

Parasoft Findings for SonarQube

View online: <https://docs.parasoft.com/x/Ny1-C>

In this section:

- Introduction
- Requirements
- Installing the Parasoft Findings Plugin
- Activating Parasoft Profiles
- Setting the Report Path for Static Analysis and Test Execution Results
- Setting the Report Path for Coverage
- Uploading Project Results
- Viewing Static Analysis Results
- Viewing Test Execution Results
- Viewing Code Coverage Results
- Third-party Acknowledgments

View online: <https://docs.parasoft.com/x/Ny1-C>

Introduction

The Parasoft Findings Plugin for SonarQube allows you to view static analysis, functional test results, unit test results, and code coverage results within SonarQube. It grants SonarQube the ability to analyze data from Parasoft XML reports and use it to report bugs, vulnerabilities, functional test results, unit test results, code coverage or code smells from within SonarQube. See [Uploading Project Results](#) for more details.

The plugin can consume the following report types:

- Static analysis, metrics analysis, code coverage, and unit test reports generated by 2022.2+ versions of C/C++test, Jtest, and dotTEST.
- Functional test reports generated by Parasoft SOAtest 2022.2+.

Requirements

- For version 10.6.1, a Parasoft product must be installed on the same machine as the SonarQube server since the plugin loads rules from a Parasoft product installation.
 - For C/C++test, the Standard edition must be installed even if you are sending C/C++test Professional reports to SonarQube.
 - This requirement does not apply to version 10.6.2+.
- SonarQube versions 8.9+ are supported.
- SonarQube Developer edition or better is required to process C/C++test reports.
- Language projects versions 2022.2+ are supported.

Installing the Parasoft Findings Plugin

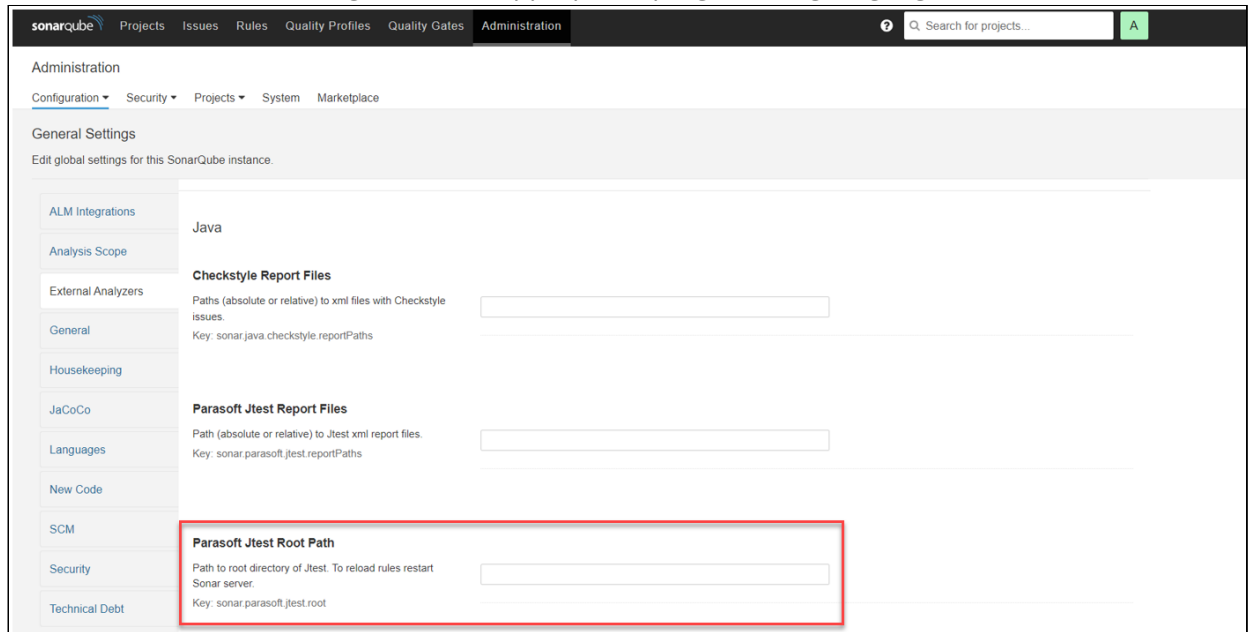
1. Either download the plugin or build it yourself (10.6.2+ only):
 - a. To download the plugin, go to <https://customerportal.parasoft.com/lightningportal/s/marketplace> and download the Parasoft SonarQube plugin jar `parasoft.findings.sonar-<VERSION>.jar`.
 - Be sure to download the version of the plugin that matches the version of your Parasoft product. For example, if you are using Jtest, dotTEST, or C++Test 2023.1, you would download `parasoft.findings.sonar-2023.1.jar`.
 - b. To build the plugin yourself, see [Building Your Own Plugin](#).
2. Copy the plugin jar into the SonarQube `extensions/plugins` directory.
3. Restart the SonarQube server.
4. For version 10.6.1: set the root path to the Parasoft product (see [Setting the Root Path to the Parasoft Product for Version 10.6.1](#)).
 - a. Starting with version 10.6.2, rule files are packaged in the plugin jar, so it is not necessary to set the root path to load rules with version 10.6.2+.

