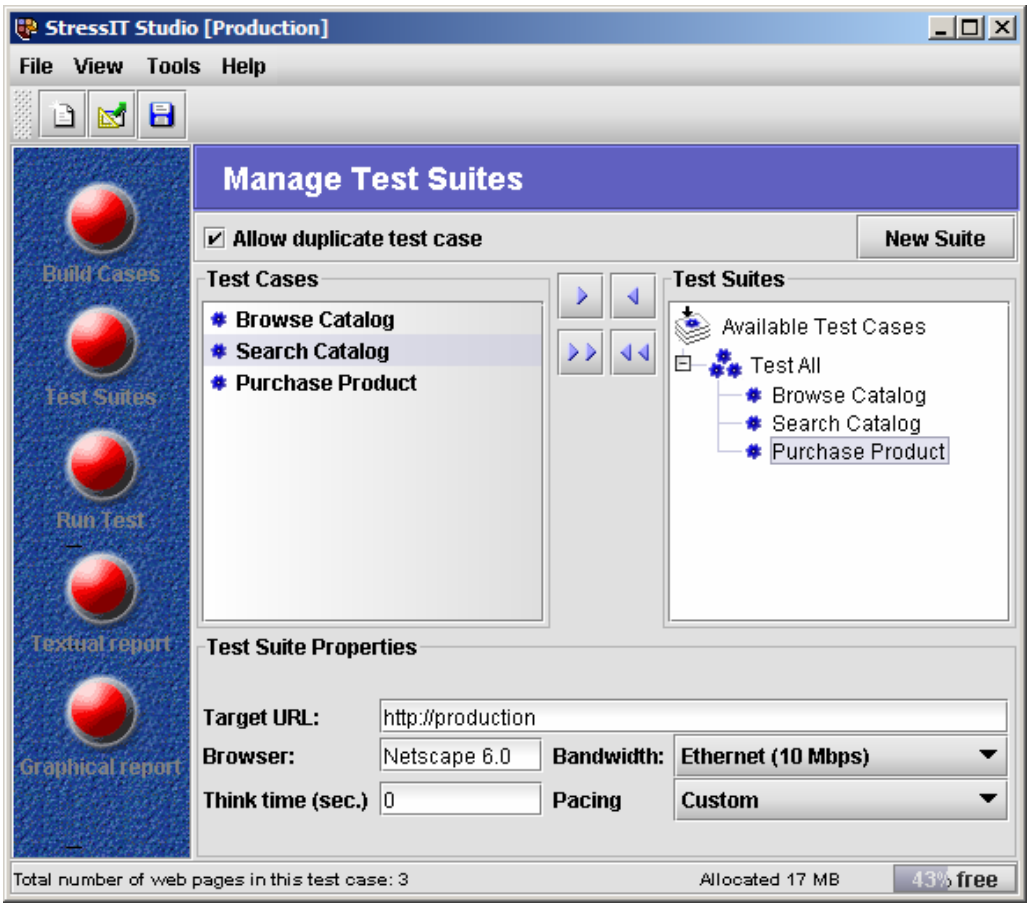
This method will get called from multiple threads. Therefore, it is important that this method is thread-safe. Moreover, if this method takes a long time for processing it may affect the overall testing time. In general this method should not take more than 100 milliseconds.

Chapter 5

Creating Test Suite

A Test Suites is a collection of Test Cases and a unit for work.

Once you create appropriate Test Cases in StressIT, you need to group them in a Test Suite. Click on the "Test Suite" button on the left or select "Manage Test Suites" from the main menu.



Test Suite is a collection of test cases. There is no limit on the number of test cases per test suite. The test cases you have defined in a project are listed on the left hand side; the test suites are on the right.

Click on New Suite, and enter a name for the test suite in the dialog box that appears. Then click on the test cases you want to be a part of the test suite, and either double click or select the right arrow button. To remove a test case from a test suite, click on the left arrow button.

When you create a new test suite, the test suite property section appears. A test suite has a number of properties, summarized in the table below.

Table 3

Field Name	Description
Host Name	This field is a convenient way of changing the host name for all the test cases. It is useful if you create your test cases against a development machine and later want to switch to another host. This way the original test cases remain intact but the tests run

	against the new host. For instance, inserting a value of "http://www.xyz.com" (without quotes) will route all the test requests to xyz.com Another use for this field is to change the port or protocol. For instance, a value of https:// will route all the calls using SSL.
Browser	Some of the server programming logic depends on what browser is being used. For example, if the user's browser is Netscape (rather than IE), you may want to display a Netscape-friendly screen. This field accepts a value for HTTP USER-AGENT. The most common values for this field are: • Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) • Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.0.1) Gecko/20020823 Netscape/7.0
Pacing	This field defines how a virtual user will pace the requests. If you select "As recorded", a virtual user will send messages to the server at the pace at which web pages are recorded. If custom is specified, the speed depends on the value of "Think time"
Think time	This is the time each virtual user spends between two consecutive requests specified in seconds. For instance if you specify 10 for this field, every web request will be sent 10 seconds apart. Notice that this does not apply to Dependent resources, which means that if there are any images or Javascript tags, they will not get affected by this value.
Bandwidth	StressIT simulates modem throughput. This allows you to simulate any bandwidth from 14.4 modem to Token ring, which can be up to 10 Mbps. Setting a smaller value for this field will leave a client connected to the server for longer time, affecting the concurrent connection a web server can handle. This field is only useful if the size of the returned data is larger than one network packet, which is usually 1.2 kb. This setting can be overridden by the value specified for a VTS.

