# Quality-time

**ICTU**

**Jan 31, 2024**

# GETTING STARTED 🖿

*Quality-time* is an automated quality system for software development and maintenance. *Quality-time* collects measurement data from sources such as GitLab, SonarQube, Jira, Azure DevOps, and OWASP Dependency Check, to provide an overview of the quality of software products and projects. It does so by comparing measurement data with metric targets and informing development teams about the metrics that need improvement actions.

# ABOUT *QUALITY-TIME*

*Quality-time* is an automated quality system for software development and maintenance. *Quality-time* collects measurement data from sources such as GitLab, SonarQube, Jira, Azure DevOps, and OWASP Dependency Check, to provide an overview of the quality of software products, processes, and projects. It does so by comparing measurement data with metric targets and informing development teams about the metrics that need improvement actions.

Technically, *Quality-time* consists of a React frontend, a Mongo database server, and a number of backend components written in Python: a worker component to collect measurement data from the sources, a worker component to send notifications, an API-server for the frontend, and an API-server for the worker components.

Users can add and configure reports, metrics, and sources (such as SonarQube and Jira) in the frontend. The collector collects metrics data from the configured metric sources. It posts the measurements to the server which in turn stores them in the database. The frontend calls the server to get the reports and the measurements and presents them to the user.

*Quality-time* is developed by ICTU, an independent consultancy and project organisation within the Dutch government, helping government organizations develop and implement digital services.

# SCREENSHOTS

Some screenshots to wet your appetite.

---

**Tip:** *Quality-time* supports both a light and a dark UI mode. Toggle the UI mode of the documentation, using the icon at the top of the page, to also toggle the UI mode of the screenshots.
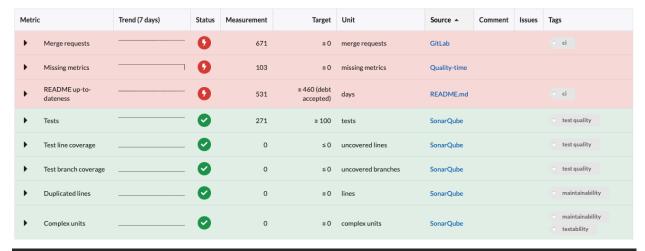
---

## 2.1 Projects dashboard

*Quality-time* shows a summary of the projects on its landing page:

## 2.2 Metrics overview

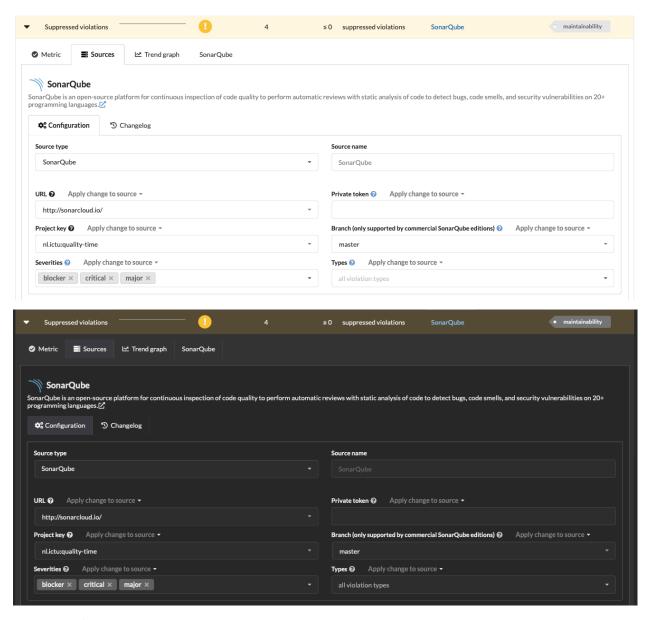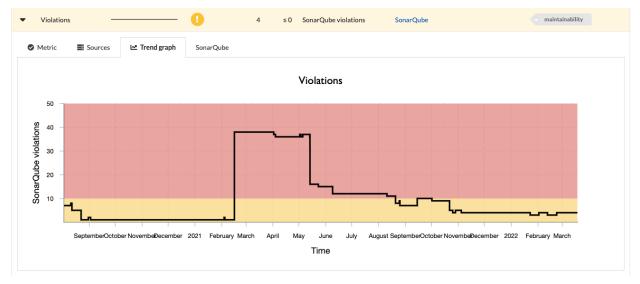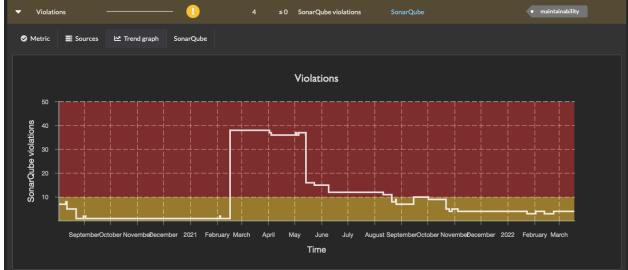For each metric, *Quality-time* displays the key data:



## 2.3 Metric details

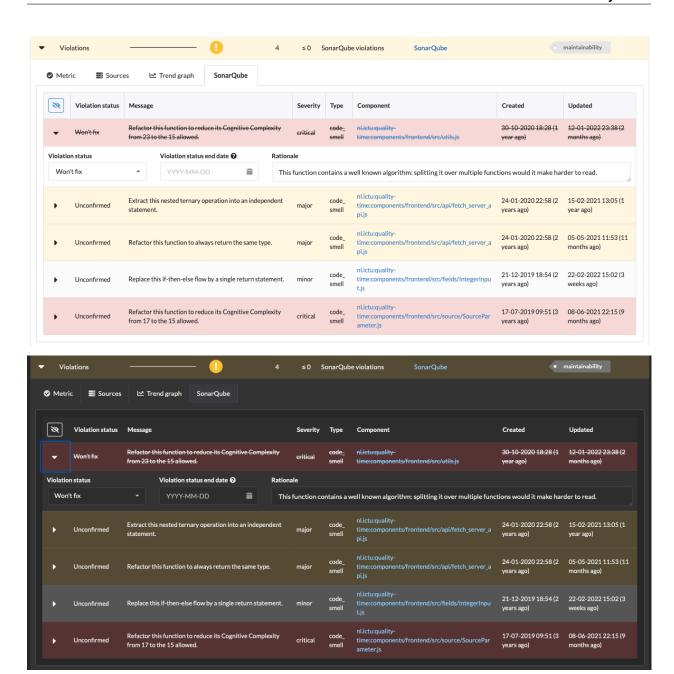Users can expand the metrics to see and configure the metric details:

Keep track of trends:

And manage false positives:

# FEATURES

Implemented features include:

- Robust data collection (the collector should never fail, even in the face of misconfigured or unavailable sources).

- Measurement history is kept in a database, allowing for time travel.

- Report configuration via the UI.

- Multiple reports in one *Quality-time* instance.

- LDAP-integration.

- Generic false-positive management.

- Metric tags can be used to summarize metrics with the same tag across subjects, e.g. to summarize all security metrics.

- Export of reports to PDF, both via the UI as well as via the API.

- Notifications of events, such as metrics turning red, to Microsoft Teams.

- Side-by-side comparison of measurements at different points in time.

- Integration with issue tracker (Jira only at the moment) to manage actions and technical debt.

- Dark and light UI mode.

**See also:**

For more plans, see the issue tracker.

# TRYING IT OUT

*Quality-time* requires Docker and Docker-compose.

Clone this repository:

```
git clone https://github.com/ICTU/quality-time.git
```

Build the containers:

```
docker-compose build
```

Start the containers:

```
docker-compose up
```

*Quality-time* is served at http://localhost. Use username `jadoe` and password `secret` to log in.

# FIVE

# WHY *QUALITY-TIME*?

ICTU developed *Quality-time* to help projects and teams within ICTU gain actionable insight into the quality of the software they are developing and maintaining. The current vision on the scope of *Quality-time* is:

- *Quality-time* collects data from sources that projects and teams are using anyway to develop and maintain software. Think version control systems, build tools, backlog management tools, test reports, security tools, and other quality tools. *Quality-time* does not analyse the software itself, but relies on information from these tools to create an integrated and actionable overview.

- The sources that *Quality-time* uses to collect its measurement data are automated. Manual sources for measurements are possible, but automated sources are preferred.

- *Quality-time* evaluates measurement data against target values. It shows users which metrics do not meet their target value, so users can take corrective action. *Informational* metrics, meaning metrics without target, are possible, but most metrics are assumed to have targets.

- *Quality-time* provides a single overview of the software measurements, but refers users to the underlying tools for detailed information.

- Technical debt management is an integral part of software quality management. Any metric that does not meet its target value for a longer period of time can be considered to represent a form of debt that needs to be managed.

- Tools come and go. Hence, an important non-functional requirement for *Quality-time* is the ability to quickly add interfacing to new sources.

- *Quality-time* provides actionable information on current quality issues and risks. Historical information is retained, but is a second-class citizen.

- *Quality-time* is not an issue tracker or task manager. It does integrate with issue trackers, however, making it easy to create issues for metrics that need action and to manage technical debt.

- Software quality is usually broken down into quality characteristics such as maintainability, testability, and accessibility, as for example in ISO 25010. Many metrics in *Quality-time* measure parts of these quality characteristics. For example, the number of 'code smells' and the 'complexity' of source code are both related to maintainability. However, *Quality-time* does not attempt to aggregate these metrics into a measurement of quality characteristics. Such aggregations typically lack a sound mathematical and scientific basis and are often not actionable.

**Note:**   The name *Quality-time* is of course a call to action: spend more quality time with the software you develop ⍰

# USER MANUAL

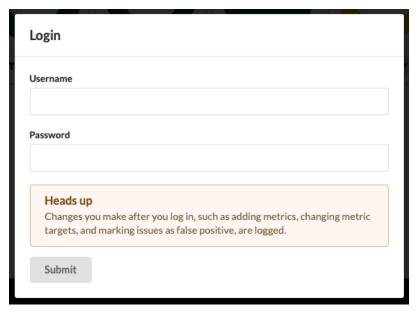**Note:** This user manual assumes *Quality-time* has been installed, is up and running, and that you have opened *Quality-time* in your browser. See the *Deployment instructions* on how to deploy *Quality-time*.
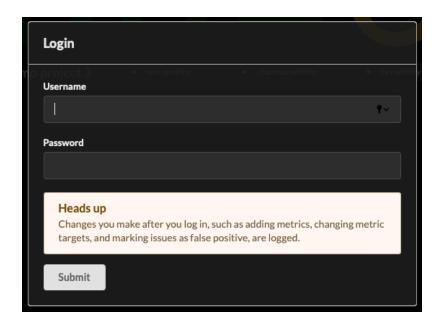
## 6.1 Logging in and out

You can view Quality reports without logging in, but to edit reports and metrics you need to log in. Click the "Login" button in the menu bar:



Enter your LDAP-credentials in the dialog:

---

**Note:** You can either use your canonical LDAP name as username or your LDAP user id. Please contact your system administrator if you don't know your LDAP credentials.

---

After hitting "Submit" you should be logged in. The menu bar shows your username. If you have a Gravatar, it will be displayed next to your username.



Clicking "Logout" logs you out from *Quality-time*. Your user session expires after 24 hours and you need to log in again to be able to edit reports.

