PassMark® Software
Management Console

Quick start guide

Edition: 1.0
Date: 28 March 2019

Overview

When testing large numbers of systems, the Management Console Web server application allows system information, test status and test result information to be managed centrally for BurnInTest and Metest86 tests.

Via a browser the Management Console allows information about tests to be displayed. This includes the status of currently running tests, system information, current and previous test results, test reports and test statistics.

Requirements

- A web server running PHP (5.3.2 or later) and MySQL (5.2.47 or later).
- If using Memtest86, Python scripting support (3.6.3 or later) and a PXE server setup to boot MemTest86 across the network is required. The systems running Memtest86 must be able to PXE boot to UEFI.

Backwards Compatibility

Please be aware the Management Console is not backwards compatible with the database from the product previously known as "BurnInTest Management Console" versions 7 & 8 and currently there is no way to port the old database contents to the new version. The database structure has changed significantly so a new empty database will need to be created.

The recommended action would be to rename your old database and BurnInTest management console folder on the webserver (and update the passmark_mgtconsole_databases.php to connect to the renamed database) so your previous results are still available. Then install the new version, create a new empty database and configure BurnInTest to point to the new install.

PassMark Products Supported

The management console supports the following products and versions;

BurnInTest Windows	8.1.1000 or newer
BurnInTest Linux	4.0.1000 or newer
MemTest86	7.5 or newer

Installation

If you are installing the Management Console on an existing Web server with PHP and MySQL, skip to step 6.

1. Install Apache 2.2.14 or later
2. Install PHP 5.3.2 or later
   Note 1: When installing PHP, select Web server setup "Apache 2.2.x module" (and install everything).
   Note 2: To install MySQL, you need to install PHP with extensions (to get ext\php_mysql.dll)
3. Install MySQL 5.2.47 or later
4. Configure Apache (httpd.conf):
   a. Create a symbolic link in the apache installation folder to 'htdocs', e.g.:

```
Alias /htdocs "C:\Passmark\Software\BIT Console\mgtconsole"

<Directory "C:\Passmark\Software\BIT Console\mgtconsole ">
   Options Indexes FollowSymLinks
   AllowOverride all
   Order allow,deny
   Allow from all
</Directory>
```

b. Configure the PHP installation, e.g.:

```
LoadModule php5_module "C:/Program Files (x86)/PHP/php5apache2_2.dll"
AddType application/x-httpd-php .php
PHPIniDir "C:\Program Files (x86)\PHP"
```

5. Configure PHP (PHP.ini):
   a. Set the default Time zone in PHP, e.g.:

```
[Date]
; Defines the default time zone used by the date functions
; http://php.net/date.timezone
date.timezone =Australia/Sydney
```

   b. Configure the MySQL extension: extension=php_mysql.dll
6. Create the Management Console database using the MySQL administration tools using the SQL shown in “Appendix A – Creating the Management Console database”. There is also an SQL script included in the install files called “database_creation_script.sql” that can be used to create the initial database.
7. Copy the PassMark supplied Management Console files to the Web server, e.g. to htdocs. Set the error log file, ManagementConsole-errors.log, as writeable to all.
8. Setup the Management Console connection to MySQL:
   a. The PHP file, settings.php, contains the host name, port and MySQL password. Open the settings.php file in a text editor and change the $database_hostname, $database_username, $database_password, $database_port and $database_name values to suit your MySQL setup.
9. You should now be able to access the dashboard via http://<server>/dashboard.php (for example).

Update existing database

Sometimes columns will be added to existing tables and require existing databases to be modified.

If you previously installed the management console and are updating to a new build then please see the database_update.sql script and the Readme.txt file to see if any changes have occurred and if the update script needs to be run.

BurnInTest – Management Console setup

To manage BurnInTest with the Management Console, the following must be selected and setup in BurnInTest: Configuration->Test Preferences->Management.

Manage BurnInTest with the Console Manager

When this option is selected, BurnInTest will send system information (on startup), test status (every 10 seconds) and test result information (at the completion of each test run) to the Management Console Web server application.

