Dashboard User’s Manual

Accenture StormTest Development Center

Document ID: ST-11028

Revision Date: November 2015

Product Version: 1.0

Web: http://www.accenturestormtest.com

Contents

1 Preface

1.1 Accenture StormTest

Accenture StormTest Development Center is the leading automated test solution for digital TV services. It is designed to reduce the cost of getting high quality digital TV services to market faster.

StormTest Development Center greatly reduces the need for time-consuming, expensive and error-prone manual testing and replaces it with a more accurate and cost-effective alternative. It scales easily to large numbers and types of devices and integrates with existing infrastructure to give much greater efficiency in testing. It can be used to verify and validate services on a virtually every piece of consumer premises equipment (CPE), from set-top boxes to games consoles and from iPads to Smart TVs. It has been specifically designed to meet the needs of developers and testers of these CPE devices and the applications which run on them.

StormTest Development Center consists of:

- A choice of hardware units that can test 1, 4, or 16 devices. Each device under test can be controlled individually and independently and the audio/video from each device can be captured and analysed to determine the outcome of the test. The StormTest hardware supports capture of audio and video over HDMI interfaces and supports all HD resolutions up to 1080p. In addition there is a hardware upgrade option for the 16 device tester that will allow native capture of UHD content.
- Server software that controls all the hardware and devices in the rack as well as managing a central repository of test scripts and a central database of test results.
- A Client API that allows test scripts to interact with the server software
- A number of graphical tools that allow the user to directly control devices connected to StormTest Development Center, to create and schedule tests to run and to view the results of those test runs.

Test scripts can be run from any location – the tester needs only a network connection to the StormTest server. Video and audio output from the devices under test can be streamed over this network to any location, allowing remote monitoring and control of testing, either within a company LAN or across a WAN. Alternatively, scheduled tests can run directly on the server, negating the need for maintaining a continuous network connection to the StormTest server.

1.2 About This Document

This document describes how to use the StormTest Dashboard web application.

1.3 Related Documentation

The StormTest user documentation set comprises of the following documents:

1) StormTest Developer Suite User’s Manual
2) StormTest Programmer’s Guide
3) StormTest Client API

4) StormTest Hardware Installation Guides (HV01, HV04, HV16)
5) StormTest Software Installation Guide
6) StormTest Server Monitor User's Manual
7) StormTest Administration Console User's Guide
8) StormTest Administration Tools User's Guide

The latest version of these documents can always be found on our support website, in the "Docs" section: https://larisa.engage.s3group.com/docman/?group_id=6.

1.4 Definitions, Acronyms and Abbreviations

Item	Description
DUT	Device Under Test
RPS	Remote Power Switch
UPS	Uninterruptible Power Supply
UI	User Interface
STB	Set Top Box
PVR	Personal Video Recorder

Table 1: Acronyms

2 Introduction

A StormTest Development Center facility has at its heart a database for storing all the results of test runs. Viewing these results can be achieved using the StormTest Developer Suite application and this is how a user that is developing test cases would normally work.

However, since there are often users of StormTest that are not involved in writing tests, but simply want to view results without the overheard of installing and using the full StormTest Developer Suite, we have developed the StormTest Dashboard – a web application for viewing StormTest script test results.

The StormTest Dashboard is installed on the StormTest Facility config server and can be accessed using a URL of a form like this:

```
http://<config_server>/stormtestdashboard
```

where <config_server> is replaced with the name/IP address of your config server.

The StormTest Dashboard can display results from both Python and Test Creator based test scripts.

2.1 High Level Functionality

The StormTest Dashboard provides the following functionality:

1. Daily over-view of slot usage in a facility
2. Script run browser, with comprehensive filters so you can view just the results of interest to you
3. Reports showing script results and trends
4. Reports showing test point results and trends
5. Test point rule definition

This document will show you how to use each of these functions provided by the Dashboard application.

2.2 Test Points

One of the new concepts introduced by the StormTest Dashboard web application is the concept of a test point.

Test points are a new way of looking at the data generated by your test script runs. Every test script will generate a pass/fail result. Within each test script it is also possible to define test steps that can also have a result and a comment which contains useful information.

Normally test scripts are written to a well-defined convention – this means that things like test step names or test script names will often reflect the intention of the test step or test script.

By looking at all the test step names recorded in the database, we can then extract useful information on the status of a particular feature that's being tested.

We will now describe a couple of concrete examples to help illustrate how this might work

2.2.1 Quality of a software feature

At a high level, there is normally a need to know how ‘good’ a particular feature is. For example, on a PVR STB, one of the major features is record and playback of events.

So a top level test point would be **PVR**

Under this test point we might want to know about how well playback is working or recording or bookmarks etc. So, you could have second level test points such as:

- **Pvr.playback**
- **Pvr.record**
- **Pvr.bookmarks**
- **Pvr.trick_modes**

And so on. At a high level, you would for example like to report that **Pvr.playback** is working well, but there is an issue with recordings.

However, if you are actually running say 100 tests, each with 20 test steps in them, then it’s likely that a lot of those tests have steps that actually start a recording. Or steps that playback a recording. However the script itself might not be specifically testing that function.

How then do you give a report on a high level feature’s status?

This is where test points come in. If you name your test steps consistently, then you can use test points to extract the overall status of each of your second level test points, and hence the status of the top level **PVR** test point.

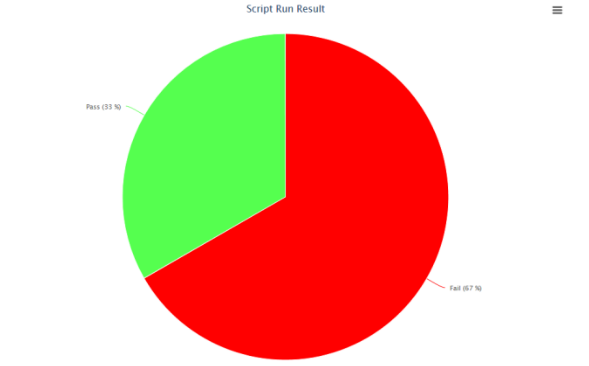


Figure 1 : Script Based Reporting

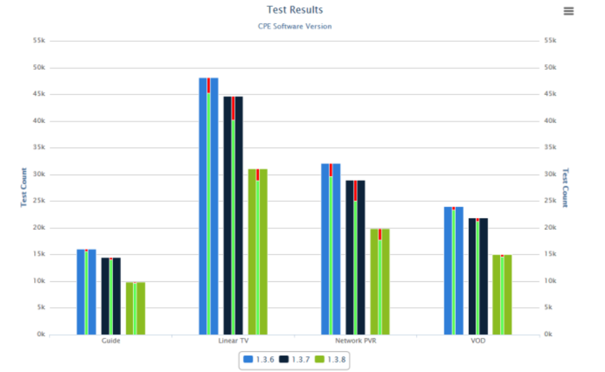


Figure 2 : Test Point Based Reporting

In this case, we aggregate the status of every test step in every test execution that was marked ‘pvr.playback’ and this will tell us the status of the playback feature. This is known as a *status* test point.

2.2.2 Performance metrics for a feature

Non-functional requirements are another important aspect of testing. In this case, you’re not so interested in whether a function works or not – you are looking at how fast (or slow) it works.

This is not a status test point – this is a *numeric* test point.

So, for example, if you have a suite of tests that are measuring the performance of your device, they can log the measured performance in the comment of a test step.

The StormTest Dashboard can then extract this data from across all the test runs and display useful information such as the maximum, minimum and average values for a particular metric.

In this way, it is quite easy to compare performance across multiple test runs.

2.2.3 User Defined Meta-data

It is quite possible to use Test Points to attach any piece of meta-data to a test script or test run. This meta-data can be any ‘key, value’ pair, but some examples of what might be used are:

- Software version number
- Middleware version number
- Test campaign ID

Then, the status of various functions (such as PVR playback) can be shown split across these user defined meta-data points.

In this case we are creating *text* test points. These test points are not interpreted in anyway by StormTest (unlike the status and numeric test points) as the meaning of them is completely defined by the end user.

2.3 Writing tests for Dashboard

In order for the Dashboard to be most useful, there are a couple of requirements on test script developers, although only one of them is mandatory

2.3.1 Call ReturnTestResult()

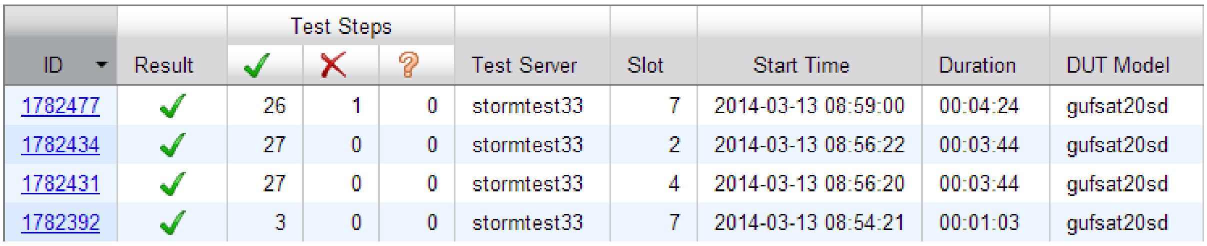
This is the only mandatory requirement. All python test scripts must call the StormTest API `ReturnTestResult()` at the end of the test to log the result into the StormTest database

2.3.2 Test Steps

Test steps are intermediate steps within a test that can have a result of their own independent of the overall test result.

If you wish to view test step results in the Dashboard, then you must begin each step in your test with a call to `BeginTestStep()` and end each step with a call to `EndTestStep()`.

Once you are doing this in your tests, the Dashboard will tell you how many steps were in each test and how many passed/failed, for example:



		Test Steps							
ID	Result	✓	✗	?	Test Server	Slot	Start Time	Duration	DUT Model
1782477	✓	26	1	0	stormtest33	7	2014-03-13 08:59:00	00:04:24	gufsat20sd
1782434	✓	27	0	0	stormtest33	2	2014-03-13 08:56:22	00:03:44	gufsat20sd
1782431	✓	27	0	0	stormtest33	4	2014-03-13 08:56:20	00:03:44	gufsat20sd
1782392	✓	3	0	0	stormtest33	7	2014-03-13 08:54:21	00:01:03	gufsat20sd

In this example, each test passed, although in the first case, one of the 27 test steps failed. This was not a critical failure as the overall test passed.

2.3.3 Test Points

To use test points, there are no specific requirements on the test script developers, other than to use some consistency in the naming of tests steps and/or test scripts so that appropriate rules can be created.

To report on test points in Dashboard, you must define test point rules. These rules use regular expressions to extract the required data from the test steps or test scripts. While any regular expression can be used, in reality a few common types are all that you need.

2.3.3.1 Regular Expression for status test point

In our earlier example, we said that the test point for PVR playback would be Pvr.playback. As long as test script writers always use this in the name or the comment of a test step that tests PVR playback, then you can setup a test point rule with a regular expression of

Pvr.playback

and this will match all those test steps across all test runs.

2.3.3.2 Regular expression for numeric test points

When logging numeric metrics, it's recommended to embed the numbers into the comment of a test step. For example, if a test step was of the form:

Step 19: Zap HD to SD : Zap measure=1.8s

Then a regular expression like this:

Zap measure=(.+)s

Will match the value reported in the comment. Everything inside the () brackets is extracted.

2.3.3.3 Regular expressions for text test points

For text test points something similar to numeric test points is used. For example, if a test step was of the form:

Step 1: CPESW = 1.3.8 TestCycle = Cycle_2014

Then two test points could be created to extract two items of information:

CPESW = ([^\s]+)

TestCycle = ([^\s]+)

There are more specific examples of how to use test points in section 6.1. Please read this section closely before using test points.

3 Dashboard Overview

The StormTest Dashboard[1] is a web application for viewing the test results stored in the database of a StormTest Development Centre facility.

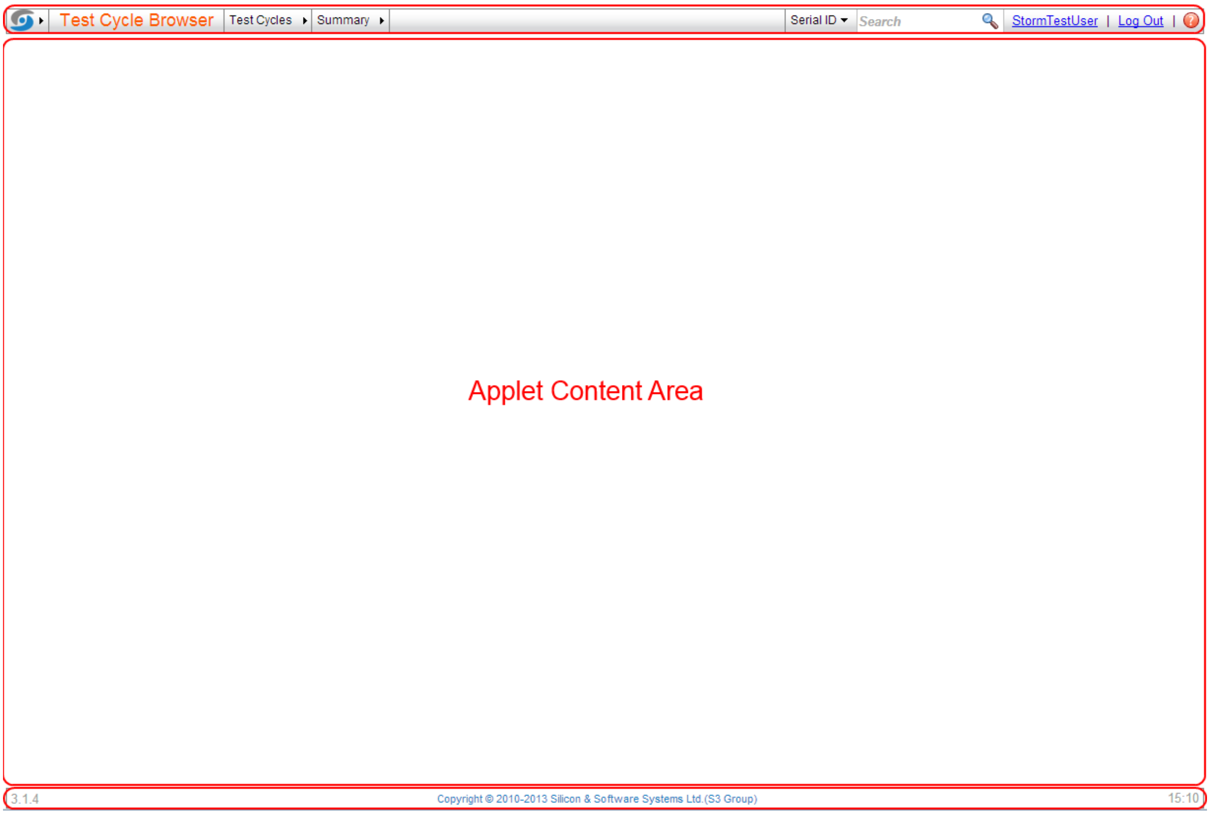

Test Cycle Browser
Test Cycles
Summary
Serial ID
Search
StormTestUser | Log Out |
Applet Content Area
3.1.4
Copyright © 2010-2013 Silicon & Software Systems Ltd.(S3 Group)
15:10

Figure 3: Dashboard Overview

The Dashboard screen consists of three sections:

- Navigation bar – this is located at the top of the screen
- Applet content area – the middle area of the screen
- Status bar – this is located at the bottom of the screen

3.1 Navigation bar

The navigation bar is located at the top of the screen and contains the following elements

- Applet Menu – user to choose the active applet. This can be *Daily Overview*, *Script Run Browser* or *Administration*. Each of these applets is dealt with individually in a section in this document.
- Selected Applet Title – displays the active applet
- Selected Applet Navigation – used for navigation within the active applet

[1] It is recommended to use Chrome or Firefox to view the Dashboard.

- Script Run Search – Search script runs by script run id
- Login User Details – view/edit details for the logged in user (refer to section 6.2.2)
- Logout

3.2 Applet content area

The applet content area fills the middle section of the screen.

