

# Configuring User Journeys to Ensure Correct Testing

User's Guide



Version 5.0

## Copyright

Copyright © 2004, 2005, 2006, 2007, 2008 Reflective Solutions Ltd.  
All rights reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms and conditions of the Reflective Solutions Ltd. license agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Reflective Solutions Ltd.

Information in this document is subject to change without notice and does not represent a commitment on the part of Reflective Solutions Ltd.

THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER REFLECTIVE SOLUTIONS LTD. DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY OR OTHERWISE.

## Trademarks or Service Marks

jTune™ and StressTester™ are registered trademarks of Reflective Solutions Ltd.

All other trademarks are the property of their respective companies.

# Contents

Introduction .....	5
User Journey Workspace .....	6
Journey Navigation Tree .....	6
Right-Click Menus .....	7
Quick Glance Icons .....	9
User Journey Properties .....	11
Advanced Settings .....	12
Extra Settings .....	13
Debug .....	13
Step and Step Group Properties .....	15
Advanced Settings .....	17
Headers .....	18
Authentication .....	18
Debug .....	18
Checking Application Responses.....	20
When to Specify Success Strings.....	20
Success String Formats .....	21
Sleep and Simulated Wait Times .....	23
Estimating Sleep Times.....	23
Configuring Sleep Times .....	24
Sleep Time Variation .....	24
Sleep Time Multipliers .....	25
Varying Data Values .....	26
Dynamic Data Introduction .....	27
Auto-Correlated Dynamic Data.....	29
Auto-Increment Dynamic Data .....	30
Constant Dynamic Data .....	32
Date Dynamic Data .....	32
Delimited File Dynamic Data .....	34
Java Class Dynamic Data .....	36
Response Dynamic Data.....	37
Configuring Siphons .....	39
Related Dynamic Data .....	42
Testing Dynamic Data Configurations .....	43
Varying Business Transaction Flow / Step Order.....	45
Introduction to Flow Control.....	46
Go To Flow Control .....	47
Fixed Loop Flow Control .....	47
Variable Loop Flow Control .....	48
Percentage Flow Control.....	48
Response Based Flow Control.....	49
Dynamic Data Loop Flow Control.....	50

## StressTester™ : Configuring User Journeys to be Correct and Realistic

Import / Export a User Journey .....	52
Import a User Journey .....	52
Export a User Journey .....	53

# Introduction

The [Recording HTTP User Journeys](#) user guide describes the process of recording a new User Journey.

However, if the recorded User Journey is simply played back without alteration against the application under test, this would not be *correct testing*.

*Correct testing* ensures that data provided in a business transaction will be varied as it would be between users in real life. Data includes form field values, selections from drop-down lists, checkboxes and similar.

Also, the user “thinking” time between the different steps in a business transaction should be varied to ensure different user speeds are correctly simulated.

Correct testing also considers the varying routes through an application that a user may take – for example not all users on an airline reservation system purchase the flight shown as a result of their first search; they may do many searches, check available seats, etc. before proceeding with a ticket purchase.

Finally, correct testing ensures that all responses from the application are checked to confirm the application processed the request correctly. This is very important as it is relatively common for applications under heavy load to return results that are well formed yet do not contain the expected data.

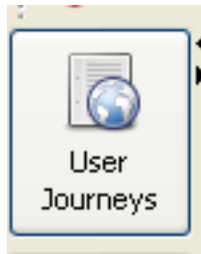
This manual describes how User Journeys can be configured to ensure they are correct in terms of:

- data variation ([Varying Data Values](#) section below)
- loops, forks, and other different routes and variations ([Varying Business Transaction Flow](#))
- timing ([Sleep and Simulated Wait Times](#)), and
- verification of responses ([Checking Application Responses](#)).

It is suggested that before you read this manual, you read the [Quick Start](#) user guide to gain an understanding of the recommended approach to creating correct User Journeys in an efficient manner.

## User Journey Workspace

All tasks related to configuring a User Journey are performed within the User Journey workspace of the StressTester™ user interface.



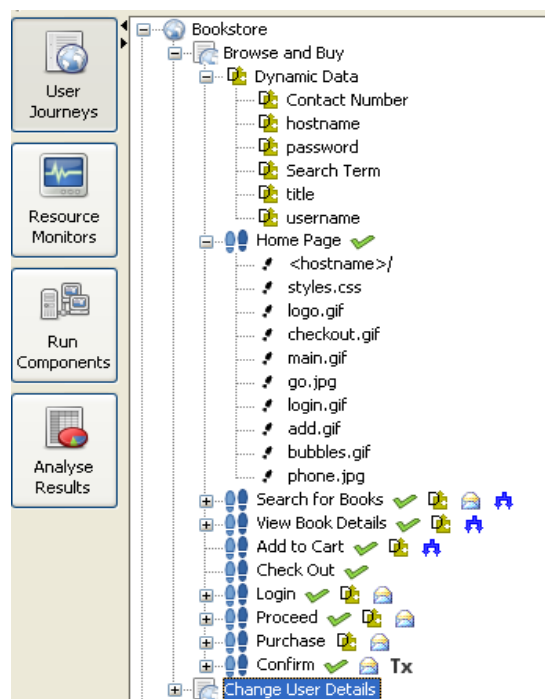
To access this workspace, simply click the “User Journeys” button on the left hand side of the StressTester™ user interface.

## Journey Navigation Tree

When the workspace is entered, the workspace navigation tree and introductory help page are displayed.

**Note:** It is highly recommended that you read the introductory help page the first time you use the workspace – it contains valuable information.

By clicking a node in the navigation tree, the introductory page will be changed to a data panel - allowing the selected node’s information to be viewed and updated (license permitting).




In the screen shot example shown, there are two User Journeys (represented by the globe icon) called “Browse and Buy” and “Change User Details”.


These User Journeys test the Bookstore application (represented by the globe icon).

Within the “Browse and Buy” User Journey there are many step groups (analogous with web pages). Step groups are

represented by the  icon.

Step groups consist of steps – individual requests made from the browser to the server – and these are represented by the  icon.

The other nodes shown within the navigation tree relate to Dynamic Data – the method by which StressTester™ varies the data sent to the application during a performance test.

Dynamic Data items (parameters supplied to the application) are grouped under the “Dynamic Data” node and are represented by the  icon.

## Right-Click Menus

The majority of nodes within the User Journey workspace navigation tree have context sensitive right-click menus – providing easy access to the majority of the StressTester™ user interface’s functionality.

The sections below detail the different functionality available on the different tree nodes.

### Application

Menu Item	Description
Record a User Journey	Same as clicking the “Record User Journey” button on the user interface tool bar.
Create User Journey	Creates a new User Journey; initially with no steps or step groups.
Import User Journey	Imports a User Journey from a previously created export file.  <b>Note:</b> When importing the application using the right-click menu, the User Journey will be imported into the application selected. However, if the User Journey import file has a different application name, you will be warned and asked to confirm you wish to continue.

### User Journey

Menu Item	Description
Insert Step	Inserts a new step at the beginning of the User Journey.
Search/Replace	Provides standard search and replace functionality within a User Journey.
Quick Run	Invokes the Execute Test Run dialogue and run the User Journey for one cycle, with one simulated user, logging all requests and responses.
Export	Exports the selected User Journey to a file.
Delete	Deletes the selected User Journey.  <b>Note:</b> When a User Journey is deleted, all results for performance tests within which the User Journey participated are deleted, and the User Journey is deleted from any Test Run Configurations.

### “Dynamic Data” Node

Menu Item	Description
New	Create a new Dynamic Data item.

### Dynamic Data Item

Menu Item	Description
Delete	Deletes a Dynamic Data item.

### Step Group

Menu Item	Description
Insert	Inserts a new step immediately after the ‘lead’ step of the step group (the step on which the right-click is occurring).
Revert to Steps	Converts the steps within the step group to ‘orphaned’ steps (i.e. not within a step group).

### Step


Menu Item	Description
Insert	Inserts a new step immediately after the selected step.
Delete	<p>Deletes the currently selected step.</p> <p>You can delete multiple steps at the same time by selecting the steps required (using the Shift and Control keys) and then right-clicking the selection.</p> <p><b>Note:</b> You cannot delete a step that participates in a Dynamic Data or Flow Control configuration.</p>
Convert to Step Group	<p>Converts the selected step(s) to a step group.</p> <p><b>Note:</b> This option only appears when steps not already within a step group are selected.</p>







## Quick Glance Icons

Up to eight icons may be displayed after a step's name in the navigation tree.

These icons provide an easy way of identifying the steps that have been configured in some manner following the recording of the User Journey.

The table below shows the icons in the order they may appear and provides a description of each.