## 6.2 Configuring permissions

*Quality-time* implements a limited permissions system. Anybody (authenticated and not authenticated) can always view all the information in *Quality-time*. By default, anybody who is logged in can edit reports, subjects, metrics, sources and measured entities. However, this access can be restricted to certain users. On the homepage, expand the reports overview title to see two input fields to grant users report editing rights or entity editing rights.

### 6.2.1 Report edit permission

Report edit permission allows a user to edit the reports in this *Quality-time* instance. That means to edit the add, edit, and delete reports, subjects, metrics, and sources.

If you forget to add yourself, your username will be added automatically. This means that you can't retract your own report editing rights: add another user and ask them to remove your username or email address.

To restore the default situation where every logged-in user can edit reports, subjects, metrics, and sources, remove all usernames and email addresses.

---

## 6.2.2 Entity edit permission

Entity edit permission will allow a user to update the status of measured entities. A user with this permission can for example mark violations as false positives.

Unlike for the report edit permission, it's possible to retract yourself from the list of entity editors.

To restore the default situation where every logged-in user can edit entities, remove all usernames and email addresses.

# 6.3 Configuring quality reports

Each *Quality-time* instance can serve multiple quality reports. A quality report consists of one or more subjects - things such as software products, projects, and processes - that you want to measure the quality of. Each subject has one or more metrics that tell you something about the quality of the subject. For example, the number of failing test cases of an application, or the number of ready user story points for a Scrum team. To collect the measurement data, each metric has one or more sources that *Quality-time* will use to measure the metric.

> **Attention:** You need to be logged in to be able to edit quality reports.

## 6.3.1 Configuring reports

### Adding reports

To add a new report, be sure to be logged in and click the "Add report" button on the home page. This will create a new empty report. Click the report card in the dashboard to navigate to it.

### Editing reports

To change the title or subtitle of a report, expand the report header and enter a new title and/or subtitle in their respective fields. For the desired reaction times, see the *Customizing quality reports* section below. For the issue tracker, see the *Issue tracker* section below. For notifications, see the *Notifications* section below.

The "Comment" field can be used to describe the report, or any other information. HTML and URLs are supported. The entered comments are shown between the report title and the dashboard.

▼ **New report**

| ⚙ Configuration | 🕐 Desired reaction times | 🔔 Notifications | ☰ Issue tracker | 🕘 Changelog | ↗ Share |

**Report title**

New report

**Report subtitle**

**Comment**

Enter comments here (HTML allowed; URL's are transformed into links)

🗋 Download report as pdf                    🗑 Delete report

## Deleting reports

To delete a report expand the report header and click the "Delete report" button. The report and all its subjects is deleted.

> **Danger:** Be careful, there's no way to undo your action via the user interface.

## 6.3.2 Configuring subjects

### Adding subjects



Each quality report consists of "subjects". Subjects are the things being measured by *Quality-time*. A subject can for example be a software product or component, a software process, or a continuous integration pipeline.

**See also:**

See the *reference manual* for the list of supported subject types.

To add a new subject, be sure you are logged in and are on a report page. Click the "Add subject" button to select a subject type and add a new subject. The subject is added to the report dashboard.

Alternatively, you can also copy an existing subject or move an existing subject to the report. Clicking the "Copy subject" or "Move subject" button shows a drop down menu with all of the subjects to choose from. Copying or moving a subject also copies or moves the metrics and sources of the subject.

### Editing subjects

To change the subject type and name expand the subject header if it's not already expanded. The subject type can be changed by means of the "Subject type" dropdown.

▼    **Software application**

**Software**
A custom software application or component.

| ⚙ Configuration | ⏱ Changelog | ↪ Share |

**Subject type**

| Software ▾ |

**Subject title**

| Software application |

**Subject subtitle**

| |

**Comment**

| Enter comments here (HTML allowed; URL's are transformed into links) |

| ≫ | ⌃ | ⌄ | ≫ |    🗑 Delete subject

▼    **Software application**

**Software**
A custom software application or component.

| ⚙ Configuration | ⏱ Changelog | ↪ Share |

**Subject type**

| Software ▾ |

**Subject title**

| Software application |

**Subject subtitle**

| | |

**Comment**

| Enter comments here (HTML allowed; URL's are transformed into links) |

| ≫ | ⌃ | ⌄ | ≫ |    🗑 Delete subject

---

**Note:**   Currently, changing the type of the subject does not affect what you can do with the subject.

---

To change the title or subtitle of the subject, enter a new title and/or subtitle in their respective fields.

The "Comment" field can be used to describe the subject, or any other information. HTML and URLs are supported. The entered comments are shown between the subject title and the table with metrics.

### Deleting subjects

To delete a subject expand the subject header and click the "Delete subject" button. The subject and all its metrics is deleted.

> **Danger:** Be careful, there's no way to undo your action via the user interface.

### Reordering subjects

To reorder subjects, expand the subject title and use the buttons on the lower left-hand side to move the subject up or down, or to the first or last position in the report. The order is saved on the server.

## 6.3.3 Configuring metrics

### Adding metrics

▶ **Software application**

| Metric | Trend (7 days) | Status | Measurement | Target | Unit | Source | Comment | Issues | Tags |
|---|---|---|---|---|---|---|---|---|---|

➕ Add metric ▾   📋 Copy metric ▾   ⤬ Move metric ▾

▶ **Software application**

| Metric | Trend (7 days) | Status | Measurement | Target | Unit | Source | Comment | Issues | Tags |
|---|---|---|---|---|---|---|---|---|---|

➕ Add metric ▾   📋 Copy metric ▾   ⤬ Move metric ▾

To add a metric to a subject, hit the "Add metric" button to select a metric type and create a new metric. Only metric types that can measure the subject are listed.

**See also:**

See the *reference manual* for the list of supported metric types.

*Quality-time* adds the selected metric to the report that you can next configure. It is immediately displayed in the metric table (and in the report dashboard) as white and with a question mark because *Quality-time* has no data on this metric yet.
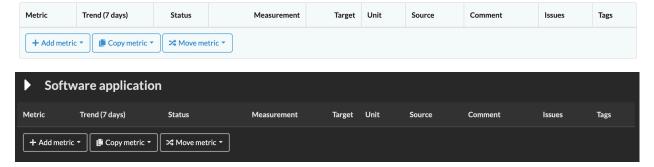
Alternatively, you can also copy an existing metric or move an existing metric to the subject. Clicking the "Copy metric" or "Move metric" button shows a drop down menu with all of the metrics to choose from. Copying or moving a metric also copies or moves the sources of the metric.

## Editing metrics

After you've added a metric, the metric is visible in the subject's metric table. You can change the metric configuration in the metric tab.





The first parameter is the "Metric type". The metric type determines what gets measured. By default, the name of the metric is equal to its type, "Accessibility violations" in the example above, but you can change the metric name using the "Metric name" field. When you change the metric type, the sources you can select in the "Sources" tab change accordingly.

> **Warning:** If you change the type of a metric that has sources configured, sources that do not support the new metric type will be removed.

Metrics can have zero or more arbitrary "Tags". Most metric have a default tag, but you can remove it and/or add more if you like. For each tag, the report dashboard at the top of the page shows a summary of the metrics with that tag:



The "Metric scale" field determines what scale to use to measure the metric. Most metrics support either the "Count" scale, the "Percentage" scale, or both. In the example of the duplicated lines metric above, setting the metric scale to "Percentage" means that the percentage of lines that are duplicated is shown instead of the count of duplicated lines.

The "Metric direction" determines whether lower measurement values are considered to be better or worse. Usually, the default direction is correct. An example of a metric where you might want to change the direction is the "tests" metric. When used to measure the number of tests, more tests is better. But, when used to measure the number of failing tests, fewer is better.

The "Metric unit" derives its default value from the metric type. Override as needed.

The "Metric target" determines at what value a measurement is below or above target. In the example below only measurement values of 0 are on target. The "Metric near target" determines when the measurement value is sufficiently close to the target to no longer require immediate action. Metrics near their target are yellow.

If you don't want to evaluate the metric against targets, but only want to track its measurement value, you can set the "Evaluate metric targets?" field to "No". The metric status will always be "Informative", unless the source data is missing.

### Managing technical debt

If a metric doesn't meet the target value, but your team isn't able to fix the situation in the short run, you can accept the deviation as *technical debt*.

To accept technical debt, navigate to the "Technical debt" tab of the metric and set the "Accept technical debt?" field to "Yes". Enter the value you're accepting for the time being in the "Metric debt target" field. If you want to pay off the debt before a certain date, this can be registered in the "Technical debt end date" field.

The "Issue identifiers" field can be used to enter the identifier of one or more issues in an issue tracker system. This can be used to track progress on resolving technical debt. See the *Issue tracker* section below on how to configure the issue tracker.

The "Create new issue" button can be used to create a new issue in the configured issue tracker. *Quality-time* will use the

issue tracker's API to create a new issue and will add the new issue's id to the tracked issue identifiers. The created issue is opened in a new browser tab for further editing. You may have to allow *Quality-time* to open popup windows in your browser.

---

**Note:** Metrics with accepted technical debt are displayed with a money icon and grey background as long as their measurement value is worse than their target and equal to or better than their technical debt target.

However, measurement values of a metric with accepted technical debt will *not* be evaluated against the technical debt target when:

- the metric has a technical debt end date that is in the past, or

- the metric has issues associated with it, and the issue tracker reports that all these issues have been resolved.

If any of these situations apply, the technical debt target is ignored and the measurement value is evaluated against the target values. Depending on the evaluation, the metric is shown as green, yellow, or red, as usual.

Also, when the technical debt target is ignored, the target value is shown with a grey background in the target column and has a popup explaining *why* the accepted technical debt target is being ignored.

---

The "Comment" field can be used to capture the rationale for accepting technical debt, or any other information. HTML and URLs are supported.

### Reordering metrics

To reorder metrics, expand the metric in the metric table and use the buttons on the lower left-hand side to move the metric one row higher or lower, or to the top or bottom of the table. The order is saved on the server. You can temporarily override the default ordering of the metrics by clicking a column headers to sort the metrics by the values in that column.
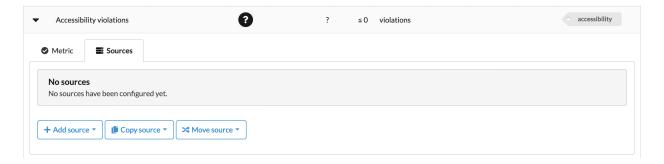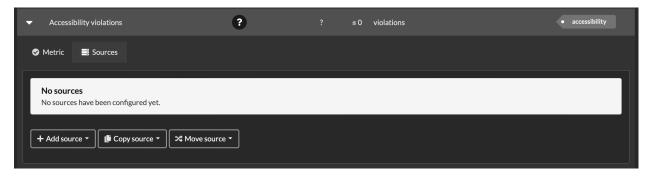
### Deleting metrics

To delete a metric, expand the metric in the metric table and click the "Delete metric" button. The metric and its sources are deleted.

---

**Danger:** Be careful, there's no way to undo your action via the user interface.

---

## 6.3.4 Configuring sources

### Adding sources

To add a source to a metric, expand the metric in the metric table and then click the "Sources" tab. In the "Sources" tab, click the "Add source" button and select a source type. Only sources that can support the metric type are listed.

**See also:**

See the *reference manual* for the list of supported source types.

Alternatively, you can also copy an existing source or move an existing source to the metric. Clicking the "Copy source" or "Move source" button shows a drop down menu with all of the sources to choose from.