3.3 Status bar

The status bar is located at the bottom of the screen and contains the following elements

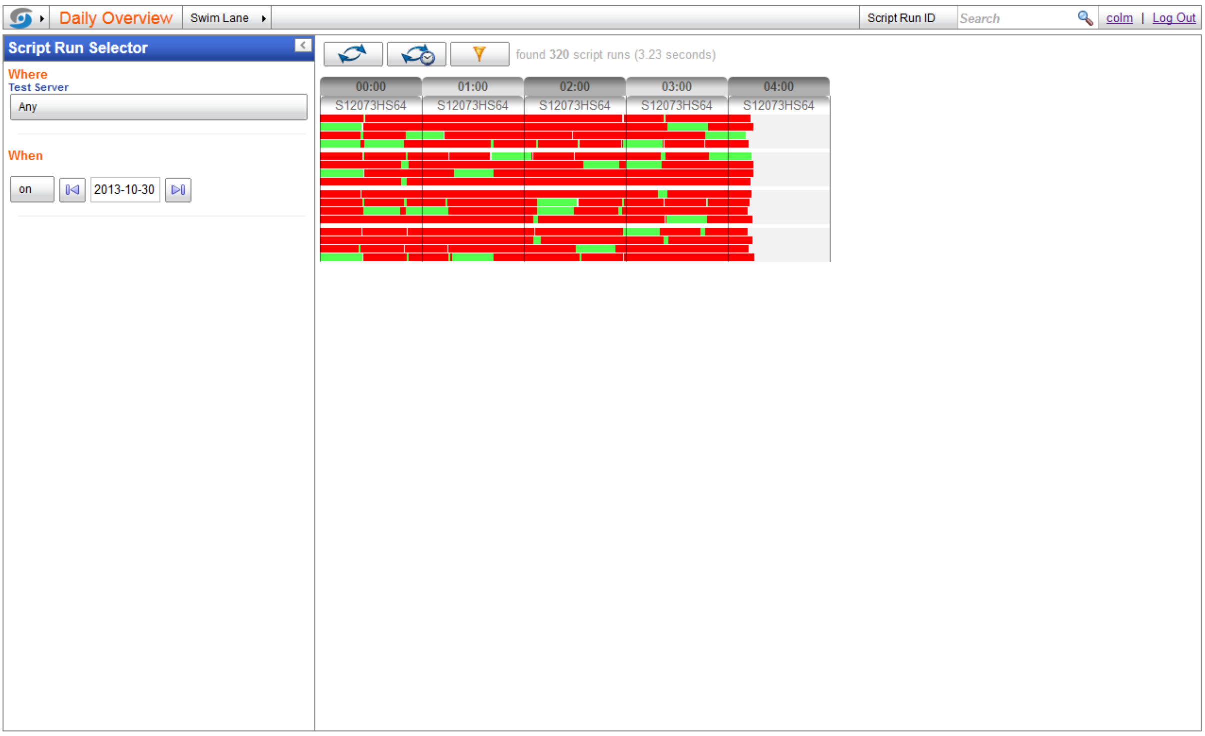
- Dashboard Version Number
- Copyright notice
- Test Facility Name
- Test Facility Local Time

4 Daily Overview

The Daily Overview applet shows an overview of all script run in a facility on a given day.

4.1 Swim Lane

The Swim Lane is a graphical overview of all script runs matched by the selection criteria. Script runs are colour coded according to their status (pass/fail).



Where

Test Server: Any

When: on 2013-10-30

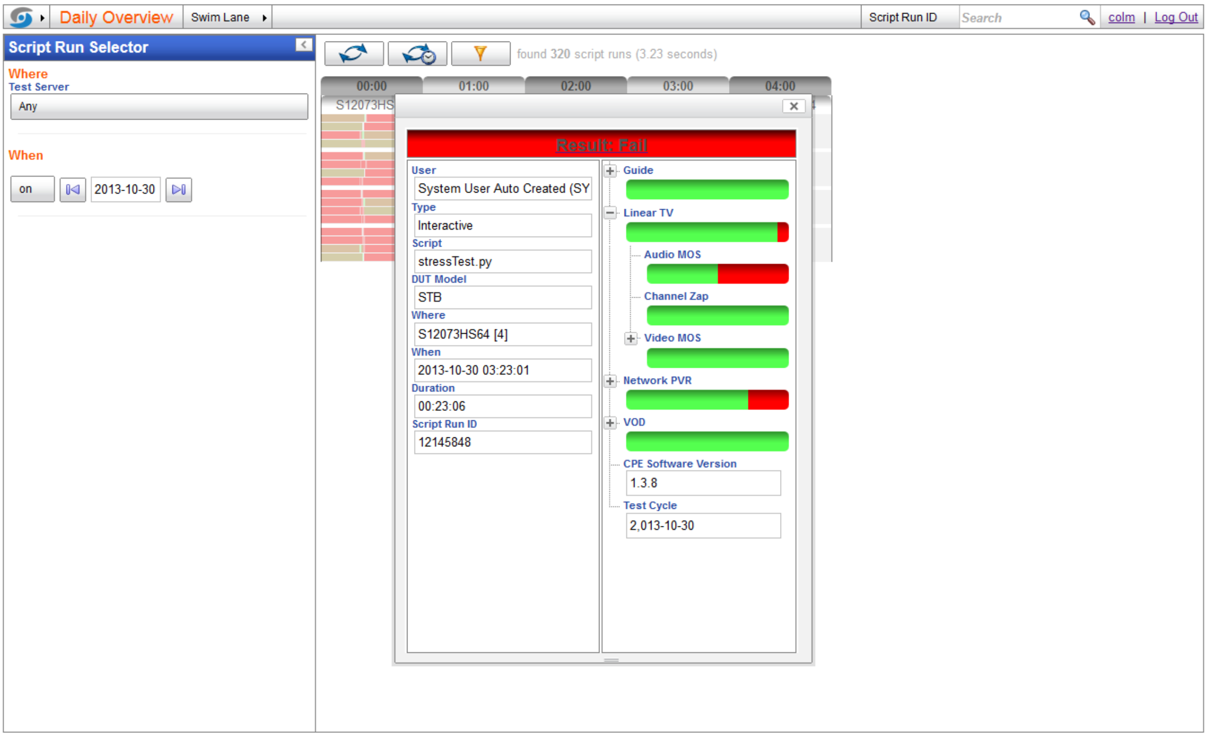
found 320 script runs (3.23 seconds)

Figure 4: Daily Overview - Swim Lane

The following features are available in the Swim Lane:

- Auto scroll on refresh
- Auto refresh
- Click and Drag to scroll
- Click on a test cycle to see more details
- Show/Hide Script Runs by clicking on Swim Lane Legend/Filter

Clicking on a script run brings up a dialog showing details for the selected test cycle. An example is shown in Figure 5.



Where
Test Server
Any

When
on 2013-10-30

User
System User Auto Created (SY
Type
Interactive
Script
stressTest.py
DUT Model
STB
Where
S12073HS64 [4]
When
2013-10-30 03:23:01
Duration
00:23:06
Script Run ID
12145848

CPE Software Version
1.3.8
Test Cycle
2,013-10-30

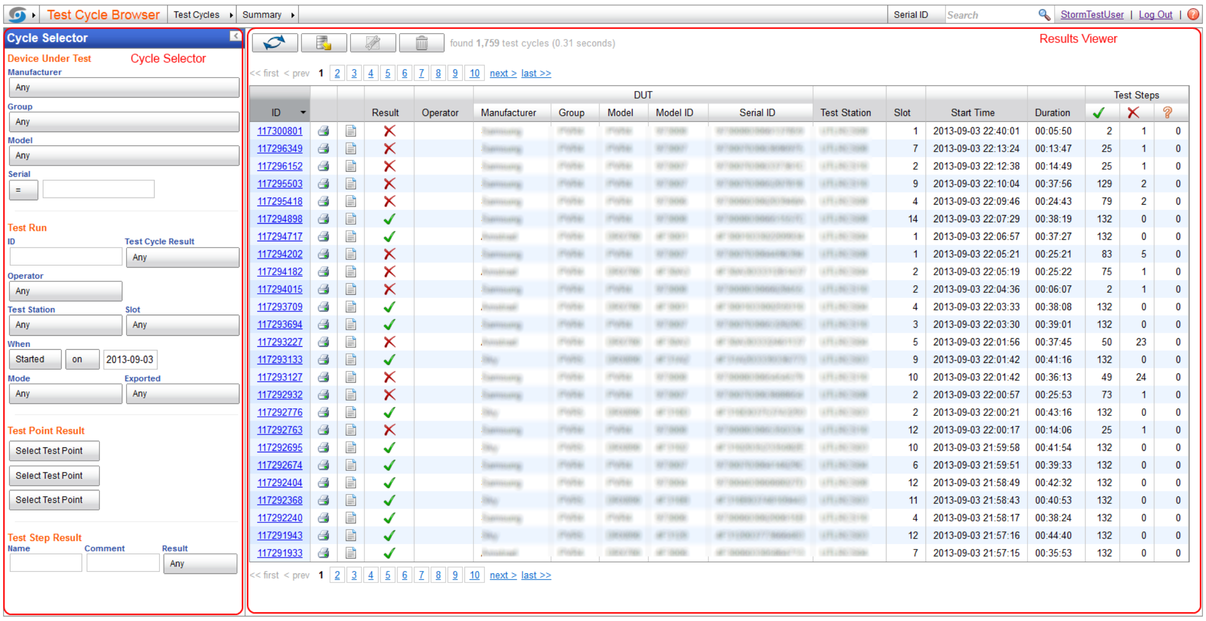
Figure 5: Daily Overview - Swim Lane Script Run Details Dialog

Clicking on the script run result opens the script run in the Script Run Browser.

5 Script Run Browser

The Script Run Browser applet as shown in **Error! Reference source not found.** allows the user to:

- Select script runs based on a range of search criteria
- Choose from multiple views of the selected script runs



Device Under Test

Manufacturer: Any

Group: Any

Model: Any

Serial: =

Test Run

ID:

Test Cycle Result: Any

Operator: Any

Test Station: Any

Slot: Any

When: Started on 2013-09-03

Mode: Any

Exported: Any

Test Point Result

Select Test Point

Select Test Point

Select Test Point

Test Step Result

Name:

Comment:

Result: Any

found 1,759 test cycles (0.31 seconds)

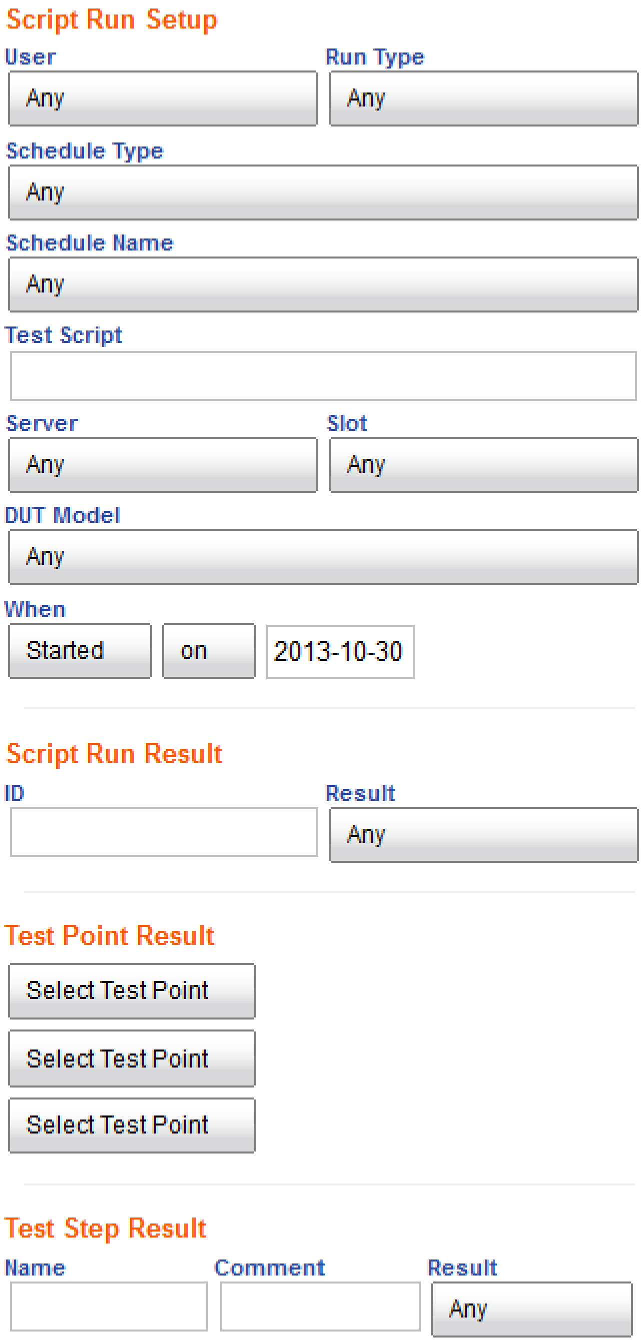
<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

					DUT										Test Steps		
ID			Result	Operator	Manufacturer	Group	Model	Model ID	Serial ID	Test Station	Slot	Start Time	Duration		✓	✗	?
117300801			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	2013-09-03 22:40:01	00:05:50		2	1	0
117296349			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	7	2013-09-03 22:13:24	00:13:47		25	1	0
117296152			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	2013-09-03 22:12:38	00:14:49		25	1	0
117295503			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	9	2013-09-03 22:10:04	00:37:56		129	2	0
117295418			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	2013-09-03 22:09:46	00:24:43		79	2	0
117294898			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	14	2013-09-03 22:07:29	00:38:19		132	0	0
117294717			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	2013-09-03 22:06:57	00:37:27		132	0	0
117294202			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	2013-09-03 22:05:21	00:25:21		83	5	0
117294182			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	2013-09-03 22:05:19	00:25:22		75	1	0
117294015			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	2013-09-03 22:04:36	00:06:07		2	1	0
117293709			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	2013-09-03 22:03:33	00:38:08		132	0	0
117293694			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	2013-09-03 22:03:30	00:39:01		132	0	0
117293227			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	5	2013-09-03 22:01:56	00:37:45		50	23	0
117293133			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	9	2013-09-03 22:01:42	00:41:16		132	0	0
117293127			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	10	2013-09-03 22:01:42	00:36:13		49	24	0
117292932			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	2013-09-03 22:00:57	00:25:53		73	1	0
117292776			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	2013-09-03 22:00:21	00:43:16		132	0	0
117292763			✗		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12	2013-09-03 22:00:17	00:14:06		25	1	0
117292695			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	10	2013-09-03 21:59:58	00:41:54		132	0	0
117292674			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6	2013-09-03 21:59:51	00:39:33		132	0	0
117292404			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12	2013-09-03 21:58:49	00:42:32		132	0	0
117292368			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	11	2013-09-03 21:58:43	00:40:53		132	0	0
117292240			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4	2013-09-03 21:58:17	00:38:24		132	0	0
117291943			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12	2013-09-03 21:57:16	00:44:40		132	0	0
117291933			✓		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	7	2013-09-03 21:57:15	00:35:53		132	0	0

<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

Figure 6: Script Run Selector & Results Viewer

5.1 Script Run Selector



Script Run Setup

User

Any

Run Type

Any

Schedule Type

Any

Schedule Name

Any

Test Script

Server

Any

Slot

Any

DUT Model

Any

When

Started on 2013-10-30

Script Run Result

ID

Result

Any

Test Point Result

Select Test Point

Select Test Point

Select Test Point

Test Step Result

Name

Comment

Result

Any

Figure 7: Cycle Selector

The Script Run Selector allows the user to retrieve script run results using any combination of the search criteria described in the following sections.

5.1.1 User

The User drop down list contains a list of users in the local facility. Multiple values can be selected.

5.1.2 Run Type

The Run Type drop down allows users to select script runs based on how the script was launched. The following options are available:

- Client Daemon (i.e. show tests that were executed using the StormTest Scheduler)
- Command Line (i.e. show tests run in some other way, either from a command line or from within a development environment such as eclipse)

5.1.3 Schedule Type

The Schedule Type drop down list allows users to select scripts based on schedule type. Multiple values can be selected.

This selector is only available if “Run Type” is set to “Any” or “Client Daemon”.

5.1.4 Schedule Name

The Schedule Name drop down list allows users to select scripts based on schedule name. Multiple values can be selected.

The list of available schedules is pre-filtered by the User and Schedule Type selectors.

This selector is only available if “Run Type” is set to “Any” or “Client Daemon”.

5.1.5 Test Script

The Test Script input field where uses can select scripts based on script name. Partial or full script names can be used.

5.1.6 Server

The Server selector allows users to select script runs based on which server the script was run on. Multiple values can be selected.

5.1.7 Slot

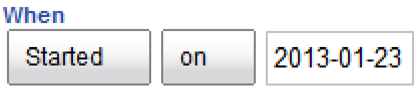
The Slot selector allows users to select script runs based on which slot number the script was run on. Multiple values can be selected.

5.1.8 DUT Model

The DUT Model selector allows users to select script runs based on which DUT model the script was run on. Multiple values can be selected.

5.1.9 When

This set of selectors allows the user to specify when a script was run.



When

Started

on

2013-01-23

Figure 8: Script Run Selector - When

The when drop down list Date drop down list contains a list of date options. For options involving a date choice an input field with a popup calendar is shown.

5.1.10 Script Run ID

The Script Run ID input field is a text field where you can specify the Script Run ID using one of the following formats:

- Exact ID e.g. 7117424

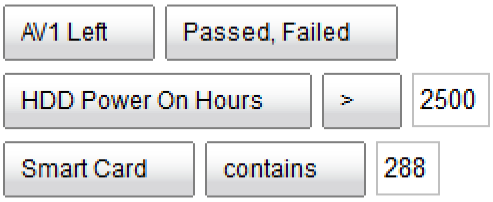
- List of IDs e.g. 7117424,7117425,7117426
- Range of IDs e.g. 7114025-7117424
- Min ID e.g. >7114025
- Max ID e.g. <7117424

5.1.11 Script Run Result

The Script Run Result selector allows users to select script runs based on script run result. Multiple values can be selected.

5.1.12 Test Point

This set of selectors allows you to specify script runs by test point result.



AV1 Left | Passed, Failed

HDD Power On Hours | > | 2500

Smart Card | contains | 288

Figure 9: Some Example Test Point Selectors

5.1.12.1 Test Point

The Test Point drop down contains list of available test points. The required test point can be found by selecting from the list or by typing part of the name and then selecting from the reduced list.

5.1.12.2 Operator

The list of available operators depends on the type of the chosen test point.

5.1.12.3 Result

Depending on the type of the chosen test point the user will be presented with either a drop down list or an input field. For definition of Test Point Types, refer to section 0.

In the drop down list multiple values can be selected.

Test Point Type	Available Operators	Result
Status	=, !=	Any/Pass/Warn/Fail
Text	=, !=, starts with, ends with, contains	Input Field
Numeric	=, !=, <, >	Input Field

Table 2: Test Point Selector options

Note that if a test point has been fired multiple times in a script run the overall result is used.