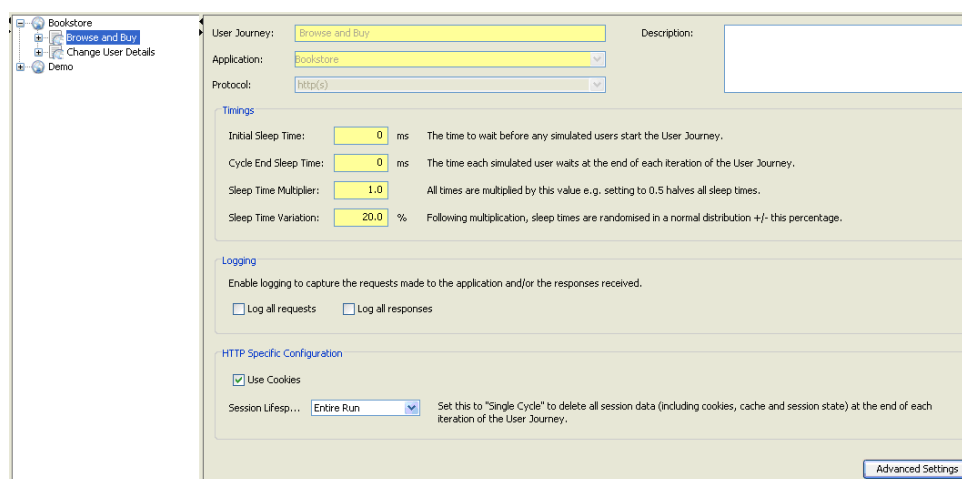
Icon	Description
	<p>Indicates that a success criterion has been specified for the step.</p> <p>Please refer to the section <a href="#">Checking Application Responses</a> section below for more details on specifying success criteria.</p>

Icon	Description
	<p>Indicates that the step contains Dynamic Data items.</p> <p>Please refer to the section <a href="#">Varying Data Values</a> below for more details about Dynamic Data.</p> <p>This icon will not display if the step only contains items that are listed in the exclusion file – <code>&lt;stresstester&gt;\properties\dynamicdataicon.exclude</code>.</p> <p>This file ensures that the tree is not cluttered for common items such as <code>hostname</code>.</p>
	<p>Indicates that the step is a POST and that POST data has been specified.</p>
	<p>Indicates that Flow Control has been configured for the step.</p> <p>Please refer to the section <a href="#">Varying Business Transaction Flow / Step Order</a> below for more details about Dynamic Data.</p>
	<p>Indicates that this step is configured to be executed only for the first cycle of a performance test.</p>
	<p>Indicates that this step is configured to be executed only for the last cycle of a performance test.</p>
<p><b>Tx</b></p>	<p>Indicates that the step should be counted as a transaction when calculating throughput figures.</p>
	<p>Indicates that StressTester™ is configured to call an external Response Processor program following the execution of the step.</p> <p>Please contact Reflective Solutions Support – <a href="mailto:support@reflective.com">support@reflective.com</a> – for more details about Response Processing.</p>

## User Journey Properties

One of the first things to consider, after ensuring your recorded User Journey replays successfully, is the generic properties for the User Journey.

These are accessed by selecting the User Journey in the navigation tree.



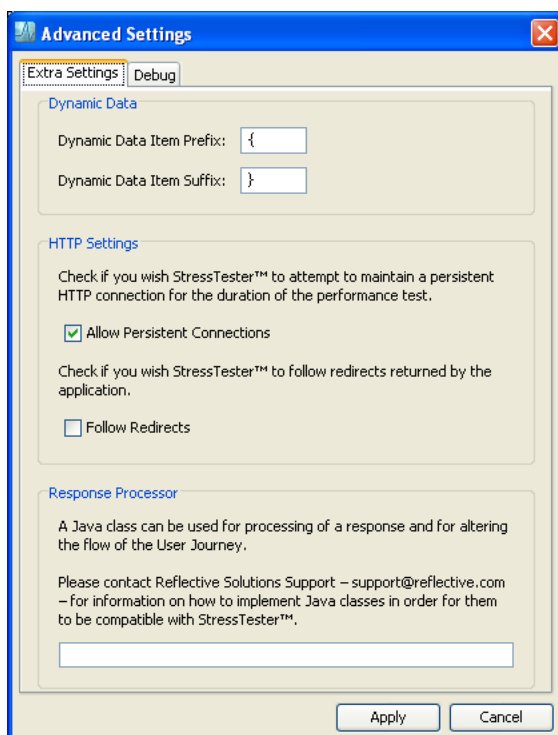
The fields on the screen are as follows:

Field	Description
User Journey Name	The name of the User Journey.
Application Name	The name of the application that the User Journey tests.
Protocol	The protocol StressTester™ uses to communicate with the application.
Description	Free text description of the User Journey.
Initial Sleep Time	See <a href="#">Sleep and Simulated Wait Times</a> section below for details on this field.
Cycle End Sleep Time	See <a href="#">Sleep and Simulated Wait Times</a> section below for details on this field.
Sleep Time Multiplier	See <a href="#">Sleep and Simulated Wait Times</a> section below for details on this field.

Field	Description
Sleep Time Variation	See <a href="#">Sleep and Simulated Wait Times</a> section below for details on this field.
Log all requests	Instructs StressTester™ to save every request made to a file. Each request will be saved in an individual file and will be stored within the StressTester™ Injector's log directory.
Log all responses	The same as above but for every application response.
Use Cookies	Specifies that the simulated users should behave in the same way as a browser with cookies enabled.
Session lifespan	Specifies how long StressTester™ should maintain each simulated user's session – specifically when cookies are deleted and HTTP connections reset.

## Advanced Settings

By clicking the “Advanced Settings” button on the User Journey properties screen, you can access advanced properties.



The fields on the different tabs on this pop-up are described in the sections below.

### Extra Settings

The table below describes the fields on this tab.

Field	Description
Dynamic Data Item Prefix	The character(s) used as the starting delimiter of a Dynamic Data parameter in the User Journey.
Dynamic Data Item Suffix	The character(s) used as the ending delimiter of a Dynamic Data parameter in the User Journey.  <b>Note:</b> The prefix and suffix cannot be the same character(s).
Allow Persistent Connections	Specifies whether StressTester™ should allow persistent connections if the application server requests such during a performance test.
Follow Redirects	Specifies whether StressTester™ should follow redirects if these are returned by the application during a performance test.  <b>Note:</b> This defaults to being disabled as the redirects were probably captured during the recording of the User Journey.
Response Processor	The specification of the external Java class that will be called for every step that has the Process Response field selected.  This is an advanced option and if you need to process application responses, please contact Reflective Solutions Support – <a href="mailto:support@reflective.com">support@reflective.com</a> – for further details.

### Debug

Under the instruction of Reflective Solutions Support, you may be asked to configure debug flags in order that more information about a reported issue can be captured.

## StressTester™ : Configuring User Journeys to be Correct and Realistic

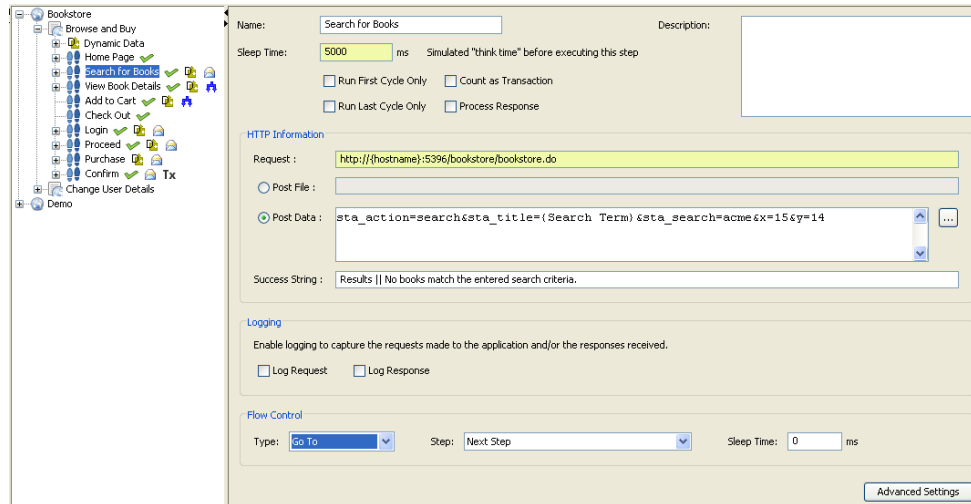
This is the screen where these flags are configured and it should only be used following guidance from Reflective Solutions Support.

When you have updated the User Journey properties, click the “Save” toolbar button to persist the changes.

**Note:** For further details on the fields on this screen, press the Help button when on the screen to access the in-tool context specific help.

## Step and Step Group Properties

When you select a step within the navigation tree (whether it be one that starts a step group or just a normal step), StressTester™ will display that step's details.



A step group is a set of steps that make up one unit of work – typically this is analogous to a web page for web applications.

However, StressTester™ provides the flexibility for you to group together application requests as you like, in order that you can obtain the information you need from your performance testing.

To change a set of selected “orphaned” steps into a step group, select the steps and then right-click in the navigation tree and choose the “Convert to Step Group” option.

To split a step group into individual steps belonging to no step group, simply right-click on the step group in the navigation tree and choose the “Revert to Steps” option.