Server address

Specifies the address of the Management Console Web server application. For example:

http://<Server address>/htdocs/mgtconsolemsghandler.php

Notes:

1) mgtconsolemsghandler.php is the Management Console web application that receives XML based connect, status and test result information from BurnInTest (via HTTP POST messages).
2) Configuration->Report information->Machine ID uniquely identifies the system in the Management Console. The MachineID is a unique identifier automatically generated and saved by BurnInTest for the system and must be between 8 and 15 characters. While it can be modified by a user, this is not recommended. If the MachineID is changed, then this system will appear as a different system in the Management Console.
3) When connected to the Management Console, BurnInTest will display "Connected to Management Console" in the status bar at the bottom of the BurnInTest window.

MemTEST86 – Management Console setup

In order for Memtest86 to communicate with the management console it needs to have access to the network while running, currently to do this Memtest86 needs to boot over the network vix PXE. It can then upload status messages via TFTP to a folder on the PXE server. This folder is watched by a Python script for changes, on detection of a new file it will then send the XML contents of the file to the management server (the mgtconsolemsghandler.php file) via a HTTP POST request.

Setting up the PXE Server

In order to configure PXE booting of MemTest86, a DHCP/PXE server must be present on the network to host the MemTest86 boot image for PXE boot-enabled client machines to acquire. Network booting of MemTest86 has been tested successfully with Serva PXE Server but other PXE servers should work as well. See the manual for your DHCP/PXE server for configuration instructions. The configuration instructions for Serva PXE Server is included in the following section.

Once the PXE server is configured, extract the files from the MemTest86 package to the appropriate directory for your PXE server configuration. For most cases, this is the TFTP root directory configured in the TFTP server. In the PXE/DHCP server settings, specify the boot image file to "BOOTX64.efi" for x86-64 client machines and "BOOTIA32.efi" for x86 client machines.

On the client machine, the UEFI BIOS must support booting from the network. In the BIOS setup, ensure that the “UEFI Network Stack” and “IPv4 PXE Support' features are enabled. If the PXE Server was successfully set up, the client machines should automatically boot MemTest86 on power-up.

Setting up and running the Python script

The Python scripting environment must be installed on the system used for the PXE boot. A Python script (memtest_status_watcher.py) needs to be running in order to transfer messages from the TFTP upload folder (where Metest86 uploads its status messages) to the management console PHP files.

It contains several user configurable options, they can be edited by opening the file in a text editor;

MANAGEMENT_CONSOLE_URL
This is the address of the management console server eg
http://localhost/mgtconsole/mgtconsolemsghandler.php

USER
User name for web server authentication if required (leave empty if no authentication)

PASSWD
Password for web server authentication if required (leave empty if no authentication)

WATCHDIR
Directory to watch for files uploaded from Memtest86 (tftp server upload directory)

This script requires the “requests” and “watchdog” Python libraries be installed, which can be done using the pip command eg “pip install requests” (on windows pip.exe is located in scripts directory of the Python install directory).

Configuring Serva for MemTest86 PXE Boot

Serva is a light-weight but powerful Windows PXE server that bundles all required services (eg. DHCP, TFTP) in order to support UEFI-based network booting. Serva does not require an installation and can be setup in minutes.

Configuring Serva for Single-Image Boot is ideal for servers that require only a simple setup and do not need to distribute software images other than MemTest86. All necessary settings are configured within the Serva application and do not require any additional configuration files.

1. Open Serva and select 'Settings'
2. Click on the TFTP tab to setup the TFTP server
   a) Ensure that 'TFTP Server' is checked
   b) Specify the TFTP root directory. This should be the location where the files in the MemTest86 are to be extracted.
   c) Set the TFTP Security to 'Standard' to allow MemTes86 report files to be uploaded to the server
3. Click on the DHCP tab to setup the DHCP server
   a) If your network already has a DHCP server, check 'proxyDHCP'. Otherwise, check 'DHCP Server'.

b) If 'DHCP Server' is selected, specify the 'IP Pool 1st Addr', 'Pool size' and 'Subnet Mask' for the DHCP server.

c) Specify the 'Boot File' to be retrieved by the client. For 64-bit clients (most systems), enter 'BOOTX64.efi' as the boot file. For 32-bit clients, enter 'BOOTIA32.efi'

4. Press OK to save the settings.
5. Extract all files in the MemTest86 package in the folder specified in Step 2b.
6. Close and restart Serva to apply the settings.

