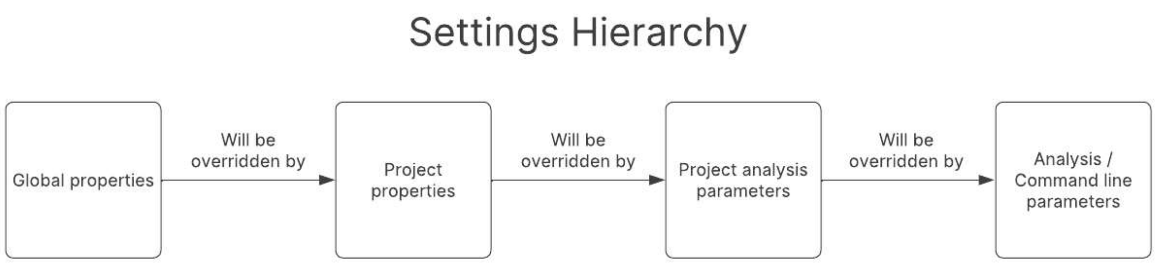
Sonar Project Properties

Analysis parameters (36)

We can configure project analysis settings at multiple places in the SonarQube environment. Each plugin and language analyzer adds properties to the SonarQube UI; we can define those properties as analysis parameters.

The best place to set the descriptions of those properties is in the UI when possible. Structurally, only parameters set through the UI are reusable for subsequent analysis in a way the parameters apply by the scanner- the **SonarScanner**:

> Here is the hierarchy in order of precedence:
>
> 
>

- *Global / Admin properties*: Apply to all projects. Defined in the UI in **Administration > Configuration > General Settings**
- *Project properties*: Apply to one project only. At project level, defined in the UI in **Project Settings > General Settings**
- *Project analysis parameters*: Defined in a project analysis configuration file or scanner configuration file
- *Analysis / Command line parameters*: Defined when launching an analysis with -D on the command line

> Notice only the stored parameters set through the UI are in the database. For example, if we override `the sonar.exclusions` parameter via the command line for a specific project; we do not store the settings in the database.

> Subsequent analyses, or analyses in **SonarLint** with connected mode, would still be executed with the exclusions defined in the UI. Thus, we can store the settings in the DB.

> The interface shows the property keys at both global and project levels. We can also set them as analysis parameters, but set the parameters listed below can only be at analysis time.

> **Code Coverage Reporting (CCR) and Code Inspection Analysis Reporting (CIA)**: For language-specific parameters related to test code coverage and execution, see test coverage. For language-specific parameters related to external issue reports, see external issues. And to learn more about controlling our analysis, see the page on Narrowing the focus.

MANDATORY PARAMETERS (2)

Server (1)

Key	Description	Default
`sonar.host.url`	The server URL	http://localhost:9000

Project configuration (1)

Key	Description	Default
`sonar.projectKey`	The project's unique key. Allowed characters are: letters, numbers, -, _, . and :, with at least one non-digit.	

OPTIONAL PARAMETERS (34)

Project Identity (2)

Key	Description	Default
`sonar.projectName`	Name of the project that will be displayed on the web interface	If not provided and there is already a name in the DB, it won't be overwritten.
`sonar.projectVersion`	The project version	Do not use build number as sonar.projectVersion

Authentication (1)

> By default, user authentication requires preventing anonymous users from browsing and analyzing projects on our instance, and they need to authenticate when running analyses.

> When authenticating, the "Anyone" pseudo-group doesn't have permission to perform analyses, and they need to supply the user's credentials with Executive Analysis permissions to run the analysis.

Key	Description	Default
`sonar.token`	The authentication token of a SonarQube user with either Execute Analysis permission on the project or Global Execute Analysis permission.	

Web services(1)

Key	Description	Default
`sonar.ws.timeout`	Maximum time to wait for the response of a Web Service call (in seconds). Modifying this value from the default is useful only when experiencing timeouts during analysis while waiting for the server to respond to Web Service calls.	60

Project configuration (23)

Key	Description	Default
`sonar.projectDescription`	The project description	
`sonar.links.homepage`	Project home page	
`sonar.links.ci`	Continuous Integration	
`sonar.links.issue`	Issue tracker	
`sonar.links.scm`	Project source repository	
`sonar.sources`	Comma-separated paths to directories containing main source files	
`sonar.tests`	Comma-separated paths to directories containing test source files	
`sonar.sourceEncoding`	Encoding of the source files. For example: `UTF-8`, `MacRoman`, `Shift_JIS`	System encoding
`sonar.externalIssuesReportPath`	Comma-delimited list of paths to Generic Issue reports	
`sonar.externalIssuesReportPath`	Comma-delimited list of paths to SARIF reports	
`sonar.projectDate`	Assign a date to the analysis. This parameter is only useful when we need to retroactively create the history of a not-analyzed-before project. The format is YYYY-MM-DD, for	Current date