The fields on the step screen are as follows:

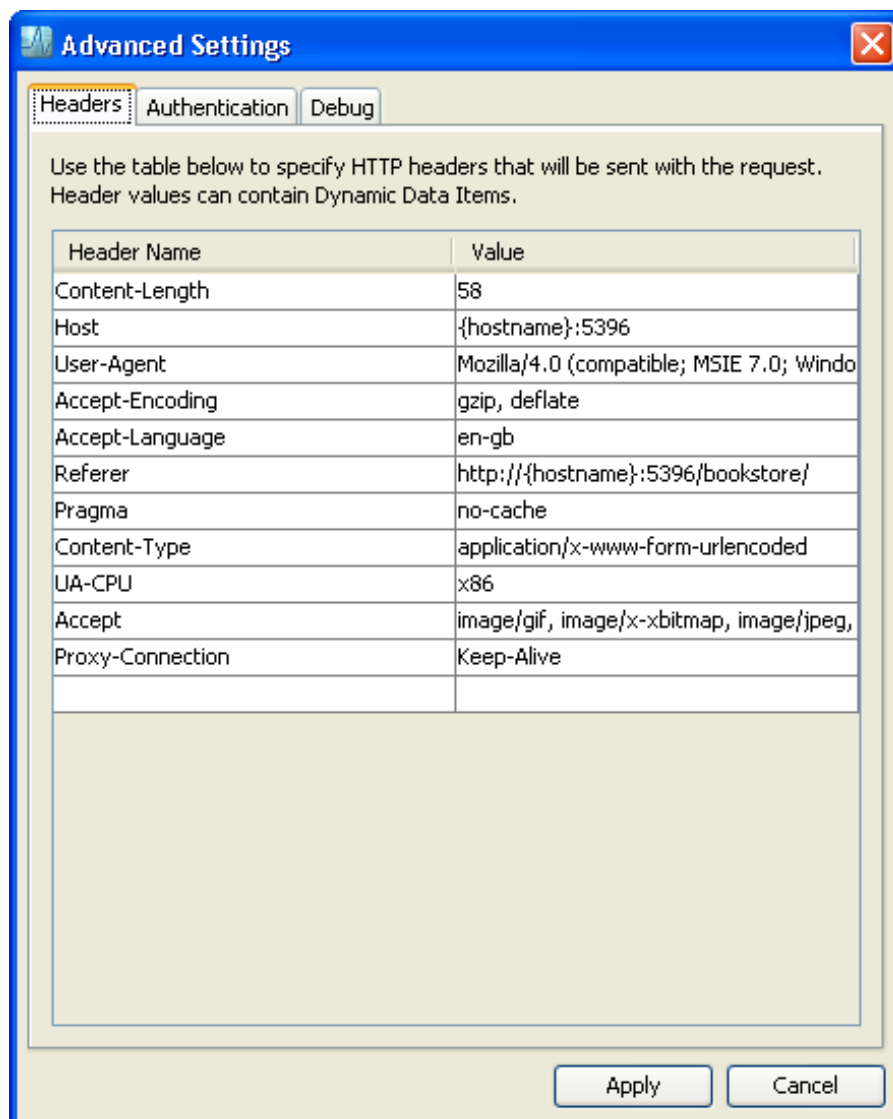
Field	Description
Name	The name given to the step.
Description	Free text description of the step.

Field	Description
Sleep Time	The time in milliseconds that StressTester™ should wait before executing this step.  <b>Note:</b> The time shown after recording is the time during the recording session between the last byte of the previous response being received and the first byte of the request this step represents being sent.
Run First Cycle Only	If checked, this step is only run on the first cycle of the performance test.
Run Last Cycle Only	If checked, this step is only run on the last cycle of the performance test.
Count as Transaction	If checked, this step is counted as a transaction in throughput graphs and data tables.  <b>Note:</b> This value can be set before or after a performance test, allowing different throughput figures to be provided to different business audiences.
Process Response	Please contact Reflective Solutions Support for details of this advanced feature that allows the response of this step to be passed to an external program for special processing.
Request	The http(s) request.
Post File	If specified, this step is a POST request and the named file contains the data to be sent as the body of the POST request.
Post Data	If specified, this step is a POST request and the field contains the data to send as the body of the POST request.
Success String	See the <a href="#">Checking Application Responses</a> section below for details on using this field.
Log Request	If checked, StressTester™ will save the step's request to an individual file in the Injector's log directory.
Log Response	If checked, StressTester™ will save the step's response to an individual file in the Injector's log directory.

Field	Description
Flow Control	See the <a href="#">Varying Business Transaction Flow / Step Order</a> section below for details of this part of the step screen.

## Advanced Settings

By clicking the “Advanced Settings” button on the step properties screen, you can access advanced properties.



The fields on the different tabs on this pop-up are described in the sections below.

### Headers

The table below describes the fields on this tab.

Field	Description
Header Name	The name of the HTTP header to be sent with the request to the application server.
Header Value	The value for the header being sent. Values can contain Dynamic Data item parameters.

### Authentication

The fields on this tab are concerned with the basic authentication method of a user authenticating itself to an application.

The values specified on a step will be sent with the request for that step, and for all subsequent steps in the User Journey.

The table below describes the fields on this tab.

Field	Description
Realm	The realm of the user authenticating to the server.
User	The user name of the authenticating user.
Password	The password credential corresponding to the provided user name.

**Note:** All the fields on this tab may contain Dynamic Data item parameters; allowing different simulated users to authenticate using different credentials.

### Debug

Under the instruction of Reflective Solutions Support, you may be asked to configure debug flags in order that more information about a reported issue may be captured.

This is the screen where these flags are configured and it should only be used following guidance from Reflective Solutions Support.

When you have updated the step properties, click the “Save” toolbar button to persist the changes.

**Note:** For further details on the fields on this screen, press the Help button when on the screen to access the in-tool context specific help.

## Checking Application Responses

It is recommended that you configure StressTester™ to check application responses wherever possible in the User Journey.

The specified response checks should be at as fine a level of granularity as possible. It is not uncommon for applications to start showing incorrect details under load, yet behave correctly under a smaller load.


StressTester™ makes it is possible to check the response of any request within a User Journey. This is achieved by specifying a Success String for the step.

The example below shows that the success criteria for the step is that the response contains either the literal `Results` or `No books match the entered search criteria`.

```
Success String : Results || No books match the entered search criteria.
```

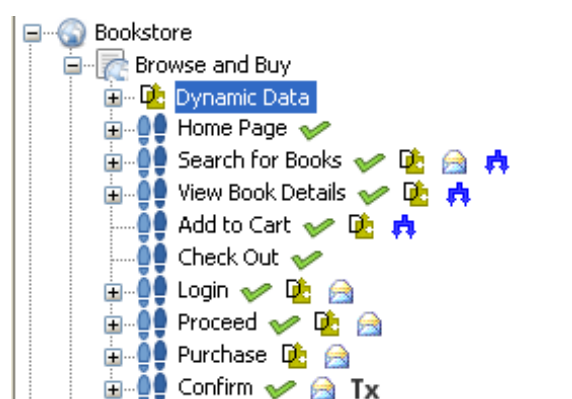
## When to Specify Success Strings

As a rule of thumb, if you are going to name a step (and hence identify it is something of relevance within the User Journey) you should always specify a Success String on the step.

When a step has a Success String specified, StressTester™ shows the step in the navigation tree with a tick mark  following the step.

This allows quick identification of steps that are missing validation criteria.

It can be seen in the User Journey below, that only the “Purchase” step does not have a Success String specified – this is obviously a mistake and should be rectified before testing using this User Journey is performed.



## Success String Formats

Success strings consists of literals and/or Dynamic Data items (allowing you to check that the response contains a parameter value) separated by operators.

**Note:** Success strings are matched to the response of the application – this includes any cookies, HTTP headers, etc. Also, it should be remembered that responses are encoded so certain characters (such as the space character) will have a different representation (in this example '&nbsp;').

### AND Operator

By using the characters `&&` to separate two values, you are testing that both are true.

Therefore in the example below, it is being tested that the response contains both the current value of the “title” Dynamic Data item and the literal ISBN-13.

```
Success String : <title> && ISBN-13
```

### OR Operator

By using the characters `|` to separate two values, you are testing that one or both are true.

Therefore in the example below, it is being tested that the response contains the literals Results and / or No books match the entered search criteria.

Success String : Results || No books match the entered search criteria.

### NOT Operator

By preceding a string or Dynamic Data variable with the exclamation character **!**, you are testing that the response does not contain the specified string or Dynamic Data value.

Therefore in the example below, it is being tested that the response does not contain the string “Sorry – this book is currently out of stock”.

Success String : !Sorry - this book is currently out of stock.

### Operator Order

When a Success String contains more than one operator, StressTester™ validates the string from left to right.

Therefore in the example below, the response must contain either “results” or “no books found”, and also must contain the current value of the ‘title’ Dynamic Data item.

Success String : results || no books found && <title>

## Sleep and Simulated Wait Times

The next important thing to consider, in order to ensure your User Journeys are correct, is the amount of time between steps that will be simulated to represent the time it takes a user to decide what to do, maybe complete some screen fields, and then submit the request.