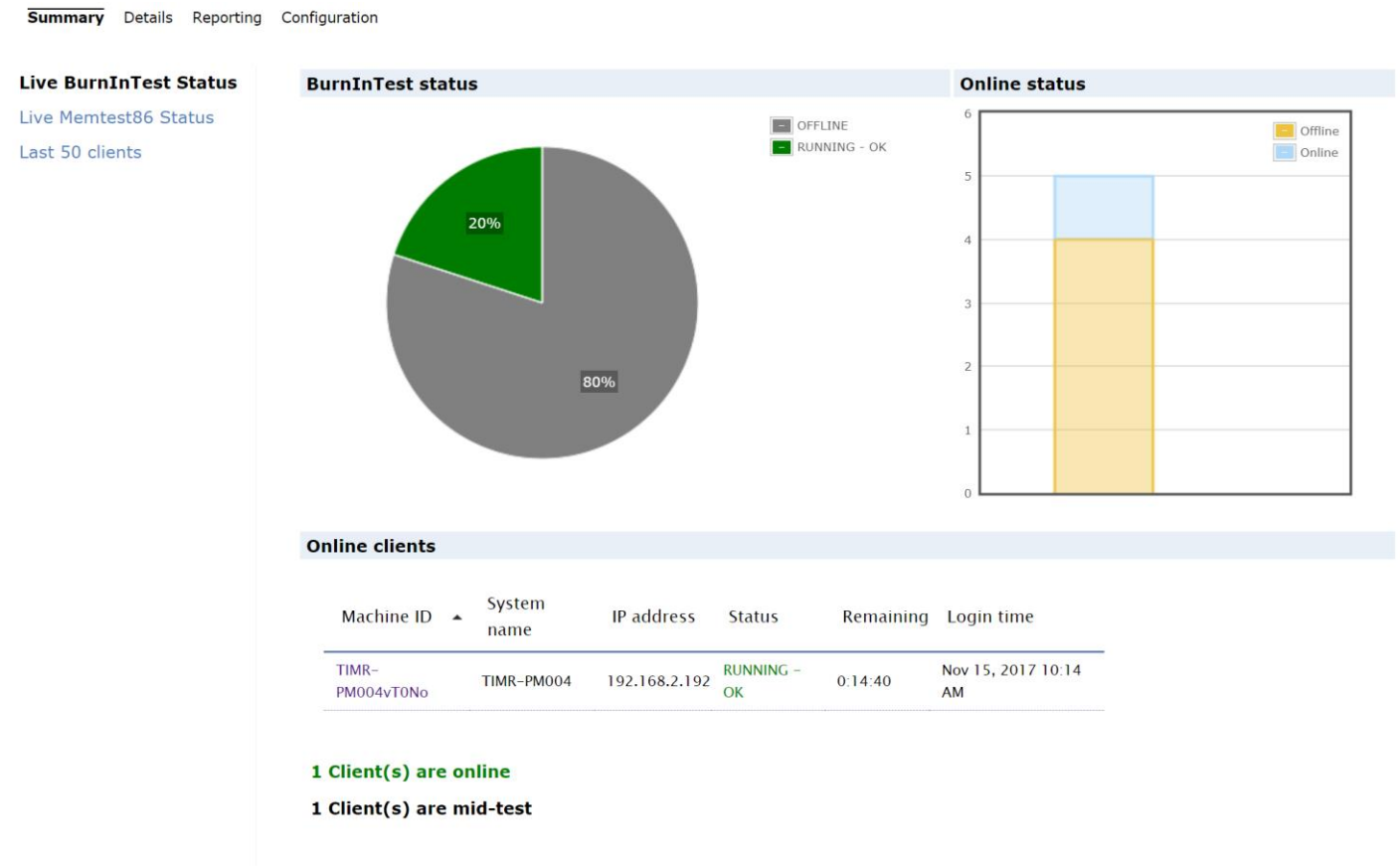
Management Console – usage

The Management Console user interface (dashboard.php) has 4 main options: Summary, Details, Reporting and Configuration.

Summary

The Summary option allows an overview of online and offline BurnInTest and MemTest86 clients.