If you add multiple sources for one metric the measurement values of each source are combined to get one measurement value for the metric. Usually this means adding up the values, but for some metrics this doesn't make sense and the minimum or maximum value of the sources is used as the metric value.

## Editing sources

After you've added a source, you can change the source type using the "Source type" drop-down menu. The available source types depend on the metric type. E.g. OWASP Dependency Check supports the security warnings metric type, but GitLab does not so GitLab is not shown.

By default, the name of the source equals the source type but this can be overridden using the "Source name" field.

The parameters that sources need differ per source type. Most sources need a URL, and optionally take either a username and password or a token so that *Quality-time* can access the source. If a parameter is required, this is indicated with a red outline as shown below.



---

Source parameter (URLs, usernames, passwords, etc.) changes can be applied to different scopes: to just the source being edited or to multiple sources that have the same type and value as the one being edited. When applying the change to multiple sources, the user can change all sources that have the same type and value of a metric, of a subject, of a report, or of all reports.

### Deleting sources

To delete the source of a metric, first expand the metric in the metric table. Then, select the "Sources" tab and click the "Delete source" button. The source is deleted and no longer used to measure the metric.

---

**Danger:** Be careful, there's no way to undo your action via the user interface.

---

### Reordering sources

To reorder sources, expand the metric in the metric table and click the sources tab. Use the buttons on the lower left-hand side of each source to move the source up or down, or to the top or bottom of the list of sources. The order is saved on the server.

## 6.3.5 Configuring entities

An entity is a measured entity like for example one single failed job in GitLab for a metric that measures failed GitLab jobs or a single violation in SonarQube for a metric that measures violations. What exactly an entity is, and what properties it has depends on what the metric in question is measuring. Not every metric will have entities.

To add a source to a metric, expand the metric in the metric table and then click the tab with the source name. It will show a list of entities with all its details.

When clicking on one of the entities, it can be expanded and edited. Options are for example mark an entity as false positive or as fixed. Entities marked as fixed, false positive, or won't fix are crossed out and subtracted from the measurement

---

value. For example, if a source reports 14 security warnings and two are marked as false positive, the measurement value will be 12.

When setting the status of an entity, the end date of that status is also set. By default, the status end date is set to 180 days after the current date for confirmed, false positive, and won't fix. The status end date for fixed is 7 days as 'fixed' means the entity should disappear shortly. These defaults can be customized, see the *Desired reaction times* section below.

## 6.4 Customizing quality reports

You can customize quality reports by changing the dashboard layout, by changing the desired metric reaction times, by filtering metrics, and by filtering metric entities.

---

**Note:** Settings that you change via the 'Settings' panel are not shared with other users.

---

### 6.4.1 Customizing dashboards

Both the reports dashboard on the *Quality-time* landing page and the dashboard of individual projects can be customized by dragging and dropping the cards.

The dashboard layout is persisted in the database and thus shared with other users.

### 6.4.2 Desired reaction times

The default desired response times for metrics and measurement entities (violations, issues, security warnings, etc.) can be changed via the report's title. Expand the report title and navigate to the 'Desired reaction times' tab.

#### Desired metric response times

Each of the metric states that require action - target not met (red), near target (yellow), accepted technical debt (grey), and status unknown (white) - has a desired reaction time in days that can be changed.

For each metric that requires action, *Quality-time* shows the time left to respond in the time left column. When the deadline is missed, the time left column shows '0 days' with a red background. Hovering over the time left value shows the deadline.

When showing multiple dates in the metric table (this can be done via the Settings panel), *Quality-time* shows an 'overrun' column with the number of days that the metric deadline was missed in the displayed period. The purpose of this information is to gain an understanding of how well the team is responding to metrics that require action.

#### Desired time after which to review measurement entities

Measurement entities (violations, issues, security warnings, etc.) can have one of five possible status: unconfirmed, confirmed, fixed, false positive, and won't fix. When setting the state of a measurement entity to a state other than unconfirmed, the end date of the status is also set. The reason is to encourage the status of measurement entities to be periodically reviewed. The default values can be changed.

### 6.4.3 Measurement trend

By default, subjects show the current measurement value of each metric, together with other details such as the target value, comments and tags. Subjects can also list multiple recent measurement values of each metric to show the measurement trend. Use the 'Settings' panel in the menu bar to increase the number of dates displayed. The 'Settings' panel can also be used to configure the number of days or weeks between dates.

### 6.4.4 Sorting metrics

Metrics can be sorted by clicking on the table column headers. The sort order cycles between the default order, sorted ascending by the column click, and sorted descending by the column clicked. The sort order can also be changed via the 'Settings' panel in the menu bar.

### 6.4.5 Filtering metrics by tag

In a report's dashboard, click on a tag card to show only metrics that have the selected tag. Click the selected tag again to turn off the filtering. To filter on multiple tags, use the 'Settings' panel.

### 6.4.6 Filtering metrics by status

The 'Settings' panel in the menu bar can be used to hide metrics that need no action.

### 6.4.7 Hiding columns

The 'Settings' panel in the menu bar can be used to hide specific columns from the metric tables.

## 6.5 Export reports as PDF

*Quality-time* reports can be downloaded as PDF. To create PDFs, *Quality-time* has a rendering service included to convert the HTML report into PDF.

As *Quality-time* has to open the report in a (headless) browser and load all the metrics, creating the PDF can take some time. Especially for big reports.

---

**Tip:** The report title in the footer of the PDF will link to the online version of the same report.

---

To manually download a PDF version of a report, navigate to the report and expand the report's title. Click the "Download report as PDF" button to create and download the PDF report.

The exported PDF report has the same metric table rows and columns hidden as in the user interface, and has the same metrics expanded as in the user interface. The exported PDF report also has the same date as the report visible in the user interface.

---

**Tip:** It is also possible to download a PDF version of the reports overview. Navigate to the reports overview and expand the title of the reports overview. Click the "Download overview as PDF" button to create and download the PDF report.

---

**See also:**

See the *API-documentation* for exporting reports via the API.

## 6.6 Export and import reports as JSON

*Quality-time* provides functionality for importing and exporting reports in JSON format. This functionality can be used for backing up reports or for transferring reports from one *Quality-time* instance to another one. Currently, this functionality is only available via the *API*, with one endpoint for importing and one for exporting the JSON reports.

## 6.7 Issue tracker

To track work being done on metrics, for example to resolve technical debt, it's possible to add (identifiers of) issues to metrics. *Quality-time* uses these issue identifiers to check the status of the issue with an issue tracker. For this to work, an issue tracker needs to be added to the report. Expand the report header and configure the issue tracker in the issue tracker tab. Currently, only Jira can be used as issue tracker. Please consider submitting a pull request if you need support for other issue trackers such as Azure DevOps Server or GitLab.

Multiple issues can be linked to one metric. At most one issue tracker can be configured per report.

## 6.8 DORA metrics

DORA metrics are a set of four key metrics for measuring the performance of software delivery, first described by the DevOps Research & Assessment (DORA) team in the 2016 State of DevOps report. See the DORA research program for more information.

*Quality-time* can monitor these metrics in the following manner:

- Deployment Frequency: measure "Job runs within time period" filtered on deployment jobs.

- Lead Time for Changes: measure "Average issue lead time" filtered on issues marked as change.

- Time to Restore Services: measure "Average issue lead time" filtered on issues marked as failure.

- Change Failure Rate: measure "Issues" filtered on issues marked as failure in production.

## 6.9 Notifications

*Quality-time* can send notifications about metrics that change status to Microsoft Teams channels. To enable notifications for a report, expand the report header and paste a Microsoft Teams webhook.

If a webhook has been configured, *Quality-time* will check for changes in the status of metrics every minute. As soon as one or more metrics in the report change status, a notification will be sent to the Microsoft Teams channel configured by the webhook.

# REFERENCE MANUAL

This is an overview of all *subjects* that *Quality-time* can measure, all *metrics* that *Quality-time* can use to measure subjects, and all *sources* that *Quality-time* can use to collect data from to measure the metrics. For each supported *combination of metric and source*, the parameters that can be used to configure the source are listed.

## 7.1 Subjects

This is an overview of all the subjects that *Quality-time* can measure. For each subject, the metrics that can be used to measure the subject are listed.

### 7.1.1 CI-environment

A continuous integration environment.

---

**Supporting metrics**

- *Failed CI-jobs*
- *Job runs within time period*
- *Merge requests*
- *Performancetest duration*
- *Software version*
- *Source up-to-dateness*
- *Source version*
- *Unmerged branches*
- *Unused CI-jobs*

---

## 7.1.2 Process

A software development and/or maintenance process.

---

**Supporting metrics**

- *Average issue lead time*
- *Change failure rate*
- *Issues*
- *Manual test duration*
- *Manual test execution*
- *Merge requests*
- *Source up-to-dateness*
- *Time remaining*
- *Unmerged branches*
- *User story points*
- *Velocity*
- *Sentiment*

---

## 7.1.3 Quality report

A software quality report.

---

**Supporting metrics**

- *Metrics*
- *Missing metrics*

---

## 7.1.4 Software

A custom software application or component.

---

**Supporting metrics**

- *Average issue lead time*
- *Change failure rate*
- *Commented out code*
- *Complex units*
- *Dependencies*
- *Duplicated lines*
- *Issues*

---

- *Job runs within time period*

- *Size (LOC)*

- *Long units*

- *Manual test duration*

- *Manual test execution*

- *Many parameters*

- *Merge requests*

- *Performancetest duration*

- *Performancetest stability*

- *Violation remediation effort*

- *Scalability*

- *Security warnings*

- *Slow transactions*

- *Software version*

- *Source up-to-dateness*

- *Source version*

- *Suppressed violations*

- *Test cases*

- *Tests*

- *Todo and fixme comments*

- *Test branch coverage*

- *Test line coverage*

- *Unmerged branches*

- *User story points*

- *Violations*

## 7.2 Metrics

This is an overview of all the metrics that *Quality-time* can use to measure subjects. For each metric, the default target, the supported scales, and the default tags are given. In addition, the sources that can be used to collect data from to measure the metric are listed.

### 7.2.1 Average issue lead time

The average lead time for changes completed in a certain time period.

*Why measure average issue lead time?* The shorter the lead time for changes, the sooner the new features can be used. Also, the shorter the lead time for changes, the fewer changes are in progress at the same time. Less context switching is needed and the risk of interfering changes is reduced.

**Default target**
> ≦ 10 days

**Scales**
> count

**Default tags**
> process efficiency

---

**Supported subjects**

- *Process*

- *Software*

---

**Supporting sources**

- *Azure DevOps Server*

- *Jira*

---

### 7.2.2 Change failure rate

The percentage of deployments causing a failure in production.

*Why measure change failure rate?* The change failure rate is an indicator of the DevOps effectiveness of a team.

*How to configure change failure rate?* Because this metric is a "rate", it needs a numerator (the number of failed deployments in the look back period) and a denominator (the number of deployments in the look back period). The metric can be configured with a combination of Jira, and either Jenkins or GitLab CI as source. Jira is used as source for the number of failures (numerator). Either Jenkins or GitLab CI is used as source for the number of deployments (denominator).

**Default target**
> ≦ 0% of the failed deployments

**Scales**
> percentage

**Default tags**
> process efficiency

---

**Supported subjects**

- *Process*

- *Software*

**Supporting sources**

- *Jenkins*

- *Jira*

- *GitLab*

### 7.2.3 Commented out code

The number of blocks of commented out lines of code.