It is important that inter-step sleep times are considered carefully.

If you specify too little time between steps, you will end up putting greater load on the server than will happen in the real world and hence may spend time diagnosing and correcting performance 'defects' that would never have occurred once the application was given to the users.

Conversely, if you simulate too much time between steps, you risk the chance of placing the application under lesser load than will be experienced in the real world. The possible ramification of this could be that the application passes testing and then fails when given to the users – completely negating the purpose of performance testing!

## Estimating Sleep Times

### So, how do you go about estimating a sleep time?

The first thing to bear in mind is that the sleep time you specify, unless you override the default StressTester™ behaviour, will be varied at run time to ensure that the distribution of sleep times between steps is realistic when considering that some users work at a faster rate than others.

To determine a correct sleep time for an existing application, if at all possible, a tool such as Reflective Solutions' LoadMonitor™ ([www.reflective.com/loadAnalysis.html](http://www.reflective.com/loadAnalysis.html)) should be used to monitor an application over a period of time and provide empirical information.

This information may also be available from web server log files.

Of course, for a new application, there is no way of monitoring a production environment and therefore estimates need to be made.

## Configuring Sleep Times

The sleep time for a step – that is the time that StressTester™ will wait before running the step – is configured on the step's properties screen. This screen is accessed by selecting the step's node in the navigation tree.

The example below shows that the step has a sleep time of 5906 milliseconds, or slightly less than six seconds.

The screenshot shows the configuration interface for a step named "Search for Books". The "Sleep Time" field is set to 5906 ms. Below this, there are four checkboxes: "Run First Cycle Only", "Run Last Cycle Only", "Count as Transaction", and "Process Response", all of which are currently unchecked. The "HTTP Information" section is expanded, showing the "Request" field with the URL "http://<hostname>:5396/bookstore/bookstore.do". The "Post Data" field is selected and contains the query string "sta\_action=search&sta\_title=<Search Term>&sta\_search=acm". The "Success String" field contains the text "Results || No books match the entered search criteria."

When you first inspect a step following the recording of a User Journey, the sleep time shown in the StressTester™ user interface will represent the time during the recording session from the last byte of the previous response being received and the first byte of this step's request being transmitted.

To change a step's sleep time, set the Sleep Time field to the value required and click "Save".

## Sleep Time Variation

It is obvious that all users of an application do not work at the same pace. Some may be more familiar with the application, be able to type quicker, or already know the data they need to enter.

The result of this is that if a performance test is to realistically simulate the load an application will experience in the real world, the inter-step sleep times need to be varied.

StressTester™ allows the sleep times specified in the step properties to be varied by a random amount within limits defined by the user.

The Sleep Time Variation field on the User Journey property screen allows a percentage to be specified – by default 20%.

Timings			
Initial Sleep Time:	<input type="text" value="0"/>	ms	The time to wait before any simulated users start the User Journey.
Cycle End Sleep Time:	<input type="text" value="0"/>	ms	The time each simulated user waits at the end of each iteration of the User Journey.
Sleep Time Multiplier:	<input type="text" value="1.0"/>		All times are multiplied by this value e.g. setting to 0.5 halves all sleep times.
Sleep Time Variation:	<input type="text" value="20.0"/>	%	Following multiplication, sleep times are randomised in a normal distribution +/- this percentage.

StressTester™ will calculate a random variance for each simulated user and each cycle that is between the Sleep Time specified in the step properties, plus or minus the specified Sleep Time Variation percentage.

The values actually generated will be in the form of a normal distribution between the lowest and highest possible values.

**Example:** For a step Sleep Time of 5000ms (five seconds), and a Sleep Time Variation of 20%, the actual sleep time for the step will be a random value between 4000ms and 6000ms milliseconds.

## Sleep Time Multipliers

All sleep times within StressTester™ can be further varied by multiplying the times calculated by a set amount.

This is achieved by changing the Sleep Time Multiplier field on the User Journey properties screen from its default value of 1.

**Example:** If Sleep Time Multiplier is set to 0.5, all sleep times will be halved. In the same manner, if it is set to 2, all sleep times will be doubled. Finally, setting the Sleep Time Multiplier to 0 will cause all sleep times to be 0 i.e. no sleep time will be simulated by StressTester™.

## Varying Data Values

A key aspect of making sure that the performance test is correct is to ensure the data passed to the application is as varied as that which will be supplied by the user community.

If data is not sufficiently varied, various caches in the application may come into play, which will mean the results of performance testing will not be the same as those that will be experienced by the users in the real world.

### How do you know which data to vary?

The rule of thumb is:

*If you are not sure whether varying the data will affect how the application executes the request, vary it to be safe.*

When you have specific access to information about how the application functions internally, you should ensure all values that affect how the application processes the request (for example, the type of insurance policy being applied for), or how application data is stored (for example, all values that form part of database keys) are varied.

### What data values should be used?

Again, this requires information about how the application is currently used in the real world, or how it will be used (for a new application).

Once the data values for a particular field are known, StressTester™ should be used to match the real world variation as much as is possible (for example, when testing a book store, the infinite variation in search terms cannot obviously be simulated – but it should be ensured that the data set is as large as possible).

### How do I provide data to a User Journey?

StressTester™ provides access to eight different data sources:

- Auto-Correlated
- Auto-Increment
- Constant
- Delimited File

- Date
- Java Class
- Response
- Related (to another defined Dynamic Data item)

This set will cover all data requirements and hence ensure that a User Journey's data can be correctly varied without having to write any test script.

## Dynamic Data Introduction

StressTester™ uses the term Dynamic Data to represent a value that is parameterised within a User Journey – and hence has varied data supplied during a performance test.

Dynamic Data items (parameters) can be created in one of two ways:

- by entering the name of a new Dynamic Data within a step (in this case StressTester™ creates a new Dynamic Data item automatically), or
- by right-clicking on the Dynamic Data group node and selecting “New”

When a Dynamic Data item is created, it is displayed in the Dynamic Data section of the navigation tree. By selecting the node, the Dynamic Data item screen is shown.

The screenshot displays the configuration interface for a Dynamic Data item. On the left, a navigation tree shows the 'Dynamic Data' group expanded. The main configuration area includes the following fields and options:

- Name:** title
- Description:** (empty)
- Source:** Response
- Step:** Search For Books
- Location:** A table defining filters for the response.
- Selection:** Random
- Column Number:** 1
- Encode Value?:** No
- Shared?:** One User Only
- Refresh?:** Every Cycle

Filter Type	Start	End	Index
Text	<h2>	</h2>	All
Text	sta_jsbn=	"	1

## StressTester™ : Configuring User Journeys to be Correct and Realistic

The fields on the Dynamic Data item screen will differ depending on the selected source value. The table below details the fields that are common to all Dynamic Data sources.

<b>Field</b>	<b>Description</b>
Name	The unique name (within the User Journey) for the Dynamic Data item.
Description	A free text description of the Dynamic Data item.
Encode Value?	Specifies whether the value StressTester™ retrieves from the source is already encoded for http transmission (choose "No") or needs to be encoded first (choose "Yes").
Shared	Indicates whether each simulated user has its own copy of the Dynamic Data item's source ("One User Only") or that all simulated users share one copy ("All Users of the User Journey").
Refresh	<p>Specifies how often an individual user's current value of the Dynamic Data item should be updated.</p> <p><b>Once per Run</b> - the first user that needs a value for this item will be provided one by StressTester™ and then all subsequent requests for values for this item, from all simulated users, will be given the same value.</p> <p><b>Once per User</b> - once the simulated user has been provided with a value, it will continue to use that value for the remainder of the performance test execution.</p> <p><b>Every Cycle</b> - once the simulated user has been provided with a value in a User Journey cycle (iteration), it will continue to use that value for the remainder of the User Journey cycle.</p> <p><b>Every Time</b> - every time the User Journey refers to the Dynamic Data item, the simulated user will be provided with a new value.</p>

The sections below describe each of the Dynamic Data sources and detail fields specific to each source.

## Auto-Correlated Dynamic Data

The auto-correlated source is used to configure parameters that are usually automatically handled "behind the scenes" by the browser.

Typically these hold some kind of session state, or a unique identifier allowing the application to retrieve the user's state on the server.

Name:  Description:

Source:

Auto-Correlate Dynamic Data is used when you wish StressTester™ to maintain the value of a data item (typically session state field) that is passed between the browser and the application server. Choose a Field Type of "Known Values" or "Repeated Fields" and then select the application field to auto-correlate.

It is always recommended that you configure any Known Fields.

See Help for more information on this type of Dynamic Data.

Field Type:

Field Name:

Appears In:  All  
 URL  
 Post  
 Headers

Encode Value?  Yes  No

Shared:  One User Only  All Users of User Journey

Refresh:  Once per Run  Once per User  Every Cycle  Every Time

To configure an auto-correlated Dynamic Data item choose the Source of "Auto-Correlated" and then complete the displayed fields - which are described below.