Chapter 6

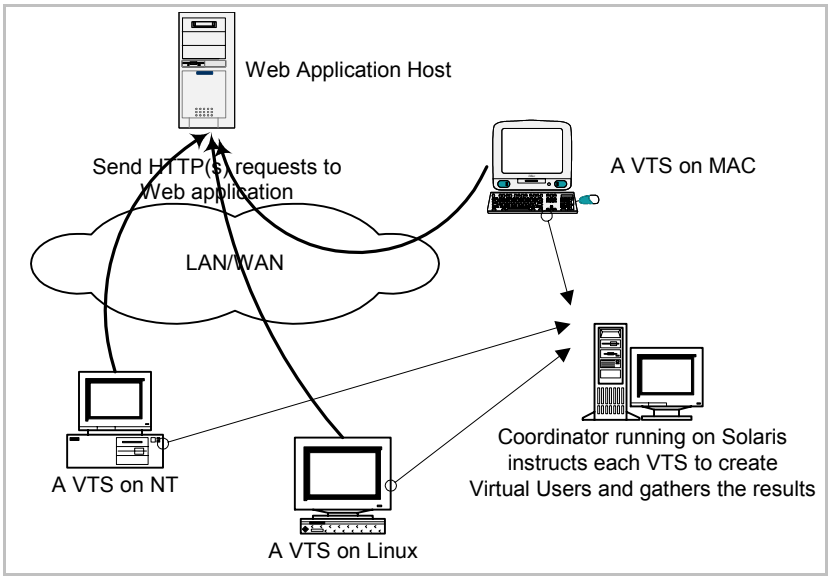
Phase II – Testing Phase

One you have satisfactorily captured what you want to record, you are ready to send the captured request to the server simulating users numbering from the hundreds to the hundreds of thousands..

Preparing for the Testing phase

Running test to simulate thousands of users is a memory and CPU intensive task. Quite often if you run these tests on a low-end machine, the client computer runs out of memory and/or CPU cycles before the server reaches its limit.

Therefore, StressIT allows you to run the actual tests from multiple machines. For instance if your server machine is a heavy duty E15K from Sun Microsystems, you will need quite a few desktop machines to stress the server.



Every machine that participates in running a test is called a Virtual Test Site (VTS). All VTS machines are connected to a centralized control center called the Coordinator.

Preparing a VTS

This section assumes that you have successfully installed VTS on necessary machines.

Since a running VTS is a memory, CPU and network-intensive operation, you should make sure that there are no background processes running on the machine, particularly processes that communicate through the same network channel.

Tip: The StressIT Studio has a built-in VTS, which can be disabled during testing

Memory usage

By default every VTS is configured for a maximum of 512 MB ram. In certain cases, when you want to run the tests for a longer period of time, you may want to increase this limit. The maximum amount of memory you can allocate to a VTS is given below.

Solaris	4 GB
Windows NT	2 GB
Linux	2 GB
Mac	2 GB
Other Platforms	Check JDK documentation

In order to increase the memory usage, you need to modify the vts.bat (used on Windows) or vts.sh (used on Unix) file and change the -Xmx512m parameter. For example, in order to modify the maximum memory to 1 GB, change this parameter to –Xm1g.

Preparing the Coordinator

The Coordinator is a built-in module in StressIT studio. You do not have to run any other process. When you start StressIT studio, the coordinator automatically starts in the background.

Memory usage

By default every coordinator is configured for a maximum of 1GB ram. In certain cases, when you want to run the tests for a longer period of time with many virtual users, you may want to increase this limit. The maximum amount of memory you can allocate for coordinator is given below.

Solaris	4 GB
Windows NT	2 GB
Linux	2 GB

Mac	2 GB
Other Platforms	Check JDK documentation

In order to increase the memory usage, you need to modify the vts.bat (used on Windows) or vts.sh (used on Unix) file and change the -Xmx512m parameter. For example, in order to modify the maximum memory to 1 GB, change this parameter to –Xm1g.

Performing test

To start testing, click the "Run Test" button or select "Testing console" from the main menu. This brings up the testing console window.

On the left you see a list of Test Suites that are defined in the project. Remember, you can only run Test Suites, not Test Cases.

Tip: You can only run Test Suites not Test Cases

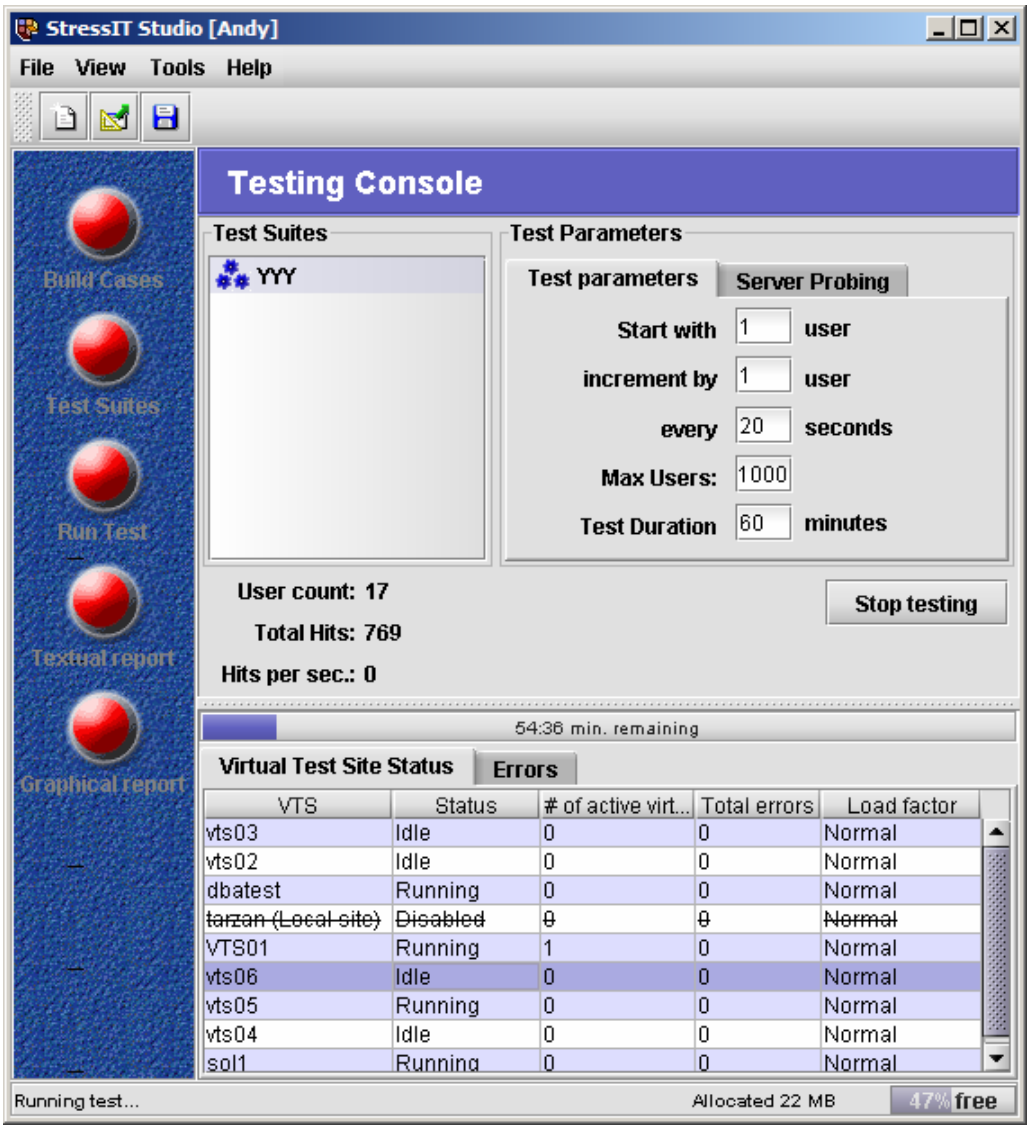
Select the appropriate test case that you want to test and specify an appropriate value for the remaining fields. The first column of the fields that appear in the Test Parameters should be read like a sentence. For instance:

Start with 10 users, increment by 2 users every 30 seconds.

The remaining fields on the screen are described in the table below.

Table 4

Field	Description
Max Users	This is the maximum number of Users StressIT will attempt to create. You can specify any number in this field up to a number that is specified in your license file. If you have purchased StressIT for unlimited virtual users, there is no limit to what you put here.
Test Duration	Duration of the test specified in minutes. If you need to specify a number in hours, you need to multiply that by 60 to achieve a number in minutes. The duration of the test depends on your testing goals.



The above screen shows the testing console.

We recommend that you start with 1 user and increment the number of users as you go along.

Server Probing

When you run a test with thousands of virtual users, it is important to find out how the web/app server behaves – that is how much memory is being used, what is the size of the database connection pool, etc. Server Probing is a mechanism in StressIT that allow you to capture different statistics on your web and application server.

Server probing is achieved by implementing a customized program on the web server, which is usually a JSP, ASP, PHP or similar script. StressIT sends a request to the web server sending the URL for this script, which then calculates different parameters within the server and sends the result back to StressIT.

The scripts that run on the server must send a response back in a particular format. The rules are listed below

- Every line in the HTML script that starts with "$PV_" is considered to hold a value
- Any string that is after "$PV_" and before the equal sign is considered to be a variable. This means that variable names can have spaces as well as punctuation characters.
- Any string after the equal sign on the same line is considered to be a value of the variable
- There is no limit on the number of variables a page can have
- Although the value part can be an alpha numeric values, StressIT includes those fields in the graph where the value part is only numeric.

EXAMPLE

Assume that you are running your web application in a J2EE-based application server. The Servlet container resides on the web tier and EJBs are hosted on the application server tier.

You can write a simple JSP file on the web tier that calculates the following information and sends it back to StressIT.

- Memory usage on the web tier
- Memory usage on the EJB tier
- Size of the database connection pool
- Any business specific parameter such as transaction queues.

Here is a sample HTML that is returned by this script running on the server.

```
<html>
<body bgcolor="#ffffff">

$PV Memory Used Web Tier(KB)=16646
$PV Memory Used EJB Tier (KB)=512187
$PV_Connection Pool Size=10
$PV_Loan Transaction Queue=24

</body>
</html>
```

Any line in the HTML script that starts with "$PV_" is considered a line that holds a name-value pair. In the above example there are four name-value pairs defined:

- Total Memory (KB) – 16646. Shows 16 MB used on Web Server
- Free Memory (KB) – 512187. Shows 512 MB used on the Application Server
- Connection Pool Size – 10. Shows size of the database connection pool
- Loan Transaction Queue – 24. This is a business specific parameter

Every time StressIT sends a request for this script, the web server returns a different value for these variables, which are stored by StressIT and can be used for analysis once testing is over.

The appendix section of this guide includes some sample scripts that you can use with a J2EE server with .NET.

Important: You can only run 1 VTS per machine

Starting a VTS

Although not required, it is recommended that you start the Coordinator before you start a VTS. You can start a VTS by running vts.bat file on Windows platform or vts.sh on a UNIX platform.

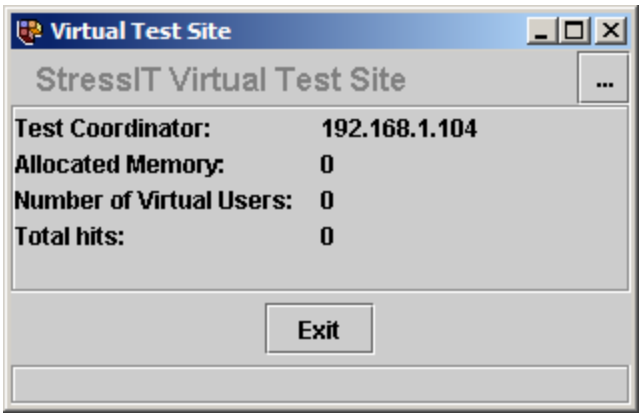
VTS can be run in two modes:

- Graphical interface
- Console based

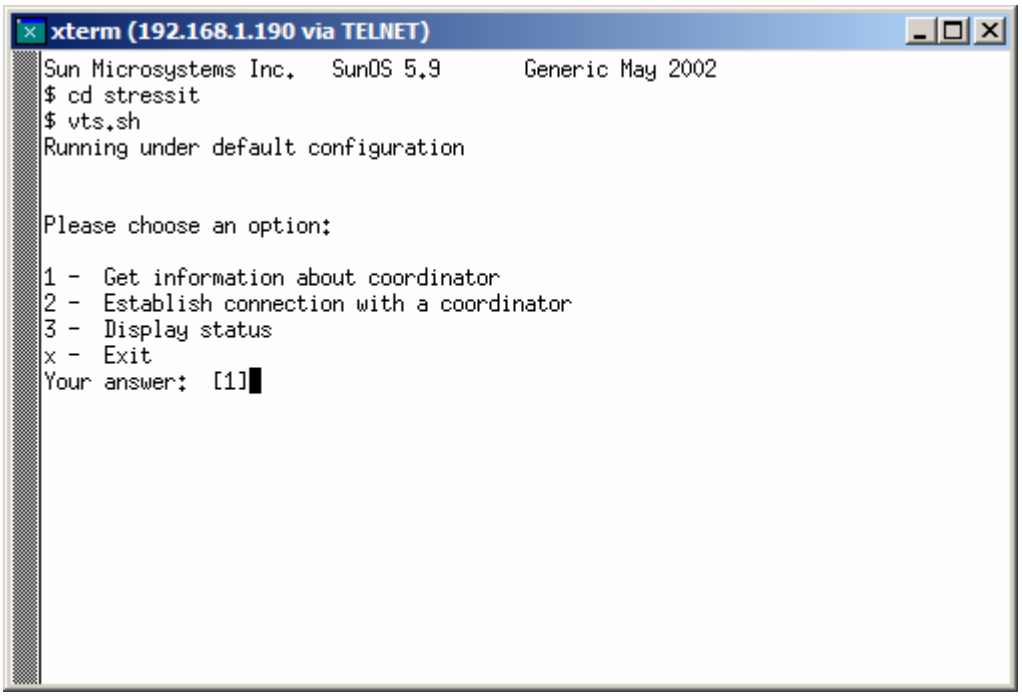
When VTS is first started it asks the user if he/she want to start VTS in graphical or console mode. This question can be avoided by putting a command line argument in

vts.bat or vts.sh file. Put a –c for console or a –g from GUI in the file at the end of the command.

Following screen is displayed in the Graphical mode.



In console mode the screen looks like:



Which mode is best for you to run? Well, this depends on your need and preferences. In general you should run VTS in console mode if:

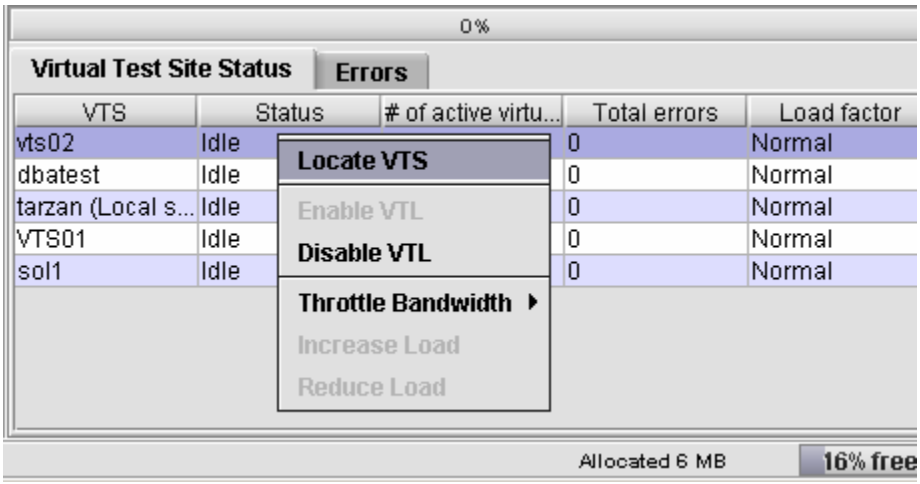
- You are running on a remote UNIX machine and X-Windows is not available
- You want to conserve memory for the real test. Although the difference in memory usage for the graphical mode and console mode is a very small when compared to the actual work a VTS does, it can help a little. Beside, in console mode the VTS will use fewer threads to run.

One disadvantage in running the VTS in console mode is that all information messagages are sent to only one target, producing a cluttered screen. However, in GUI mode you can redirect the information messages to a file.

Connecting the Coordinator with VTS

When VTS is loaded into memory, it checks for the existence of a coordinator on the network, and if one is available it will connect to it. This is why we recommend that you start the coordinator before a VTS. This way you do not have to make an explicit connection. However, this automatic connection will not work if the coordinator is running on a different subnet on your network or on the Internet.

When a VTS establishes connection with a coordinator, you see it listed in the "Virtual Test Site Status" grid.



If, however, a VTS is started before the coordinator, you will have to manually connect to it. This is done by selecting the "Locate VTS" from the pop-up menu.

Local VTS

The Coordinator has a built-in VTS that is denoted by a string "(Local site)" appended to its name in the grid. If you have enough VTSs running on other machines, we recommend that you disable the local VTS. This leaves the coordinator with only two

jobs when the test is run: send commands that gather information from other VTS on the network and send probe messages to the web/application server.

Enable/Disable VTS

You can enable or disable any VTS from the coordinator, which done by selecting the appropriate value from the pop-up menu. IMPORANT: You cannot enable a VTS once a test begins. You can, however, disable it during the test. This is useful if you see a VTS is under very heavy load or is generating too many errors. When a VTS is disabled, you will see its name crossed out in the VTS status grid.

Throttle bandwidth

You can configure each VTS to use a different bandwidth. This simulates a real-world scenario where some users use a higher bandwidth connection than others who come over a slower connection. Select the desired VTS from the grid, click the right mouse button to bring up the pop-up menu, and change the bandwidth.

A lower bandwidth leaves the TCP/IP connection between client and server open for a longer period of time, which may affect the total number of concurrent connections served by the web server.