*Why measure commented out code?* Code should not be commented out because it bloats the sources and may confuse the reader as to why the code is still there, making the source code harder to understand and maintain. Unused code should be deleted. It can be retrieved from the version control system if needed.

**See also:**

1. https://rules.sonarsource.com/python/RSPEC-125

2. https://kentcdodds.com/blog/please-dont-commit-commented-out-code

**Default target**
 ≤ 0 blocks

**Scales**
 count

**Default tags**
 maintainability

**Supported subjects**

- *Software*

**Supporting sources**

- *Manual number*

- *SonarQube*

### 7.2.4 Complex units

The number of units (classes, functions, methods, files) that are too complex.

*Why measure complex units?* Complex code makes software harder to test and harder to maintain. Complex code is harder to test because there are more execution paths that need to be tested. Complex code is harder to maintain because it is harder to understand and analyze.

**See also:**

1. https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainabil pdf

2. https://blog.codacy.com/an-in-depth-explanation-of-code-complexity/

3. https://blog.sonarsource.com/cognitive-complexity-because-testability-understandability

**Default target**
≦ 0 complex units

**Scales**
count (default)

percentage

**Default tags**
maintainability

testability

**Supported subjects**

- *Software*

**Supporting sources**

- *Manual number*

- *SonarQube*

### 7.2.5 Dependencies

The number of (outdated) dependencies.

*Why measure dependencies?* Dependencies that are out of date can be considered a form of technical debt. On the one hand, not upgrading a dependency postpones the work of testing the new version. And, if the new version of a dependency has backwards-incompatible changes, it also postpones making adaptations to cater for those changes. On the other hand, upgrading the dependency may fix bugs and vulnerabilities, and unlock new features. Measuring the number of outdated dependencies provides insight into the size of this backlog.

**Default target**
≦ 0 dependencies

**Scales**
count (default)

percentage

**Default tags**
maintainability

**Supported subjects**

- *Software*

**Supporting sources**

- *Composer*

- *Manual number*

- *npm*

- *OWASP Dependency-Check*

- *pip*

## 7.2.6 Duplicated lines

The number of lines that are duplicated.

*Why measure duplicated lines?* Duplicate code makes software larger and thus potentially harder to maintain. Also, if the duplicated code contains bugs, they need to be fixed in multiple locations.

**See also:**

1. https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainabil
   pdf

**Default target**
> ≦ 0 lines

**Scales**
> count (default)
>
> percentage

**Default tags**
> maintainability

**Supported subjects**

- *Software*

**Supporting sources**

- *Manual number*

- *SonarQube*

## 7.2.7 Failed CI-jobs

The number of continuous integration jobs or pipelines that have failed.

*Why measure failed CI-jobs?* Although it is to be expected that CI-jobs or pipelines will fail from time to time, a significant number of failed CI-jobs or pipelines over a longer period of time could be a sign that the continuous integration process is not functioning properly. Also, having many failed CI-jobs or pipelines makes it hard to see that additional jobs or pipelines start failing.

**Default target**
> ≦ 0 CI-jobs

**Scales**
> count

**Default tags**
ci

**Supported subjects**

- *CI-environment*

**Supporting sources**

- *Azure DevOps Server*

- *Jenkins*

- *GitLab*

- *Manual number*

### 7.2.8 Issues

The number of issues.

*Why measure issues?* What exactly issues are, depends on what is available in the source. The issues metric can for example be used to count the number of open bug reports, the number of ready user stories, or the number of overdue customer service requests. For sources that support a query language, the issues to be counted can be specified using the query language of the source.

**Default target**
≤ 0 issues

**Scales**
count

**Supported subjects**

- *Process*

- *Software*

**Supporting sources**

- *Azure DevOps Server*

- *Jira*

- *Manual number*

- *Trello*

## 7.2.9 Job runs within time period

The number of job runs within a specified time period.

*Why measure job runs within time period?* Frequent deployments are associated with smaller, less risky deployments and with lower lead times for new features.

**Default target**
≧ 30 CI-job runs

**Scales**
count

**Default tags**
ci

---

**Supported subjects**

- *CI-environment*

- *Software*

---

**Supporting sources**

- *Azure DevOps Server*

- *GitLab*

- *Jenkins*

- *Manual number*

---

## 7.2.10 Long units

The number of units (functions, methods, files) that are too long.

*Why measure long units?* Long units are deemed harder to maintain.

**See also:**

1. https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainabil
pdf

**Default target**
≦ 0 long units

**Scales**
count (default)

percentage

**Default tags**
maintainability

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Manual number*

- *SonarQube*

### 7.2.11 Manual test duration

The duration of the manual test in minutes.

*Why measure manual test duration?* Preferably, all regression tests are automated. When this is not feasible, it is good to know how much time it takes to execute the manual tests, since they need to be executed before every release.

**Default target**
≦ 0 minutes

**Scales**
count

**Default tags**
test quality

**Supported subjects**

- *Process*

- *Software*

**Supporting sources**

- *Jira*

- *Manual number*

### 7.2.12 Manual test execution

Measure the number of manual test cases that have not been tested on time.

*Why measure manual test execution?* Preferably, all regression tests are automated. When this is not feasible, it is good to know whether the manual regression tests have been executed recently.

**Default target**
≦ 0 manual test cases

**Scales**
count

**Default tags**
test quality

**Supported subjects**

- *Process*

- *Software*

---

**Supporting sources**

- *Jira*

- *Manual number*

---

## 7.2.13 Many parameters

The number of units (functions, methods, procedures) that have too many parameters.

*Why measure many parameters?* Units with many parameters are deemed harder to maintain.

**See also:**

1. https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainabil
   pdf

**Default target**
   ≦ 0 units with too many parameters

**Scales**
   count (default)

   percentage

**Default tags**
   maintainability

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Manual number*

- *SonarQube*

---

## 7.2.14 Merge requests

The number of merge requests.

*Why measure merge requests?* Merge requests need to be reviewed and approved. This metric allows for measuring the number of merge requests without the required approvals.

*How to configure merge requests?* In itself, the number of merge requests is not indicative of software quality. However, by setting the parameter "Minimum number of upvotes", the metric can report on merge requests that have fewer than the minimum number of upvotes. The parameter "Merge request state" can be used to exclude closed merge requests, for example. The parameter "Target branches to include" can be used to further limit the merge requests to only count merge requests that target specific branches, for example the "develop" branch.

---

**Default target**
≦ 0 merge requests

**Scales**
count (default)

percentage

**Default tags**
ci

**Supported subjects**

- *CI-environment*

- *Process*

- *Software*

**Supporting sources**

- *Azure DevOps Server*

- *GitLab*

- *Manual number*

## 7.2.15  Metrics

The number of metrics from one or more quality reports, with specific states and/or tags.

*Why measure metrics?* Use this metric to monitor other quality reports. For example, count the number of metrics that don't meet their target value, or count the number of metrics that have been marked as technical debt for more than two months.

*How to configure metrics?* After adding *Quality-time* as a source to a "Metrics"-metric, one can configure which statuses to count and which metrics to consider by filtering on report names or identifiers, on metric types, on source types, and on tags.

**Note:** If the "Metrics" metric is itself part of the set of metrics it counts, a peculiar situation may occur: when you've configured the "Metrics" to count red metrics and its target is not met, the metric itself will become red and thus be counted as well. For example, if the target is at most five red metrics, and the number of red metrics increases from five to six, the "Metrics" value will go from five to seven. You can prevent this by making sure the "Metrics" metric is not in the set of counted metrics, for example by putting it in a different report and only count metrics in the other report(s).

**Default target**
$\leqq$ 0 metrics

**Scales**
count (default)

percentage

**Supported subjects**

- *Quality report*

**Supporting sources**

- *Manual number*

- *Quality-time*

## 7.2.16 Missing metrics

The number of metrics that can be added to each report, but have not been added yet.

*Why measure missing metrics?* Provide an overview of metrics still to be added to the quality report. If metrics will not be added, a reason can be documented.

**Default target**
$\leqq$ 0 missing metrics

**Scales**
count (default)

percentage

**Supported subjects**

- *Quality report*

**Supporting sources**

- *Manual number*

- *Quality-time*

## 7.2.17 Performancetest duration

The duration of the performancetest in minutes.

*Why measure performancetest duration?* Performance tests, especially endurance tests, may need to run for a minimum duration to give relevant results.

**Default target**
$\geqq$ 30 minutes

**Scales**
count

**Default tags**
performance

---

**Supported subjects**

- *CI-environment*

- *Software*

---

**Supporting sources**

- *Gatling*

- *JMeter CSV*

- *Manual number*

- *Performancetest-runner*

---

## 7.2.18 Performancetest stability

The duration of the performancetest at which throughput or error count increases.

*Why measure performancetest stability?* When testing endurance, the throughput and error count should remain stable for the complete duration of the performancetest. If throughput or error count starts to increase during the performancetest, this may indicate memory leaks or other resource problems.

**Default target**
$\geqq$ 100% of the minutes

**Scales**
percentage

**Default tags**
performance

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Manual number*

- *Performancetest-runner*

---

### 7.2.19 Scalability

The number of virtual users (or percentage of the maximum number of virtual users) at which ramp-up of throughput breaks.

*Why measure scalability?* When stress testing, the load on the system-under-test has to increase sufficiently to detect the point at which the system breaks, as indicated by increasing throughput or error counts. If this breakpoint is not detected, the load has not been increased enough.

**Default target**
$\geqq$ 75 virtual users

**Scales**
count (default)

percentage

**Default tags**
performance

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Manual number*

- *Performancetest-runner*

---

### 7.2.20 Security warnings

The number of security warnings about the software.

*Why measure security warnings?* Monitor security warnings about the software, its source code, dependencies, or infrastructure so vulnerabilities can be fixed before they end up in production.

**Default target**
$\leqq$ 0 security warnings

**Scales**
count

**Default tags**
security

---

**Supported subjects**

- *Software*

---

**Supporting sources**

---

- *Anchore*

- *Anchore Jenkins plugin*

- *Bandit*

- *Cargo Audit*

- *Checkmarx CxSAST*

- *JSON file with security warnings*

- *Harbor*

- *Harbor JSON*

- *Manual number*

- *OpenVAS*

- *OWASP Dependency-Check*

- *OWASP ZAP*

- *Pyupio Safety*

- *SARIF*

- *Snyk*

- *SonarQube*

- *Trivy JSON*

### 7.2.21 Sentiment

How are the team members feeling?

*Why measure sentiment?* Satisfaction is how fulfilled developers feel with their work, team, tools, or culture; well-being is how healthy and happy they are, and how their work impacts it. Measuring satisfaction and well-being can be beneficial for understanding productivity and perhaps even for predicting it. For example, productivity and satisfaction are correlated, and it is possible that satisfaction could serve as a leading indicator for productivity; a decline in satisfaction and engagement could signal upcoming burnout and reduced productivity.

**See also:**

1. https://queue.acm.org/detail.cfm?id=3454124

**Default target**
$\geq 10$

**Scales**
count

**Supported subjects**

- *Process*

**Supporting sources**

- *Manual number*

## 7.2.22 Size (LOC)

The size of the software in lines of code.

*Why measure size (LOC)?* The size of software is correlated with the effort it takes to maintain it. Lines of code is one of the most widely used metrics to measure size of software.