See <https://docs.sonarqube.org/latest/setup/install-plugin/> for more details.

Setting the Root Path to the Parasoft Product for Version 10.6.1

1. In the SonarQube web UI go to **Administration > Configuration > General Settings > External Analyzers**.

2. Locate the **Root Path** setting under the appropriate programming language.



3. In the field enter the absolute path to the Parasoft product installation and click **Save**.
4. Restart the SonarQube server so that the rule definitions are loaded.
5. After the server has restarted, go to **Quality Profiles** in the web UI and select the built-in Parasoft profile under the appropriate supported language. Verify that the rule definitions are loaded successfully under the profile.

If zero or only one rule is loaded, then loading of rules failed. In this case:

- a. Check that the root path is correct.
- b. Review the SonarQube web server logs for any error messages.

Building Your Own Plugin

You can build your own plugin for your Parasoft products. To do so, you will need JDK 11+, Maven 3.3+, and your Parasoft products installed on the same machine.

1. Clone the source code from GitHub found here: <https://github.com/parasoft/parasoft-findings-sonar>.

View online: <https://docs.parasoft.com/x/Ny1-C>

2. Run a Maven package command that is appropriate for the Parasoft products you have installed. The example below includes Jtest, dotTEST, and C++Test; if you don't have one or more of these installed, remove its root path reference:

```
mvn clean package -DjtestRootPath="<JTEST-INSTALL-ROOT-PATH>" -DdottestRootPath="<DOTTEST-INSTALL-ROOT-PATH>" -DcpptestRootPath="<CPPTTEST-INSTALL-ROOT-PATH>"
```

- For C/C++test, the root path must be set to Standard edition, even if you are sending C/C++test Professional reports to SonarQube.

The plugin jar that is created will be in the `<SOURCE-CODE-ROOT-PATH>/target` folder and will include the rule files for your Parasoft product.

Activating Parasoft Profiles

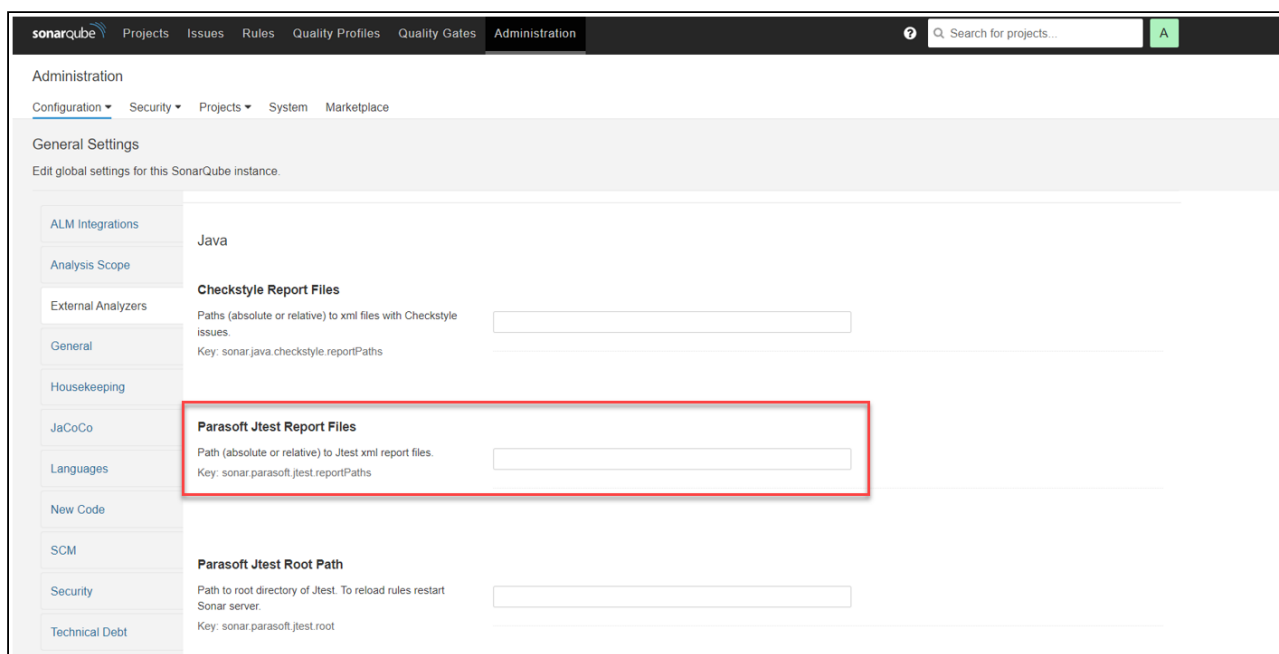
There are two ways to enable the Parasoft profile for your projects:

