SonarQube for Testwell CTC++ plugin

Application note

Date: 12 June 2015

Technologiepark Offenburg
In der Spoeck 10-12
77656 Offenburg
Germany

http://www.verifysoft.com

CONTENT

Overview of SonarQube for Testwell CTC++ plug-in

The SonarQube for Testwell CTC++ plug-in allows the import of the Testwell CTC++ coverage data into the SonarQube database and their visualisation within the SonarQube web interface.

The import of all Testwell CTC++ coverage levels (up to MC/DC and Multicondition Coverage) is possible. Testwell CTC++ and the SonarQube plug-in can be used for projects written in C, C++, Java, and C#.

In case of problems using this plug-in, please note there is an issue tracker on github.

In our example below, we are using sonarrunner, please note this is optional, and if you are a more experienced user, you would rather use the maven or ant-approach, as this would be the more automated way:

- SonarQube Runner: generic runner
- SonarQube Ant Task: runner for Ant projects
- SonarQube Maven Plugin: runner for Maven projects
- SonarQube Gradle Plugin: runner for Gradle projects
- SonarQube MSBuild Runner: runner for .Net projects
- CI Engines

Background

SonarQube™ is an open source platform for continuous inspection of code quality.

Quality is central

SonarQube is a web-based application. Rules, alerts, thresholds, exclusions, settings… can be configured online. By leveraging its database, SonarQube not only allows to combine metrics altogether but also to mix them with historical measures.

It can be extended with plug-ins, covering new languages, adding rules engines, computing advanced metrics can be done through a powerful extension mechanism.

Testwell CTC++

Testwell CTC++ is a leading test coverage tool for measuring Code Coverage on host and all embedded targets (even very small ones). The tool is compliant to Safety Standards. It works seamlessly with many tool chains, testing tools and test environments.

Main features

The Testwell-CTC++ reports are used to import coverage data.
Statement-Coverage could be selected as measure (e.g. for the treemap-widget).
The coverage data is linked to the source files.
Use of custom metrics per default
Standard way of integration (package and drop the plug-in, restart the server)
Invasive-Mode (further explanation later)
Usable API for parser and metrics
Mapping of Files to HTML-Reports in the Database used by widget

Invasive mode

You have the option to overwrite some of the Core metrics to fit your needs.
If you do this, your TER-Data will be saved into the Coverage Metric normally used for Condition Coverage in the Test-Domain. Please consider, that it may happen, that different Coverage-Plug-in may overwrite each other! Also notice, that if you use the invasive mode, you may experience a plus on LineCoverage. Due to some wacky behaviour of Sonar, you could only show branchcoverage on the source code, if the line in question is covered. This solution will probably be fixed with the next iteration.

Prerequisites

For this tutorial, we used the following:

-**Java Virtual Machine** (Might already be installed or available for free from Oracle)

-A **Windows Native compiler/linker** (Microsoft Visual Studio Express 2008 recommended)

-**SonarQube Server** (4.5.4) + **Runner** (2.4)

http://www.sonarqube.org/downloads/

-**Cxx plug-in** (0.9.3)

https://github.com/wenns/sonar-cxx

-**CTC++ plug-in** (1.0.1)

https://github.com/Londran/sonar-ctc/releases

Installation

✓ Unzip SonarQube server and runner zip files

e.g in C:\sonarqube-4.5.4 and C:\sonar-runner-dist-2.4

✓ Drop the *.jar files into your plug-in-folder of your SonarQube server.

e.g C:\sonarqube-4.5.4\extensions\plugins

✓ add path for runner\bin

e.g C:\sonar-runner-dist-2.4\bin

- ✓ add env variable to Windows SONAR_RUNNER_HOME =>$RunnerHome

e.g C:\sonar-runner-dist-2.4

The following screen dumps illustrate how to update the variables.