Increasing/decreasing load

Although the coordinator decides which VTS should get the next virtual user, you may want to manually decrease or increase the load. You would want to increase the load on a particular VTS if it is running on a bigger machine with faster network access then others. You can change the load factor for any VTS while the test is running.

Start Testing

Once you are satisfied with the testing options, click the "Begin Test" button on the Testing console.

Preparing VTS

When a user clicks the Begin Test button, the coordinator sends a test package to every VTS. This is called the preparation stage. This package includes: the test suite which is being tested, and a copy of any datasets that are required. Note that the actual datasets resides on the coordinator machine. Therefore, you do not need to copy any datasets on the VTS machine. Coordinator will take care of the data transfer automatically.

Once every VTS is prepared, the coordinator instructs the VTS to create a user. The Coordinator keeps track of the load each VTS has and how many virtual users are running at a given time in every VTS. Based on this information, it decides which VTS should create the next virtual user.

Birth certificate

Each virtual user that is created during runtime is issued a birth certificate, which contains:

- Birth Time with respect to the beginning of the test
- The name of the VTS a particular user is being created
- Total number of virtual user in that VTS at the time of creation
- A unique identifier, which is a number starting with 1

This birth certificate is unique to a virtual user and used in the analysis phase. Ironically, no virtual user is issued a death certificate ☹

Monitoring test progress

Besides sending requests to the web server, every VTS sends a status message to the coordinator in a timely fashion. This status message includes the number of active users, error count and total hits. The Coordinator displays a summary of the status on its screen.

A test is stopped when: a) the duration of the test is over, and b) if the user manually stops it. Regardless of how the test is stopped, at the end of testing each VTS sends the result back to the VTS. If there are many VTSs running on the system and each has many hits, it can take a while to collect the data from each VTS. StressIT, by default, compresses the data that is sent back to the Coordinator. Although compression and decompression take some CPU cycles, this feature saves a considerable amount of network bandwidth.

The Virtual Test Site Status grid containts the following columns

Table 5

Column Name	Description
VTS	Name of the virtual test site that is participating the test
Status	Show the state of the VTS. The possible values are: Idle – When no test is running Running – Signifies that a VTS has created virtual users and is send requests Disabled – Signifies that a VTS is disabled and all the virtual users are sleeping

	Error – This occurs in case of an error.
VU Count	Shows the number of active users in the VTS at a particular time
Total Error	Number of errors encountered by the VTS
Load Factor	This number is used by the Coordinator to determine which VTS should get the next virtual user. It can change two ways: 1 – The user changes the load factor manually 2 – The VTS sends a message notifying that is too busy and unable to cope with the load.

The Errors grid has the following columns

Table 6

Field Name	Description
VTS	Virtual test site
Time	Timestamp of the error with respect to the beginning of the test
Error Type	Type of error. Possible values are: Logical, Server, and Network
ID	Resource ID of the URL that generated the eror
URL	URL string
Error message	The actual message

A VTS continuously sends status messages to the coordinator. However, error messages are not sent to the coordinator until the test is over. This is done to minimize the communication between the coordinator and a VTS. Remember, the objective here is to test the web server, not the VTS or the coordinator machine.

Double click the error grid to retrieve a detailed error message.

Error messages

There are three types of errors that can occur during a test:

- Logical – This error occur due to a failure in a rule. For instance, during the capturing phase you specified that the result of a particular URL should be greater than a certain value. If this condition is not met, StressIT treats it as a logical error. Besides, all errors that are flagged by a custom error handler are treated as logical errors.

- Server – This error occurs if the web server returns an error code, for example a 404 for file not found. Please refer to the Appendix for details on HTTP error codes.

- Network – This kind of error occurs due to a network problem. For instance, when the web server is not accepting any client connection or it is not found.

Chapter 7

Test Analysis and Reporting

One a test is complete you are ready to analyze the results. This chapter defines what to look for in the reports and defines a description of every report.

StressIT can display the data in two ways: Textually and Graphically. Every report can be drilled down by a VTS name and Test Case, which helps you identify the worst test cases in your system.

Before we define a description of each report in StressIT, we would like you to get familiar with some terminology that is being used in StressIT.

Response Time (RT)
Response time is defined as the time between the last byte of request and first byte of response. This is the most important factor when measuring a server's scalability.

Throughput (TP)
This is the number of bytes transferred in one second

Iteration
Every virtual user runs a test suite starting from the first test case that is found. However, if the duration of the test is longer, it is possible that a particular virtual will finish running all the test cases in a test suite, which is also called an iteration. Once an iteration is complete, the virtual user will again start from the beginning of the test suite. This time, however, it will begin with a different iteration number.

Transaction
A test case at runtime is called a transaction, which contains a number of web requests.

Textual Reports

This section discusses the available reports in StressIT and explains how to interpret them. .

Worst Case Scenario

This report has two parts, each represented by a grid control. The top grid control shows the degradation in performance as the number of users is increased. This helps you identify the upper limit on the number of users.

The columns in the first grid are:

Table 7

Field Name	Description
# of users	Total number of Users found in the VTS
Avg. RT	Average RT
Worst RT	Worst Response Time
# of hits	Number of hits by this VTS

The URL grid on the other hand displays the worst 20 URLs in the system. Notice that the text of a URL may appear more than once. However, the resource ID should only appear once. The columns in the URL grid are:

Table 8

Field Name	Description
ID	Resource ID of the URL
URL	The URL string for the request
Worst RT	Worst Response time
Avg RT	Average Response time
BestRT	Best Response time
Best TP	Best Throughput
Worst TP	Worst Throughput
Avg TP	Average Throughput
# of hits	Number of hits for this particular URL

Http Error Report

This report also displays two grid controls. The first grid display a summary of errors that occurred and the second grid displays the error in detail. The following table shows the field description:

Table 9

Field Name	Description
ID	Resource ID
URL	URL of resource
Test Case Aborted	Shows if the test case was aborted due to the error. A test case is aborted if during the capture phase the user selects the "Abort test case if an error occurs" check box in the Error Handling tab in the HTTP Runtime Parameters windows
Total Errors	Total error for this resource ID

The following table explains the fields in the Error Detail Grid.