Field	Description
Field Type	Drop-down field allowing you to select application fields that StressTester™ knows should be auto-correlated ("Known Fields") or those that are unknown but are repeating in the request-response interaction between the browser and the server. Such fields may need to be auto-correlated ("Repeated Fields") in order that the User Journey functions correctly.
	When you select either option, the Field Name drop-down is populated.

Field	Description
Field Name	A drop-down list of the User Journey's fields that match the Field Type selected. Simply select the field you wish to auto-correlated.
Appears In	<p>Specifies where in the application request the field appears.</p> <p>This is a tuning parameter for the Save of this Dynamic Data item (at which point StressTester™ scans the User Journey and replaces the recorded values of the identified field with the Dynamic Data item parameter).</p> <p>If you are unsure, always choose "All"; otherwise configure the Dynamic Data item appropriately.</p>

When you click the "Save" toolbar button, StressTester™ will prompt you to confirm you wish to update the User Journey to substitute recorded values with the Dynamic Data item parameter, and when you click "OK" will proceed to do such.

One auto-correlated Dynamic Data item should be created for each field that needs to be automatically handled by StressTester™.

## Auto-Increment Dynamic Data

The auto-increment source is used to configure a parameter that is based on a numeric value that increases or decreases every time it is requested.

The value can be padded with leading zeroes and also have a literal suffix and/or prefix.

This Dynamic Data source is ideal for generating phone numbers, email addresses, etc.

Name:  Description:

Source:

Auto-Increment Dynamic Data is used when you wish StressTester™ to generate a sequence of numbers. You can choose the first number in the sequence, the positive or negative increment, the minimum length the generated number should be (if under this length it will be padded with leading zeros) and also specify a prefix and/or suffix.

See Help for more information on Auto-Increment Dynamic Data.

Starting Value:  Increment:

Prefix:  Suffix:

Minimum Length:

Encode Value?  Yes  No

Shared:  One User Only  All Users of User Journey

Refresh:  Once per Run  Once per User  Every Cycle  Every Time

To configure an auto-increment Dynamic Data item choose the Source of "Auto-Increment" and then complete the displayed fields - which are described below.

Field	Description
Starting Value	The first value to be used in the series of generated numbers.
Increment	The value - positive or negative - by which the value should be altered every time it is requested.
Prefix	The string that should prefix the generated value.
Suffix	The string that should suffix the generated value.
Minimum Length	If the generated number (excluding the Prefix and Suffix) is less than this length, the value is preceded by zeroes until it is this minimum length i.e. if the generated number was 23 and the Minimum Length 3, the value generated would be 023.

Click "Save" on the toolbar to save the configured Dynamic Data item.

## Constant Dynamic Data

The constant source is used to configure a parameter that does not change during a performance test.

This is typically used to represent values that are changed when a User Journey is passed between StressTester™ users (see [Import / Export a User Journey](#) below). Therefore, this type of data is typically used to represent 'global' values such as the server and port hosting the application.

The screenshot shows a configuration form for Constant Dynamic Data. It includes the following fields and options:

- Name:** A text input field containing "hostname".
- Description:** An empty text area.
- Source:** A dropdown menu set to "Constant".
- Value:** A text input field containing "rs1-laptop-088".
- Encode Value?:** Radio buttons for "Yes" and "No", with "No" selected.
- Shared:** Radio buttons for "One User Only" and "All Users of User Journey", with "All Users of User Journey" selected.
- Refresh:** Radio buttons for "Once per Run", "Once per User", "Every Cycle", and "Every Time", with "Once per Run" selected.

Below the "Source" dropdown, there is explanatory text: "Constant Dynamic Data is used when you wish to set the value of something to a value that does not change during the test. A typical usage is to replace the host names against which the User Journey was recorded with a Dynamic Data item - making the User Journey easily portable." and a link to "See Help for more information on Constant Dynamic Data."

## Date Dynamic Data

The date source is used to configure a date or time - in any format required - that is based on the date of the performance test or the time at which the value is requested by a simulated user.

The value generated can be offset in the future or the past by a set number of seconds, minutes, hours or days.

This source is very useful when you need to ensure entered dates and times are in the future, and need to specify dates in exact formats for the application to function correctly.

## StressTester™ : Configuring User Journeys to be Correct and Realistic

Name:  Description:

Source:

Date Dynamic Data allows you to generate a date or time relative to the current date or time - with a negative or positive offset. The data generated can be formatted as required.

See Help for more information on Date Dynamic Data.

Starting Point:  Now  Today

Offset:

Date Format:

Encode Value?  Yes  No

Shared:  One User Only  All Users of User Journey

Refresh:  Once per Run  Once per User  Every Cycle  Every Time

To configure a date Dynamic Data item choose the Source of "Date" and then complete the displayed fields - which are described below.

Field	Description
Starting Point	If you wish to base the generated date/time on 00:00am of the date of the performance test, choose "Today". If you wish to base the date/time on the time at which the date is requested by a simulated user, choose "Now".
Offset	Choose either positive ("+") or negative ("-") offset and the number of seconds, minutes, hours or days to offset with relation to the Starting Point.
Date Format	You can generate any format of date or time required.  The format of this field is specified as described on the following web page:  <a href="http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html">http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html</a>

Click "Save" on the toolbar to save the configured Dynamic Data item.

## Delimited File Dynamic Data

The delimited file source is used to configure StressTester™ to retrieve the Dynamic Data item's values from a column within a delimited file.

StressTester™ can access any file that has separate columns delimited by the same character, and a new line indicating a new set of data.

It is recommended that this type of Dynamic Data is only used to represent data that is truly entered into the application by the real world user; for any other user action (selecting a page option, a drop-down value, check box on a screen, random item on a search results page, etc.) the “Response” source should be used instead.

**Note:** Even if more than one Dynamic Data item's values will be retrieved from a single delimited file (each item from a different column in the file), only one item should be configured using this source and the remainder of the items should be configured using the “Related” source

The screenshot shows a configuration form for a Delimited File Dynamic Data item. The form includes the following fields and options:

- Name:** Search Term
- Description:** (Empty text box)
- Source:** Delimited File (Selected from a dropdown menu)
- File Name:** data/search.txt (Text input with a browse button)
- Column Index:** 1 (Text input)
- Delimiter:** , (Text input)
- Selector:** Random (Selected from a dropdown menu)
- Encode Value?** No (Selected radio button)
- Shared:** All Users of User Journey (Selected radio button)
- Refresh:** Every Time (Selected radio button)

Below the Source dropdown, there is explanatory text: "Delimited File Dynamic Data allows values to be retrieved from any file delimited file. More than one value can be retrieved from each row of a file by creating subsequent Dynamic Data Items which are related to this one (using the Related type of Dynamic Data)." and a link to "See Help for more information on Delimited and Related Dynamic Data."

To configure a delimited file Dynamic Data item choose the Source of "Delimited File" and then complete the displayed fields - which are described below.

<b>Field</b>	<b>Description</b>
File Name	<p>The name (either relative to the location in which the StressTester™ Injector will be started or a full explicit path name) of the file containing the data values.</p> <p>If the file is available on an accessible file system, the file chooser button ".." can be clicked to locate the file and auto-complete the File Name field on the screen.</p>
Column Index	<p>The index of the column to be read to obtain the Dynamic Data item's value set.</p> <p>Column numbering starts at '1'.</p>
Delimiter	<p>The delimiter used within the file to separate columns on each line.</p>
Selector	<p>Determines which line in the file is used to obtain the Dynamic Data item's value:</p> <p><b>First</b> - use the first line in the file</p> <p><b>Last</b> - use the last line in the file</p> <p><b>Random</b> - pick a value at random. Potential exists for the same value to be used more than once by different or the same simulated users in a performance test.</p> <p><b>Random Unique</b> - pick a value at random but never use the same value twice. This therefore ensures the same value will never be used more than once in a performance test.</p> <p><b>Sequential</b> - iterate over the list of values held in the appropriate column and, when the list is exhausted, start from the beginning again.</p> <p><b>Sequential Unique</b> - iterate over the list of values once and then stop.</p>

Click "Save" on the toolbar to save the configured Dynamic Data item.

## Java Class Dynamic Data

The Java class source is used to configure a parameter that is based on values returned by a Java class.

This source of Dynamic Data is only used on the rare occasions when the other sources cannot provide the required values.

**Note:** Please contact Reflective Solutions Support – [support@reflective.com](mailto:support@reflective.com) - for information on how to implement Java classes in order for them to be compatible with StressTester™. In addition, it is possible to call other programming languages such as VB, C#, etc. – again please contact Support for further details.

The screenshot shows a configuration form for a 'Java Class' dynamic data source. The 'Name' field is 'Stock Ticker Feed' and the 'Description' is 'Latest stock movements over an RS232 interface.' The 'Source' is set to 'Java Class'. Below this, there is explanatory text: 'Java Class Dynamic Data allows you to call out to a Java program implementing a simple interface to retrieve data. It should only be used in the rare times when the other types of Dynamic Data are not appropriate. If you wish to call out to another programming language (VB, C#, etc.) please contact Reflective Solutions Support for details on how to do this. See Help for more information on Java Dynamic Data.' The 'Classname' field is 'com.acmecorp.ticker.Reader'. The 'Encode Value?' section has 'No' selected. The 'Shared:' section has 'One User Only' selected. The 'Refresh:' section has 'Every Cycle' selected.