5.1.13 Test Step

This set of selectors allows you to specify script runs containing certain test steps. Use of this selector can cause longer than normal search times and should be used sparingly.

5.1.13.1 Name

Input field where you can specify the test step name. The search is case insensitive and sub-string matching is used. For example “Left” will match the following:

- TID_AVOUT_AV_ANALYSIS_DVI_ANALOGAUDIO_**LEFT**_RUN2

- TID_AVOUT_DIGITALCOAXIAL_**LEFT**_AUDIO_ANALYSIS_RUN1

5.1.13.2 Comment

Input field where you can specify the test step comment. The search is case insensitive and sub-string matching is used.

5.1.13.3 Result

Drop down list of all possible test step result values. Multiple values can be selected.

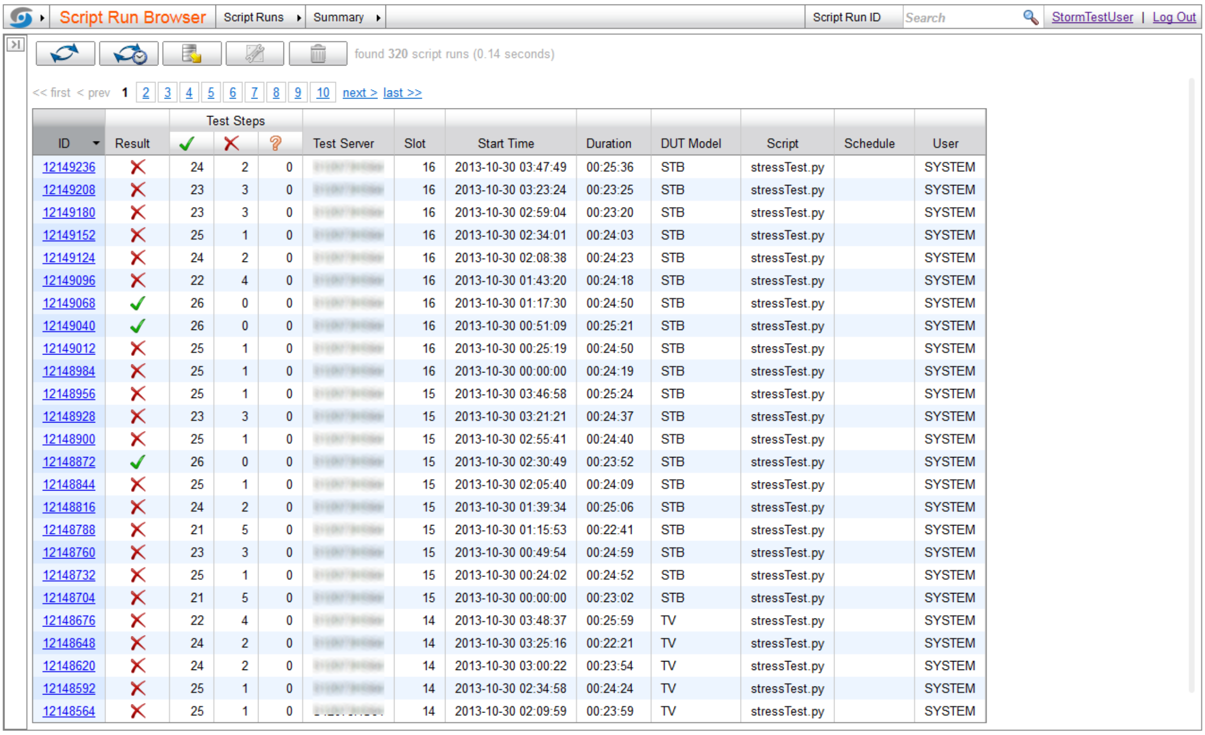
5.2 Results Viewer

The Results Viewer provides the following views into the script runs found by the Script Run Selector

- Script Runs
- Test Point Results
- Test Point Trends
- Script Run Count
- Script Run Result
- Script Run Trends

5.2.1 Script Runs View

The Script Runs View shows results for script runs matched by the selector in a table format referred to here as Script Runs Table.



found 320 script runs (0.14 seconds)

<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

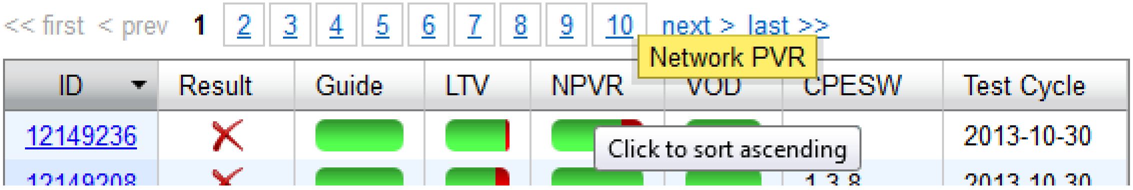
ID	Result	Test Steps			Test Server	Slot	Start Time	Duration	DUT Model	Script	Schedule	User
		✓	✗	?								
12149236	✗	24	2	0	[redacted]	16	2013-10-30 03:47:49	00:25:36	STB	stressTest.py		SYSTEM
12149208	✗	23	3	0	[redacted]	16	2013-10-30 03:23:24	00:23:25	STB	stressTest.py		SYSTEM
12149180	✗	23	3	0	[redacted]	16	2013-10-30 02:59:04	00:23:20	STB	stressTest.py		SYSTEM
12149152	✗	25	1	0	[redacted]	16	2013-10-30 02:34:01	00:24:03	STB	stressTest.py		SYSTEM
12149124	✗	24	2	0	[redacted]	16	2013-10-30 02:08:38	00:24:23	STB	stressTest.py		SYSTEM
12149096	✗	22	4	0	[redacted]	16	2013-10-30 01:43:20	00:24:18	STB	stressTest.py		SYSTEM
12149068	✓	26	0	0	[redacted]	16	2013-10-30 01:17:30	00:24:50	STB	stressTest.py		SYSTEM
12149040	✓	26	0	0	[redacted]	16	2013-10-30 00:51:09	00:25:21	STB	stressTest.py		SYSTEM
12149012	✗	25	1	0	[redacted]	16	2013-10-30 00:25:19	00:24:50	STB	stressTest.py		SYSTEM
12148984	✗	25	1	0	[redacted]	16	2013-10-30 00:00:00	00:24:19	STB	stressTest.py		SYSTEM
12148956	✗	25	1	0	[redacted]	15	2013-10-30 03:46:58	00:25:24	STB	stressTest.py		SYSTEM
12148928	✗	23	3	0	[redacted]	15	2013-10-30 03:21:21	00:24:37	STB	stressTest.py		SYSTEM
12148900	✗	25	1	0	[redacted]	15	2013-10-30 02:55:41	00:24:40	STB	stressTest.py		SYSTEM
12148872	✓	26	0	0	[redacted]	15	2013-10-30 02:30:49	00:23:52	STB	stressTest.py		SYSTEM
12148844	✗	25	1	0	[redacted]	15	2013-10-30 02:05:40	00:24:09	STB	stressTest.py		SYSTEM
12148816	✗	24	2	0	[redacted]	15	2013-10-30 01:39:34	00:25:06	STB	stressTest.py		SYSTEM
12148788	✗	21	5	0	[redacted]	15	2013-10-30 01:15:53	00:22:41	STB	stressTest.py		SYSTEM
12148760	✗	23	3	0	[redacted]	15	2013-10-30 00:49:54	00:24:59	STB	stressTest.py		SYSTEM
12148732	✗	25	1	0	[redacted]	15	2013-10-30 00:24:02	00:24:52	STB	stressTest.py		SYSTEM
12148704	✗	21	5	0	[redacted]	15	2013-10-30 00:00:00	00:23:02	STB	stressTest.py		SYSTEM
12148676	✗	22	4	0	[redacted]	14	2013-10-30 03:48:37	00:25:59	TV	stressTest.py		SYSTEM
12148648	✗	24	2	0	[redacted]	14	2013-10-30 03:25:16	00:22:21	TV	stressTest.py		SYSTEM
12148620	✗	24	2	0	[redacted]	14	2013-10-30 03:00:22	00:23:54	TV	stressTest.py		SYSTEM
12148592	✗	25	1	0	[redacted]	14	2013-10-30 02:34:58	00:24:24	TV	stressTest.py		SYSTEM
12148564	✗	25	1	0	[redacted]	14	2013-10-30 02:09:59	00:23:59	TV	stressTest.py		SYSTEM

Figure 10: Script Runs Table – Summary View

The table is paged displaying only 25 rows at a time. The pagination controls can be used to navigate between pages.

By default, the table is sorting by script run ID is descending order. Sorting can be changed by clicking on any of the column headers.

Tooltips are provided on column headers as shown in **Error! Reference source not found.**.



<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

ID	Result	Guide	LTV	NPVR	VOD	CPESW	Test Cycle
12149236	✗	🟩	🟩	🟩	🟩		2013-10-30

Figure 11 : Test Cycle Browser - Test Cycle Table Tooltip

The table columns are grouped into the following views:

- Summary
- Test Points

The selection of which view is displayed is made using the drop down menu in the navigation bar as shown in **Error! Reference source not found.**.

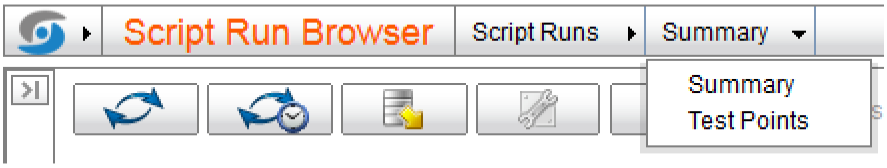


Figure 12: Script Runs Table: View Selector

5.2.1.1 *Summary View*

This view shows summary information for each script run, such as what was tested, where and when it was tested and what the result was.

The ID column shows the script run ID.

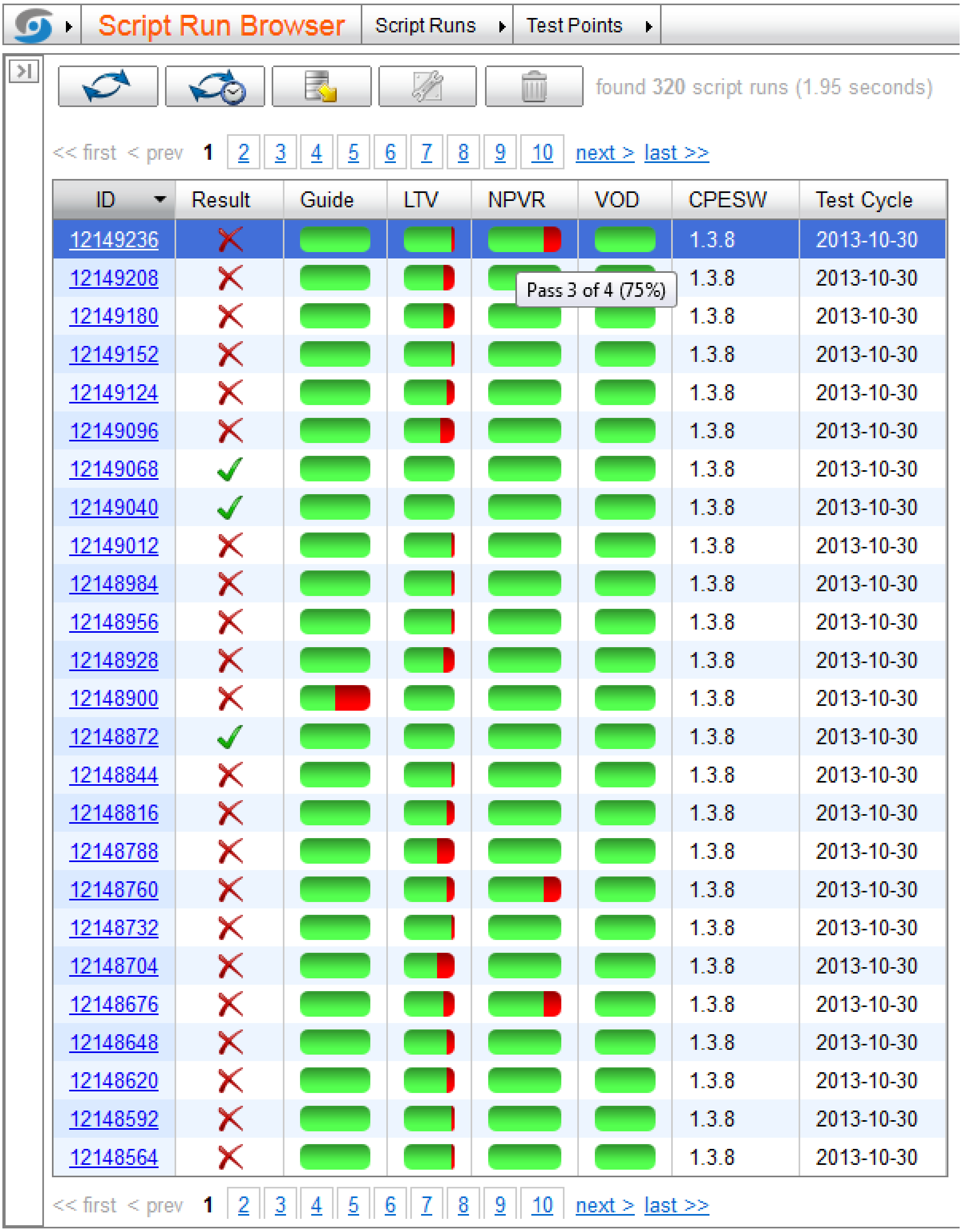
The Result column contains icons indicating the result for each script run. The meaning of the icons is as shown in **Error! Reference source not found.**.

Icon	Result
▯▮	Running
✓	Passed
✗	Failed
⊗	Killed
⚠	Partial Pass
💣	Ran out of time
🛑	Error/Unknown

Table 3: Script Run Result Icons

5.2.1.2 *Test Points View*

This view shows Test Point results for the script runs that match the selection criteria.



found 320 script runs (1.95 seconds)

<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

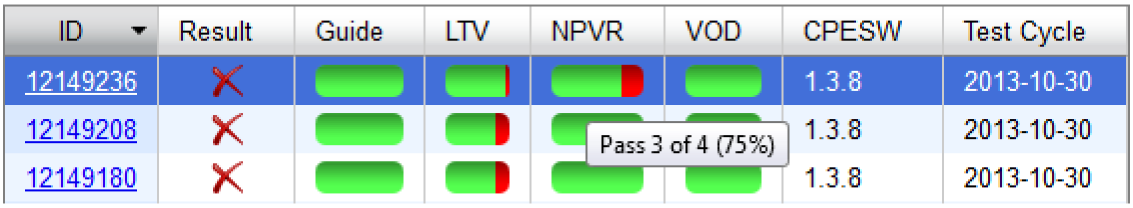
ID	Result	Guide	LTV	NPVR	VOD	CPESW	Test Cycle
12149236	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12149208	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149180	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149152	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149124	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149096	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149068	✓	🟩	🟩	🟩	🟩	1.3.8	2013-10-30
12149040	✓	🟩	🟩	🟩	🟩	1.3.8	2013-10-30
12149012	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148984	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148956	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148928	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148900	✗	🟩🟥	🟩	🟩	🟩	1.3.8	2013-10-30
12148872	✓	🟩	🟩	🟩	🟩	1.3.8	2013-10-30
12148844	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148816	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148788	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148760	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12148732	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148704	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148676	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12148648	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148620	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148592	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148564	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30

Pass 3 of 4 (75%)

<< first < prev 1 2 3 4 5 6 7 8 9 10 next > last >>

Hovering over a status Test Point result shows pass/warn/fail details.

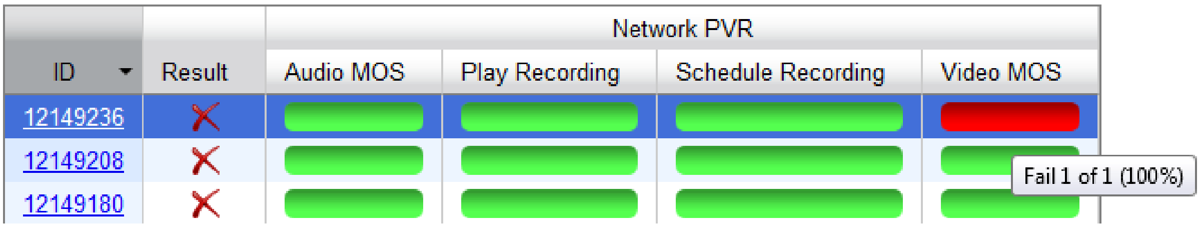
Clicking on a status Test Point result selects the next level down in the Test Point tree. An example of this is shown in the following sequence of images.