- To set a Parasoft profile as the default for new projects, go to **Quality Profiles** and select **Set as Default** from the gear menu to the right of the profile.
- To set a Parasoft profile for a given project, go to **Quality Profiles** section in the project settings. Select the **Change Profile** button on the right side of the appropriate language.

Setting the Report Path for Static Analysis and Test Execution Results

For Parasoft results to be uploaded, the path to the report files must be set. There are several ways to configure this.

- To configure the path globally go to **Administration > Configuration > General Settings > External Analyzers** and find the **Report Files** setting for your Parasoft product.



Enter the path to the report.xml file of your project. For example:

```
target/jtest/report.xml
```

You can add multiple paths, each in a separate field. When the Parasoft scanner runs, results will be loaded from each valid report file path.

- To configure the path for a project, go to the **External Analyzers** section in the project's settings.
- To configure the path when running the sonar scanner, include the report path's settings key. For example:

View online: <https://docs.parasoft.com/x/Ny1-C>

- Jtest using Maven:

```
mvn sonar:sonar -Dsonar.token=<your sonar token> -Dsonar.host.url="http://  
<your sonar server>:<your sonar server port>/"  
-Dsonar.parasoft.jtest.reportPaths=target/report.xml...<additional sonar  
settings>
```

- Jtest using sonar-scanner:

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your  
sonar token> -Dsonar.host.url="http://<your sonar server>:<your sonar  
server port>/" -Dsonar.parasoft.jtest.reportPaths=target/  
report.xml...<additional sonar settings>
```

- C/C++test using sonar-scanner:

```
build-wrapper-win-x86-64.exe --out-dir  
build_wrapper_output_directory...<your build commands i.e.: make clean all  
"C:/cpptest/examples/FlowAnalysisCpp">
```

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your  
sonar token> -Dsonar.host.url="http://<your sonar server>:<your sonar  
server port>/" -Dsonar.parasoft.cpptest.reportPaths=target/  
report.xml...<additional sonar settings>
```

- dotTEST using .NET's sonar-scanner (**Note:** Sonar-scanner must be installed via the command: `dotnet tool install --global dotnet-sonarscanner`):

```
dotnet sonarscanner begin /k:"<your solution>" /d:sonar.token="<your sonar  
token>" /d:sonar.parasoft.dottest.reportPaths="target/report.xml"...<additi  
onal sonar settings>
```

```
dotnet build <your solution>
```

```
dotnet sonarscanner end /d:sonar.token="<your sonar token>"
```

- dotTEST using Sonar's provided sonar-scanner:

Setting the Report Path for Static Analysis and Test Execution Results

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your sonar token> -Dsonar.host.url="http://<your sonar server>:<your sonar server port>/" -Dsonar.parasoft.dottest.reportPaths=target/report.xml...<additional sonar settings>
```

- SOAtest using Sonar's provided sonar-scanner:

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your sonar token> -Dsonar.host.url="http://<your sonar server>:<your sonar server port>/" -Dsonar.parasoft.soatest.reportPaths=target/report.xml...<additional sonar settings>
```