To configure a Java class Dynamic Data item, choose the Source of "Java Class" and then complete the displayed field - which is described below.

Field	Description
Classname	The full class name of the Java class to be executed. The specified class must be on the class path of the Injector.

Click "Save" on the toolbar to save the configured Dynamic Data item.

## Response Dynamic Data

The response source is used to extract Dynamic Data item values from a previous application response.

This is a very powerful feature allowing User Journeys to simulate real world users selecting from the options an application presents them – whether it is a link on a page, a value from a drop-down or any other type of displayed option.

In script-based tools this generally requires making function calls or writing elaborate scripts; in StressTester™ it is simply a case of configuring one Dynamic Data screen – an example of which is shown below.

The screenshot shows the configuration interface for a Response Dynamic Data item. The 'Name' field is 'title' and the 'Source' is 'Response'. The 'Step' is 'Search for Books'. The 'Location' table is as follows:

Filter Type	Start	End	Index
Text	<h2>	</h2>	All
Text	sta_isbn=	"	1

The 'Selection' is 'Random', 'Column Number' is '1', and 'Encode Value?' is 'No'. The 'Shared' option is 'One User Only' and the 'Refresh' option is 'Every Cycle'.

To configure a response Dynamic Data item choose the Source of "Response" and then complete the displayed fields - which are described on the next page.

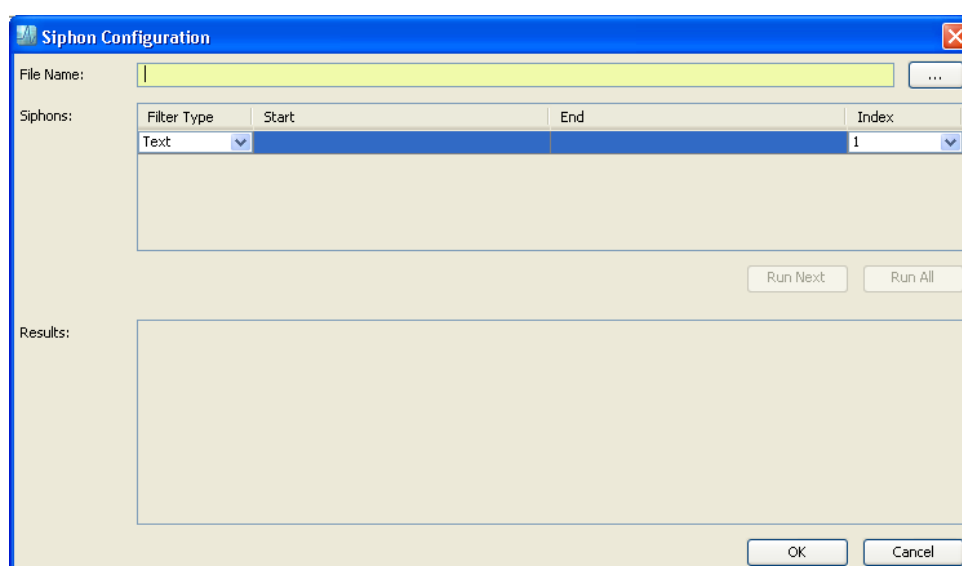
Field	Description
Step	<p>The name of the step whose response should be processed to obtain the Dynamic Data item's value.</p> <p><b>Note:</b> Only steps that have been named by the user are shown in the drop-down menu.</p>
Location	<p>A table describing the text or position filters that should be applied to the response. These table lines are termed "siphons" as they siphon values from either the response, or the preceding table line's output.</p> <p>Please refer to the Configuring Siphons section after this table for information on how to configure siphons.</p>
Selection	<p>When the siphons have been processed and a set of potential values has been computed, the Selection option defines which value will be used by the simulated user.</p> <p><b>First</b> - use the first value in the result set produced by the siphons</p> <p><b>Last</b> - use the last value in the result set produced by the siphons</p> <p><b>Random</b> - pick a value at random. Potential exists for the same value to be used more than once by different or the same simulated users in a performance test.</p> <p><b>Random Unique</b> - pick a value at random but never use the same value twice.</p> <p><b>Sequential</b> - iterate over the list of values held in the appropriate column and, when the list is exhausted, start from the beginning again.</p> <p><b>Sequential Unique</b> - iterate over the list of values once and then stop.</p>
Column Number	<p>It is possible for the siphons to return a two-dimensional array of values; in such a case the column number identifies the column from which the value should be selected.</p>

**Note:** The data set for a response Dynamic Data item is refreshed every time the response that contains the data is received for each simulated user.

### Configuring Siphons

To configure the siphons for your response Dynamic Data, first of all ensure you have executed your User Journey using Quick Run (this ensures a response log file exists).

Then, click on the “Configure Siphons” button to open the configuration pop-up.



This screen allows you to configure and test siphons before saving the Dynamic Data item; ensuring the Dynamic Data is correctly configured before you execute a performance test.

First, you should select a log file for the response you wish to siphon using the file chooser button. You can identify the appropriate file in a log directory as its file name will contain the name of the response step that you want the Dynamic Data item to be based upon.

### Defining Siphons

Siphons are the means by which specific parts of a response are isolated. Two types of siphons can be specified: "Text" and "Index".

Text siphons can be thought of as a 'substring function'; they return the string(s) that exists between the values in the "Start" and "End" column of the Location table.

Index siphons can be thought of as a positional search; they return the string that exists between the numeric values in the "Start" and "End" columns.

**Note:** The first character in a response is at position '1'.

The last part of configuring each siphon line is to determine, if the siphon can return more than one value, which value(s) should be passed to the next siphon in the table.

This is specified in the Index column and the potential values are:

- "All" (pass all values to the next siphon),
- "Random" (pick one at random to pass to the next siphon), or
- an integer value which specifies which item should be passed on.

**Note:** If two rows in the table have an Index value of "All", this will generate a two-dimensional array of values and the Column Number field on the screen will become active.

### A Worked Example

In the response to a search for books, various books are listed. Each listed book has a visible title that when clicked shows a page of details for the selected book.

The next page shows an extract of the HTML for the search results page.

```
<h2><a href="http://localhost:5396/bookstore/bookstore.do?
sta_action=getDetails&sta_isbn=978-0072253603">SCJP
Sun Certified Programmer for Java 5 Study Guide</a></h2>
...
<h2><a href="http://localhost:5396/bookstore/bookstore.do?
sta_action=getDetails&sta_isbn=978-0072263855">Java:
The Complete Reference</a></h2>
...
<h2><a href="http://localhost:5396/bookstore/bookstore.do?
sta_action=getDetails&sta_isbn=978-0130449689">SOA
Using Java(TM) Web Services</a></h2>
```

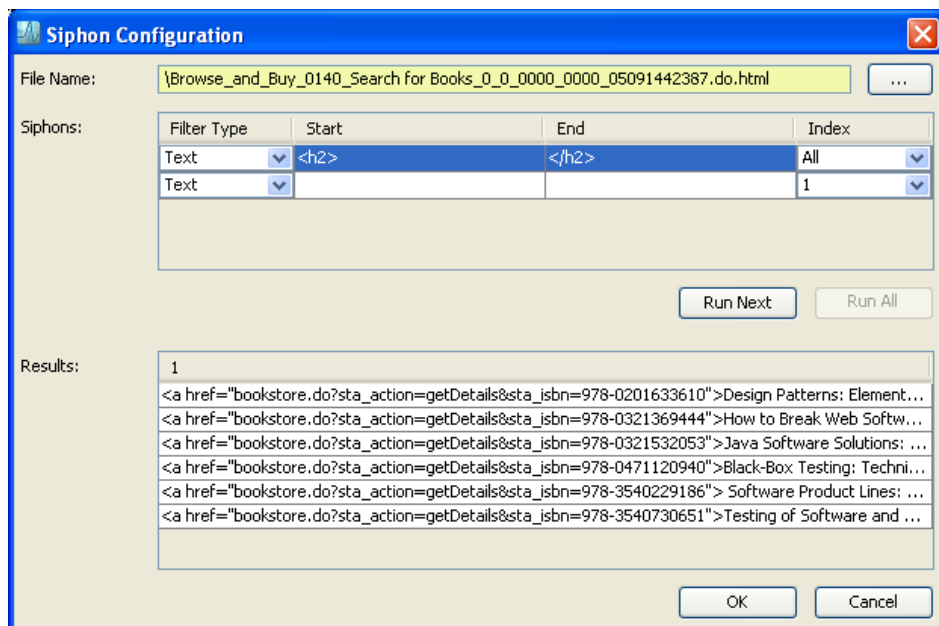
The only part of the link URL that is varying is the number that follows 'sta\_isbn=' that is highlighted above. The text siphon gives complete freedom to select as much of the URL as desired and extracting the entire link would be an equally valid choice.