Tip: Double-click the grid to bring up a window that shows further details about the error message

Table 10

Field Name	Description
VTS	Name of the virtual test site
Time	Timestamp when error occurred. This is with respect to the beginning of the test
Error Type	Type of the error
ID	Resource ID
URL	URL of the resource
Error Message	A description of the error message

Graphical Reports

Click the "Graphical Report" button in the side panel or "View Graphical Reports" from the view menu.

There are several reports that come built-in with StressIT . Their descriptions are given below.

Performance degradation by users per iteration

This report plots the performance degradation of the first iteration of every virtual user. It summarizes the response time for every request in the first iteration and displays the result in the graph. The vertical axis holds the time in seconds it took for the user and the horizontal axis contains the number of users. The fields in the grid are explained below

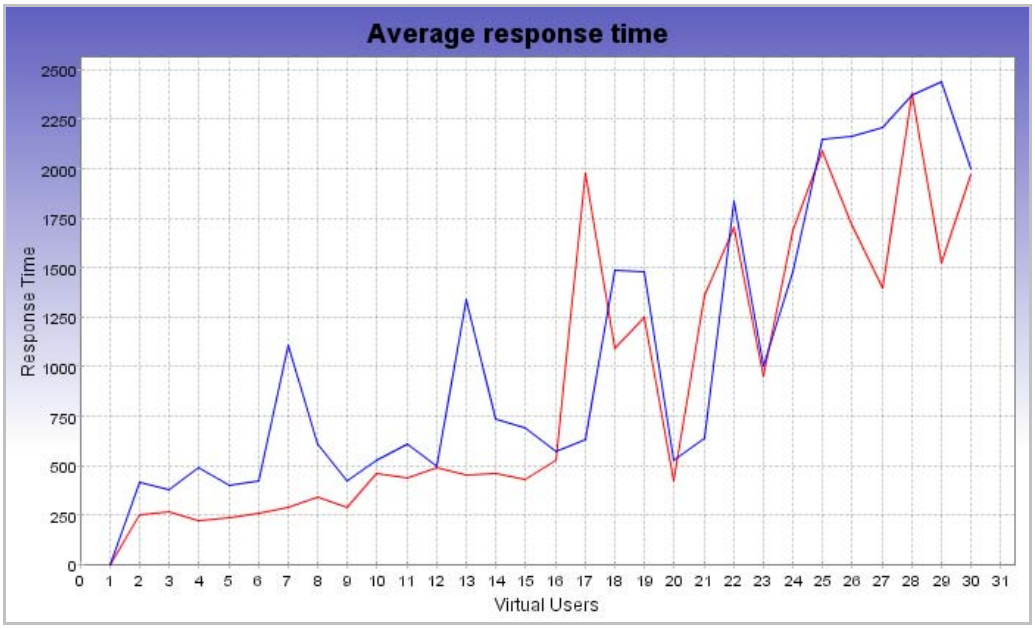
Table

Field Name	Description
Virtual User	Label for virtual user
Response time	The sum of response time for the first iteration
Completed	Show Yes if the iteration was completed and No if the test was stopped before the iteration could complete
Error	Total number of errors in the report

Average response time by virtual users

This report displays the response time of the worst URLs. It prompts a user to enter a number, which represents the number of URLs to display. For instance, if the user chooses 2, the 2 worst URLs (unique resource id) will be displayed.

This reports plots the response time on the vertical axis and virtual users on the horizontal. There can be more than one request for every resource ID. This report computes the average response time as the number of virtual user increases.

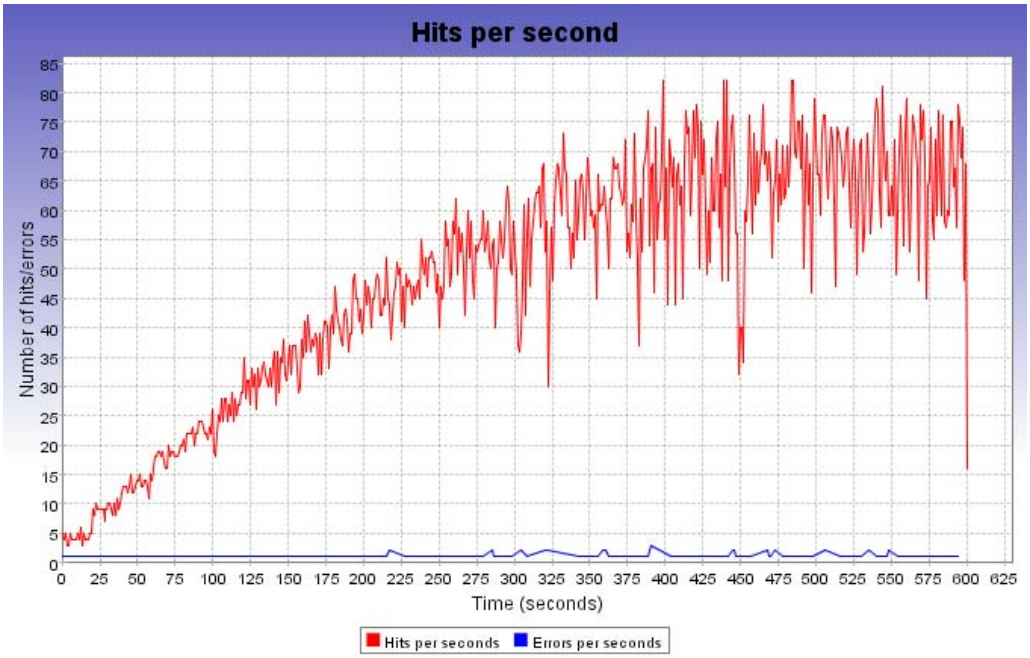


Worst response time by virtual users

This report displays similar information as the previous report, except it shows all the requests rather than an average.