Live BurnInTest Status



Last 50 Clients

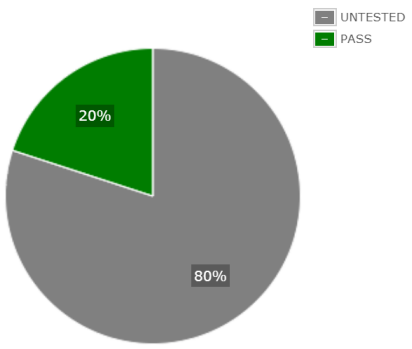
Summary Details Reporting Configuration

Live BurnInTest Status

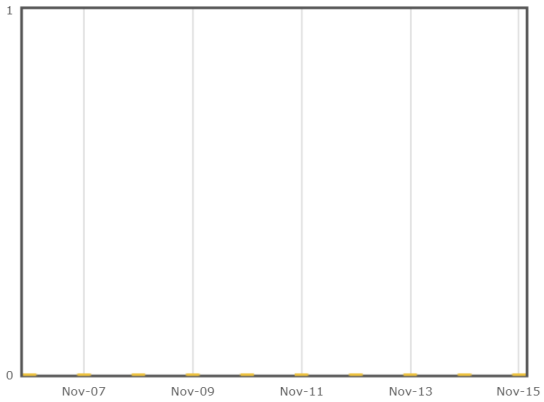
Live Memtest86 Status

Last 50 clients

Pass Rate



# of Errors - Past Week



Last 50 clients

Machine ID	Program	System name	Last result	# of errors	Last test date
01-0C-29-23-7F-60	MemTest86			0	Oct 30, 2017 3:16 PM
71 -D4-35-F8-F8-1A	MemTest86			0	Nov 7, 2017 11:15 AM
74-D1-35-F8-F8-1A	MemTest86		PASS	0	Nov 7, 2017 11:42 AM
74-D4-35-F8-F8-1A	MemTest86			0	Nov 2, 2017 1:23 PM
TIMR-PM004vT0No	BurnInTest	TIMR-PM004		0	Nov 2, 2017 1:56 PM