ID	Result	Guide	LTV	NPVR	VOD	CPESW	Test Cycle
12149236	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12149208	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149180	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30

Pass 3 of 4 (75%)

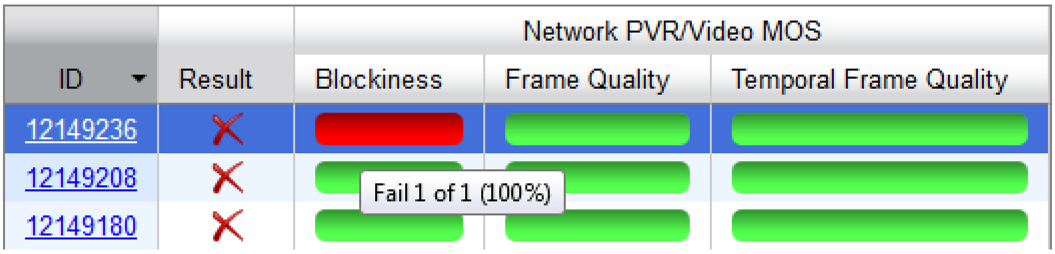
Figure 13: Script Run Browser - Test Point Level 1



		Network PVR			
ID	Result	Audio MOS	Play Recording	Schedule Recording	Video MOS
12149236	✗	🟩	🟩	🟩	🟥
12149208	✗	🟩	🟩	🟩	🟩
12149180	✗	🟩	🟩	🟩	🟩

Fail 1 of 1 (100%)

Figure 14: Script Run Browser - Test Point Level 2



		Network PVR/Video MOS		
ID	Result	Blockiness	Frame Quality	Temporal Frame Quality
12149236	✗	🟥	🟩	🟩
12149208	✗	🟩	🟩	🟩
12149180	✗	🟩	🟩	🟩

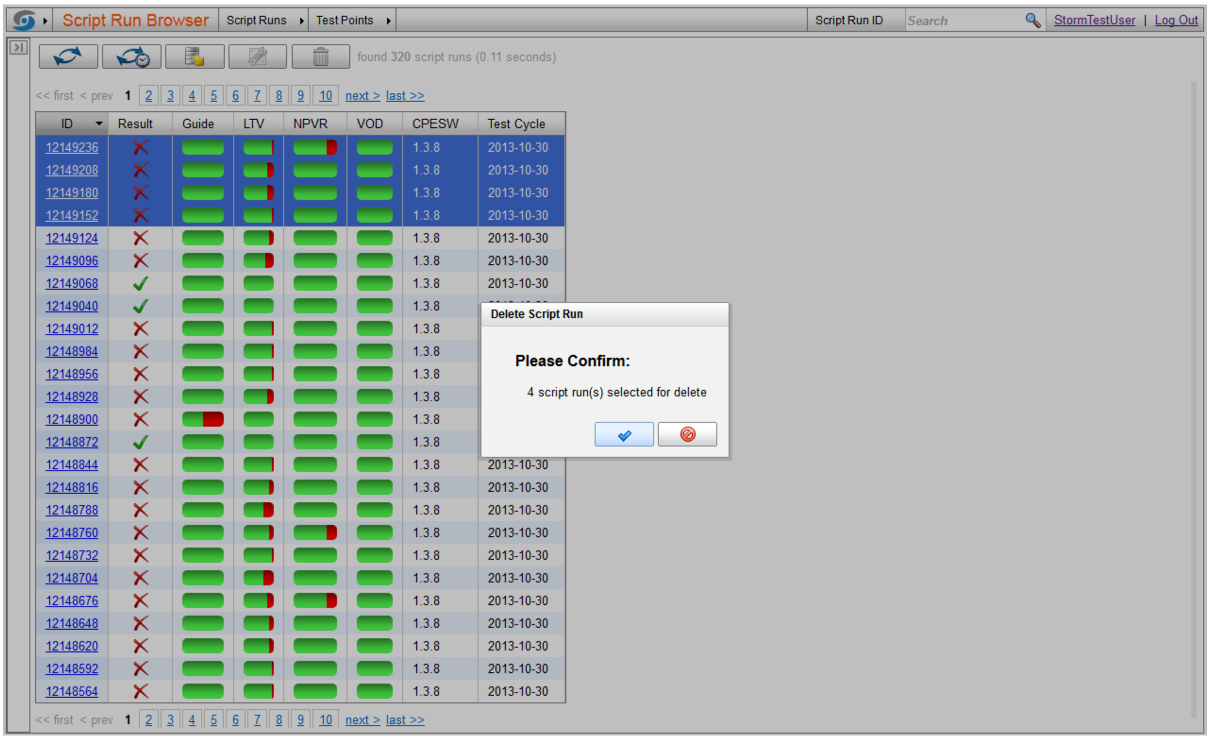
Fail 1 of 1 (100%)

Figure 15: Script Run Browser - Test Point Level 3

Clicking on the ⬆ button selects the next level up in the Test Point tree.

5.2.2 Deleting Script Runs

Script runs can be deleted by selecting them and then clicking on the 🗑 button. Multiple script runs can be selected by using CTRL+click or using SHIFT+click.



found 320 script runs (0.11 seconds)

ID	Result	Guide	LTV	NPVR	VOD	CPESW	Test Cycle
12149236	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12149208	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149180	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149152	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149124	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149096	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12149068	✓	🟩	🟩	🟩	🟩	1.3.8	2013-10-30
12149040	✓	🟩	🟩	🟩	🟩	1.3.8	
12149012	✗	🟩	🟩🟥	🟩	🟩	1.3.8	
12148984	✗	🟩	🟩🟥	🟩	🟩	1.3.8	
12148956	✗	🟩	🟩🟥	🟩	🟩	1.3.8	
12148928	✗	🟩	🟩🟥	🟩	🟩	1.3.8	
12148900	✗	🟩🟥	🟩	🟩	🟩	1.3.8	
12148872	✓	🟩	🟩	🟩	🟩	1.3.8	
12148844	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148816	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148788	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148760	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12148732	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148704	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148676	✗	🟩	🟩🟥	🟩🟥	🟩	1.3.8	2013-10-30
12148648	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148620	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148592	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30
12148564	✗	🟩	🟩🟥	🟩	🟩	1.3.8	2013-10-30

Delete Script Run

Please Confirm:

4 script run(s) selected for delete

Figure 16: Deleting test cycles

Please note the following

- Deleting a script run completely removes all record of the script run
- Deleting test cycles is only available to Admin users

5.2.3 Rebuilding Script Run Results

Script run results can be rebuilt by selecting them and then clicking on the button. Multiple script runs can be selected by using CTRL+click or using SHIFT+click. Note that this functionality is only available to administrators.

5.2.4 Exporting Script Runs

Details for all script runs matched by the cycle selector can be exported to a CSV file by clicking on the button.

5.2.5 Script Run Detail View

Clicking on the Script Run ID displays one of the following three Script Run Details views:

- Test Steps
- Test Points
- Test Log

The selected view can be changed using the menu in the navigation bar as shown in **Error! Reference source not found.**.

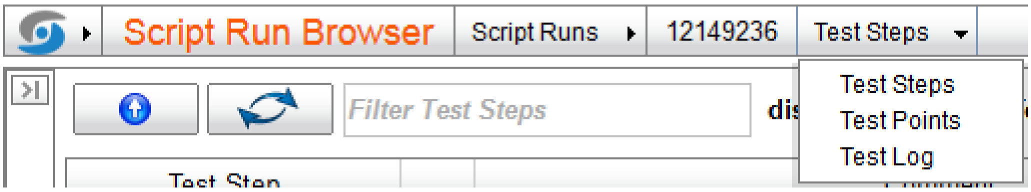
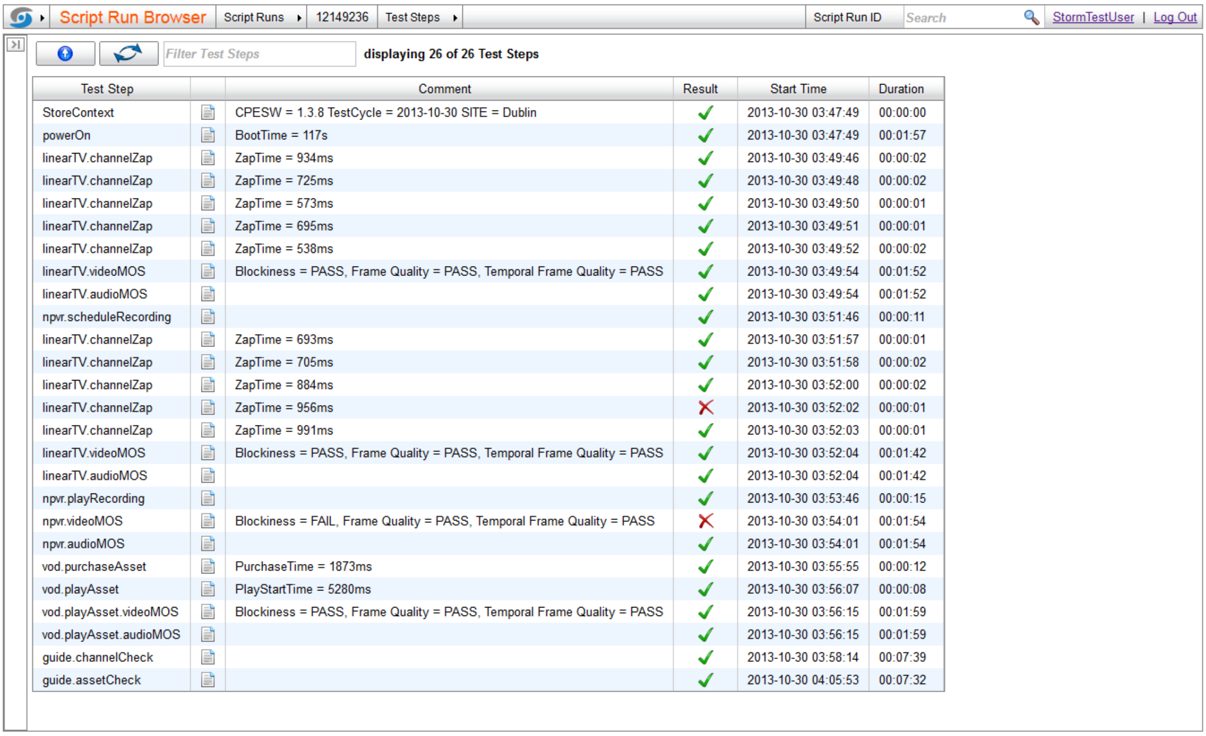


Figure 17: Script Run Detail View Selector

5.2.5.1 *Script Run Detail - Test Steps View*

The Test Steps view displays all test steps for the currently selected script run.



displaying 26 of 26 Test Steps

Test Step		Comment	Result	Start Time	Duration
StoreContext	📄	CPESW = 1.3.8 TestCycle = 2013-10-30 SITE = Dublin	✓	2013-10-30 03:47:49	00:00:00
powerOn	📄	BootTime = 117s	✓	2013-10-30 03:47:49	00:01:57
linearTV.channelZap	📄	ZapTime = 934ms	✓	2013-10-30 03:49:46	00:00:02
linearTV.channelZap	📄	ZapTime = 725ms	✓	2013-10-30 03:49:48	00:00:02
linearTV.channelZap	📄	ZapTime = 573ms	✓	2013-10-30 03:49:50	00:00:01
linearTV.channelZap	📄	ZapTime = 695ms	✓	2013-10-30 03:49:51	00:00:01
linearTV.channelZap	📄	ZapTime = 538ms	✓	2013-10-30 03:49:52	00:00:02
linearTV.videoMOS	📄	Blockiness = PASS, Frame Quality = PASS, Temporal Frame Quality = PASS	✓	2013-10-30 03:49:54	00:01:52
linearTV.audioMOS	📄		✓	2013-10-30 03:49:54	00:01:52
npvr.scheduleRecording	📄		✓	2013-10-30 03:51:46	00:00:11
linearTV.channelZap	📄	ZapTime = 693ms	✓	2013-10-30 03:51:57	00:00:01
linearTV.channelZap	📄	ZapTime = 705ms	✓	2013-10-30 03:51:58	00:00:02
linearTV.channelZap	📄	ZapTime = 884ms	✓	2013-10-30 03:52:00	00:00:02
linearTV.channelZap	📄	ZapTime = 956ms	✗	2013-10-30 03:52:02	00:00:01
linearTV.channelZap	📄	ZapTime = 991ms	✓	2013-10-30 03:52:03	00:00:01
linearTV.videoMOS	📄	Blockiness = PASS, Frame Quality = PASS, Temporal Frame Quality = PASS	✓	2013-10-30 03:52:04	00:01:42
linearTV.audioMOS	📄		✓	2013-10-30 03:52:04	00:01:42
npvr.playRecording	📄		✓	2013-10-30 03:53:46	00:00:15
npvr.videoMOS	📄	Blockiness = FAIL, Frame Quality = PASS, Temporal Frame Quality = PASS	✗	2013-10-30 03:54:01	00:01:54
npvr.audioMOS	📄		✓	2013-10-30 03:54:01	00:01:54
vod.purchaseAsset	📄	PurchaseTime = 1873ms	✓	2013-10-30 03:55:55	00:00:12
vod.playAsset	📄	PlayStartTime = 5280ms	✓	2013-10-30 03:56:07	00:00:08
vod.playAsset.videoMOS	📄	Blockiness = PASS, Frame Quality = PASS, Temporal Frame Quality = PASS	✓	2013-10-30 03:56:15	00:01:59
vod.playAsset.audioMOS	📄		✓	2013-10-30 03:56:15	00:01:59
guide.channelCheck	📄		✓	2013-10-30 03:58:14	00:07:39
guide.assetCheck	📄		✓	2013-10-30 04:05:53	00:07:32

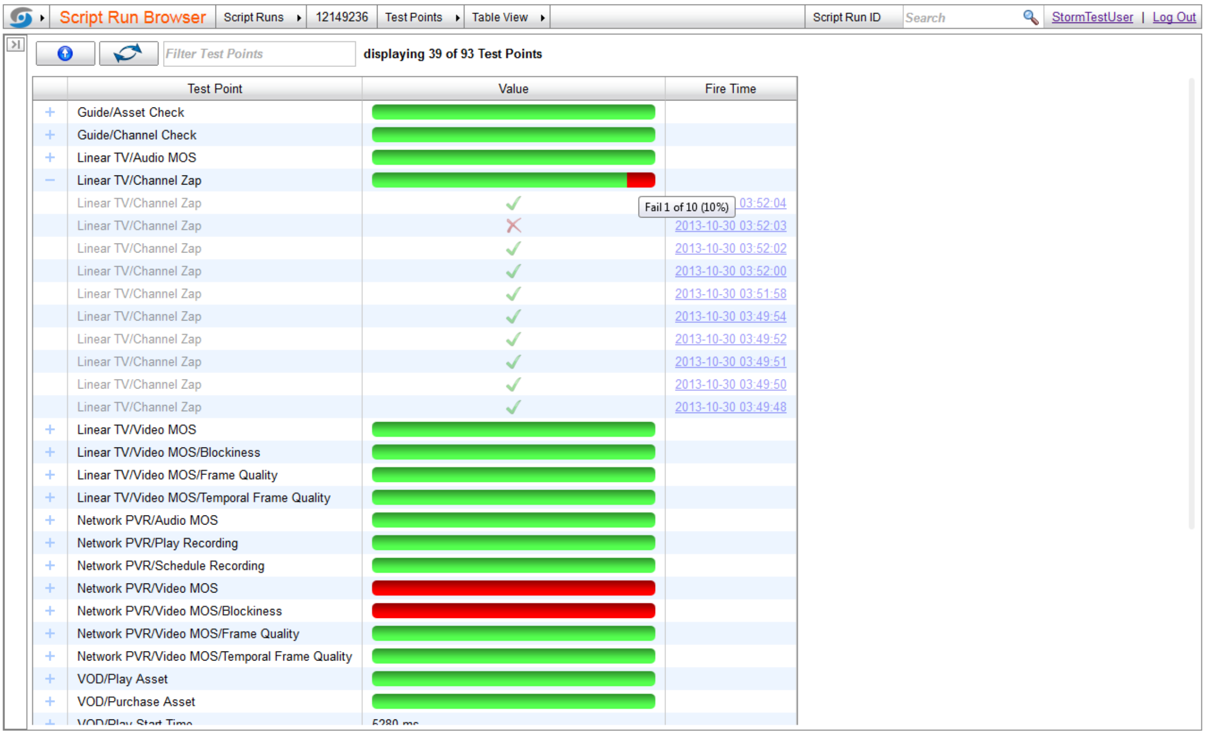
Figure 18: Script Run Detail - Test Steps View

The 📄 icon switches to the "Test Log" view and opens the node for the selected Test Step. Please note that test logs are only available when the test cycle has finished.

5.2.5.2 *Script Run Detail - Test Points View*

The Script Run Detail - Test Points view displays Test Points for the currently selected script run. Test Points can be displayed as either a table or tree view.