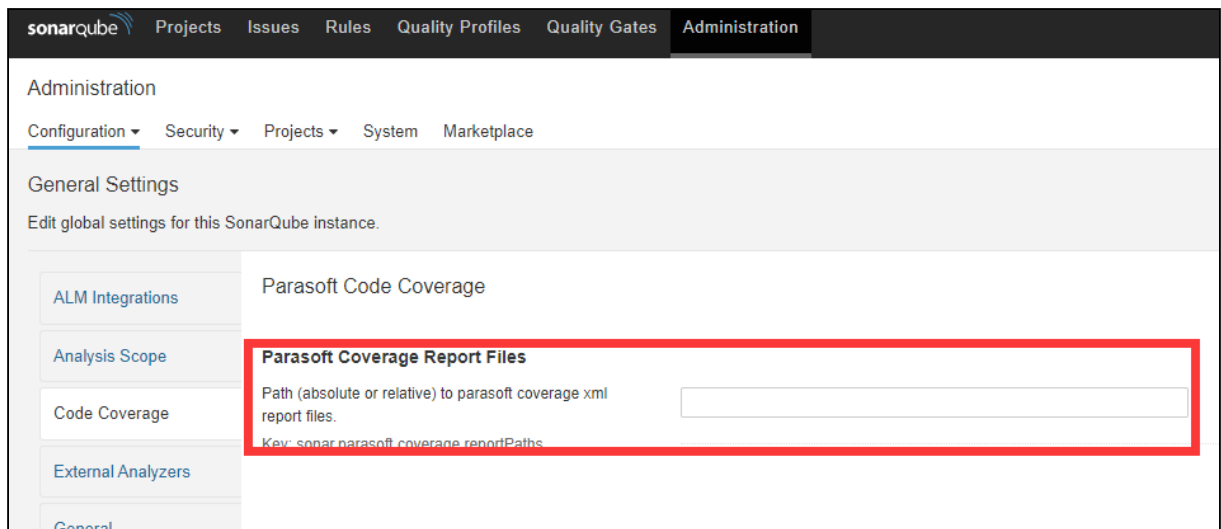
- Sonar-scanner can be downloaded from: <https://docs.sonarqube.org/latest/analyzing-source-code/scanners/sonarscanner/>
- Jtest unit test results will be skipped if Maven Surefire reports exist in target/surefire-reports folder for the project, which will be parsed automatically by the built-in sensor of Sonar.
- You can include multiple report files:

```
mvn sonar:sonar -Dsonar.parasoft.jtest.reportPaths=target/report.xml -Dsonar.parasoft.jtest.reportPaths=target/report_2.xml...<additional sonar settings>
```

Setting the Report Path for Coverage

For Parasoft results to be uploaded, the path to the report files must be set. There are several ways to configure this.

- To configure the path globally go to **Administration > Configuration > General Settings > Code Coverage** and find the **Parasoft Coverage Report Files** setting for your Parasoft product.



Enter the path to the report.xml file of your project. For example:

```
report/coverage.xml
```

or

```
D:\project\report\coverage.xml
```

- To configure the path when running the sonar scanner, include the report path's settings key. For example:

- Jtest using Maven:

```
mvn sonar:sonar -Dsonar.token=<your sonar key> -Dsonar.host.url="http://
<your sonar server>:<your sonar server port>/"
-Dsonar.parasoft.coverage.reportPaths=target/coverage.xml...<additional
sonar settings>
```

- Jtest using sonar-scanner:

View online: <https://docs.parasoft.com/x/Ny1-C>

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your sonar key> -Dsonar.host.url="http://<your sonar server>:<your sonar server port>/" -Dsonar.parasoft.coverage.reportPaths=target/coverage.xml...<additional sonar settings>
```

- C/C++ test using sonar-scanner:

```
build-wrapper-win-x86-64.exe --out-dir build_wrapper_output_directory...<your build commands i.e.: make clean all "C:/cpptest/examples/FlowAnalysisCpp">
```

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your sonar key> -Dsonar.host.url="http://<your sonar server>:<your sonar server port>/" -Dsonar.parasoft.coverage.reportPaths=target/coverage.xml...<additional sonar settings>
```

- dotTEST using .NET's sonar-scanner (**Note:** Sonar-scanner must be installed via the command: `dotnet tool install --global dotnet-sonarscanner`):

```
dotnet sonarscanner begin /k:"<your solution>" /d:sonar.token="<your sonar key>" /d:sonar.parasoft.coverage.reportPaths="target/coverage.xml"...<additional sonar settings>
```

```
dotnet build <your solution>
```

```
dotnet sonarscanner end /d:sonar.token="<your sonar key>"
```