**See also:**

1. https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainabil pdf

*How to configure size (LOC)?* To track the **absolute** size of the software, set the scale of the metric to 'count' and add one or more sources.

To track the **relative** size of sources, set the scale of the metric to 'percentage' and add cloc as source. Make sure to generate cloc JSON reports with the `--by-file` option so that filenames are included in the JSON reports. Use the 'files to include' field to select the sources that should be compared to the total amount of code. For example, to measure the relative amount of test code, use the regular expression `.*test.*`.

---

**Note:** Unfortunately, SonarQube can only used to measure the absolute size of the software. SonarQube does not report the sizes of files with test code for all programming languages. Hence it cannot be used to measure the relative size of test code.

---

**Default target**
   ≦ 30000 lines

**Scales**
   count (default)

   percentage

**Default tags**
   maintainability

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *cloc*

- *Manual number*

- *SonarQube*

### 7.2.23 Slow transactions

The number of transactions slower than their target response time.

*Why measure slow transactions?* Transations slower than their target response time indicate performance problems that need attention.

**Default target**
≦ 0 transactions

**Scales**
count

**Default tags**
performance

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Gatling*

- *Manual number*

- *JMeter CSV*

- *JMeter JSON*

- *Performancetest-runner*

---

### 7.2.24 Software version

The version number of the software as analyzed by the source.

*Why measure software version?* Monitor that the version of the software is at least a specific version or get notified when the software version becomes higher than a specific version.

*More information* Quality-time uses the packaging library (1) to parse version numbers. The packaging library expects version numbers to comply with PEP-440 (2). PEP is an abbrevation for Python Enhancement Proposal, but this PEP is primarily a standard for version numbers. PEP-440 encompasses most of the semantic versioning scheme (3) so version numbers that follow semantic versioning are usually parsed correctly.

**See also:**

1. https://pypi.org/project/packaging/

2. https://peps.python.org/pep-0440/

3. https://semver.org

**Default target**
≧ 1.0

**Scales**
version_number

**Default tags**
    ci

**Supported subjects**

- *CI-environment*

- *Software*

**Supporting sources**

- *Performancetest-runner*

- *SonarQube*

## 7.2.25  Source up-to-dateness

The number of days since the source was last updated.

*Why measure source up-to-dateness?* If the information provided by sources is outdated, so will be the metrics in Quality-time. Hence it is important to monitor that sources are up-to-date.

**Default target**
    ≦ 3 days

**Scales**
    count

**Default tags**
    ci

**Supported subjects**

- *CI-environment*

- *Process*

- *Software*

**Supporting sources**

- *Anchore*

- *Anchore Jenkins plugin*

- *Axe-core*

- *Azure DevOps Server*

- *Bandit*

- *Calendar date*

- *Cobertura*

- *Cobertura Jenkins plugin*

- *Checkmarx CxSAST*

- *Gatling*

- *GitLab*

- *JaCoCo*

- *JaCoCo Jenkins plugin*

- *Jenkins*

- *Jenkins test report*

- *JMeter CSV*

- *JUnit XML report*

- *NCover*

- *Robot Framework*

- *OpenVAS*

- *OWASP Dependency-Check*

- *OWASP ZAP*

- *Performancetest-runner*

- *Quality-time*

- *Robot Framework Jenkins plugin*

- *SonarQube*

- *TestNG*

- *Trello*

### 7.2.26 Source version

The version number of the source.

*Why measure source version?* Monitor that the version of a source is at least a specific version or get notified when the version becomes higher than a specific version.

*How to configure source version?* If you decide with your development team that you are only interested in the major and minor updates of tools and want to ignore any patch updates, then the following settings can be helpful.

---

**Tip:** Leave the metric direction default, i.e. a higher version number is better. Set the metric target to the first two digits of the latest available software version. So if the latest is 3.9.4, set it to 3.9. Set the metric near target to only the first digit, so in this example that will be 3.

The result will be that when there is a minor update, this metric will turn yellow. So, suppose the version of your tool is at version 2.12.15 and the latest version available is 2.13.2, then this metric will turn yellow. If there is only a patch update, the metric will stay green. If there is a major update, the metric will turn red. So when we have 2.9.3 and the version available is 3.9.1, then the metric will turn red.

---

*More information* Quality-time uses the packaging library (1) to parse version numbers. The packaging library expects version numbers to comply with PEP-440 (2). PEP is an abbrevation for Python Enhancement Proposal, but this PEP is primarily a standard for version numbers. PEP-440 encompasses most of the semantic versioning scheme (3) so version numbers that follow semantic versioning are usually parsed correctly.

**See also:**

1. [https://pypi.org/project/packaging/](https://pypi.org/project/packaging/)

2. [https://peps.python.org/pep-0440/](https://peps.python.org/pep-0440/)

3. [https://semver.org](https://semver.org)

**Default target**
$\geqq 1.0$

**Scales**
version_number

**Default tags**
ci

**Supported subjects**

- *CI-environment*
- *Software*

**Supporting sources**

- *Axe-core*
- *cloc*
- *Cobertura*
- *Checkmarx CxSAST*
- *Gatling*
- *GitLab*
- *Jenkins*
- *Jira*
- *OpenVAS*
- *OWASP Dependency-Check*
- *OWASP ZAP*
- *Quality-time*
- *Robot Framework*
- *SonarQube*

## 7.2.27 Suppressed violations

The number of violations suppressed in the source.

*Why measure suppressed violations?* Some tools allow for suppression of violations. Having the number of suppressed violations violations visible in Quality-time allows for a double check of the suppressions.

**Default target**
≦ 0 suppressed violations

**Scales**
count (default)

percentage

**Default tags**
maintainability

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Manual number*

- *SonarQube*

## 7.2.28 Test branch coverage

The number of code branches not covered by tests.

*Why measure test branch coverage?* Code branches not covered by tests may contain bugs and signal incomplete tests.

**See also:**

1. https://martinfowler.com/bliki/TestCoverage.html

**Default target**
≦ 0 uncovered branches

**Scales**
count (default)

percentage

**Default tags**
test quality

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Cobertura*

- *Cobertura Jenkins plugin*

- *JaCoCo*

- *JaCoCo Jenkins plugin*

- *Manual number*

- *NCover*

- *SonarQube*

### 7.2.29  Test cases

The number of test cases.

*Why measure test cases?* Track the test results of test cases so there is traceability from the test cases, defined in Jira, to the test results in test reports produced by tools such as Robot Framework or Junit.

*How to configure test cases?* The test cases metric reports on the number of test cases, and their test results. The test case metric is different than other metrics because it combines data from two types of sources: it needs one or more sources for the test cases, and one or more sources for the test results. The test case metric then matches the test results with the test cases.

Currently, only Jira is supported as source for the test cases. JUnit, TestNG, and Robot Framework are supported as source for the test results. So, to configure the test cases metric, you need to add at least one Jira source and one JUnit, TestNG, Robot Framework source. In addition, to allow the test case metric to match test cases from Jira with test results from the JUnit, TestNG, or Robot Framework XML files, the test results should mention Jira issue keys in their title or description.

Suppose you have configured Jira with the query: `project = "My Project" and type = "Logical Test Case"` and this results in these test cases:

| Key  | Summary     |
|------|-------------|
| MP-1 | Test case 1 |
| MP-2 | Test case 2 |
| MP-3 | Test case 3 |
| MP-4 | Test case 4 |

Also suppose your JUnit XML has the following test results:

```xml
<testsuite tests="5" errors="0" failures="1" skipped="1">
<testcase name="MP-1; step 1">
    <failure />
</testcase>
<testcase name="MP-1; step 2">
    <skipped />
</testcase>
<testcase name="MP-2">
    <skipped />
</testcase>
<testcase name="MP-3; step 1"/>
<testcase name="MP-3; step 2"/>
</testsuite>
```

The test case metric will combine the JUnit XML file with the test cases from Jira and report one failed, one skipped, one passed, and one untested test case:

| Key | Summary | Test result |
| --- | --- | --- |
| MP-1 | Test case 1 | failed |
| MP-2 | Test case 2 | skipped |
| MP-3 | Test case 3 | passed |
| MP-4 | Test case 4 | untested |

If multiple test results in the JUnit, TestNG, or Robot Framework XML file map to one Jira test case (as with MP-1 and MP-3 above), the 'worst' test result is reported. Possible test results from worst to best are: errored, failed, skipped, and passed. Test cases not found in the test results are listed as untested (as with MP-4 above).

**Default target**
> $\geqq$ 0 test cases

**Scales**
> count (default)
>
> percentage

**Default tags**
> test quality

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Jenkins test report*

- *Jira*

- *JUnit XML report*

- *Manual number*

- *Robot Framework*

- *TestNG*

## 7.2.30 Test line coverage

The number of lines of code not covered by tests.

*Why measure test line coverage?* Code lines not covered by tests may contain bugs and signal incomplete tests.

**See also:**

1. https://martinfowler.com/bliki/TestCoverage.html

**Default target**
> $\leqq$ 0 uncovered lines

**Scales**

> count (default)

> percentage

**Default tags**

> test quality

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Cobertura*

- *Cobertura Jenkins plugin*

- *JaCoCo*

- *JaCoCo Jenkins plugin*

- *Manual number*

- *NCover*

- *SonarQube*

---

## 7.2.31 Tests

The number of tests.

*Why measure tests?* Keep track of the total number of tests or the number of tests with different states, for example failed or errored.

**Default target**

> ≧ 0 tests

**Scales**

> count (default)

> percentage

**Default tags**

> test quality

---

**Supported subjects**

- *Software*

---

**Supporting sources**

- *Azure DevOps Server*

- *Gatling*

- *Jenkins test report*

---

- *JMeter CSV*

- *JMeter JSON*

- *JUnit XML report*

- *Manual number*

- *Performancetest-runner*

- *Robot Framework*

- *Robot Framework Jenkins plugin*

- *SonarQube*

- *TestNG*

## 7.2.32 Time remaining

The number of days remaining until a date in the future.

*Why measure time remaining?* Keep track of the time remaining until for example a release date, the end date of a policy, or the next team building retreat.

**Default target**
   ≧ 28 days

**Scales**
   count

**Supported subjects**

- *Process*

**Supporting sources**

- *Calendar date*

## 7.2.33 Todo and fixme comments

The number of todo and fixme comments in source code.

*Why measure todo and fixme comments?* Code should not contain traces of unfinished work. The presence of todo and fixme comments may be indicative of technical debt or hidden defects.

**See also:**

1. https://rules.sonarsource.com/python/RSPEC-1135/

2. https://cwe.mitre.org/data/definitions/546

**Default target**
   ≦ 0 todo and fixme comments

**Scales**
   count

**Default tags**
    maintainability

**Supported subjects**

  • *Software*

**Supporting sources**

  • *Manual number*

  • *SonarQube*

### 7.2.34 Unmerged branches

The number of branches that have not been merged to the default branch.

*Why measure unmerged branches?* It is strange if branches have had no activity for a while and have not been merged to the default branch. Maybe commits have been cherry picked, or maybe the work has been postponed, but it also sometimes happen that someone simply forgets to merge the branch.

*How to configure unmerged branches?* To change how soon *Quality-time* should consider branches to be inactive, use the parameter "Number of days since last commit after which to consider branches inactive".

What exactly is the default branch is configured in GitLab or Azure DevOps. If you want to use a different branch as default branch, you need to configure this in the source, see the documentation for GitLab or Azure DevOps.