5.2.5.2.1 Table View



displaying 39 of 93 Test Points

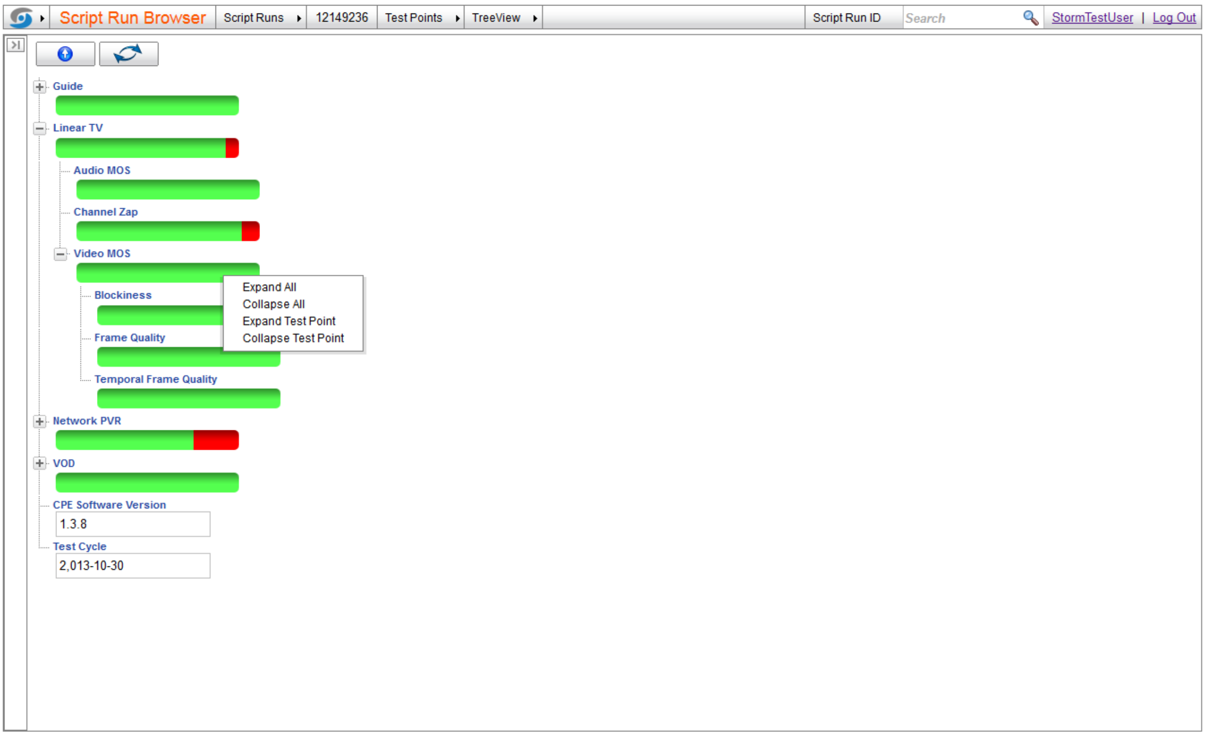
	Test Point	Value	Fire Time
+	Guide/Asset Check	🟩	
+	Guide/Channel Check	🟩	
+	Linear TV/Audio MOS	🟩	
−	Linear TV/Channel Zap	🟩🟥	
	Linear TV/Channel Zap	✓	Fail 1 of 10 (10%) 03:52:04
	Linear TV/Channel Zap	✗	2013-10-30 03:52:03
	Linear TV/Channel Zap	✓	2013-10-30 03:52:02
	Linear TV/Channel Zap	✓	2013-10-30 03:52:00
	Linear TV/Channel Zap	✓	2013-10-30 03:51:58
	Linear TV/Channel Zap	✓	2013-10-30 03:49:54
	Linear TV/Channel Zap	✓	2013-10-30 03:49:52
	Linear TV/Channel Zap	✓	2013-10-30 03:49:51
	Linear TV/Channel Zap	✓	2013-10-30 03:49:50
	Linear TV/Channel Zap	✓	2013-10-30 03:49:48
+	Linear TV/Video MOS	🟩	
+	Linear TV/Video MOS/Blockiness	🟩	
+	Linear TV/Video MOS/Frame Quality	🟩	
+	Linear TV/Video MOS/Temporal Frame Quality	🟩	
+	Network PVR/Audio MOS	🟩	
+	Network PVR/Play Recording	🟩	
+	Network PVR/Schedule Recording	🟩	
+	Network PVR/Video MOS	🟥	
+	Network PVR/Video MOS/Blockiness	🟥	
+	Network PVR/Video MOS/Frame Quality	🟩	
+	Network PVR/Video MOS/Temporal Frame Quality	🟩	
+	VOD/Play Asset	🟩	
+	VOD/Purchase Asset	🟩	
+	VOD/Play Start Time	5280 ms	

Figure 19: Script Run Detail - Test Points Table View

The overall result for each test point is displayed. Details of individual test point firing can be seen by expanding a test point as shown in XXX.

For test points that are using a Test Step trigger clicking on the Fire Time link will jump to the “Test Step” view and highlight the test step that fired the test point.

5.2.5.2.2 Tree View



CPE Software Version: 1.3.8

Test Cycle: 2,013-10-30

Test Point nodes or be expanded by clicking on the ⊞ icon or collapsed by clicking on the ⊟ icon.

Right-clicking brings up a context menu where the user can expand or collapse the whole tree or individual nodes.

Note that for numeric test points the average value for the selected script run is displayed.

5.2.5.3 Script Run Detail – Test Log View

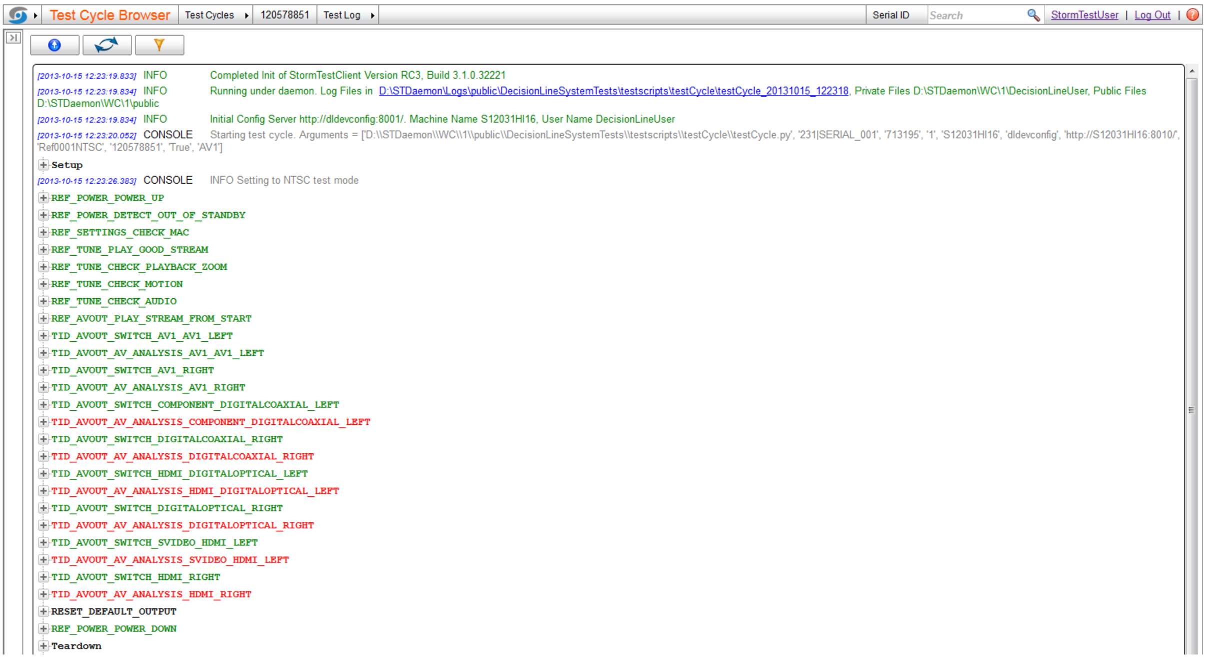
The Script Run Detail - Test Log view contains detailed debug information from the test cycle run. The log is presented as a tree structure of expandable Test Steps nodes.

Test Step nodes or log region nodes can be expanded[2] by clicking on the ⊞ icon or collapsed by clicking on the ⊟ icon.

The log viewer also provides the following features:

- Inline viewing of captured images
- Highlighting of OCR area in captured images
- Highlighting of area of interest used in image comparison
- Image Zoomer for captured images

[2] Right-clicking within a node brings up a context menu where the user can expand or collapse the node



```
[2013-10-15 12:23:19.833]  INFO      Completed Init of StormTestClient Version RC3, Build 3.1.0.32221
[2013-10-15 12:23:19.834]  INFO      Running under daemon. Log Files in  D:\STDaemon\Logs\public\DecisionLineSystemTests\testscripts\testCycle\testCycle_20131015_122318, Private Files D:\STDaemon\WC\1\DecisionLineUser, Public Files
D:\STDaemon\WC\1\public
[2013-10-15 12:23:19.834]  INFO      Initial Config Server http://dldevconfig:8001/. Machine Name S12031HI16, User Name DecisionLineUser
[2013-10-15 12:23:20.052]  CONSOLE   Starting test cycle. Arguments = ['D:\\STDaemon\\WC\\1\\public\\DecisionLineSystemTests\\testscripts\\testCycle\\testCycle.py', '231|SERIAL_001', '713195', '1', 'S12031HI16', 'dldevconfig', 'http://S12031HI16:8010/',
'Ref0001NTSC', '120578851', 'True', 'AV1']
+ Setup
[2013-10-15 12:23:26.383]  CONSOLE   INFO Setting to NTSC test mode
+ REF_POWER_POWER_UP
+ REF_POWER_DETECT_OUT_OF_STANDBY
+ REF_SETTINGS_CHECK_MAC
+ REF_TUNE_PLAY_GOOD_STREAM
+ REF_TUNE_CHECK_PLAYBACK_ZOOM
+ REF_TUNE_CHECK_MOTION
+ REF_TUNE_CHECK_AUDIO
+ REF_AVOUT_PLAY_STREAM_FROM_START
+ TID_AVOUT_SWITCH_AV1_AV1_LEFT
+ TID_AVOUT_AV_ANALYSIS_AV1_AV1_LEFT
+ TID_AVOUT_SWITCH_AV1_RIGHT
+ TID_AVOUT_AV_ANALYSIS_AV1_RIGHT
+ TID_AVOUT_SWITCH_COMPONENT_DIGITALCOAXIAL_LEFT
+ TID_AVOUT_AV_ANALYSIS_COMPONENT_DIGITALCOAXIAL_LEFT
+ TID_AVOUT_SWITCH_DIGITALCOAXIAL_RIGHT
+ TID_AVOUT_AV_ANALYSIS_DIGITALCOAXIAL_RIGHT
+ TID_AVOUT_SWITCH_HDMI_DIGITALOPTICAL_LEFT
+ TID_AVOUT_AV_ANALYSIS_HDMI_DIGITALOPTICAL_LEFT
+ TID_AVOUT_SWITCH_DIGITALOPTICAL_RIGHT
+ TID_AVOUT_AV_ANALYSIS_DIGITALOPTICAL_RIGHT
+ TID_AVOUT_SWITCH_SVIDEO_HDMI_LEFT
+ TID_AVOUT_AV_ANALYSIS_SVIDEO_HDMI_LEFT
+ TID_AVOUT_SWITCH_HDMI_RIGHT
+ TID_AVOUT_AV_ANALYSIS_HDMI_RIGHT
+ RESET_DEFAULT_OUTPUT
+ REF_POWER_POWER_DOWN
+ Teardown
```

Figure 20: Test Cycle Detail - Test Log View

5.2.5.3.1 Image Compare Screen Capture

Screen captures used for image compare are displayed alongside the expected image. The region used for comparison is highlighted.

5.2.5.3.2 OCR Screen Captures

Screen captures used for OCR are displayed with the OCR region highlighted. An example of this is shown in **Error! Reference source not found.**.

Figure 21: Log Viewer OCR Example

In the example above OCR was used to check the channel number on the flip bar following navigation to channel 300.

5.2.5.3.3 Image Zoomer

Clicking on a screen capture image brings up the image zoomer where you can zoom into specific regions of interest in the captured image.

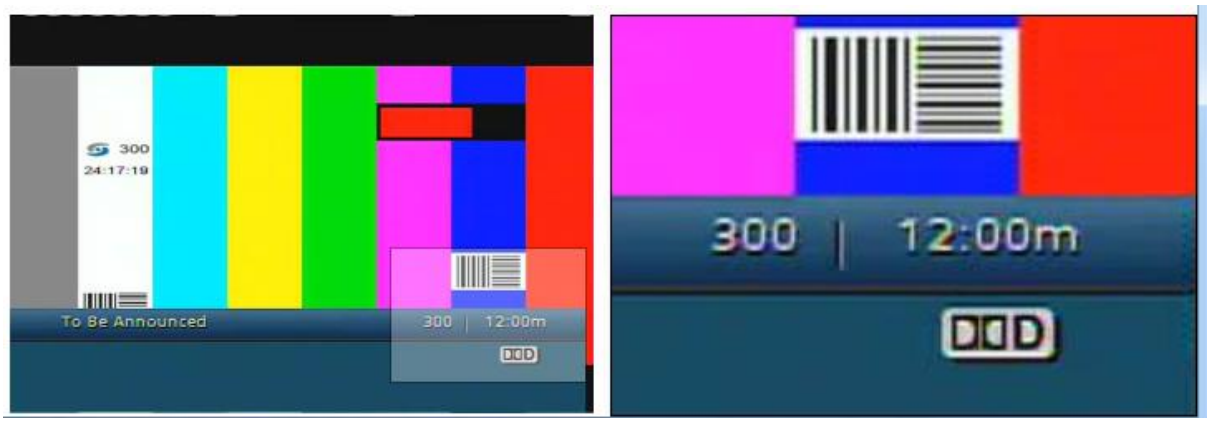


Figure 22: Image Zoomer

5.2.6 Test Point Results

This view shows test point results across the set of script runs matched by the script run selector. An example is shown in XXX.

Clicking on a column or a column name selects the next level down in the Test Point Tree. Clicking on the ⬆ button selects the next level up in the Test Point tree.

If a clicking on the 📊 icon. An example is shown in **Error! Reference source not found.**

5.2.7 Test Point Trends

5.2.8 Script Run Count View

This view shows script run count spreads across the set of script runs matched by the script run selector. An example is shown in **Error! Reference source not found.**.

Figure 23: Script Run Browser – Script Run Count Spread

The menus in the navigation bar can be used to change which spread is displayed.

5.2.9 Script Run Result View

This view shows the script run result spreads across the set of script runs matched by the script run selector. An example is shown in **Error! Reference source not found.**.

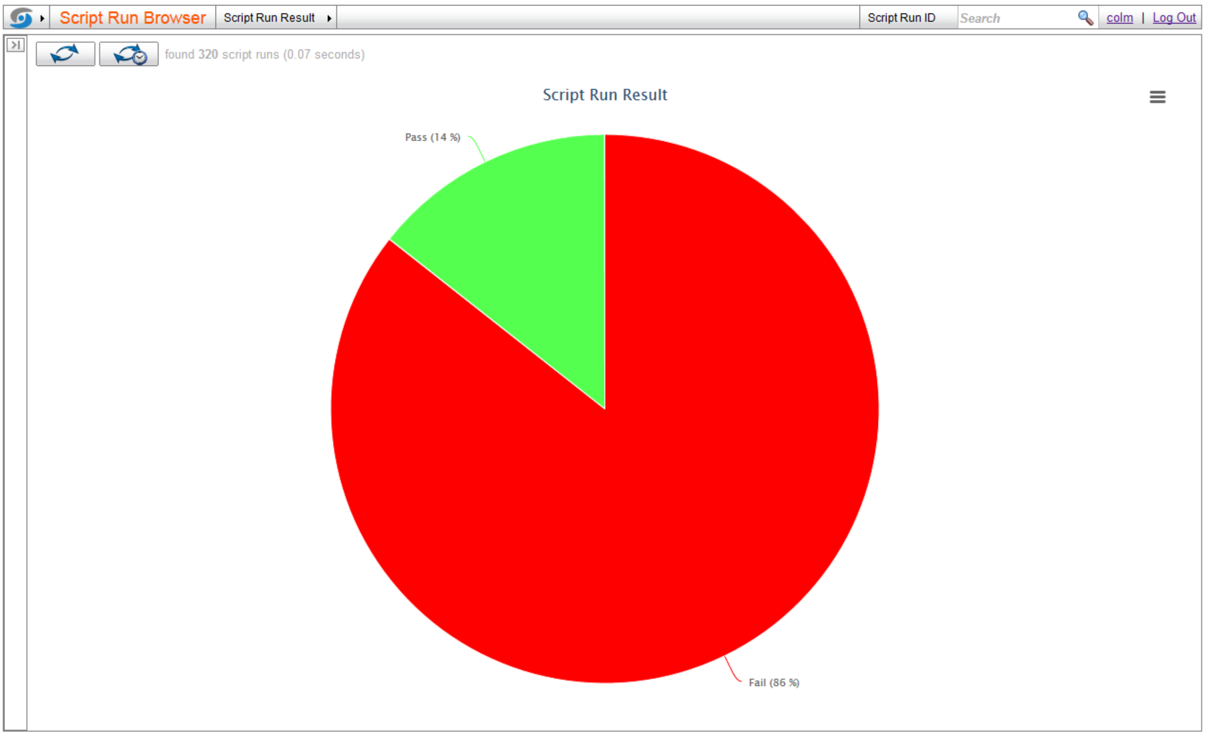


Figure 24: Script Run Browser – Script Run Result Spread

Clicking on a segment in the chart displays the spread for the selected result as shown in **Error! Reference source not found.**.