- dotTEST using Sonar's provided sonar-scanner:

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your sonar key> -Dsonar.host.url="http://<your sonar server>:<your sonar server port>/" -Dsonar.parasoft.coverage.reportPaths=target/coverage.xml...<additional sonar settings>
```

- SOAtest using Sonar's provided sonar-scanner:

```
sonar-scanner-<version>-windows/bin/sonar-scanner.bat -Dsonar.token=<your  
sonar key> -Dsonar.host.url="http://<your sonar server>:<your sonar server  
port>/" -Dsonar.parasoft.soatest.reportPaths=target/  
report.xml...<additional sonar settings>
```

- Sonar-scanner can be downloaded from: <https://docs.sonarqube.org/latest/analyzing-source-code/scanners/sonarscanner/>

Uploading Project Results

To upload Parasoft results for a project, first run the command which generates the report.xml file, and then run the usual Sonar scanner command. For example, with a Maven project setup using Jtest, a single command can be run from the root directory of your project:

```
mvn jtest:jtest sonar:sonar -Dsonar.projectKey=<project key> -Dsonar.host.url=<server url> -Dsonar.token=<login token>
```

See the documentation for your Parasoft product for details on how to run test cases and generate a test execution report:

- For Jtest: <https://docs.parasoft.com/display/JTEST20232/Running+Unit+Tests>
- For dotTEST: <https://docs.parasoft.com/display/DOTTEST20232/Running+Unit+Tests>
- For C/C++test: <https://docs.parasoft.com/display/CPPTTEST20232/Integrating+with+CppUnit+and+CppUtest>
- For SOAtest: <https://docs.parasoft.com/display/SOAVIRT20232/CLI+Usage>

See the documentation for your Parasoft product for details on how to run test cases and generate a coverage report:

- For Jtest: <https://docs.parasoft.com/display/JTEST20232/Coverage+for+Unit+Test>
- For dotTEST: <https://docs.parasoft.com/display/DOTTEST20232/Coverage+for+Unit+Test>
- For C/C++test: <https://docs.parasoft.com/display/CPPTTEST20232/Collecting+Application+Coverage+for+CMake+Projects>

See <https://docs.sonarqube.org/latest/analysis/overview/> for details on how to run the Sonar scanner on projects using different build systems.

View online: <https://docs.parasoft.com/x/Ny1-C>

Viewing Static Analysis Results

After the results are uploaded, they can be viewed in the project in the **Issues** tab.

The screenshot displays the SonarQube web interface. At the top, the navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar and a green 'A' icon are on the right. Below the navigation bar, the 'Demo Project' is selected, with a 'master' branch. A status bar indicates 'Last analysis had 1 warning' on August 8, 2022, at 10:16 AM, version 1.0.0. The 'Issues' tab is active, showing a list of issues. On the left, a 'Filters' sidebar allows filtering by 'Type' (BUG, Vulnerability, Code Smell) and 'Severity' (Blocker, Critical, Major, Minor, Info). The main area shows a list of issues for the file 'src/_/java/examples/flowanalysis/AlwaysCloseGSS.java'. The issues are categorized by type and severity, with options to filter by 'Bug', 'Blocker', 'Open', and 'Not assigned'. The issues are sorted by '4 minutes ago' and include a 'Why is this an issue?' link. The issues are: 'Security context not disposed: GSSManager.getInstance().createContext(tokens)', 'context may possibly be null', 'Image resource not closed: imgReader', and 'imgReader may possibly be null'. The bottom of the page shows a 'View online' link: <https://docs.parasoft.com/x/Ny1-C>.

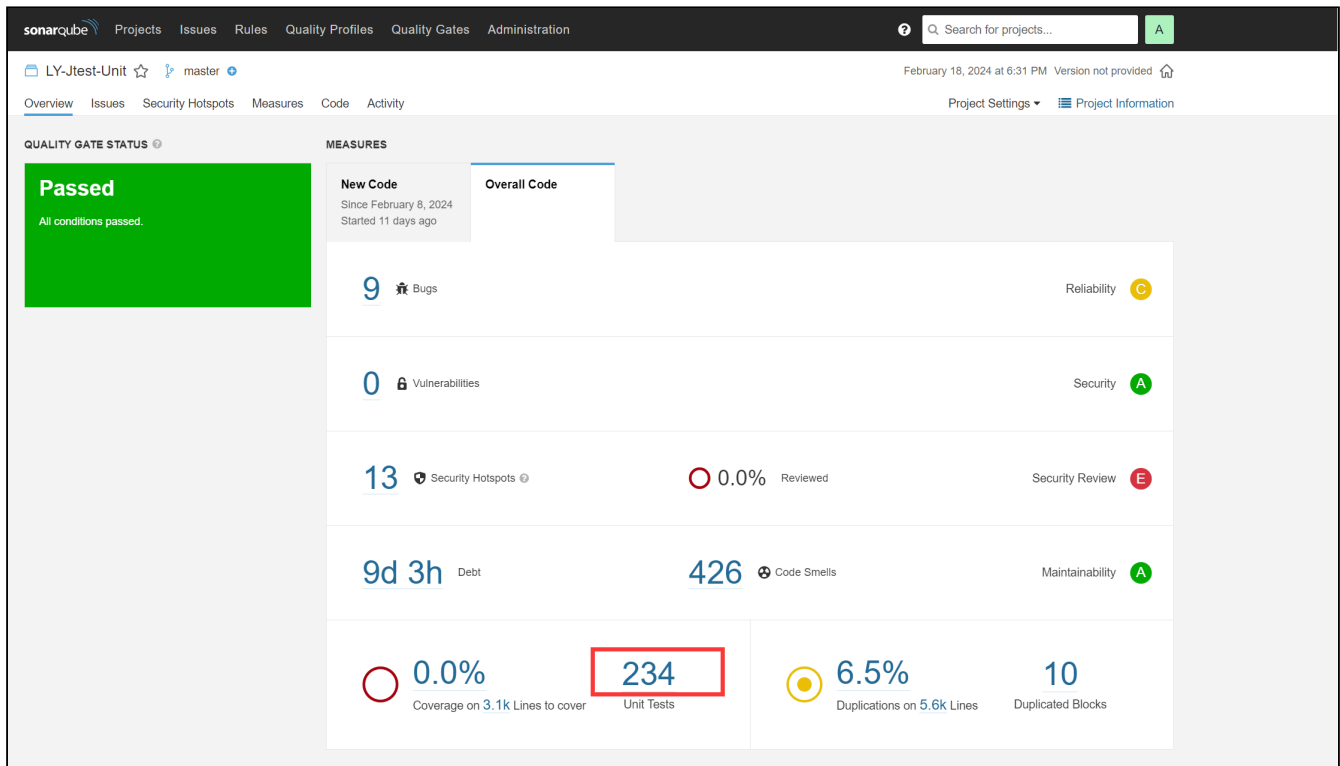