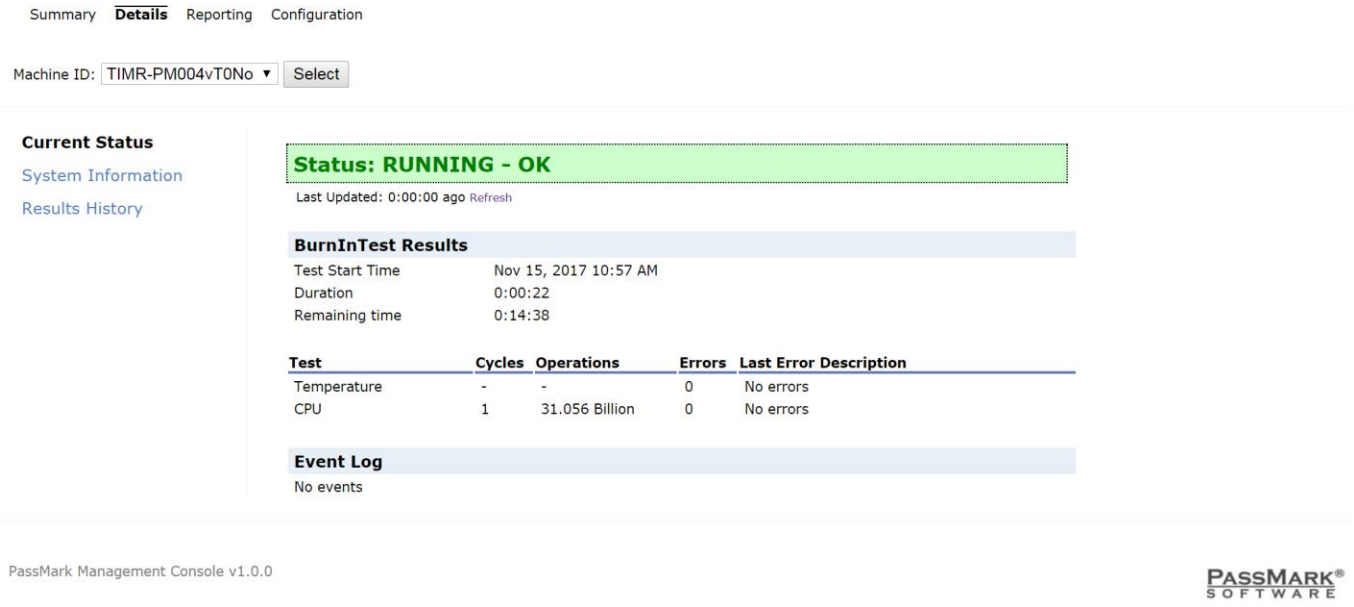
0 client(s) with FAIL result

5 client(s) in total

Details

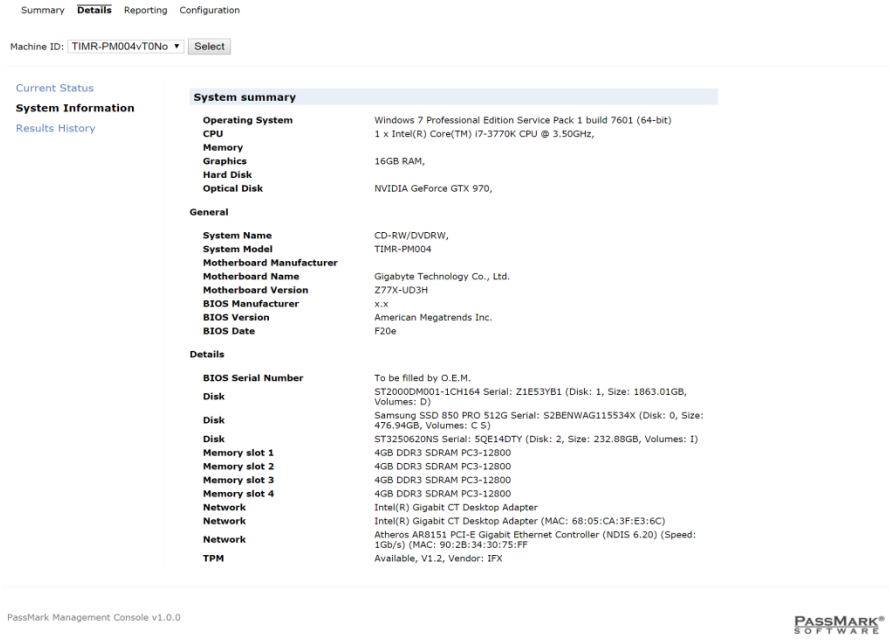
The Details option provides information for a selected test system, including the Current Status, System Information, Last Test Result and a searchable results history and per test run deletion.

Current Status



System Information

The system information for the system is stored each time a system connected to the management console so hardware changes will be stored across test runs. This will display the system information from the most recent test session.