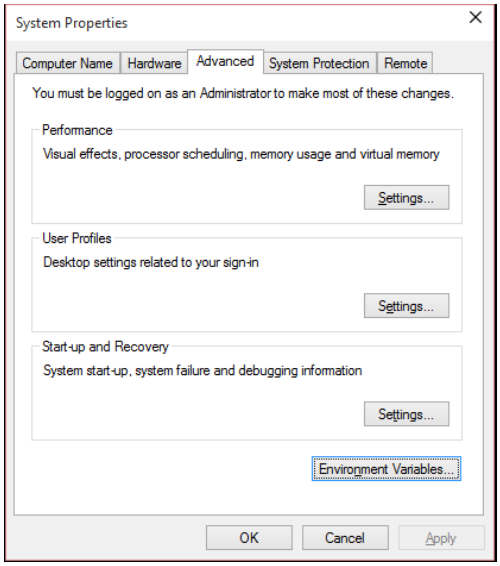


Figure1: Update Windows Environment Variables

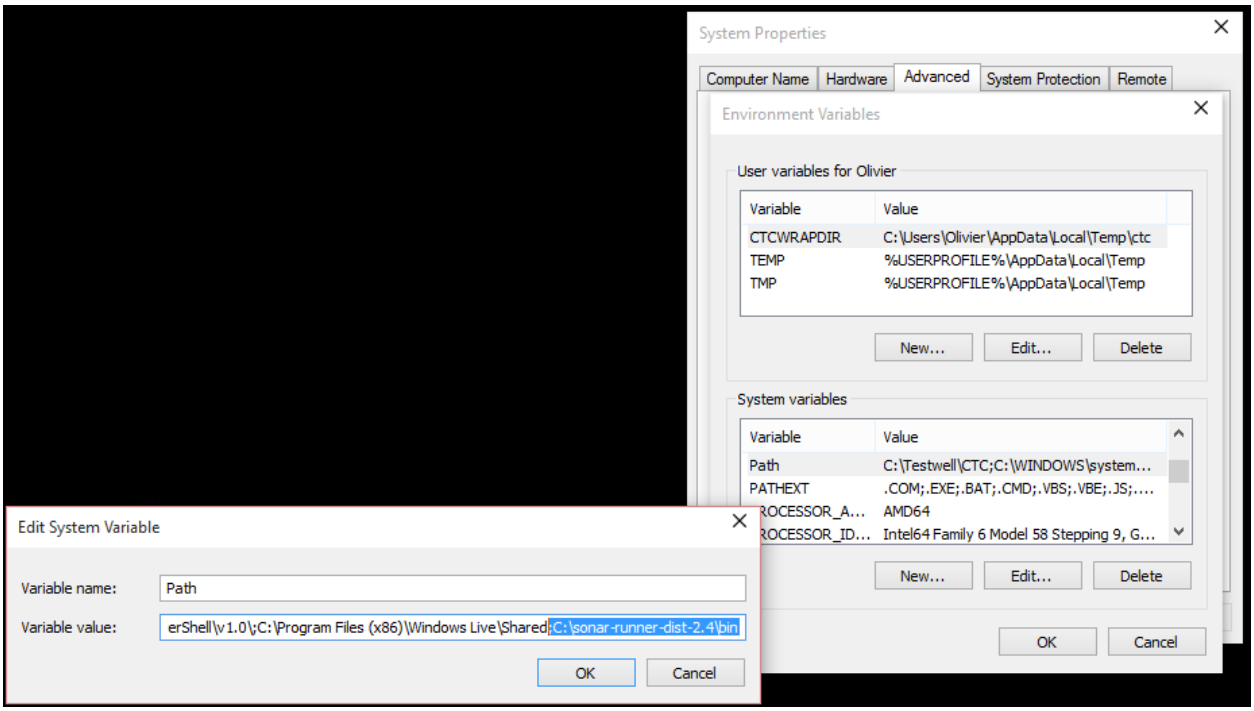
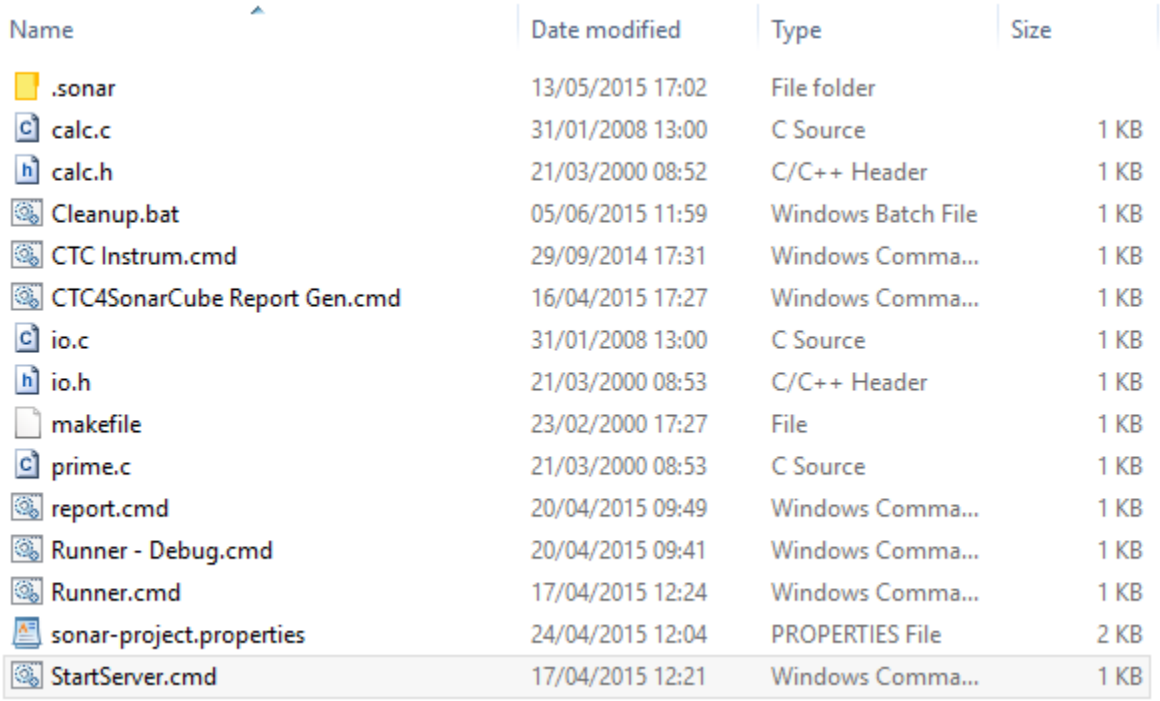