Our goal is to pick a book at random; so we need a list of values. Therefore, we need to isolate a set of fragments containing the value required, and then further isolate so we end up with a list of values.

The starting string of '<h2>' is an ideal choice here as it does not occur in the page anywhere other than for the book detail links.

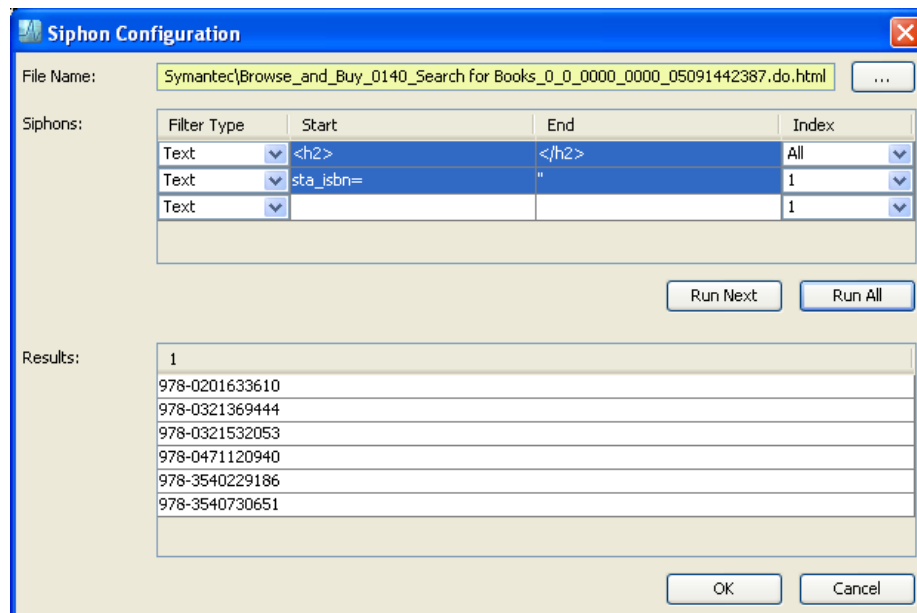
So, our first siphon has a Start of "<h2>", and End of "</h2>" and an Index of "All" – so that we get all of the books listed.

If we enter this siphon line into the Configure Siphons screen and click "Run Next", the screen will update as shown below.



We now need a second siphon line to extract the identifier.

By specifying a second siphon with a Start of “sta\_isbn=” and an End of a double quotation mark, and then clicking “Run All”, we can see that the correct identifiers have been extracted from the response.



## Related Dynamic Data

The related source is used to configure a parameter that is based on a source that has been defined for a different Dynamic Data item - and hence this item is *related* to the other one.

This is useful for multi-column delimited files, Java classes and response Dynamic Data items that return two-dimensional data sets.

A real advantage of the related source is that when StressTester™ picks a row in the file or siphon results to determine the value for the first item, the related item will use the same row. So, for a file containing unique user names and passwords, StressTester™ would ensure that even if it is configured to pick a user name at random, the password will come from the same row in the file and hence the simulated user will log into the application correctly.

Name:  Description:

Source:

Related Dynamic Data allows subsequent Dynamic Data items to be defined that use the same source as an item defined with either the type of Delimited File or Response. It saves having to specify the values again and also ensures the Related item will be taken from the same location (in the case of a file, the same line) as the item to which it is related.

See Help for more information on Related Dynamic Data.

Item:

Column Number:

Encode Value?  Yes  No

Shared:  One User Only  All Users of User Journey

Refresh:  Once per Run  Once per User  Every Cycle  Every Time

To configure a related Dynamic Data item choose the Source of "Related" and then complete the displayed fields - which are described below.

Field	Description
Item	A list of Dynamic Data items that are configured with a source type of Delimited File, Java Class or Response.  Choose the item you wish to relate this item to.
Column Number	The column number within the file, Java class or siphon results that StressTester™ should use to obtain this item's value set.

Click "Save" on the toolbar to save the configured Dynamic Data item.

## Testing Dynamic Data Configurations

If you are configuring your first User Journey, you will probably have been using StressTester™'s Quick Run functionality up until now to check that your User Journey continues to execute successfully after each modification.

Obviously, the whole point of Dynamic Data is to vary data supplied to an application, and therefore you would have to repeat Quick Run tests a number of times to ensure the variation you require is occurring.

It is now time to create a Test Run Configuration (the definition of how StressTester™ executes a test) that runs your User Journey for more than one cycle.

Please refer to the [Executing Performance Tests](#) user guide for details on how to create Test Run Configurations.

When you have created such a configuration, select the "Log Request" option for either the steps on which you have configured Dynamic Data, or for the User Journey (to log all requests).

You can now execute a test and check the requests sent to ensure that data variation is occurring as you intended.

## Varying Business Transaction Flow / Step Order

Ensuring that a User Journey's simulated users vary the data they supply to an application does still not mean that the User Journey will exercise the application correctly – that is, in the manner the real world user community will do so.

This is because real users will vary the route taken through the business transaction, will go back and repeat steps, and will react to application responses when deciding how to proceed.

For example, consider users searching for a book using an online book store application.

Some will know the exact book they want, immediately find it, add it to their cart and check out.

However, others will need more than one search to find books of interest, need to look at one or more book's details to check which is the best match for their needs, may perform a "quick checkout" or may go through many steps to complete the same.

Simulating these different routes through a User Journey is imperative if the load applied to the application is to be the same as the application will experience in the real world.

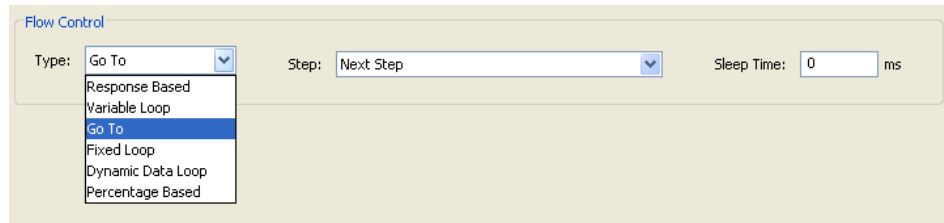
All of the above scenarios could be called "Buying a Book" and with StressTester™ there is no need to develop multiple User Journeys for each possible route – one User Journey can simulate all the different routes needed.

The importance of ensuring the User Journey is correct in this manner should not be underestimated – web sites commonly fail following "successful performance tests" because the real world users' routes through the application were not correctly simulated.

In StressTester™ terminology, the variation of a route from the sequential order the User Journey was recorded to being something different is called Flow Control.

## Introduction to Flow Control

Flow Control is configured on the step screen – the configured Flow Control determines the simulated user’s next interaction with the application under test after the current step has completed.



The fields on the Flow Control section of the step properties screen will differ depending on the type of Flow Control selected.

The table below details the fields that are common to all Flow Control types.

Field	Description
Type	The type of Flow Control.
Step	<p>May appear once or many times in a Flow Control configuration – specifies the step the simulated user should execute next.</p> <p>Only steps that have been named by the user will appear in the drop-down list.</p> <p>In addition to named steps, there are two special destinations:</p> <p><b>Next Step</b> – the simulated user will run the next step in the User Journey.</p> <p><b>End Cycle</b> – the simulated user will end the current cycle of the User Journey and start a new cycle.</p> <p><b>End User</b> – the simulated user will stop and make no more requests during the performance test.</p>

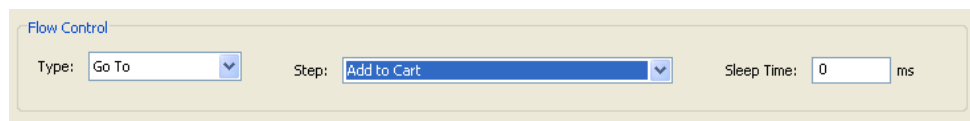
Field	Description
Sleep Time	<p>Specifies a sleep time that should occur before the simulated user is redirected to the destination step.</p> <p><b>Note:</b> If the destination step also specifies a sleep time, the actual sleep time (which will be varied as described in the <a href="#">Sleep and Simulated Wait Times</a> section above) will be the total of the two specified values.</p>

The sections below describe each of the Flow Control types and detail fields specific to each type.

## Go To Flow Control

Go to Flow Control specifies that the simulated user jumps to the specified step.

This can be useful when you have effectively created two routes through a section of a User Journey, and having finished the first route, wish to jump over the steps that make up the second route to continue with the main flow of the User Journey.



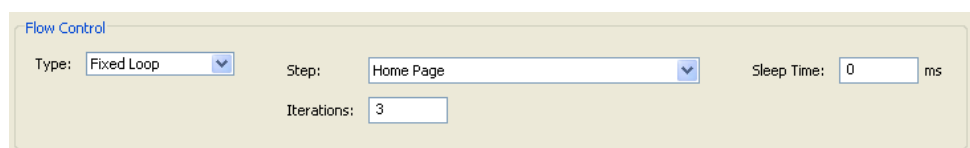
The screenshot shows a 'Flow Control' configuration panel. It contains three fields: 'Type' is a dropdown menu set to 'Go To'; 'Step' is a dropdown menu set to 'Add to Cart'; and 'Sleep Time' is a text input field set to '0' with 'ms' to its right.