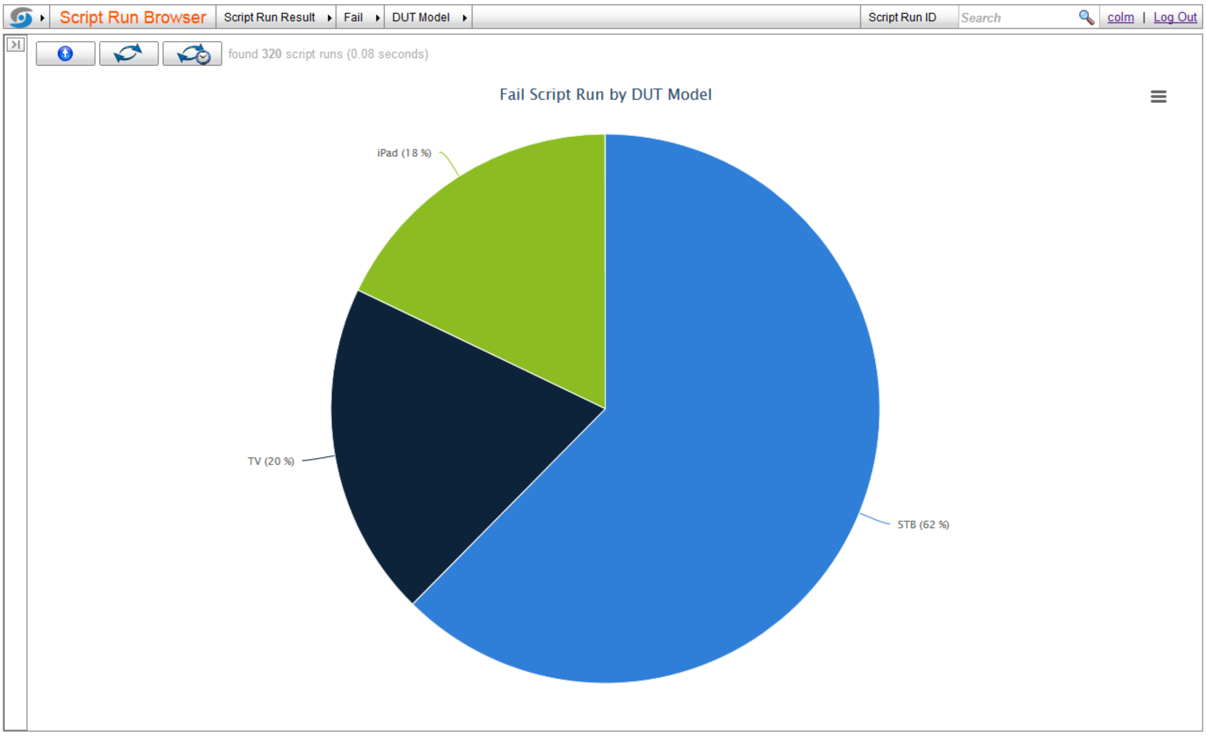


Figure 25: Script Run Browser - Failed Script Run Spread

The menus in the navigation bar can be used to change which spread is displayed.

5.2.10 Script Run Trends View

This view shows script run trends for the set of script runs matched by the script run selector. An example is shown in **Error! Reference source not found.**.

Figure 26: Script Run Browser – Script Run Trends View

The menu in the navigation bar can be used to change which trend is displayed.

Certain metrics may be displayed as cumulative by clicking on the 📊 icon. An example is shown in **Error! Reference source not found.**.

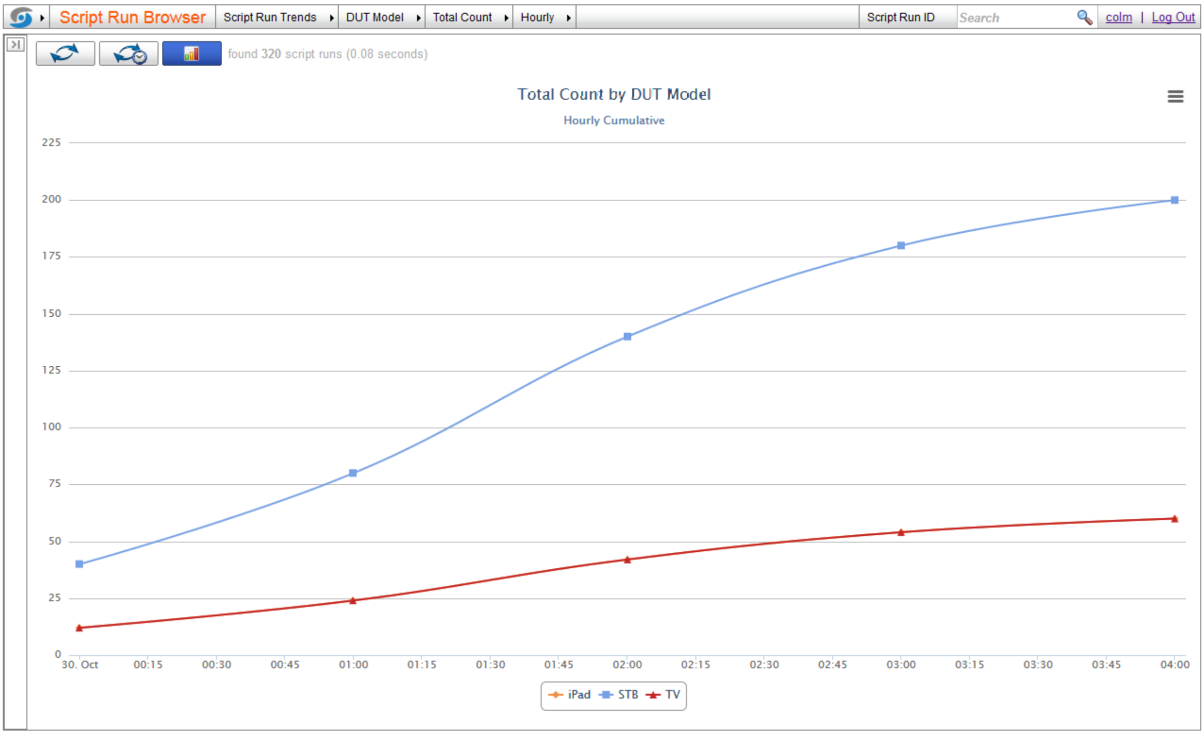


Figure 27: Script Run Browser – Script Run Trends Cumulative View

6 Administration

The Administration applet allows various aspects of the StormTest Decision Line system to be configured.

The applet has the following views:

- Test Point Trees
- Users

Under each view, applicable actions such as add, edit, delete become available.

6.1 Test Points

Test Points are used to provide a layer of reporting on top of the raw data provided by test scripts and test steps. In particular they try to address the issues of test intent and text context.

Instead of reporting which scripts passed or failed sometimes it is more useful to be able to tell which features of the system begin tested passed or failed, i.e. the test intent.

In addition it is useful to know some contextual information about the testing, for example software version, device under test.

The figures below show a comparison of script based and test point based reporting for the same set of script runs.

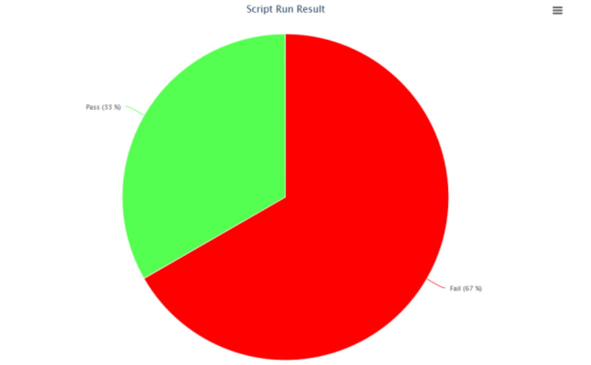


Figure 28 : Script Based Reporting

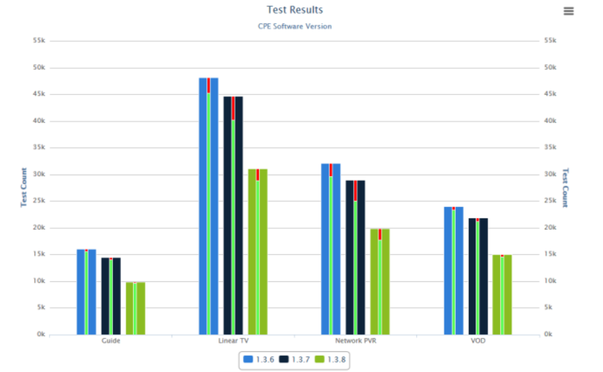


Figure 29 : Test Point Based Reporting

Test Points are key-value pairs organised into a tree structure. Each Test Point has a set of rules defining how it generates the values using the raw data from test scripts and test steps.

Every time a script is run a set of Test Point values is generated using these rules.

To cater for different types of data there are three Test Point Types:

- **Status** – For capturing pass/warn/fail type values
- **Text** – For capturing textual values e.g. software version
- **Numeric** – For capturing numeric values e.g. channel zap time

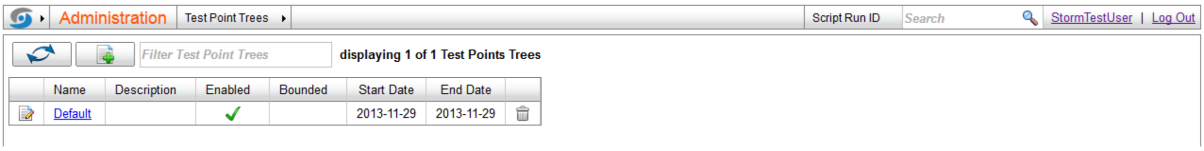
Some important things to note about Test Points:

- Multiple Test Point trees can be created based on the same raw data
- Test Points can have multiple values per script run
- Status Test Point values automatically bubble up through the tree
- Test Point results viewing is limited to one tree at a time
- A trigger (Test Step or Test Script) can fire multiple test points

Test Point values are automatically generated as each test script is run. If the Test Point definition is changed it might be required to rebuild test point values for previously run test scripts. This can be done for individual test scripts from the Dashboard. If a full rebuild is required there is a rebuild_results.cmd script in StormTestDashboard/php directory. Note that a full rebuild might take some time depending on how many test scripts need to be rebuilt.

The following sections describe how to create Test Points using the Dashboard.

6.1.1 Test Point Trees

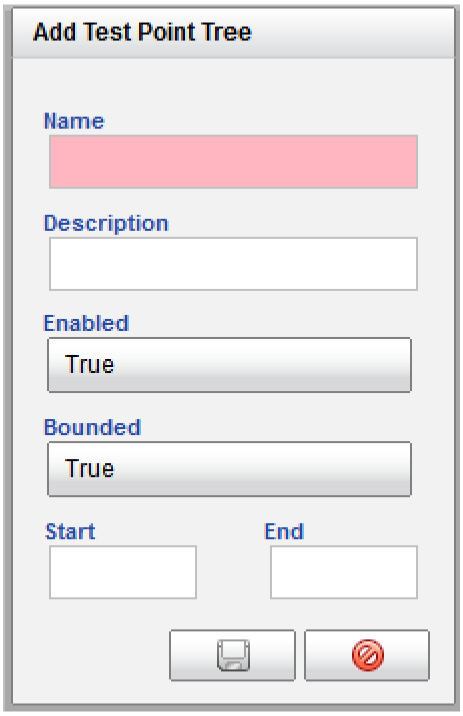


displaying 1 of 1 Test Points Trees

	Name	Description	Enabled	Bounded	Start Date	End Date	
	Default		✓		2013-11-29	2013-11-29	

Figure 30: Administration - Test Points

Test Points are organised into Test Point Trees. To add Test Point Tree, click on the button. A dialog box is displayed, as shown in Figure 31.



Add Test Point Tree

Name:

Description:

Enabled: True

Bounded: True

Start:

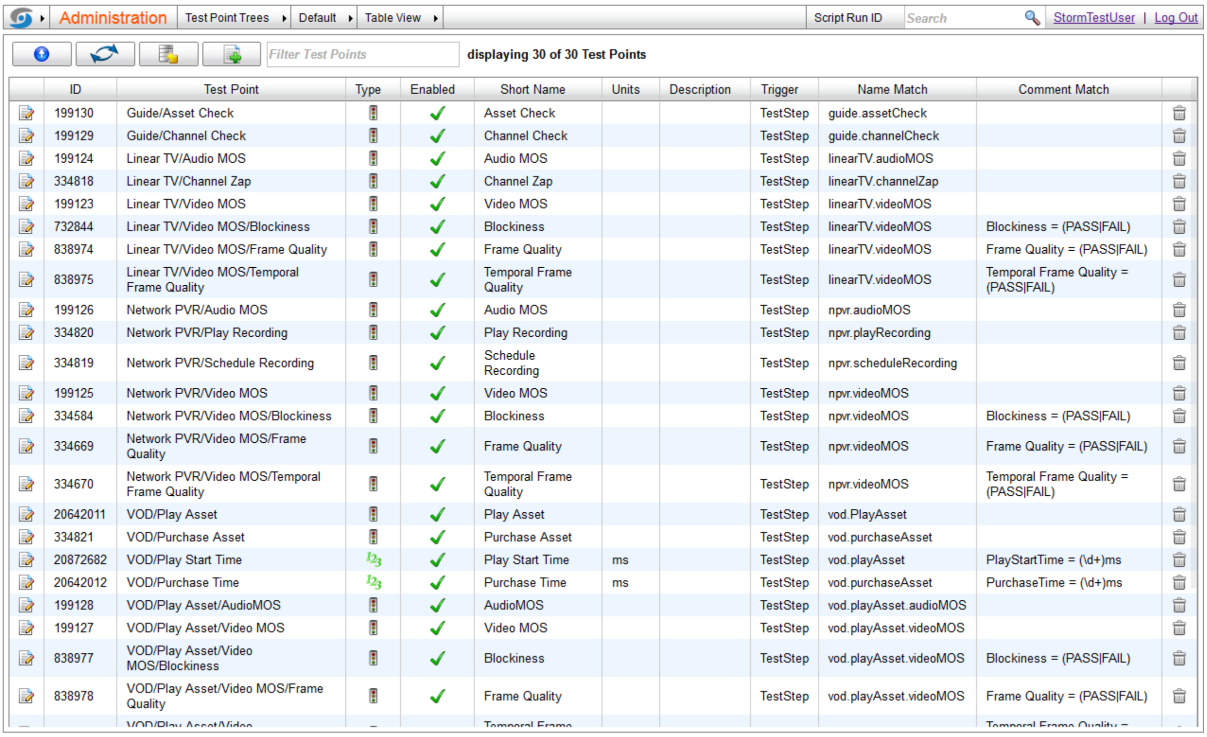
End:

Figure 31: Administration - Add Test Point Tree

Test Point Trees can be edited by clicking on the icon or deleted by clicking on the icon. Note that deleting a Test Point Tree will delete all Test Points results for that tree.

6.1.2 Test Points

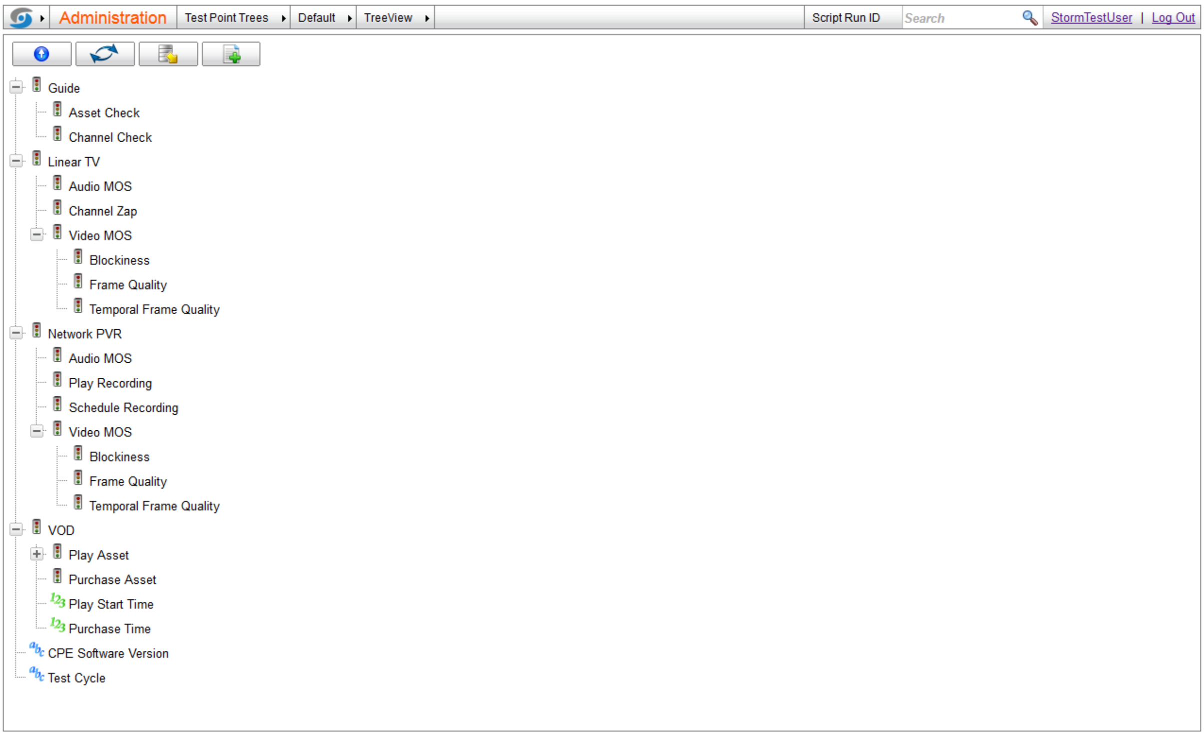
Selecting a Test Point Tree will display its Test Points. Table and Tree views are available via the navigation bar.