Results History

Summary **Details** Reporting Configuration

Machine ID: TIMR-PM004vT0No ▾ Select

Current Status
System Information
Results History

Filter results history

Date Range: From: 2016-Nov-15 To: 2017-Nov-15

Components tested: Any ▾

Result: Any ▾

Filter

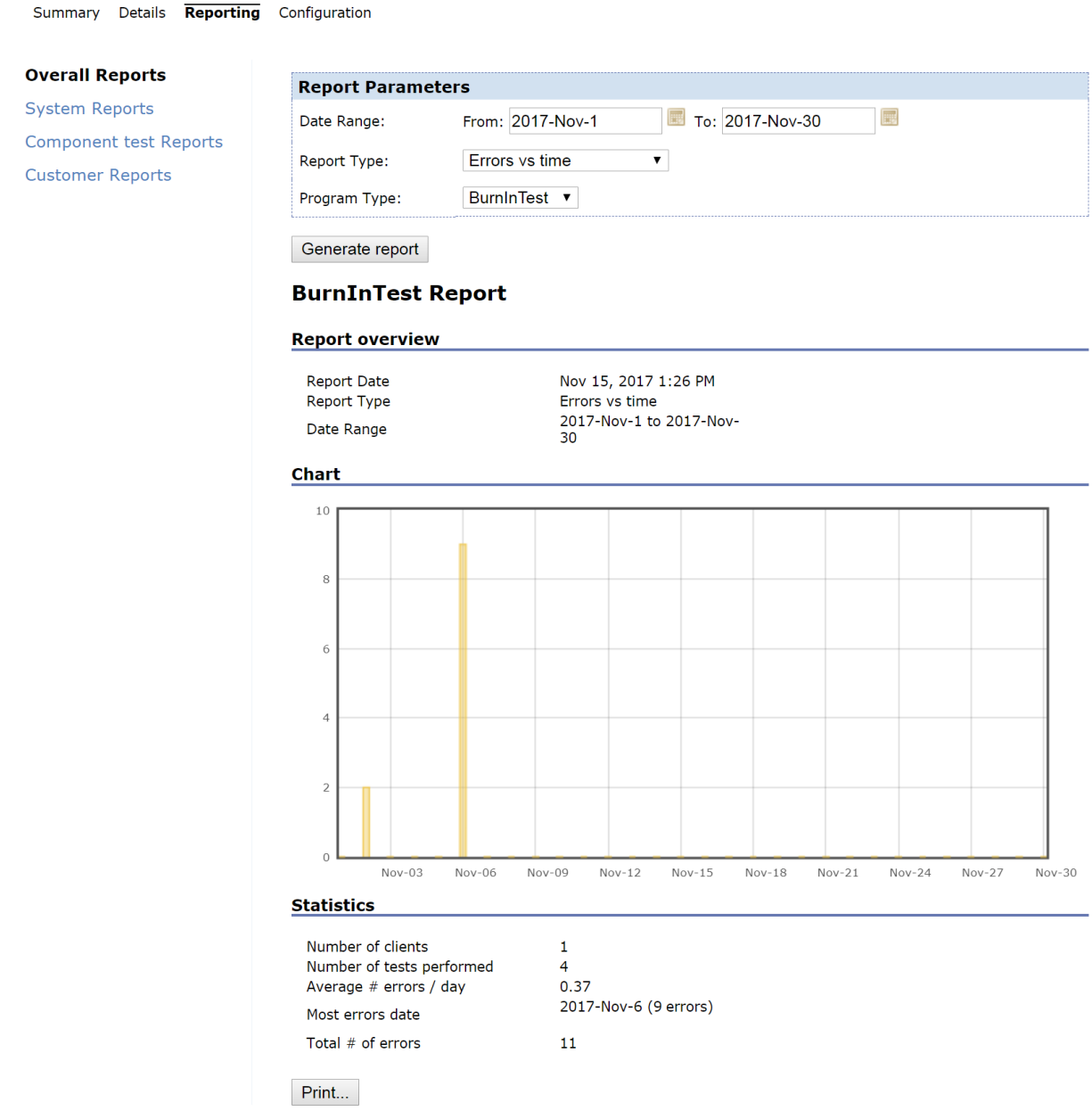
Test date	Test duration	# of errors	Result	Delete
Nov 15, 2017 10:13 AM	0:04:13	0 (Event Log)	PASS	☐
Nov 6, 2017 9:23 AM	0:00:00	9 (Event Log)	FAIL	☐
Nov 2, 2017 1:50 PM	0:00:00	2 (Event Log)	PASS	☐

Delete

Reporting

The Reporting option provides statistical reports across all tested systems, filtered by date, system, component type or customer. The report types include Errors vs. time, Failures vs. time, Tests performed vs. time and the Pass rate.