Figure2: Update Windows Environment Variables- continuing

Steps

You first need to install the provided files stored in SonarQube_CTC.zip.

You should get something very similar



Name	Date modified	Type	Size
.sonar	13/05/2015 17:02	File folder	
calc.c	31/01/2008 13:00	C Source	1 KB
calc.h	21/03/2000 08:52	C/C++ Header	1 KB
Cleanup.bat	05/06/2015 11:59	Windows Batch File	1 KB
CTC Instrum.cmd	29/09/2014 17:31	Windows Comma...	1 KB
CTC4SonarCube Report Gen.cmd	16/04/2015 17:27	Windows Comma...	1 KB
io.c	31/01/2008 13:00	C Source	1 KB
io.h	21/03/2000 08:53	C/C++ Header	1 KB
makefile	23/02/2000 17:27	File	1 KB
prime.c	21/03/2000 08:53	C Source	1 KB
report.cmd	20/04/2015 09:49	Windows Comma...	1 KB
Runner - Debug.cmd	20/04/2015 09:41	Windows Comma...	1 KB
Runner.cmd	17/04/2015 12:24	Windows Comma...	1 KB
sonar-project.properties	24/04/2015 12:04	PROPERTIES File	2 KB
StartServer.cmd	17/04/2015 12:21	Windows Comma...	1 KB

- ✓ Compile & link the prime example using the CTC Instrum.cmd script.

e.g. Open a Visual Studio 2008 Command prompt, and launch the script

You should get as a result a couple of new files (*.obj, MON.sym, and prime.Exe)

- ✓ Run prime.exe and enter a few values in order to get some coverage values
- ✓ Enter 0 to finish; you should get a MON.dat file

Next step is to generate a coverage report compatible with our plug-in

- ✓ Run it using the CTC4SonarCube Report Gen.cmd script.
- ✓ Now launch SonarQube server, from a command line for instance.

e.g C:\sonarqube-4.5.4\bin\windows-x86-64\StartSonar.bat

When ready, you should see the following message: `Process[web] is up`

The project folder provides the requested files, a full working example based on the prime demo.

If you prefer to work on your own example, here are below some hints:

- ✓ Create sonar-project.properties with:

```
# Required metadata
sonar.projectKey=prime
sonar.projectName=Testwell CTC ++ C project analyzed with the SonarQube Runner
sonar.projectVersion=1.0
# Comma-separated paths to directories with sources (required)
sonar.sources=.
# Language, needed for SonarQube < 4.2
sonar.language=c++
# The build-wrapper output dir
# sonar.cfamily.build-wrapper-output=/path/to/build-wrapper/output/dir
# Optional comma-separated list of additional libraries folders (e.g. /usr/include)
# sonar.cfamily.library.directories=lib
# Optional specific predefined macros
# sonar.cfamily.predefinedMacros=#define MY_MACRO(a) ((a)+1),#define DEBUG
# Encoding of the source files
sonar.sourceEncoding=UTF-8
```

- ✓ edit runner.cmd with:

$Home\\sonar-runner-dist-2.4\sonar-runner-2.4\bin\sonar-runner.bat

- ✓ Run it !
- ✓ Open a Web Browser
- ✓ Type http://localhost:9000/

You should view SonarQube dashboard now

- ✓ Click on our Project (top right)
- ✓ Log in (admin – admin is default)
- ✓ Click “Configure widgets”

- ✓ Add a configurable widget

e.g. a Timeline one

- ✓ Click Edit
- ✓ Type in CTC in the Metrics field, pick up one

Enjoy! You should get a screen similar to this