The screenshot shows the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is on the right. The main content area is divided into two panels. The left panel shows a list of issues for the 'Demo Project' (master branch). The right panel shows the code for the 'AlwaysCloseGSS' class, with two issues highlighted in red boxes. The issues are 'Security context not disposed: GSSManager.getInstance().createContext(tokens)' and 'Security context not disposed: GSSManager.getInstance().createContext(tokens)'. Both issues are marked as 'Bug' and have a severity of 'Blocker'. The code is a Java class with two methods: 'process' and 'processClose'. The 'process' method calls 'GSSManager.getInstance().createContext(tokens)' and 'context.initSecContext(inputBuff, 0, 256)'. The 'processClose' method calls 'GSSManager.getInstance().createContext(tokens)' and 'context.initSecContext(inputBuff, 0, 256)'. The issues are highlighted in red boxes, showing the error message, severity (Bug), and a link to the issue details.

C/C++test Professional Report Settings

If you are generating static analysis reports with C/C++test Professional 2023.1 or earlier, make sure the **Add absolute file paths to XML data** option is enabled to show the issues if it is stored in the SonarQube project. You can enable this option on the command line by using the option `-property report.contexts_details=true` or by setting the `report.contexts_details=true` property in the settings file.

Viewing Test Execution Results

After the results are uploaded, in addition to SOAtest test execution results, they can be viewed in the project in the **Overview** tab.



Viewing Test Execution Results

The screenshot displays the SonarQube web interface for the project 'LY-Jtest-Unit'. The 'Measures' tab is active, showing a list of unit tests. On the left, the 'Project Overview' sidebar is visible, with the 'UNIT TESTS' section highlighted. A red box in the sidebar highlights the 'Unit Tests' summary: 234 tests, 3 errors, 0 failures, 98.7% success, and 49s duration. The main area shows a list of 69 files, each with a corresponding test class and its execution count.

File Path	Count
src/test/java/com/parasoft/parabank/web/controller/AccountActivityControllerTest.java	1
src/test/java/com/parasoft/parabank/web/controller/AccountsOverviewControllerTest.java	1
src/test/java/com/parasoft/parabank/web/AccountTypeConverterTest.java	1
src/test/java/com/parasoft/parabank/domain/logic/impl/AvailableFundsLoanProcessorTest.java	1
src/test/java/com/parasoft/parabank/domain/logic/impl/CombinedLoanProcessorTest.java	1
src/test/java/com/parasoft/parabank/web/controller/DatabaseControllerTest.java	1
src/test/java/com/parasoft/parabank/domain/logic/impl/DownPaymentLoanProcessorTest.java	1
src/test/java/com/parasoft/parabank/web/controller/ErrorControllerTest.java	1
src/test/java/com/parasoft/parabank/dao/jdbc/JdbcSequenceDaoTest.java	1
src/test/java/com/parasoft/parabank/messaging/UmsLoanProcessorTest.java	1
src/test/java/com/parasoft/parabank/messaging/LocalLoanProviderTest.java	1
src/test/java/com/parasoft/parabank/web/LoginInterceptorTest.java	1
src/test/java/com/parasoft/parabank/web/controller/LogoutControllerTest.java	1
src/test/java/com/parasoft/parabank/web/controller/OpenAccountControllerTest.java	1
src/test/java/com/parasoft/bookstore2/OrderTest.java	1
src/test/java/com/parasoft/parabank/web/TemplateViewResolverTest.java	1
src/test/java/com/parasoft/parabank/web/controller/TransferControllerTest.java	1
src/test/java/com/parasoft/parabank/web/ViewUtilTest.java	1
src/test/java/com/parasoft/parabank/domain/validator/AddressValidatorTest.java	2
src/test/java/com/parasoft/parabank/web/form/AdminFormTest.java	2
src/test/java/com/parasoft/parabank/test/util/BeanTestCaseTest.java	2
src/test/java/com/parasoft/parabank/web/controller/CustomerLoginControllerTest.java	2

For SOAtest test execution results, it can be viewed in the project in the **Measures** tab.