displaying 30 of 30 Test Points

	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
✎	199130	Guide/Asset Check	🚦	✓	Asset Check			TestStep	guide.assetCheck		🗑
✎	199129	Guide/Channel Check	🚦	✓	Channel Check			TestStep	guide.channelCheck		🗑
✎	199124	Linear TV/Audio MOS	🚦	✓	Audio MOS			TestStep	linearTV.audioMOS		🗑
✎	334818	Linear TV/Channel Zap	🚦	✓	Channel Zap			TestStep	linearTV.channelZap		🗑
✎	199123	Linear TV/Video MOS	🚦	✓	Video MOS			TestStep	linearTV.videoMOS		🗑
✎	732844	Linear TV/Video MOS/Blockiness	🚦	✓	Blockiness			TestStep	linearTV.videoMOS	Blockiness = (PASS|FAIL)	🗑
✎	838974	Linear TV/Video MOS/Frame Quality	🚦	✓	Frame Quality			TestStep	linearTV.videoMOS	Frame Quality = (PASS|FAIL)	🗑
✎	838975	Linear TV/Video MOS/Temporal Frame Quality	🚦	✓	Temporal Frame Quality			TestStep	linearTV.videoMOS	Temporal Frame Quality = (PASS|FAIL)	🗑
✎	199126	Network PVR/Audio MOS	🚦	✓	Audio MOS			TestStep	npvr.audioMOS		🗑
✎	334820	Network PVR/Play Recording	🚦	✓	Play Recording			TestStep	npvr.playRecording		🗑
✎	334819	Network PVR/Schedule Recording	🚦	✓	Schedule Recording			TestStep	npvr.scheduleRecording		🗑
✎	199125	Network PVR/Video MOS	🚦	✓	Video MOS			TestStep	npvr.videoMOS		🗑
✎	334584	Network PVR/Video MOS/Blockiness	🚦	✓	Blockiness			TestStep	npvr.videoMOS	Blockiness = (PASS|FAIL)	🗑
✎	334669	Network PVR/Video MOS/Frame Quality	🚦	✓	Frame Quality			TestStep	npvr.videoMOS	Frame Quality = (PASS|FAIL)	🗑
✎	334670	Network PVR/Video MOS/Temporal Frame Quality	🚦	✓	Temporal Frame Quality			TestStep	npvr.videoMOS	Temporal Frame Quality = (PASS|FAIL)	🗑
✎	20642011	VOD/Play Asset	🚦	✓	Play Asset			TestStep	vod.PlayAsset		🗑
✎	334821	VOD/Purchase Asset	🚦	✓	Purchase Asset			TestStep	vod.purchaseAsset		🗑
✎	20872682	VOD/Play Start Time	123	✓	Play Start Time	ms		TestStep	vod.playAsset	PlayStartTime = (\d+)ms	🗑
✎	20642012	VOD/Purchase Time	123	✓	Purchase Time	ms		TestStep	vod.purchaseAsset	PurchaseTime = (\d+)ms	🗑
✎	199128	VOD/Play Asset/AudioMOS	🚦	✓	AudioMOS			TestStep	vod.playAsset.audioMOS		🗑
✎	199127	VOD/Play Asset/Video MOS	🚦	✓	Video MOS			TestStep	vod.playAsset.videoMOS		🗑
✎	838977	VOD/Play Asset/Video MOS/Blockiness	🚦	✓	Blockiness			TestStep	vod.playAsset.videoMOS	Blockiness = (PASS|FAIL)	🗑
✎	838978	VOD/Play Asset/Video MOS/Frame Quality	🚦	✓	Frame Quality			TestStep	vod.playAsset.videoMOS	Frame Quality = (PASS|FAIL)	🗑
		VOD/Play Asset/Video			Temporal Frame					Temporal Frame Quality =	

Figure 32 : Administration - Test Points Table View

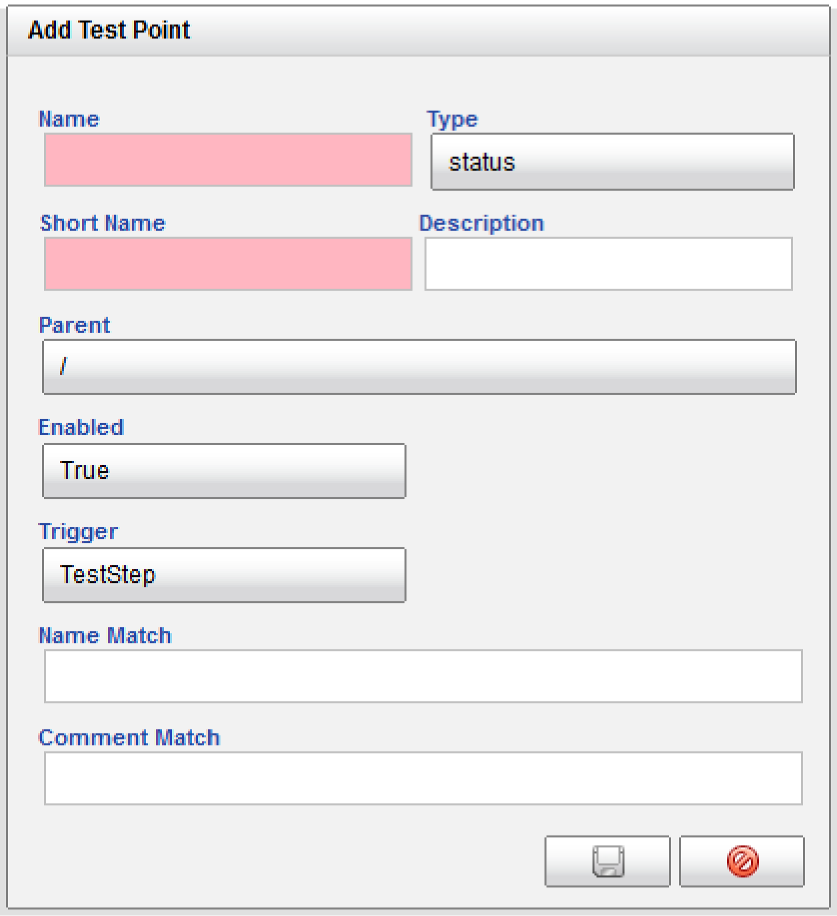


- Guide
  - Asset Check
  - Channel Check
- Linear TV
  - Audio MOS
  - Channel Zap
  - Video MOS
    - Blockiness
    - Frame Quality
    - Temporal Frame Quality
- Network PVR
  - Audio MOS
  - Play Recording
  - Schedule Recording
  - Video MOS
    - Blockiness
    - Frame Quality
    - Temporal Frame Quality
- VOD
  - Play Asset
  - Purchase Asset
  - Play Start Time
  - Purchase Time
- CPE Software Version
- Test Cycle

Figure 33: Administration - Test Points Tree View

The tree view has a context menu allowing the user to expand/collapse nodes and also to add, edit or delete test points.

To add Test Point, click on the ➕ button or select “Add Test Point” from the context menu. A dialog box is displayed, as shown in Figure 34.



Add Test Point

Name	
Type	status
Short Name	
Description	
Parent	/
Enabled	True
Trigger	TestStep
Name Match	
Comment Match	

Figure 34: Administration - Add Test Point

For each Test Point the following is defined:

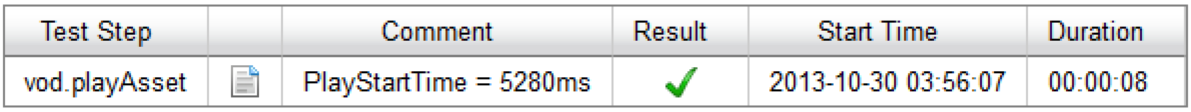
Column	Notes
Name	Name
Type	status, text or numeric
Short Name	Name used when trying to save on screen space
Description	Optional description
Parent	Parent Test Point
Enabled	Disabled Test Points don’t generate any results
Units	Optional units for text or numeric test point types
Trigger	“TestStep or “TestScript”
Name Match	Regular expression pattern using trigger name. This is used to decide which triggers will fire the test point.
Comment Match	Regular expression pattern using trigger comment. This is used to decide which triggers will fire the test point. It can also be used to decide what value will be used when firing the test point.

Table 4: Test Point Definition

6.1.3 Test Point Examples

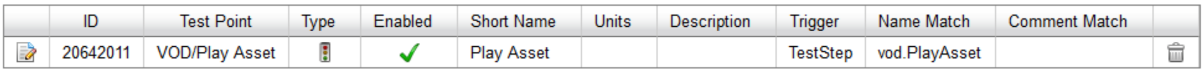
6.1.3.1 Status from Test Step

The most basic use of test points is to extract the result of a test for example playing a VOD asset. In this example the test script generates a test step like this



Test Step		Comment	Result	Start Time	Duration
vod.playAsset	📄	PlayStartTime = 5280ms	✓	2013-10-30 03:56:07	00:00:08

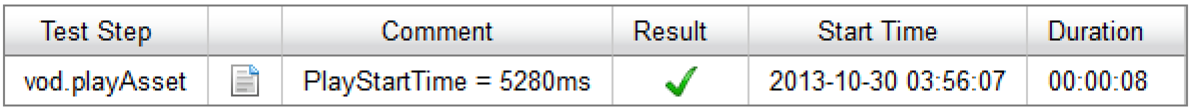
The test point is defined as



	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
✎	20642011	VOD/Play Asset	🚦	✓	Play Asset			TestStep	vod.PlayAsset		🗑

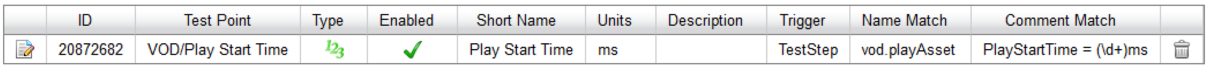
6.1.3.2 Numeric Values from Test Step

For non-functional testing it is interesting to track performance metrics for example how long it takes to start playing a VOD asset. In this example the test script generates a test step like this



Test Step		Comment	Result	Start Time	Duration
vod.playAsset	📄	PlayStartTime = 5280ms	✓	2013-10-30 03:56:07	00:00:08

The test point to extract the numeric value from the test step comment is defined as



	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
✎	20872682	VOD/Play Start Time	123	✓	Play Start Time	ms		TestStep	vod.playAsset	PlayStartTime = (\d+)ms	🗑

Note that in this case a simple rule meaning more than one digit is used to extract the value. If the value will contain decimal points or negative sign then the rule could be extended to be (-{0,1}[0-9]+(\.[0-9]+){0,1}).

Tracking numeric values allows us to display min/avg/max values across a range of test results as shown in XXX.

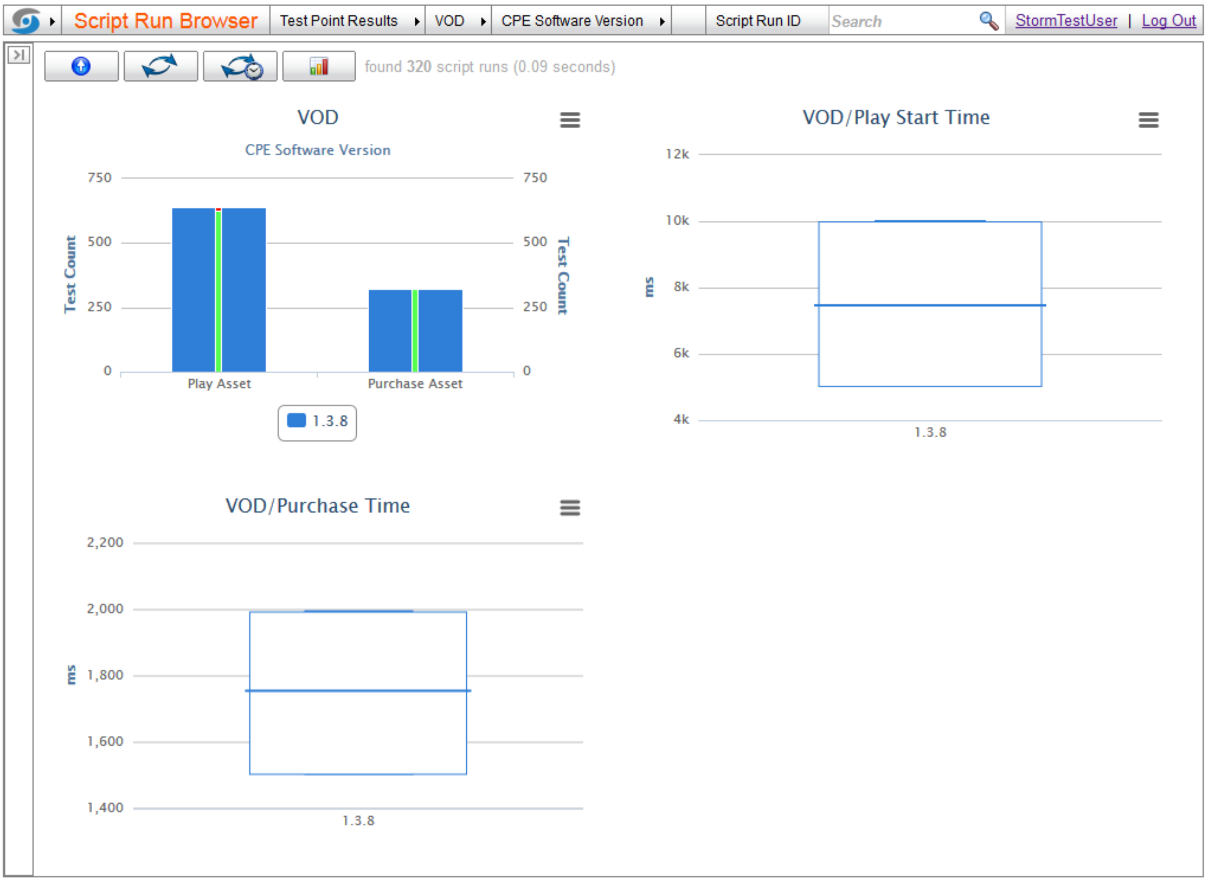
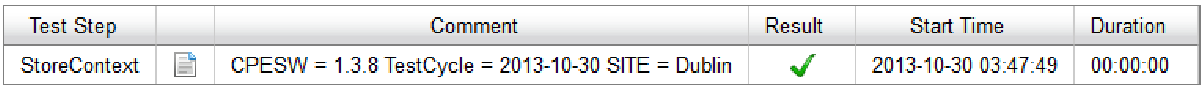


Figure 35: Test Points – Example showing Numeric Test Points

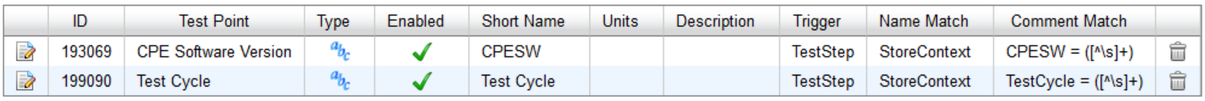
6.1.3.3 Software Version from Test Step

To track contextual information like software version we can create a test step to report the information and the extract this info using a number of test points. For example the test script generates a test step like this



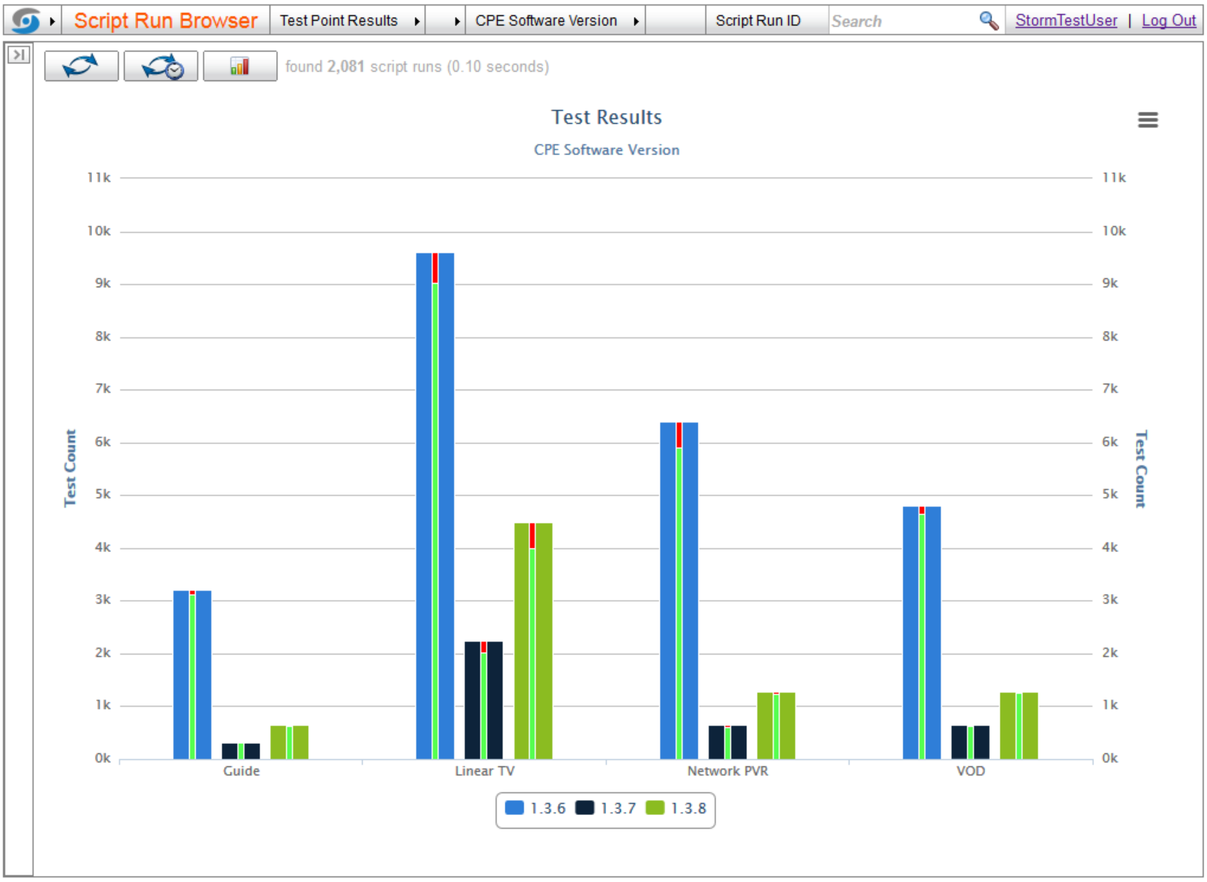
Test Step		Comment	Result	Start Time	Duration
StoreContext	📄	CPESW = 1.3.8 TestCycle = 2013-10-30 SITE = Dublin	✓	2013-10-30 03:47:49	00:00:00

We can create two test points to extract as follows



	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
✎	193069	CPE Software Version	abc	✓	CPESW			TestStep	StoreContext	CPESW = ([^\s]+)	🗑
✎	199090	Test Cycle	abc	✓	Test Cycle			TestStep	StoreContext	TestCycle = ([^\s]+)	🗑

This allows us to track test point results across different software versions as shown below.