Hits per second

This report displays the hit per second ratio graphically. You can see that as time passes, the number of hits increases



Exporting results to a database

StressIT allows you to export the result of the test to a relational database, which gives you the capability to create custom reports using a reporting tool of your choice. Supported databases are:

- IBM DB2/UDB
- Oracle
- MS SQL Server
- Sybase
- Informix
- MS Access
- PostgreSQL
- MySQL

If you are using any other database other than shown onthe list above, you can still use StressIT to export its results. However, you will need to modify the genericDB.xml file in the config directory. Change this file and replace the SQL scripts with the correct data types.

Report Export Wizard

Select Tools → Export to Database to bring up the Report Export Wizard

JDBC Driver – This is the class name of the JDBC driver. Select the appropriate driver from the list

Connection URL – This is the connection string for the database, which is specific to a database. Please refer to your database documentation for correct syntax.

User ID – Login ID for the database. Ask your DBA if you don't have one

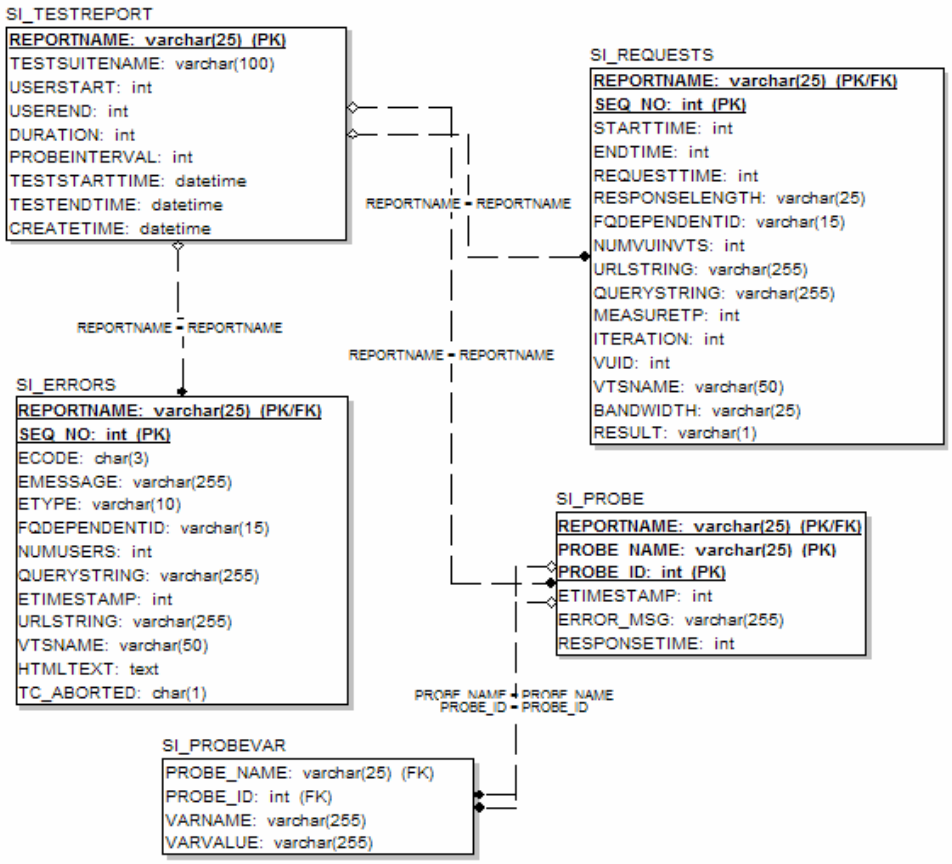
Password – User password. Ask your DBA if you don't have one

Report Id – This is a user-defined value and is used as a primary key in the tables. Use a different value every time you export the data

Database Type – Select the appropriate target database.

Database Design

Notice that you do not have to create necessary tables in the database. StressIT will do that for you. However, it is important to know the schema of these tables so that you can create the reports easily. The entity/relationship diagram of the tables is given below.



The following table describes each field and its purpose

SI_TESTREPORT table

Table

Field Name	Description
ReportName	This is the name of the report, which you specified in the Report Id column. This name serves as a primary key and is used as a foreign key in other tables

TestSuiteName	Name of the test suite this report is based on
UserStart	Number of virtual user when testing began
UserEnd	Number of virtual user when testing ended
Duration	Duration of the test
ProbeInterval	Probe Interval. -1 if probing is not enabled
TestStartTime	Timestamp when test started
TestEndTime	Timestamp when test stopped
CreateTime	Creation time of the report

SI_REQUESTS

This table holds the HTTP requests that were sent. Each row represents a request

Table

Field Name	Description
ReportName	Name of the report. This is a foreign key referring to SI_TESTREPORT table.
SEQ_NO	An integer. This field along with ReportName constitutes a primary key of the table. This field has no other significance besides being part of the primary key
StartTime	This is the time at which a virtual user was finished sending the last byte of the request.. The unit of measurement is millisecond
EndTime	This is the time when a virtual user received the first byte of the response. Response time is calculated by subtracting EndTime from StartTime. These numbers are specified in milliseconds.
RequestTime	This is the time when a virtual user started preparing for a request. There is usually a small time difference between RequestTime and StartTime. This is the time in which a virtual user prepares the HTTP request, assigns any necessary

	parameters either from a dataset or previous page.
ReponseLength	Length of the response
FqDependentId	Resource ID, which is also known as a fully qualified dependent id
NumVuInVts	Number of virtual users in the VTS at the time of request
URLString	The URL string for this request. Although this information can be retrieved from the resource id, this information is given here for convenience
QueryString	This is the actual query string that was sent to the server. Query string can be different for each request depending on the configuration in the capturing phase
MeadureTP	Measured throughput. Throughput is defined as the number of bytes of data transferred per second.
Iteration	Iteration number of the request
VUId	Refers to the birth certificate ID of the user
VTSName	The VTS in from which this request was sent
Bandwidth	Bandwidth of the request
Result	This can either be: 0 – For success 1 – Error 2 – Fatal Error, which caused a test case to abort

SI_ERRORS

This table holds error information. Each record in the table represents an error.

