Analysis Parameters

Parameters to configure project analysis can be set in multiple places. Here is the hierarchy of parameters:

- Global analysis parameters, defined in the UI, apply to all the projects (From the top bar, go to Settings > General Settings)
- Project analysis parameters, defined in the UI, override global parameters (At a project level, go to Configuration > Settings)
- Project analysis parameters, defined in a project analysis configuration file or an analyzer configuration file, override the ones defined in the UI
- Analysis / Command line parameters, defined when launching an analysis, override project analysis parameters

Note that only parameters set through the UI are stored in the database. For example, if you override the `sonar.exclusions` parameter via command line for a specific project, it will not be stored in the database. Local analyses in Eclipse, for example, would still be executed with the exclusions defined in the UI and therefore stored in the DB.

Note that the list of parameters below is not exhaustive. The property keys shown in the interface, at both global and project levels, can also be set as analysis parameters.

Mandatory Parameters

**Server**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.host.url</td>
<td>Server URL</td>
<td><a href="http://localhost:9000">http://localhost:9000</a></td>
</tr>
</tbody>
</table>

**Database**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.jdbc.url</td>
<td>JDBC Connection URL</td>
<td>jdbc:h2:tcp://localhost:9092/sonar</td>
</tr>
<tr>
<td>sonar.jdbc.username</td>
<td>User for the JDBC Connection</td>
<td>sonar</td>
</tr>
<tr>
<td>sonar.jdbc.password</td>
<td>Password for the JDBC Connection</td>
<td>sonar</td>
</tr>
</tbody>
</table>

**Project Configuration**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.projectKey</td>
<td>The project key that is unique for each project. Set through <code>&lt;groupId&gt;:&lt;artifactId&gt;</code> when using Maven.</td>
<td></td>
</tr>
<tr>
<td>sonar.projectName</td>
<td>Name of the project that will be displayed on the web interface. Set through <code>&lt;name&gt;</code> when using Maven.</td>
<td></td>
</tr>
<tr>
<td>sonar.projectVersion</td>
<td>The project version. Set through <code>&lt;version&gt;</code> when using Maven.</td>
<td></td>
</tr>
<tr>
<td>sonar.language</td>
<td>Set the language of the source code to analyze. Browse the Plugin Library page to get the list of all available languages. If not set, a multi-language analysis will be triggered.</td>
<td></td>
</tr>
<tr>
<td>sonar.sources</td>
<td>Comma-separated paths to directories containing source files. Compatible with Maven since SonarQube 4.2. If not set, the source code is retrieved from the default Maven source code location.</td>
<td></td>
</tr>
</tbody>
</table>
## Optional Parameters

### Web Services

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.ws.timeout</td>
<td>Maximum time to wait for the response of a Web Service call (in seconds)</td>
<td>60</td>
</tr>
</tbody>
</table>

### Project Configuration

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.projectDescription</td>
<td>The project description. Not compatible with Maven, which uses the <code>&lt;description&gt;</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>sonar.binaries</td>
<td>Comma-separated paths to directories containing the binary files (directories with class files, in the case of Java). Not compatible with Maven, which retrieves binaries from the default location for Java Maven projects.</td>
<td></td>
</tr>
<tr>
<td>sonar.tests</td>
<td>Comma-separated paths to directories containing tests. Not compatible with Maven, which retrieves test from the default location for Java Maven projects.</td>
<td></td>
</tr>
<tr>
<td>sonar.libraries</td>
<td>Comma-separated paths to files with third-party libraries (JAR files in the case of Java). Patterns can be used. Example: sonar.libraries=path/to/specific/library/myLibrary.jar,path/to/library/*.jar Note that the * wildcard character is not supported for directories, only for files. This property is used by rule engines during issues detection (mainly the SonarQube and FindBugs engines, which both rely on bytecode). Having the bytecode of these libraries allows the rules engines to get more information on coupling, possible null parameters when calling external APIs, etc., thus getting more accuracy during issue detection.</td>
<td></td>
</tr>
<tr>
<td>sonar.analisis.mode</td>
<td>Set the analysis mode. See Concepts. Possible values: analysis preview incremental</td>
<td>analysis</td>
</tr>
<tr>
<td>sonar.previeuw.readTimeout</td>
<td>This property is only relevant in the context of preview analysis. In a preview analysis certain information about the project is downloaded from the server into a local database. This property is the timeout value in seconds for the reading of that data. Typically the default value is fine, but it may need adjusting in very large or busy environments.</td>
<td>60</td>
</tr>
<tr>
<td>sonar.sourceEncoding</td>
<td>Set the source file encoding. Encoding of source files. Example of values: UTF-8, MacRoman, Shift_JIS. This property can be replaced by the standard property <code>project.build.sourceEncoding</code> in Maven projects. The list of available encodings depends on your JVM. See <a href="http://docs.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html">http://docs.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html</a>.</td>
<td>System encoding</td>
</tr>
<tr>
<td>sonar.importSources</td>
<td>Allow or suppress the import of the text of source files into SonarQube. For security or other reasons there are times when project sources must not be stored and displayed. Set this value to false to prevent the text of a project's source files from being available via the SonarQube interface to anyone at all. Issue tracking mechanism will not properly work if sources are not imported into SonarQube. For example if you modify a file where some issues were previously flagged as false-positive, those issues will come back again as new issues. For this reason, this property has been deprecated in SonarQube 4.5.1 and the related feature will be removed in SonarQube 5.0.</td>
<td>true</td>
</tr>
</tbody>
</table>
### Assign a date to the analysis.