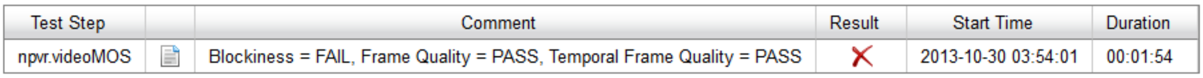


found 2,081 script runs (0.10 seconds)

Figure 36: Test Points – Example showing Context Test Points

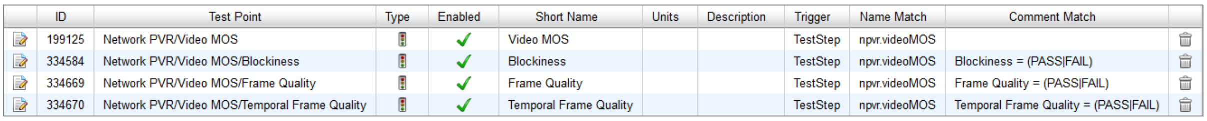
6.1.3.4 Multiple Metrics from Test Step

Sometimes it is interesting to report multiple metrics from a single test step. In the following example the test step fails but also reports results for individual metrics within the test.



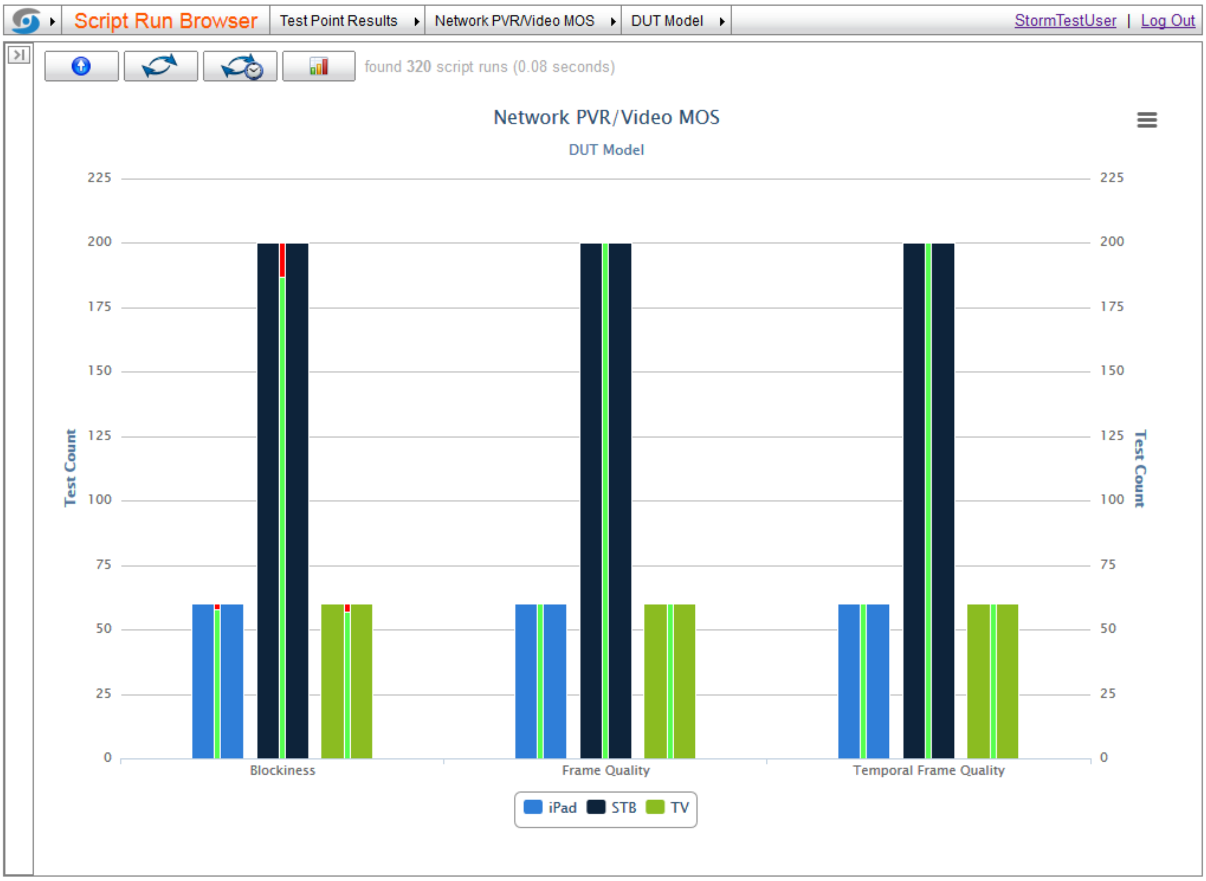
Test Step		Comment	Result	Start Time	Duration
npvr.videoMOS		Blockiness = FAIL, Frame Quality = PASS, Temporal Frame Quality = PASS	✗	2013-10-30 03:54:01	00:01:54

We can then create four test points, one to track the overall result and three to track the individual metrics



	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
	199125	Network PVR/Video MOS		✓	Video MOS			TestStep	npvr.videoMOS		
	334584	Network PVR/Video MOS/Blockiness		✓	Blockiness			TestStep	npvr.videoMOS	Blockiness = (PASS|FAIL)	
	334669	Network PVR/Video MOS/Frame Quality		✓	Frame Quality			TestStep	npvr.videoMOS	Frame Quality = (PASS|FAIL)	
	334670	Network PVR/Video MOS/Temporal Frame Quality		✓	Temporal Frame Quality			TestStep	npvr.videoMOS	Temporal Frame Quality = (PASS|FAIL)	

This example would allow us to track reasons for video MOS failures across DUT Models.

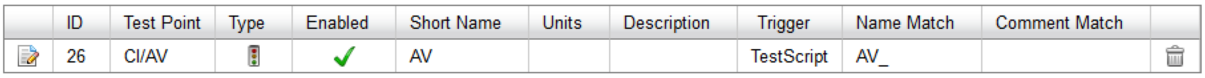


found 320 script runs (0.08 seconds)

Figure 37: Test Points - Example showing multiple Test Points from a Test Step

6.1.3.5 *Functional Area from Test Script*

Sometimes information about what is being tested is contained in the name of the test script. For example if all of your AV scripts contain AV_ in the script name you can create a test point to report an all your AV testing.



	ID	Test Point	Type	Enabled	Short Name	Units	Description	Trigger	Name Match	Comment Match	
	26	CI/AV	🚦	✓	AV			TestScript	AV_		

6.1.4 Exporting/Importing Test Points

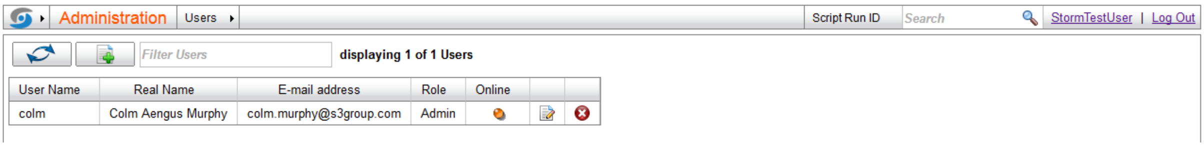
If you wish to move your test point definitions to another StormTest facility, you can export them by selecting the relevant ones and clicking on the button. This will generate an xml file.

To import this file into another facility, you need to run a php script on the config server.

From a command shell, go to the `C:\Program Files (x86)\Apache for StormTest\htdocs\StormTestDashboard\php` directory and run the `importTestPoints.php` file, passing in the test point xml file as a parameter.

6.2 Users

This view shows details of the user accounts which are used to login to the Dashboard.



displaying 1 of 1 Users

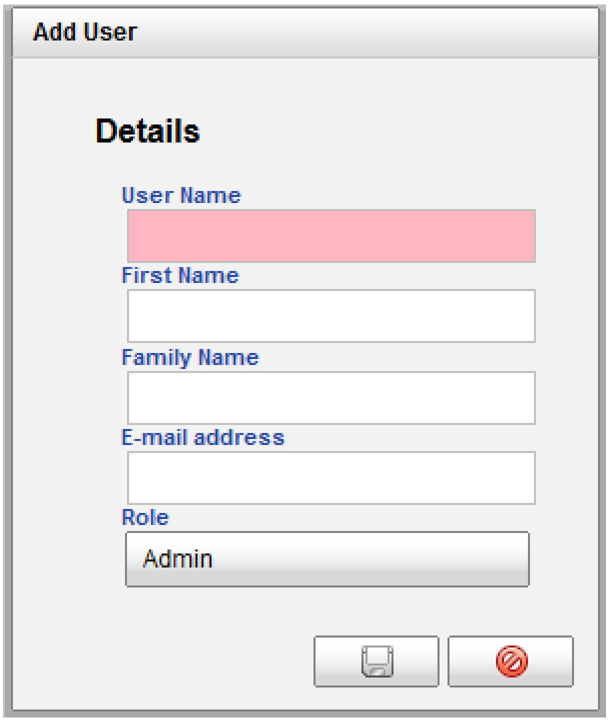
User Name	Real Name	E-mail address	Role	Online		
colm	Colm Aengus Murphy	colm.murphy@s3group.com	Admin	●	✎	✖

Figure 38: Administration - Users

A user account is required to use the Dashboard and it is recommended that accounts are not shared between users.

Clicking on the ✖ icon will delete all sessions for the selected account. This will force anyone using this account to re-login.

To add a user, click on the ➕ button. This brings up the “Add User” dialog as shown in Figure 40.

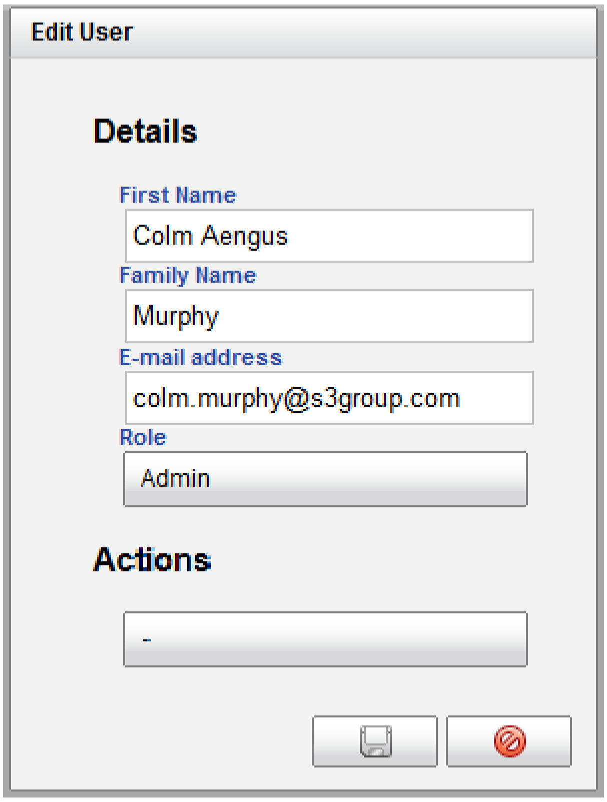


Role: Admin

Figure 39: Add User Dialog

To edit a user account, click on the ✎ icon. This brings up the “Edit User” dialog as shown in Figure 39.

It is not possible to delete a user.



Edit User

Details

First Name: Colm Aengus

Family Name: Murphy

E-mail address: colm.murphy@s3group.com

Role: Admin

Actions

-

Figure 40: Edit User Dialog

6.2.1 User Roles

6.2.1.1 Admin

This role has full access to the Dashboard.

6.2.1.2 Analyst

This role has access to all Dashboard applets except the Administration applet. The following limitations also apply

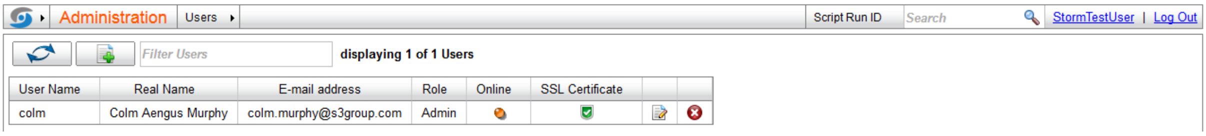
- Not allowed to delete script runs
- Not allowed to rebuild script run results

6.2.2 SSL Certificate Management

The Dashboard supports connection via http or https. When https is used client certificates are used to authenticate users.

Note that https support requires a change to the standard Apache for StormTest installation.

When connecting to the Dashboard via https and extra column appears in the Users table. Refer to Figure 41.



displaying 1 of 1 Users

User Name	Real Name	E-mail address	Role	Online	SSL Certificate		
colm	Colm Aengus Murphy	colm.murphy@s3group.com	Admin	●	✓		

Figure 41: Administration - Users SSL Connection

6.2.2.1 Creating a Client Certificate

Clicking on an empty “SSL Certificate” column will create a client certificate.

6.2.2.2 Downloading a Client Certificate

Clicking 🛡 will download the client certificate. This should then be installed into the browsers certificate store.

6.2.2.3 Revoking a Client Certificate

To revoke a client certificate select the “Revoke User SSL Certificate” action in the Edit User dialog.

6.2.3 User Actions

When editing a user there are a number of actions that can be performed.

6.2.3.1 Reset User Password

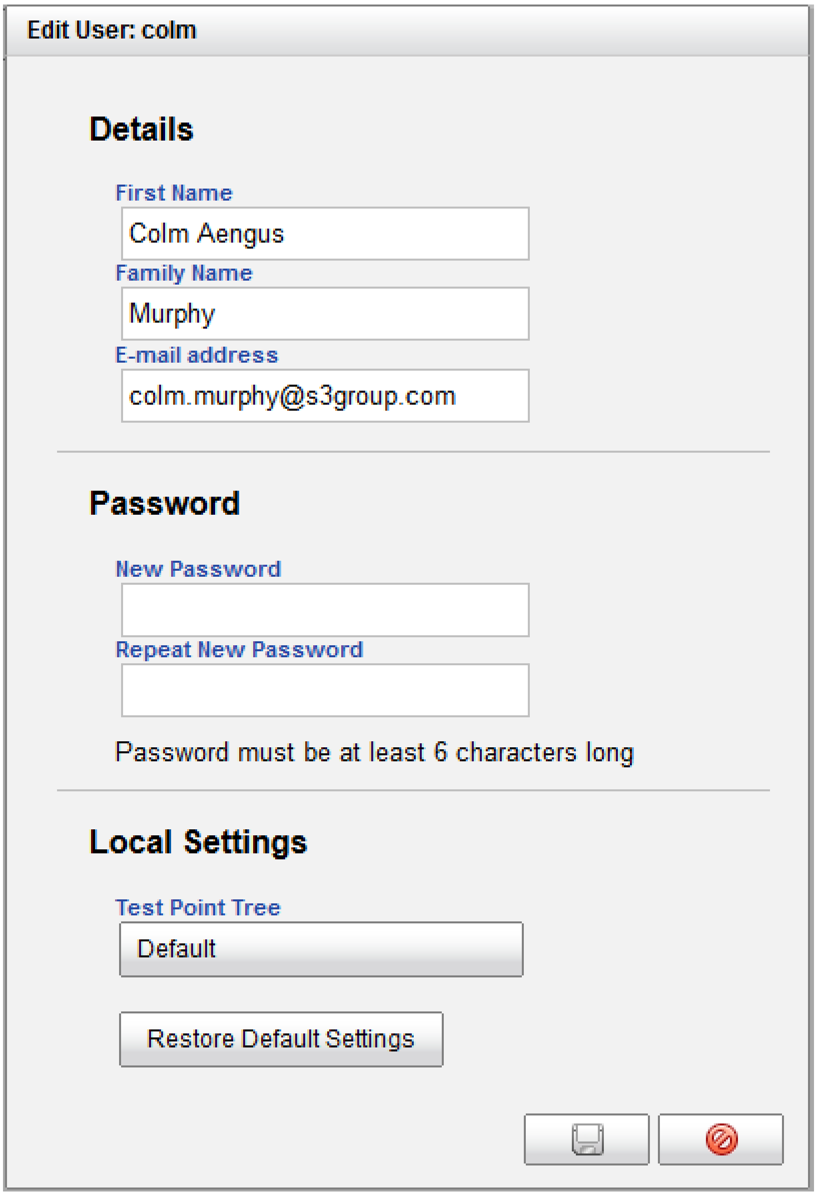
This action will reset the user password to a random string. The user will be forced to change this password after they login.

6.2.3.2 Revoke User SSL Certificate

This action will revoke a user’s SSL certificate.

6.2.4 Edit Logged in User

Clicking on the logged in user name in the navigation bar brings up the “Edit User” dialog as shown in Figure 42.



Edit User: colm

Details

First Name: Colm Aengus

Family Name: Murphy

E-mail address: colm.murphy@s3group.com

Password

New Password

Repeat New Password

Password must be at least 6 characters long

Local Settings

Test Point Tree: Default

Figure 42: Edit Logged in User Dialog

In this dialog the user can change the following

- First Name, Family Name, E-mail address
- Password which must be at least 6 characters long
- Test Point Tree – Select which Test Point Tree is used for the browsing results. Changing to a different tree will restart the application
- Restore Default Settings Restore can be used to clear all locally stored settings. This will restart the application