**Default target**
    ≦ 0 branches

**Scales**
    count

**Default tags**
    ci

**Supported subjects**

  • *CI-environment*

  • *Process*

  • *Software*

**Supporting sources**

  • *Azure DevOps Server*

  • *GitLab*

  • *Manual number*

## 7.2.35 Unused CI-jobs

The number of continuous integration jobs that are unused.

*Why measure unused CI-jobs?* Removing unused, obsolete CI-jobs helps to keep a clear overview of the relevant CI-jobs.

**Default target**
$\leqq$ 0 CI-jobs

**Scales**
count

**Default tags**
ci

---

**Supported subjects**

- *CI-environment*

---

**Supporting sources**

- *Azure DevOps Server*
- *GitLab*
- *Jenkins*
- *Manual number*

---

## 7.2.36 User story points

The total number of points of a selection of user stories.

*Why measure user story points?* Keep track of the number of user story points so the team has sufficient 'ready' stories to plan the next sprint.

**Default target**
$\geqq$ 100 user story points

**Scales**
count

**Default tags**
process efficiency

---

**Supported subjects**

- *Process*
- *Software*

---

**Supporting sources**

- *Azure DevOps Server*
- *Jira*

- *Manual number*

## 7.2.37 Velocity

The average number of user story points delivered in recent sprints.

*Why measure velocity?* Keep track of the velocity so the team knows how many story points need at least be 'ready' to plan the next sprint

**Default target**
  ≧ 20 user story points per sprint

**Scales**
  count

**Default tags**
  process efficiency

**Supported subjects**

- *Process*

**Supporting sources**

- *Jira*

- *Manual number*

## 7.2.38 Violation remediation effort

The amount of effort it takes to remediate violations.

*Why measure violation remediation effort?* To better plan the work to remediate violations, it is helpful to have an estimate of the amount of effort it takes to remediate them.

**See also:**

1. https://docs.sonarqube.org/latest/user-guide/metric-definitions/

**Default target**
  ≦ 60 minutes

**Scales**
  count

**Default tags**
  maintainability

**Supported subjects**

- *Software*

**Supporting sources**

- *Manual number*

- *SonarQube*

## 7.2.39 Violations

The number of violations of rules in the software.

*Why measure violations?* Depending on what kind of rules or guideliens the source checks, violations may pose a risk for different quality characteristics of the software. For example, the more programming rules are violated, the harder it may be to understand and maintain the software (1, 2). And the more accessibility guidelines are violated, the harder it may be for users to use the software (3).

**See also:**

1. https://docs.sonarqube.org/latest/user-guide/rules/

2. https://martinfowler.com/bliki/CodeSmell.html

3. https://www.w3.org/standards/webdesign/accessibility

**Default target**
≦ 0 violations

**Scales**
count

**Supported subjects**

- *Software*

**Supporting sources**

- *Axe CSV*
- *Axe-core*
- *Axe HTML reporter*
- *Manual number*
- *OJAudit*
- *SARIF*
- *SonarQube*

## 7.3 Sources

This is an overview of all the sources that *Quality-time* can use to measure metrics. For each source, the metrics that the source can measure are listed. Also, a link to the source's own documentation is provided.

### 7.3.1 Anchore

Anchore image scan analysis report in JSON format.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*

**See also:**

https://docs.anchore.com/current/docs/using/integration/ci_cd/inline_scanning/

### 7.3.2 Anchore Jenkins plugin

A Jenkins job with an Anchore report produced by the Anchore Jenkins plugin.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*

**See also:**

https://plugins.jenkins.io/anchore-container-scanner/

### 7.3.3 Axe CSV

An Axe accessibility report in CSV format.

**Supported metrics**

- *Violations*

**See also:**

https://github.com/ICTU/axe-reports

### 7.3.4 Axe HTML reporter

Creates an HTML report from the axe-core library AxeResults object.

---

**Supported metrics**

- *Violations*

---

**See also:**

https://www.npmjs.com/package/axe-html-reporter

### 7.3.5 Axe-core

Axe is an accessibility testing engine for websites and other HTML-based user interfaces.

---

**Supported metrics**

- *Source up-to-dateness*
- *Source version*
- *Violations*

---

**See also:**

https://github.com/dequelabs/axe-core

### 7.3.6 Azure DevOps Server

Azure DevOps Server (formerly known as Team Foundation Server) by Microsoft provides source code management, reporting, requirements management, project management, automated builds, testing and release management.

---

**Supported metrics**

- *Average issue lead time*
- *Failed CI-jobs*
- *Issues*
- *Job runs within time period*
- *Merge requests*
- *Source up-to-dateness*
- *Tests*
- *Unmerged branches*
- *Unused CI-jobs*
- *User story points*

---

**See also:**

https://azure.microsoft.com/en-us/services/devops/server/

## 7.3.7 Bandit

Bandit is a tool designed to find common security issues in Python code.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*

**See also:**

https://github.com/PyCQA/bandit

## 7.3.8 Calendar date

Specify a specific date in the past or the future. Can be used to, for example, warn when it is time for the next security test.

**Supported metrics**

- *Source up-to-dateness*
- *Time remaining*

## 7.3.9 Cargo Audit

Cargo Audit is a linter for Rust Cargo.lock files for crates.

**Supported metrics**

- *Security warnings*

**See also:**

https://docs.rs/cargo-audit/latest/cargo_audit/

## 7.3.10 Checkmarx CxSAST

Static analysis software to identify security vulnerabilities in both custom code and open source components.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*
- *Source version*

**See also:**

https://checkmarx.com/glossary/static-application-security-testing-sast/

### 7.3.11 Cobertura

Cobertura is a free Java tool that calculates the percentage of code accessed by tests.

**Supported metrics**

- *Source up-to-dateness*
- *Source version*
- *Test branch coverage*
- *Test line coverage*

**See also:**

https://cobertura.github.io/cobertura/

### 7.3.12 Cobertura Jenkins plugin

Jenkins plugin for Cobertura, a free Java tool that calculates the percentage of code accessed by tests.

**Supported metrics**

- *Source up-to-dateness*
- *Test branch coverage*
- *Test line coverage*

**See also:**

https://plugins.jenkins.io/cobertura/

### 7.3.13 Composer

A Dependency Manager for PHP.

**Supported metrics**

- *Dependencies*

**See also:**

https://getcomposer.org/

## 7.3.14 Gatling

Gatling is an open-source load testing solution, designed for continuous load testing and development pipeline integration.

**Supported metrics**

- *Performancetest duration*
- *Slow transactions*
- *Source up-to-dateness*
- *Source version*
- *Tests*

**See also:**

https://gatling.io/

## 7.3.15 GitLab

GitLab provides Git-repositories, wiki's, issue-tracking and continuous integration/continuous deployment pipelines.

**Note:** Some metric sources are documents in JSON, XML, CSV, or HTML format. Examples include JUnit XML reports, JaCoCo XML reports and Axe CSV reports. Usually, you add a JUnit (or JaCoCo, or Axe…) source and then simply configure the same URL that you use to access the document via the browser. Unfortunately, this does not work if the document is stored in GitLab. In that case, you still use the JUnit (or JaCoCo, or Axe…) source, but provide a GitLab API URL as URL. Depending on where the document is stored in GitLab, there are two scenarios; the source is a build artifact of a GitLab CI pipeline, or the source is stored in a GitLab repository:

1. When the metric source is a build artifact of a GitLab CI pipeline, use URLs of the following format:

   ```
   https://<gitlab-server>/api/v4/projects/<project-id>/jobs/artifacts/
   <branch>/raw/<path>/<to>/<file-name>?job=<job-name>
   ```

   The project id can be found under the project's general settings.

   If the repository is private, you also need to enter an personal access token with the scope `read_api` in the private token field.

2. When the metric source is a file stored in a GitLab repository, use URLs of the following format:

   ```
   https://<gitlab-server>/api/v4/projects/<project-id>/repository/files/
   <file-path-with-slashes-%2F-encoded>/raw?ref=<branch>
   ```

   The project id can be found under the project's general settings.

   If the repository is private, you also need to enter an personal access token with the scope `read_repository` in the private token field.

**Supported metrics**

- *Change failure rate*
- *Failed CI-jobs*
- *Job runs within time period*

- *Merge requests*

- *Source up-to-dateness*

- *Source version*

- *Unmerged branches*

- *Unused CI-jobs*

**See also:**

https://about.gitlab.com/

### 7.3.16 Harbor

Harbor is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted.

**Supported metrics**

- *Security warnings*

**See also:**

https://goharbor.io/

### 7.3.17 Harbor JSON

Harbor is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted. Use Harbor JSON as source for accessing vulnerability reports downloaded from the Harbor API in JSON format.

**Supported metrics**

- *Security warnings*

**See also:**

https://goharbor.io/

### 7.3.18 JMeter CSV

Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.

**Supported metrics**

- *Performancetest duration*

- *Slow transactions*

- *Source up-to-dateness*

- *Tests*

**See also:**

https://jmeter.apache.org/

### 7.3.19 JMeter JSON

Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.

**Supported metrics**

- *Slow transactions*
- *Tests*

**See also:**

https://jmeter.apache.org/

### 7.3.20 JSON file with security warnings

A generic vulnerability report with security warnings in JSON format.

In some cases, there are security vulnerabilities not found by automated tools. Quality-time has the ability to parse security warnings from JSON files with a generic format.

The JSON format consists of an object with one key `vulnerabilities`. The value should be a list of vulnerabilities. Each vulnerability is an object with three keys: `title`, `description`, and `severity`. The title and description values should be strings. The severity is also a string and can be either `low`, `medium`, or `high`.

Example generic JSON file:

```json
{
    "vulnerabilities": [
        {
            "title": "ISO27001:2013 A9 Insufficient Access Control",
            "description": "The Application does not enforce Two-Factor
↪Authentication and therefore does not satisfy                  security best
↪practices.",
            "severity": "high"
        },
        {
            "title": "Threat Model Finding: Uploading Malicious Files",
            "description": "An attacker can upload malicious files with low
↪privileges that can perform direct API                  calls and perform unwanted
↪mutations or see unauthorized information.",
            "severity": "medium"
        }
    ]
}
```

**Supported metrics**

- *Security warnings*

### 7.3.21  JUnit XML report

Test reports in the JUnit XML format.

**Supported metrics**

- *Source up-to-dateness*
- *Test cases*
- *Tests*

**See also:**

https://junit.org/junit5/

### 7.3.22  JaCoCo

JaCoCo is an open-source tool for measuring and reporting Java code coverage.

**Supported metrics**

- *Source up-to-dateness*
- *Test branch coverage*
- *Test line coverage*

**See also:**

https://www.eclemma.org/jacoco/

### 7.3.23  JaCoCo Jenkins plugin

A Jenkins job with a JaCoCo coverage report produced by the JaCoCo Jenkins plugin.

**Supported metrics**

- *Source up-to-dateness*
- *Test branch coverage*
- *Test line coverage*

**See also:**

https://plugins.jenkins.io/jacoco

## 7.3.24 Jenkins

Jenkins is an open source continuous integration/continuous deployment server.

---

**Note:** Some metric sources are documents in JSON, XML, CSV, or HTML format. Examples include JUnit XML reports, JaCoCo XML reports and Axe CSV reports. Usually, you add a JUnit (or JaCoCo, or Axe…) source and then simply configure the same URL that you use to access the document via the browser. If the document is stored in Jenkins and *Quality-time* needs to be authorized to access resources in Jenkins, there are two options:

1. Configure a Jenkins user and password. The username of the Jenkins user needs to be entered in the "Username" field and the password in the "Password" field.