Key	Description	Default
	example, 2010-12-01. Since we cannot perform an analysis dated prior to the most recent one in the database, we must analyze and recreate our project history in chronological order, the oldest first. **Note**: We may need to adjust our housekeeping settings if we wish to create a long-running history.	
`sonar.projectBaseDir`	Use this property when you need analysis to take place in a directory other than the one from which it was launched. For example, 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 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 specified value of sonar.projectBaseDir. Note that the analysis process will need write permissions in this directory; it is where the sonar.working.directory will be created.	
`sonar.working.directory`	Set the working directory for an analysis triggered with the SonarScanner. The path must be relative, and unique for each project. Beware: the specified folder is deleted before each analysis.	`.scannerwork`
`sonar.scm.provider`	This property can be used to explicitly tell SonarQube which SCM we're using on the project (in case auto-detection doesn't work). The value of this property is always lowercase and depends on the SCM (ex. "git" if we're using Git). Check the SCM integration documentation for more.	

Key	Description	Default
`sonar.scm.forceReloadAll`	By default, blame information is only retrieved for changed files. Set this property to true to load blame information for all files. This can be useful if we feel that some SCM data is outdated but SonarQube does not get the latest information from the SCM engine.	
`sonar.scm.exclusions.disabled`	For supported engines, files ignored by the SCM, i.e. files listed in `.gitignore`, will automatically be ignored by analysis too. Set this property to true to disable that feature. SCM exclusions are always disabled if `sonar.scm.disabled` is set to true.	
`sonar.scm.revision`	Overrides the revision, for instance, the Git SHA-1, displayed in analysis results. By default value is provided by the CI environment or guessed by the checked-out sources	
`sonar.buildString`	The string passed with this property will be stored with the analysis and available in the results of api/project_analyses/search, thus allowing us to later identify a specific analysis and obtain its ID for use with `api/project_analyses/set_baseline`.	
`sonar.analysis.[yourKey]`	This property stub allows us to insert custom key/value pairs into the analysis context, which will also be passed forward to webhooks	
`sonar.newCode.referenceBranch`	Sets the new code definition to **Reference Branch** for this analysis, overriding the configuration on the server. The New Code will be calculated based on the differences between the branch under analysis and the provided branch. This parameter is intended to be set in a configuration file (ex: `sonar-project.properties`), specific to a given branch.	

Key	Description	Default
`sonar.filesize.limit`	Sets the limit in MB for files to be discarded from the analysis scope if the size is greater than specified.	20

Duplications (1)

Key	Description	Default
`sonar.cpd.${language}.miminumTokens`	A piece of code is considered duplicated as soon as there are at least 100 duplicated tokens in a row (override with sonar.cpd.${language}.minimumTokens) spread across at least 10 lines of code (override with sonar.cpd.${language}.minimumLines).	100
`sonar.cpd.${language}.miminumLines`	(see above)	10

Analysis logging (4)

Key	Description	Default
`sonar.log.level`	Control the quantity/level of logs produced during an analysis. `DEBUG`: Display `INFO` logs + more details at `DEBUG` level. Similar to `sonar.verbose=true`. `TRACE`: Display `DEBUG` logs + the timings of all ElasticSearch queries and Web API calls executed by the **SonarScanner**.	INFO
`sonar.verbose`	Add more detail to both client and server-side analysis logs. Activates DEBUG mode for the scanner, and adds client-side environment variables and system properties to the server-side log of analysis report processing. **NOTE**: There is the potential for this setting to expose sensitive information such as passwords if they	false

Key	Description	Default
	are stored as server-side environment variables.	
`sonar.scanner.dumpToFile`	Outputs to the specified file the full list of properties passed to the scanner API as a means to debug analysis.	
`sonar.scanner.metaFilePath`	Set the location where the scanner writes the report-task.txt file containing among other things the ceTaskId.	value of `sonar.working.directory`

Quality Gate (2)

Key	Description	Default
`sonar.qualitygate.wait`	Forces the analysis step to poll the SonarQube instance and wait for the Quality Gate status. If there are no other options, we can use this to fail a pipeline build when the Quality Gate is failing. See the CI integration page for more information.	
`sonar.qualitygate.timeout`	Sets the number of seconds that the scanner should wait for a report to be processed.	300

Deprecated (1)

Key	Description	Default
`sonar.links.scm_dev`	Developer connection	**Deprecated since SQ 0.7.1**

Tip Edit the Markdown in `stories/SonarProjectProperties.stories.mdx`