View online: <https://docs.parasoft.com/x/Ny1-C>

Viewing Test Execution Results

The screenshot displays the SonarQube web interface for the project 'parabank-soatest-tests-maven'. The 'Measures' tab is active, showing a table of test results. A red box highlights the 'SOATEST TESTS' section, which contains the following data:

Parasoft SOATEST TESTS	
SOATEST Tests	86
SOATEST Test Failures	37
SOATEST Test Success	57.0%
SOATEST Test Duration	15s

Below the table, a warning message states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

C/C++test Professional Unit Test Results

C/C++test reports for unit test results must be generated with the **Overview of checked files and executed tests** option enabled. You can enable this option on the command line by using the option `-property report.contexts_details=true` or by setting the `report.contexts_details=true` property in the settings file.

Starting with version 2024.1, you can also use the command line option `-property report.additional.report.dir=<REPORT_DIR>` when generating the reports and use reports generated in this directory.

View online: <https://docs.parasoft.com/x/Ny1-C>

Viewing Code Coverage Results

After the results are uploaded, they can be viewed in the project in the **Overview** tab.

The screenshot shows the JTest-cover Overview tab for the 'Jtest-cover' project. The Quality Gate Status is 'Passed' with the message 'All conditions passed.' The Measures section shows various metrics:

- New Code:** 0 Bugs, 0 Vulnerabilities, 0 Security Hotspots, 0 Debt.
- Overall Code:** Reliability (A), Security (A), Security Review (A), Maintainability (A).
- Code Smells:** 0.
- Coverage:** 64.7% (highlighted with a red box). Coverage on 3.3k Lines to cover.
- Duplications:** 2.7% (Duplications on 7.6k Lines).
- Duplicated Blocks:** 19.

The screenshot shows the JTest-cover Measures tab for the 'Jtest-cover' project. The Project Overview section on the left shows the Coverage status: 64.7%.

File	Coverage	Count
src/main/java/com/parasoft/parabank/service/LoanProcessorService.java	0.0%	1
src/main/java/com/parasoft/parabank/service/ParaBankServiceConstants.java	0.0%	25
src/main/java/com/parasoft/parabank/dao/jdbc/internal/SampleJdbcDynamicDataInserter.java	0.0%	3
src/main/java/com/parasoft/parabank/dao/jdbc/SecureJdbcCustomerDao.java	0.0%	6
src/main/java/com/parasoft/parabank/dao/jdbc/SecureJdbcTransactionDao.java	0.0%	11
src/main/java/com/parasoft/parabank/util/Transactions.java	0.0%	5
src/main/java/com/parasoft/parabank/util/AccessModeController.java	4.5%	472
src/main/java/com/parasoft/parabank/web/controller/NewsController.java	6.7%	14
src/main/java/com/parasoft/parabank/web/form/TransferForm.java	10.0%	9
src/main/java/com/parasoft/parabank/util/CustomWadlGenerator.java	12.5%	7
src/main/java/com/parasoft/parabank/web/form/OpenAccountForm.java	14.3%	6
src/main/java/com/parasoft/parabank/web/controller/TransactionController.java	14.8%	23

View online: <https://docs.parasoft.com/x/Ny1-C>

The screenshot displays the JTest-cover application interface. The top navigation bar includes tabs for Overview, Issues, Security Hotspots, Measures (selected), Code, and Activity. The breadcrumb path is Jtest-cover / src/.../com/parasoft/parabank / web / form / OpenAccountForm.java.

Project Overview

- Reliability
- Security
- Security Review
- Maintainability
- Coverage** **COVERAGE**
 - Overview
 - Overall
 - Coverage 14.3%**
 - Lines to Cover 7
 - Uncovered Lines 6
 - Line Coverage 14.3%
 - Tests
 - Errors 0
 - Failures 0
 - Skipped 0
 - Success 100%
 - Duplications
 - Size

Code Coverage Details

Coverage 14.3%

src/.../java/com/parasoft/parabank/web/form/OpenAccountForm.java