2. Configure a Jenkins user and a private token of that user. The username of the Jenkins user needs to be entered in the "Username" field and the private token in the "**Password**" field.

---

**Supported metrics**

- *Change failure rate*
- *Failed CI-jobs*
- *Job runs within time period*
- *Source up-to-dateness*
- *Source version*
- *Unused CI-jobs*

**See also:**

https://www.jenkins.io/

## 7.3.25 Jenkins test report

A Jenkins job with test results.

---

**Supported metrics**

- *Source up-to-dateness*
- *Test cases*
- *Tests*

**See also:**

https://plugins.jenkins.io/junit

---

## 7.3.26 Jira

Jira is a proprietary issue tracker developed by Atlassian supporting bug tracking and agile project management.

**Supported metrics**

- *Average issue lead time*
- *Change failure rate*
- *Issues*
- *Manual test duration*
- *Manual test execution*
- *Source version*
- *Test cases*
- *User story points*
- *Velocity*

**See also:**

https://www.atlassian.com/software/jira

## 7.3.27 Manual number

A number entered manually by a Quality-time user.

The manual number source supports all metric types that take a number as value. Because users have to keep the value up to date by hand, this source is only meant to be used as a temporary solution for when no automated source is available yet. For example, when the results of a security audit are only available in a PDF-report, a 'security warnings' metric can be added with the number of findings as manual number source.

**Supported metrics**

- *Commented out code*
- *Complex units*
- *Dependencies*
- *Duplicated lines*
- *Failed CI-jobs*
- *Issues*
- *Job runs within time period*
- *Size (LOC)*
- *Long units*
- *Manual test duration*
- *Manual test execution*
- *Many parameters*

- *Merge requests*
- *Metrics*
- *Missing metrics*
- *Performancetest duration*
- *Performancetest stability*
- *Violation remediation effort*
- *Scalability*
- *Security warnings*
- *Sentiment*
- *Slow transactions*
- *Suppressed violations*
- *Test cases*
- *Tests*
- *Todo and fixme comments*
- *Test branch coverage*
- *Test line coverage*
- *Unmerged branches*
- *Unused CI-jobs*
- *User story points*
- *Velocity*
- *Violations*

### 7.3.28 NCover

A .NET code coverage solution.

**Supported metrics**

- *Source up-to-dateness*
- *Test branch coverage*
- *Test line coverage*

**See also:**

https://www.ncover.com/

### 7.3.29  OJAudit

An Oracle JDeveloper program to audit Java code against JDeveloper's audit rules.

**Supported metrics**

- *Violations*

**See also:**

https://www.oracle.com/application-development/technologies/jdeveloper.html

### 7.3.30  OWASP Dependency-Check

OWASP Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

**Supported metrics**

- *Dependencies*
- *Security warnings*
- *Source up-to-dateness*
- *Source version*

**See also:**

https://owasp.org/www-project-dependency-check/

### 7.3.31  OWASP ZAP

The OWASP Zed Attack Proxy (ZAP) can help automatically find security vulnerabilities in web applications while the application is being developed and tested.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*
- *Source version*

**See also:**

https://owasp.org/www-project-zap/

## 7.3.32 OpenVAS

OpenVAS (Open Vulnerability Assessment System) is a software framework of several services and tools offering vulnerability scanning and vulnerability management.

**Supported metrics**

- *Security warnings*
- *Source up-to-dateness*
- *Source version*

**See also:**

https://www.openvas.org/

## 7.3.33 Performancetest-runner

An open source tool to run performancetests and create performancetest reports.

**Supported metrics**

- *Performancetest duration*
- *Performancetest stability*
- *Scalability*
- *Slow transactions*
- *Software version*
- *Source up-to-dateness*
- *Tests*

**See also:**

https://github.com/ICTU/performancetest-runner

## 7.3.34 Pyupio Safety

Safety checks Python dependencies for known security vulnerabilities.

**Supported metrics**

- *Security warnings*

**See also:**

https://github.com/pyupio/safety

### 7.3.35 Quality-time

Quality report software for software development and maintenance.

---

**Supported metrics**

- *Metrics*
- *Missing metrics*
- *Source up-to-dateness*
- *Source version*

---

**See also:**

https://github.com/ICTU/quality-time

### 7.3.36 Robot Framework

Robot Framework is a generic open source automation framework for acceptance testing, acceptance test driven development, and robotic process automation.

---

**Supported metrics**

- *Source up-to-dateness*
- *Source version*
- *Test cases*
- *Tests*

---

**See also:**

https://robotframework.org/

### 7.3.37 Robot Framework Jenkins plugin

A Jenkins plugin for Robot Framework, a generic open source automation framework for acceptance testing, acceptance test driven development, and robotic process automation.

---

**Supported metrics**

- *Source up-to-dateness*
- *Tests*

---

**See also:**

https://plugins.jenkins.io/robot/

### 7.3.38 SARIF

A Static Analysis Results Interchange Format (SARIF) vulnerability report in JSON format.

---

**Supported metrics**

- *Security warnings*

- *Violations*

---

**See also:**

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=sarif

### 7.3.39 Snyk

Snyk vulnerability report in JSON format.

---

**Supported metrics**

- *Security warnings*

---

**See also:**

https://docs.snyk.io/products/snyk-code/cli-for-snyk-code/working-with-the-snyk-code-cli-results

### 7.3.40 SonarQube

SonarQube is an open-source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages.

---

**Supported metrics**

- *Commented out code*
- *Complex units*
- *Duplicated lines*
- *Size (LOC)*
- *Long units*
- *Many parameters*
- *Violation remediation effort*
- *Security warnings*
- *Software version*
- *Source up-to-dateness*
- *Source version*
- *Suppressed violations*
- *Tests*

---

- *Todo and fixme comments*
- *Test branch coverage*
- *Test line coverage*
- *Violations*

**See also:**

https://www.sonarqube.org/

## 7.3.41  TestNG

Test reports in the TestNG XML format.

**Supported metrics**

- *Source up-to-dateness*
- *Test cases*
- *Tests*

**See also:**

https://testng.org/

## 7.3.42  Trello

Trello is a collaboration tool that organizes projects into boards.

**Supported metrics**

- *Issues*
- *Source up-to-dateness*

**See also:**

https://trello.com/

## 7.3.43  Trivy JSON

A Trivy vulnerability report in JSON format.

**Supported metrics**

- *Security warnings*

**See also:**

https://aquasecurity.github.io/trivy/latest/docs/configuration/reporting/#json

### 7.3.44  cloc

cloc is an open-source tool for counting blank lines, comment lines, and physical lines of source code in many programming languages.

**Supported metrics**

- *Size (LOC)*

- *Source version*

**See also:**

https://github.com/AlDanial/cloc

### 7.3.45  npm

npm is a package manager for the JavaScript programming language.

**Supported metrics**

- *Dependencies*

**See also:**

https://docs.npmjs.com/

### 7.3.46  pip

pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes.

**Supported metrics**

- *Dependencies*

**See also:**

https://pip.pypa.io/en/stable/

## 7.4  Metric-source combinations

This is an overview of all supported combinations of metrics and sources. For each combination of metric and source, the mandatory and optional parameters are listed that can be used to configure the source to measure the metric. If *Quality-time* needs to make certain assumptions about the source, for example which SonarQube rules to use to count long methods, then these assumptions are listed under 'configurations'.

## 7.4.1 Average issue lead time from Azure DevOps Server

*Azure DevOps Server* can be used to measure *average issue lead time*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Issue query in WIQL (Work Item Query Language)**. This should only contain the WHERE clause of a WIQL query, as the selected fields are static. For example, use the following clause to hide issues marked as done: "[System.State] <> 'Done'". See https://docs.microsoft.com/en-us/azure/devops/boards/queries/wiql-syntax?view=azure-devops.

- **Number of days to look back for work items**. Work items are selected if they are completed and have been updated within the number of days configured. The default value is: 90.

- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.2 Average issue lead time from Jira

*Jira* can be used to measure *average issue lead time*.

### Mandatory parameters

- **Issue query in JQL (Jira Query Language)**.

  See also:

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

### Optional parameters

- **Number of days to look back for selecting issues**. Issues are selected if they are completed and have been updated within the number of days configured. The default value is: 90.

- **Password for basic authentication**.

- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

### 7.4.3 Change failure rate from Jenkins

*Jenkins* can be used to measure *change failure rate*.

#### Mandatory parameters

- **URL**. URL of the Jenkins instance, with port if necessary, but without path. For example, 'https://jenkins.example.org'.

#### Optional parameters

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Jobs to include (regular expressions or job names)**. Jobs to include can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Number of days to look back for selecting job builds**. The default value is: 90.

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.4 Change failure rate from Jira

*Jira* can be used to measure *change failure rate*.

#### Mandatory parameters

- **Issue query in JQL (Jira Query Language)**.

  **See also:**

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

#### Optional parameters

- **Number of days to look back for selecting issues**. Issues are selected if they are completed and have been updated within the number of days configured. The default value is: 90.

- **Password for basic authentication**.

- **Private token**.

  **See also:**

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

## 7.4.5 Change failure rate from GitLab

*GitLab* can be used to measure *change failure rate*.

**Mandatory parameters**

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

**Optional parameters**

- **Branches and tags to ignore (regular expressions, branch names or tag names)**.

  See also:

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression.

- **Number of days to look back for selecting pipeline jobs**. The default value is: `90`.

- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

## 7.4.6 Commented out code from Manual number

*Manual number* can be used to measure *commented out code*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

## 7.4.7 Commented out code from SonarQube

*SonarQube* can be used to measure *commented out code*.

**Mandatory parameters**

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

Configurations:

- Rules used to detect commented out code:

    - abap:S125

    - c:S125

    - cpp:S125

    - csharpsquid:S125

    - flex:CommentedCode

    - java:S125

    - javascript:S125

    - kotlin:S125

    - objc:S125

    - php:S125

    - plsql:S125

    - python:S125

    - scala:S125

    - swift:S125

    - tsql:S125

    - typescript:S125

    - Web:AvoidCommentedOutCodeCheck

    - xml:S125

## 7.4.8 Complex units from Manual number

*Manual number* can be used to measure *complex units*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.9 Complex units from SonarQube

*SonarQube* can be used to measure *complex units*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

Configurations:

- Rules used to detect complex units:

    - abap:S1541

    - abap:S3776

    - c:S1541

    - c:S3776

    - cpp:S1541

    - cpp:S3776

    - csharpsquid:S1541

    - csharpsquid:S3776

    - flex:FunctionComplexity

    - go:S3776

    - java:S1541

    - java:S3776

- javascript:S1541
- javascript:S3776
- kotlin:S3776
- objc:S1541
- objc:S3776
- php:S1541
- php:S3776
- plsql:PlSql.FunctionAndProcedureComplexity
- python:FunctionComplexity
- python:S3776
- ruby:S3776
- scala:S3776
- swift:S1541
- swift:S3776
- typescript:S1541
- typescript:S3776
- vbnet:S1541
- vbnet:S3776

## 7.4.10 Dependencies from Composer

*Composer* can be used to measure *dependencies*.

### Mandatory parameters

- **URL to a Composer 'outdated' report in JSON format or to a zip with Composer 'outdated' reports in JSON format**.