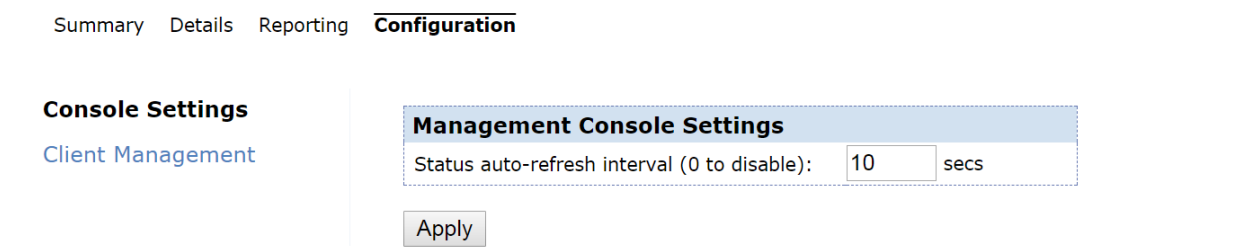
Overall Reports



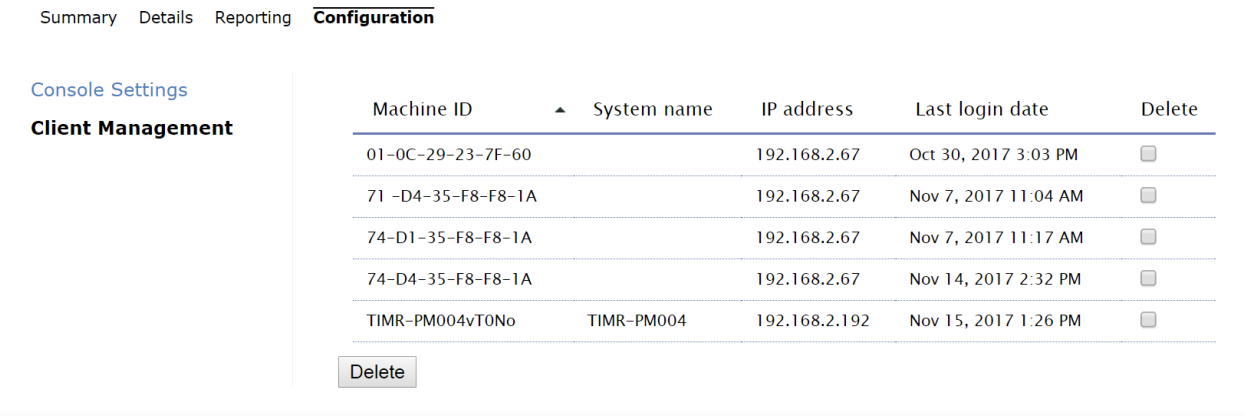
Configuration

The Configuration option allows the auto refresh of the live Management Console status web pages to be set. It also allows systems to be deleted from the Management Console database.

Console settings



Client Management



Notes:

1) Management Console errors are logged in the web server file: ManagementConsole-errors.log.
2) All date/times stored in the database are UTC offsets from January 1, 1970. When viewing date/times on the Management Console the date/times are adjusted for the time zone of the web server. If the web server time zone is different to the time zone of the systems under test, then it should be remembered that the times are not the test system's local time, but the Web server's local time.

Management Console – demonstration

PassMark Software has set up a demonstration of the Management Console:
http://www.passmark.com/burnintest/management/dashboard.php

BurnInTest 9.0.1000 or later can be managed (for demonstration purposes) by configuring the server address in BurnInTest Preferences->Management as:
http://www.passmark.com/burnintest/management/mgtconsolemsghandler.php

Troubleshooting

General Debugging

By adding the parameter "debugmode=1" to the dashboard address PHP errors and warnings will be enabled that may help troubleshoot any problems, eg;

http://localhost/mgtconsole/dashboard.php?debugmode=1

Errors can be logged to the ManagementConsole-errors.log file which can help debug database connection errors.

MySQL Database errors

If you see an error like "Warning: mysqli::mysqli(): (HY000/1045): Access denied for user 'username'@'localhost' (using password: YES)" when you navigate to the dashboard.php page then you may not have not setup the required username and password for the MySQL database. Open the settings.php file in a text editor and change the $database_hostname, $database_username, $database_password and $database_port values to suit your MySQL setup.

Version History

Here is a summary of all changes that have been made in each version of the Management Console.

Release 1.0 build 1000, 21 Nov 2017

- Rewrite of original BurnInTest Management Console
- Added support for Memtest86

Support

For technical support, questions, suggestions, please check our web page
http://www.passmark.com/support for our email address.