**Note:** This parameter is applicable to a few, special use cases, rather than being an “every day” parameter:

- When analyzing a new project, you may want to retroactively create some history for the project in order to get some information on quality trends over the last few versions.
- When moving from one database engine to another, it is highly recommended (even mandatory) to start from a fresh new database schema. In doing so, you will lose the entire history for all your projects. Which is why you may want to feed the new SonarQube database with some historical data.

To answer those use cases, you can use the `sonar.projectDate` property. The format is `yyyy-MM-dd`, for example: `2010-12-01`.

The process is the following:

- Retrieve a the oldest version of your application’s source that you wish to populate into the history (from a specific tag, whatever).
- Run a SonarQube analysis on this project by setting the `sonar.projectDate` property. Example: `sonar-runner -Dsonar.projectDate=2010-12-01`
- Retrieve the next version of the source code of your application, update the `sonar.projectDate` property, and run another analysis. And so on for all the versions of your application you’re interested in.

**Note:** You must analyze your versions in chronological order, oldest first.

### Manage SCM branches.

Two branches of the same project are considered to be different projects in SonarQube. As a consequence, issues found in a project A in a branch B1 are not linked to issues found for this project A in a branch B2. Currently, there is no way to resolve automatically issues of B2 when they are resolved in B1 as again A-B1 & A-B2 are considered as separated project.

If you are a user of Developer Cockpit, please see “Limitation” section in the Developer Cockpit Installation and Usage.

### Override the profile that would normally be used to analyze a project.

Through the web interface, you can define as many quality profiles as you want, and you can easily associate one of these quality profiles to a given project though the web interface.

**Note that this property is deprecated** (see [Unable to locate Jira server for this macro. It may be due to Application Link configuration.](http://example.com)) and will be removed in a near future.

### Skip the computation of design metrics and dependencies.

Currently only available for Java.

### Use this property when the files you need analysis to take place in a directory other than the one from which it starts. E.G. analysis begins from `jenkins/jobs/myjob/workspace` but the files to be analyzed are in `/ftpdrop/cobol/project1`. The path may be relative or absolute.

Specify not the the source directory, but some parent of the source directory. The value specified here becomes the new "analysis directory", and other paths are then specified as though the analysis were starting from the new `sonar.projectBaseDir`.

**Note that the analysis process will need write permissions in this directory; it is where the `sonar.working.directory` will be created.**

### Set the working directory for an analysis triggered with the SonarQube Runner or the SonarQube Ant Task (versions greater than 2.0).

Path must be relative and unique for each project.

Beware: the specified folder is deleted before each analysis.

### Exclusions / Inclusions

**See [Narrowing the Focus](http://example.com) to:**

- Exclude files from analysis
- Prevent some files from being checked for duplications
- Prevent some files from being taken into account for code coverage by unit tests and integration tests

---

### Table of SonarQube Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.projectDate</td>
<td>Assign a date to the analysis.</td>
</tr>
<tr>
<td>sonar.branch</td>
<td>Manage SCM branches.</td>
</tr>
<tr>
<td>sonar.profile</td>
<td>Override the profile that would normally be used to analyze a project.</td>
</tr>
<tr>
<td>sonar.skipDesign</td>
<td>Skip the computation of design metrics and dependencies.</td>
</tr>
<tr>
<td>sonar.projectBaseDir</td>
<td>Use this property when the files you need analysis to take place in a directory other than the one from which it starts. E.G. analysis begins from <code>jenkins/jobs/myjob/workspace</code> but the files to be analyzed are in <code>/ftpdrop/cobol/project1</code>. The path may be relative or absolute. Specify not the the source directory, but some parent of the source directory. The value specified here becomes the new &quot;analysis directory&quot;, and other paths are then specified as though the analysis were starting from the new <code>sonar.projectBaseDir</code>. <strong>Note that the analysis process will need write permissions in this directory; it is where the <code>sonar.working.directory</code> will be created.</strong></td>
</tr>
<tr>
<td>sonar.working.directory</td>
<td>Set the working directory for an analysis triggered with the SonarQube Runner or the SonarQube Ant Task (versions greater than 2.0). Path must be relative and unique for each project. Beware: the specified folder is deleted before each analysis.</td>
</tr>
</tbody>
</table>

---

**Disclaimer:**

- The information provided is a representation of the original content and is not intended to replace the official documentation.
- The table format ensures clarity and easy reference for the properties.
- For detailed usage and implementation, please refer to the official SonarQube documentation.
- Ignore issues on certain components and against certain coding rules

## Analyzer's Log

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonar.showProfiling</td>
<td>Display logs to see where the analyzer spends time.</td>
<td>false</td>
</tr>
<tr>
<td>sonar.log.profilingLevel</td>
<td>Set it to <strong>FULL</strong>, to display all the SQL queries executed by the analyzer.</td>
<td>NONE</td>
</tr>
<tr>
<td>sonar.verbose</td>
<td>Activate DEBUG mode for the analyzer.</td>
<td>false</td>
</tr>
</tbody>
</table>