Go to Flow Control has no additional fields.

**Note:** It is possible to create an infinite loop if this is incorrectly configured.

## Fixed Loop Flow Control

Fixed Loop Flow Control specifies that a loop will occur in the User Journey for a fixed number of iterations.



The screenshot shows a 'Flow Control' configuration panel. It contains four fields: 'Type' is a dropdown menu set to 'Fixed Loop'; 'Step' is a dropdown menu set to 'Home Page'; 'Iterations' is a text input field set to '3'; and 'Sleep Time' is a text input field set to '0' with 'ms' to its right.

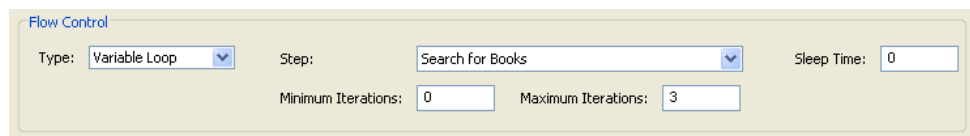
The additional field for this type of Flow Control is described in the table below.

Field	Description
Iterations	The number of times to loop back to the specified step.

## Variable Loop Flow Control

Variable Loop Flow Control is very similar to Fixed Loop Flow Control – the difference is simply that StressTester™ randomises the number of times the loop occurs.

This can be useful when simulating a user performing a task one or more times, but the number of times each user does so varies in the real world.



The screenshot shows a configuration panel titled "Flow Control". It includes a "Type" dropdown menu set to "Variable Loop", a "Step" dropdown menu set to "Search for Books", and a "Sleep Time" input field set to "0". Below these are two input fields for "Minimum Iterations" (set to "0") and "Maximum Iterations" (set to "3").

The additional fields for this type are described in the table below.

Field	Description
Minimum Iterations	The minimum number of times the simulated user will loop back to the specified step.
Maximum Iterations	The maximum number of times the simulated user will loop back to the specified step.

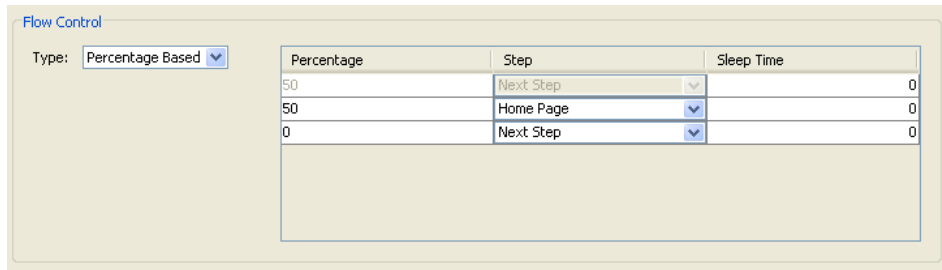
**Note:** StressTester™ picks a random number between the minimum and maximum to determine how many times the simulated user actually loops back to the specified step. This number is recalculated for every simulated user and every cycle during the performance test.

## Percentage Flow Control

Percentage Flow Control allows the simulated user community to be split and take different routes through a User Journey.

This can be useful when real world users would perform an action in two or more ways; for example, either using the 'quick check out' or 'normal check out' routes.

There is no limit on the number of different routes you can specify.



The screenshot shows the 'Flow Control' configuration window. The 'Type' is set to 'Percentage Based'. Below this is a table with three columns: 'Percentage', 'Step', and 'Sleep Time'. The table contains three rows of data.

Percentage	Step	Sleep Time
50	Next Step	0
50	Home Page	0
0	Next Step	0

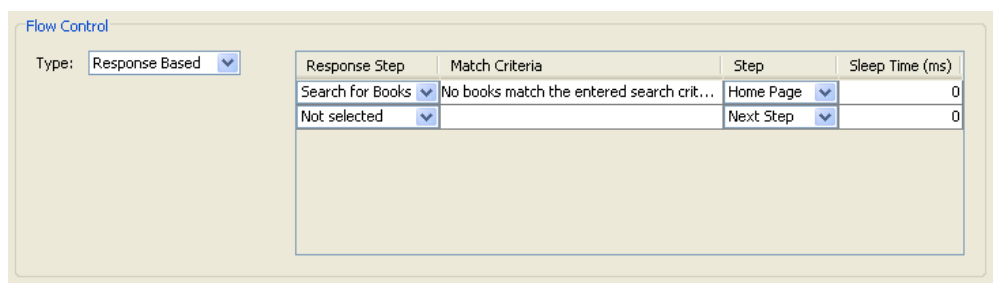
The additional field for this type of Flow Control is described in the table below.

Field	Description
Percentage	<p>The percentage of times, across all simulated users for the User Journey, that simulated users will go to the specified step.</p> <p><b>Note:</b> If the specified percentages do not equal 100, StressTester™ will create an additional line that goes to the "Next Step" for the remaining percentage available.</p>

## Response Based Flow Control

Response Based Flow Control allows the simulated user's next step to be determined based upon the application's response – either for the current step or a previously executed step.

This is very useful when needing to simulate users reacting to application responses; whether it be selecting a special offer, responding to no results being shown for a search, or one of the many other usages for this type of Flow Control.



The screenshot shows the 'Flow Control' configuration window. The 'Type' is set to 'Response Based'. Below this is a table with four columns: 'Response Step', 'Match Criteria', 'Step', and 'Sleep Time (ms)'. The table contains two rows of data.

Response Step	Match Criteria	Step	Sleep Time (ms)
Search for Books	No books match the entered search crit...	Home Page	0
Not selected		Next Step	0

The additional fields for this type of Flow Control are described in the table below.

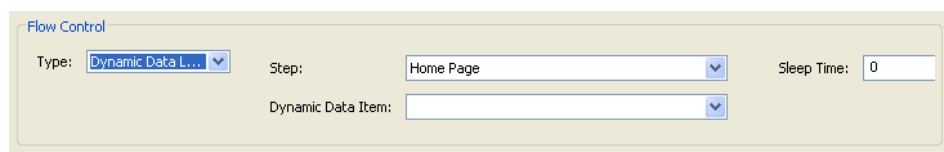
Field	Description
Response Step	The step whose response should be compared to the Match Criteria to determine if the user should go to the specified destination step.
Match Criteria	<p>The criteria to use to determine whether the response should cause the simulated user to go to the destination step.</p> <p>Match Criteria is specified in exactly the same way as Success Strings – please refer to the <a href="#">Checking Application Responses</a> section above for format and specification details.</p>

## Dynamic Data Loop Flow Control

Dynamic Data Loop Flow Control specifies that a loop will occur in the User Journey and that, for every cycle of the loop, a different value of a Dynamic Data item will be used.

The loop will iterate until the Dynamic Data values have been exhausted.

This can be useful when simulating a user processing a presented list of items – for example reading emails or processing workflow tasks.



The screenshot shows a configuration window titled "Flow Control". It contains three fields: "Type" is a dropdown menu set to "Dynamic Data Loop"; "Step" is a dropdown menu set to "Home Page"; and "Sleep Time" is a text input field containing the number "0". Below these, there is a "Dynamic Data Item" dropdown menu which is currently empty.

The additional field for this type of Flow Control is described in the table below.

Field	Description
Dynamic Data Item	<p>The Dynamic Data item whose value set should be exhausted by looping back to the specified step once for each value in the value set.</p> <p><b>Note:</b> Only Dynamic Data items that have a finite set of values will appear in the drop-down. This prevents StressTester™ getting into an infinite loop.</p> <p>Therefore, to appear in the drop-down list, an item must have a type of “Random Unique” or “Sequential Unique” and a refresh value of “Every Time”.</p>

## Import / Export a User Journey

There are a number of reasons why you may wish to save a copy of the User Journey outside of the StressTester™ database:

- For backup reasons
- To “version” the User Journey
- To store the User Journey in a source code control system – probably alongside the version of the application it tests
- To pass the User Journey to a colleague
- To send the User Journey to Reflective Solutions Support when being assisted with an issue

This section describes the quick processes of exporting and importing User Journeys.

### Import a User Journey

There are two methods to import a User Journey.

The first is to choose the “Import User Journey” option on the “File” menu.

The second is to right-click on an application node in the User Journey workspace navigation tree and select the “Import User Journey” option.

No matter whichever option is chosen, StressTester™ will present a file open dialogue, allowing the user to navigate and find the XML file containing the User Journey and import it into their StressTester™ installation.

**Note:** If you choose the second method and attempt to import a User Journey that was originally associated to a different application to the application node you selected, StressTester™ will warn you and ask you to confirm you wish to proceed with the import. If you do proceed, the User Journey will now be associated with the application you selected.

## Export a User Journey

To export a User Journey, simply right-click on the User Journey's navigation tree node and select the "Export" option.

StressTester™ will present a file save dialogue allowing you to save the exported User Journey in a location and file name of your choice.