Ordering / Registration

Visit our sales information page:
http://www.passmark.com/sales

Appendix A – Creating the Management Console database

See the database_creation_script.sql included in the management console download.

```
CREATE DATABASE `passmark_mgtconsole`;

CREATE  TABLE `passmark_mgtconsole`.`test_session` (
 `session_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
 `machineid` VARCHAR(24) CHARACTER SET latin1 COLLATE latin1_general_cs NOT NULL,
 `session_timestamp` INTEGER NULL ,
 `application_type` tinyint(1) DEFAULT '0',
 `status` VARCHAR(16) NULL ,
 `timestart` INTEGER NULL ,
 `timeupdate` INTEGER NULL ,
 `duration` INTEGER UNSIGNED NULL  ,
 `remaining` INTEGER UNSIGNED NULL  ,
 `ip` VARCHAR(16) NULL ,
 `os` VARCHAR(100) NULL ,
 `cpu` VARCHAR(100) NULL ,
 `memory` VARCHAR(16) NULL ,
 `graphics` VARCHAR(100) NULL ,
 `opticaldisk` VARCHAR(100) NULL ,
 `sysname` VARCHAR(100) NULL ,
 `sysmodel` VARCHAR(100) NULL ,
 `mbmanufacturer` VARCHAR(100) NULL ,
 `mbname` VARCHAR(100) NULL ,
 `mbversion` VARCHAR(100) NULL ,
 `biosmanufacturer` VARCHAR(100) NULL ,
 `biosversion` VARCHAR(100) NULL ,
 `biosdate` VARCHAR(100) NULL ,
 `version` VARCHAR(16) NULL ,
 `customer` VARCHAR(32) NULL ,
 `technician` VARCHAR(32) NULL ,
 PRIMARY KEY (`session_id`) );


CREATE  TABLE `passmark_mgtconsole`.`sysinfo_details` (
		`session_id` int(11) unsigned NOT NULL,
 `title` VARCHAR(32) NULL ,
 `description` VARCHAR(120) NULL ,
 KEY (`session_id`) );

CREATE  TABLE `passmark_mgtconsole`.`hdd_details` (
		`session_id` int(11) unsigned NOT NULL,
 `model` VARCHAR(32) NULL ,
 `serial` VARCHAR(120) NULL ,
 `size` VARCHAR(32) NULL ,
 KEY (`session_id`) );

 CREATE  TABLE `passmark_mgtconsole`.`ram_details` (
		`session_id` int(11) unsigned NOT NULL,
		`name` VARCHAR(32) NULL ,
 `model` VARCHAR(64) NULL ,
 `serial` VARCHAR(64) NULL ,
 `size` VARCHAR(16) NULL ,
 `date` VARCHAR(32) NULL ,
 `manufacturer_specific` VARCHAR(64) DEFAULT NULL,
 KEY (`session_id`) );

CREATE  TABLE `passmark_mgtconsole`.`bit_results` (
		`result_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
		`session_id` int(11) unsigned NOT NULL,
 `timestart` INTEGER NOT NULL ,
 `timestop` int(11) DEFAULT NULL,
```

```
  `duration` int(10) unsigned DEFAULT NULL,
  `result` VARCHAR(16) NULL ,
                PRIMARY KEY (`result_id`),
  KEY (`session_id`,`result_id`) );

  CREATE  TABLE `passmark_mgtconsole`.`memtest_results` (
  `result_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
                `session_id` int(11) unsigned NOT NULL,
  `timestart` INTEGER NOT NULL ,
  `timestop` int(11) DEFAULT NULL,
  `duration` int(10) unsigned DEFAULT NULL,
  `result` VARCHAR(16) NULL ,
  PRIMARY KEY (`result_id`),
  KEY (`session_id`,`result_id`) );

CREATE  TABLE `passmark_mgtconsole`.`events` (
  `session_id` int(11) unsigned NOT NULL,
  `result_id` int(11) unsigned NOT NULL,
  `timestamp` INTEGER NULL ,
  `level` VARCHAR(16) NULL ,
  `type` VARCHAR(16) NULL ,
  `description` VARCHAR(256) NULL ,
  KEY (`session_id`,`result_id`) );


  CREATE TABLE `passmark_mgtconsole`.`testlist` (
  `session_id` int(11) unsigned NOT NULL,
  `result_id` int(11) unsigned NOT NULL,
  `test` varchar(50) NOT NULL DEFAULT '',
  `cycles` int(11) DEFAULT NULL,
  `operations` bigint(20) DEFAULT NULL,
  `result` varchar(16) DEFAULT NULL,
  `errors` int(11) DEFAULT NULL,
  `lasterror` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`result_id`,`session_id`, `test`) );

CREATE  TABLE `passmark_mgtconsole`.`configuration` (
  `setting` VARCHAR(16) NOT NULL ,
  `value` INTEGER NULL ,
  `string` VARCHAR(256) NULL ,
  KEY (`setting`) );


#You may need to grant privelages to your user
#GRANT all on passmark_mgtconsole.* to user;
```