Table

Field Name	Description
ReportName	Name of the report. This is a foreign key referring to

	SI_TESTREPORT table.
SEQ_NO	An integer. This field, along with ReportName, constitutes a primary key of the table. This field has no other significance besides being part of the primary key
ECode	Error code. This is a 3 digit response code from the web server.
EMessage	This holds a description of the error message
EType	Type of error. There are three types of errors: Logical, Server & Network
FQDependendID	Resource ID
NumUsers	Number of users in the VTS when this error occurred
QueryString	Query string for the request. Depending on how you configured a test case, this could be a different value for every request.
ETimeStamp	The timestamp during which this error occurred
URLString	URL String.
VTSName	Name of the VTS
HTMLText	The HTML that was returned by the server
TC_Aborted	This holds either 'Y' or 'N'. A 'Y' means that this error caused the test case to abort

SI_VUBIRTH

This table holds the birth certificate information for every virtual user

Table

Field Name	Description
Report Name	Name of the report. This is a foreign key referring to SI_TESTREPORT table.

SerialNum	Serial number of the virtual user, which uniquely identifies a virtual user in the test.
SiblilngCount	Number of virtual users that existed in the VTS when this was created
CreateTime	Timestamp when this virtual user was created
VTSName	Name of the VTS where this user was created

SI_PROBE

This table holds the results of Probes that are sent to the server. Each record in this table signifies one probe request.

Table

Field Name	Description
Report Name	Name of the report. This is a foreign key referring to SI_TESTREPORT table.
Probe_Name	Name of the Probe as specified in StressIT Studio
Probe_ID	This is a sequential number which produces a primary key when combined with Probe name.
ETimeStamp	Timestamp when this probe request was sent
Error_Msg	Error message if the probe request resulted in an error
ResponseTime	Response time for probe request.

SI_PROBEVAR

The records in this table contains values for probing variables. There is a one to many relationship between SI_PROBE and SI_PROBEVAR tables

Table

Field Name	Description

Probe_Name	Name of the probe. Refers to the Probe name field in SI_PROBE table
Probe_ID	Probe ID. Refers to the Probe_ID field in the SI_PROBE table.
VarName	Name of the variable
VarValue	Value of the variable
VTSName	Name of the VTS where this user was created

Appendix A

HTTP Status Codes

The following is a list of HTTP status codes that are returned by any HTTP server.

Information Code

Code	Description
100	Continue – The client can continue the request to the server.
101	Switching Protocols – The server has changed the application protocol being used on the connection at the request of the client via the Upgrade message header field

Success Code

Code	Description
200	OK – The request completed successfully.
201	Created – The request has been fulfilled and resulted in a new resource
202	Accepted – The request has been accepted for processing, but the processing has not been completed.
203	Non-Authoritative Information – The returned meta information in the entity-header is not the definitive set as available

	from the origin server, but is gathered from a local or a third-party copy.
204	No Content – The server has fulfilled the request but does not need to return new information
205	Reset Content – The server has fulfilled the request and the client should reset the document view which caused the request to be sent to allow the user to initiate input action
206	Partial Content – The server has fulfilled the partial GET request for the resource.

Redirection Codes

Table

Code	Description
300	Multiple Choices – The requested resource corresponds to ambiguous or multiple choices.
301	Moved Permanently – The requested resource has been assigned a new permanent URI and any future references to the resource should use one of the returned URIs
302	Moved Temporarily – The requested resource resides temporarily under a different URI
303	See Other – The response to the request can be found under a different URI and should be retrieved using a GET method on that resource.
304	Not Modified – The client requested a resource that has not been modified
305	Use Proxy – The requested resource must be accessed through the proxy given by

	the Location field.
306	Not Used – Reserved
307	Temporary Redirect – The requested resource resides temporarily under a different URI

Failure Codes

Table

Code	Description
400	Bad Request – The request could not be understood by the server due to invalid syntax
401	Unauthorized – The requested resource requires user authentication
402	Payment Required – Not implemented. Reserved for future use.
403	Forbidden – The server understood the request, but is refusing to fulfill it.
404	Not found – The server has not found anything matching the Request-URI
405	Method Not Allowed – The method in the request-line is not allowed for the resource identified by the request URI
406	Not Acceptable – The response from the request has content characteristics that are not acceptable to the accept headers sent by the client
407	Proxy Authentication Required – The client must first authenticate itself with the proxy.
408	Request Time-out – The client did not produce a request within server timeout

	limit.
409	Conflict – The request could not be completed due to a conflict with the current state of the resource. The user should resolve the conflict and resubmit the request.
410	Gone – The request resource is no longer available at the server and no forwarding address is known.
411	Length Required – The server refuses to accept the request without a defined Content-Length.
412	Precondition Failed – The precondition given in one or more of the request-header fields evaluated False when tested on the server.
413	Request Entity to Large – The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
414	Request-URI Too Long – The server is refusing to service the request because the Request URI is long then the server is willing to interpret.
415	Unsupported Media Type – The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
416	Requested Range Not Satisfied – A request included a Range request-header field, and none of the range-specifier values in this field overlap the current extent of the selected resource, and the request did not include an IfRange request-header field.

417	Expected Failed – The expectation given in an Expect request-header field could not be met by this server, or, if the server is a proxy, the server as unambiguous evidence that the request could not be met by the next-hop server.

Server Error Codes

Table

Code	Description
500	Internal Server Error – The server encountered an unexpected condition that prevented it from fulfilling the request.
501	Not Implemented – The server does not support the functionality required to fulfill the request.
502	Bad Gateway – The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
503	Service Unavailable – The server is currently unable to handle the request due to a temporary overloading or maintenance of the server
504	Gateway Time-out – The server, while acting as a gateway or proxy, did not receive a time out while waiting for a response from the upstream server specified by the URI or some other auxiliary server it needed to access in attempting to complete the request.
505	HTTP Version not supported – The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

Index