```

1 sdebr... package com.parasoft.parabank.web.form;
2
3 benke... import com.parasoft.parabank.domain.Account.AccountType;
4 sdebr...
5 /**
6  * Backing class for account creation form
7  */
8 public class OpenAccountForm {
9     private AccountType type;
10    private int fromAccountId;
11
12    public AccountType getType() {
13        return type;
14    }
15
16    public void setType(AccountType type) {
17        this.type = type;
18    }
19
20    public int getFromAccountId() {
21        return fromAccountId;
22    }

```

A tooltip on line 8 states: "Fully covered by tests."

C/C++test Professional Reports

Code coverage reports for C/C++test Professional are not supported for versions prior to 2024.1.

Starting with version 2024.1, you can also use the command line option `-property report.additional.report.dir=<REPORT_DIR>` when generating the reports and use reports generated in this directory.

View online: <https://docs.parasoft.com/x/Ny1-C>

Third-party Acknowledgments

The Parasoft Findings Plugin for SonarQube uses the following third-party software:

Apache Commons Codec

This software is used under an [Apache License 2.0](#) with this [notice](#).

Apache Commons Collections

This software is used under an [Apache License 2.0](#) with this [notice](#).

Apache HttpClient

This software is used under an [Apache License 2.0](#) with this [notice](#).

Apache HttpClient Fluent API

This software is used under an [Apache License 2.0](#) with this [notice](#).

Apache HttpClient Mime

This software is used under an [Apache License 2.0](#) with this [notice](#).

Apache HttpCore

This software is used under an [Apache License 2.0](#) with this [notice](#).

Dom4j

This software is used under a [BSD License](#).

View online: <https://docs.parasoft.com/x/Ny1-C>

Jackson-annotations

This software is used under an [Apache License 2.0](#) with this [notice](#).

Jackson-core

This software is used under an [Apache License 2.0](#) with this [notice](#).

jackson-databind

This software is used under an [Apache License 2.0](#) with this [notice](#).

JRCS Diff

This software is used under a [LGPL License](#).

Saxon-HE

This software is used under an [MPL 2.0 license](#).

SLF4J API Module

This software is used under an [MIT License](#).

xmlresolver

This software is used under an [Apache License 2.0](#).

zip4j

This software is used under an [Apache License 2.0](#) with this [notice](#).

View online: <https://docs.parasoft.com/x/Ny1-C>