  **See also:**

  https://getcomposer.org/doc/03-cli.md#outdated

### Optional parameters

- **Latest version statuses**. Limit which latest version statuses to show. The status 'safe update possible' means that based on semantic versioning the update should be backwards compatible. This parameter is multiple choice. Possible statuses are: `safe update possible`, `unknown`, `up-to-date`, `update possible`. The default value is: *all statuses*.
- **Password for basic authentication**.
- **Private token**.
- **URL to a Composer 'outdated' report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Composer 'outdated' report in JSON format.

- **Username for basic authentication**.

## 7.4.11 Dependencies from Manual number

*Manual number* can be used to measure *dependencies*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.12 Dependencies from npm

*npm* can be used to measure *dependencies*.

---

**Tip:** To generate the list of outdated dependencies, run:

```
npm outdated --all --long --json > npm-outdated.json
```

To run `npm outdated` with Docker, use:

```
docker run --rm -v "$SRC":/work -w /work node:lts npm outdated --all --long --json >␣
→outdated.json"
```

---

### Mandatory parameters

- **URL to a npm 'outdated' report in JSON format or to a zip with npm 'outdated' reports in JSON format**.
  See also:

  https://docs.npmjs.com/cli-commands/outdated.html

### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a npm 'outdated' report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the npm 'outdated' report in JSON format.
- **Username for basic authentication**.

### 7.4.13 Dependencies from OWASP Dependency-Check

*OWASP Dependency-Check* can be used to measure *dependencies*.

#### Mandatory parameters

- **URL to an OWASP Dependency-Check report in XML format or to a zip with OWASP Dependency-Check reports in XML format**.

#### Optional parameters

- **Parts of file paths to ignore (regular expressions)**. Parts of file paths to ignore can be specified by regular expression. The parts of file paths that match one or more of the regular expressions are removed. If, after applying the regular expressions, multiple warnings are the same only one is reported.
- **Password for basic authentication**.
- **Private token**.
- **URL to an OWASP Dependency-Check report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP Dependency-Check report in XML format.
- **Username for basic authentication**.

### 7.4.14 Dependencies from pip

*pip* can be used to measure *dependencies*.

---

**Tip:** To generate the list of outdated packages in JSON format, run:

```
python -m pip list --outdated --format=json > pip-outdated.json
```

---

#### Mandatory parameters

- **URL to a pip 'outdated' report in JSON format or to a zip with pip 'outdated' reports in JSON format**.

  **See also:**

  https://pip.pypa.io/en/stable/cli/pip_list/

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a pip 'outdated' report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the pip 'outdated' report in JSON format.
- **Username for basic authentication**.

### 7.4.15 Duplicated lines from Manual number

*Manual number* can be used to measure *duplicated lines*.

#### Mandatory parameters

- **Number**. The default value is: `0`.

### 7.4.16 Duplicated lines from SonarQube

*SonarQube* can be used to measure *duplicated lines*.

#### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

#### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

### 7.4.17 Failed CI-jobs from Azure DevOps Server

*Azure DevOps Server* can be used to measure *failed ci-jobs*.

#### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

**Optional parameters**

- **Failure types**. Limit which failure types to count as failed. This parameter is multiple choice. Possible failure types are: `canceled`, `failed`, `no result`, `partially succeeded`. The default value is: *all failure types*.

- **Pipelines to ignore (regular expressions or pipeline names)**. Pipelines to ignore can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Pipelines to include (regular expressions or pipeline names)**. Pipelines to include can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.18 Failed CI-jobs from Jenkins

*Jenkins* can be used to measure *failed ci-jobs*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL**. URL of the Jenkins instance, with port if necessary, but without path. For example, 'https://jenkins.example.org'.

**Optional parameters**

- **Failure types**. Limit which failure types to count as failed. This parameter is multiple choice. Possible failure types are: `Aborted`, `Failure`, `Not built`, `Unstable`. The default value is: *all failure types*.

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Jobs to include (regular expressions or job names)**. Jobs to include can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Password or API token for basic authentication**.

  See also:

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.19 Failed CI-jobs from GitLab

*GitLab* can be used to measure *failed ci-jobs*.

### Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

### Optional parameters

- **Branches and tags to ignore (regular expressions, branch names or tag names)**.

  See also:

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **Failure types**. Limit which failure types to count as failed. This parameter is multiple choice. Possible failure types are: `canceled`, `failed`, `skipped`. The default value is: *all failure types*.

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression.

- **Number of days to look back for selecting pipeline jobs**. The default value is: `90`.

- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

## 7.4.20 Failed CI-jobs from Manual number

*Manual number* can be used to measure *failed ci-jobs*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.21 Issues from Azure DevOps Server

*Azure DevOps Server* can be used to measure *issues*.

**Mandatory parameters**

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

**Optional parameters**

- **Issue query in WIQL (Work Item Query Language)**. This should only contain the WHERE clause of a WIQL query, as the selected fields are static. For example, use the following clause to hide issues marked as done: "[System.State] <> 'Done'". See https://docs.microsoft.com/en-us/azure/devops/boards/queries/wiql-syntax?view=azure-devops.

- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.22  Issues from Jira

*Jira* can be used to measure *issues*.

**Mandatory parameters**

- **Issue query in JQL (Jira Query Language)**.

  See also:

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

## 7.4.23  Issues from Manual number

*Manual number* can be used to measure *issues*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

## 7.4.24 Issues from Trello

*Trello* can be used to measure *issues*.

**Mandatory parameters**

- **Board (title or id)**.

  **See also:**

  https://trello.com/1/members/me/boards?fields=name

- **URL**. The default value is: `https://trello.com`.

**Optional parameters**

- **API key**.

  **See also:**

  https://trello.com/app-key

- **Cards to count**. This parameter is multiple choice. Possible cards are: `inactive`, `overdue`. The default value is: *all cards*.

- **Lists to ignore (title or id)**.

- **Number of days without activity after which to consider cards inactive**. The default value is: `30`.

- **Token**.

  **See also:**

  https://trello.com/app-key

## 7.4.25 Job runs within time period from Azure DevOps Server

*Azure DevOps Server* can be used to measure *job runs within time period*.

**Mandatory parameters**

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

## Optional parameters

- **Number of days to look back for selecting pipeline runs**. The default value is: 90.

- **Pipelines to ignore (regular expressions or pipeline names)**. Pipelines to ignore can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Pipelines to include (regular expressions or pipeline names)**. Pipelines to include can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.26 Job runs within time period from GitLab

*GitLab* can be used to measure *job runs within time period*.

## Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

## Optional parameters

- **Branches and tags to ignore (regular expressions, branch names or tag names)**.

  See also:

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression.

- **Number of days to look back for selecting pipeline jobs**. The default value is: 90.

- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

### 7.4.27 Job runs within time period from Jenkins

*Jenkins* can be used to measure *job runs within time period*.

#### Mandatory parameters

- **URL**. URL of the Jenkins instance, with port if necessary, but without path. For example, 'https://jenkins.example.org'.

#### Optional parameters

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Jobs to include (regular expressions or job names)**. Jobs to include can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Number of days to look back for selecting job builds**. The default value is: `90`.

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.28 Job runs within time period from Manual number

*Manual number* can be used to measure *job runs within time period*.

#### Mandatory parameters

- **Number**. The default value is: `0`.

### 7.4.29 Size (LOC) from cloc

*cloc* can be used to measure *size (loc)*.

#### Mandatory parameters

- **URL to a cloc report in JSON format or to a zip with cloc reports in JSON format**.

**Optional parameters**

- **Files to include (regular expressions or file names)**. Note that filtering files only works when the cloc report is generated with the –by-file option.

- **Languages to ignore (regular expressions or language names)**.

  **See also:**

  https://github.com/AlDanial/cloc#recognized-languages-

- **Password for basic authentication**.

- **Private token**.

- **URL to a cloc report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the cloc report in JSON format.

- **Username for basic authentication**.

## 7.4.30 Size (LOC) from Manual number

*Manual number* can be used to measure *size (loc)*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

## 7.4.31 Size (LOC) from SonarQube

*SonarQube* can be used to measure *size (loc)*.

**Mandatory parameters**

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Languages to ignore (regular expressions or language names)**.

  **See also:**

  https://docs.sonarqube.org/latest/analysis/languages/overview/

- **Lines to count**. Either count all lines including lines with comments or only count lines with code, excluding comments. Note: it's possible to ignore specific languages only when counting lines with code. This is a SonarQube limitation. This parameter is single choice. Possible lines to count are: `all lines`, `lines with code`. The default value is: `lines with code`.

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

## 7.4.32 Long units from Manual number

*Manual number* can be used to measure *long units*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.33 Long units from SonarQube

*SonarQube* can be used to measure *long units*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

Configurations:

- Rules used to detect long units:

  - abap:S104
  - c:S104
  - c:S1151
  - c:S138
  - cpp:S104

- – cpp:S1151
- – cpp:S1188
- – cpp:S138
- – cpp:S6184
- – csharpsquid:S104
- – csharpsquid:S1151
- – csharpsquid:S138
- – flex:S1151
- – flex:S138
- – go:S104
- – go:S1151
- – go:S138
- – java:S104
- – java:S1151
- – java:S1188
- – java:S138
- – java:S2972
- – java:S5612
- – javascript:S104
- – javascript:S138
- – kotlin:S104
- – kotlin:S1151
- – kotlin:S138
- – kotlin:S5612
- – objc:S104
- – objc:S1151
- – objc:S138
- – php:S104
- – php:S1151
- – php:S138
- – php:S2042
- – plsql:S104
- – plsql:S1151
- – python:S104
- – python:S138
- – ruby:S104

- ruby:S1151
- ruby:S138
- scala:S104
- scala:S1151
- scala:S138
- swift:S104
- swift:S1151
- swift:S1188
- swift:S138
- swift:S2042
- tsql:S104
- tsql:S1151
- tsql:S138
- typescript:S104
- typescript:S138
- vbnet:S104
- vbnet:S1151
- vbnet:S138
- Web:FileLengthCheck
- Web:LongJavaScriptCheck

## 7.4.34 Manual test duration from Jira

*Jira* can be used to measure *manual test duration*.

**Mandatory parameters**

- **Issue query in JQL (Jira Query Language)**.

  **See also:**

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **Manual test duration field (name or id)**.

  **See also:**

  https://confluence.atlassian.com/jirakb/how-to-find-id-for-custom-field-s-744522503.html

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

## 7.4.35 Manual test duration from Manual number

*Manual number* can be used to measure *manual test duration*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.36 Manual test execution from Jira

*Jira* can be used to measure *manual test execution*.

### Mandatory parameters

- **Default expected manual test execution frequency (days)**. Specify how often the manual tests should be executed. For example, if the sprint length is three weeks, manual tests should be executed at least once every 21 days. The default value is: `21`.
- **Issue query in JQL (Jira Query Language)**.

  See also:

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

### Optional parameters

- **Manual test execution frequency field (name or id)**.

  See also:

  https://confluence.atlassian.com/jirakb/how-to-find-id-for-custom-field-s-744522503.html

- **Password for basic authentication**.
- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

## 7.4.37 Manual test execution from Manual number

*Manual number* can be used to measure *manual test execution*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.38 Many parameters from Manual number

*Manual number* can be used to measure *many parameters*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.39 Many parameters from SonarQube

*SonarQube* can be used to measure *many parameters*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

Configurations:

- Rules used to detect units with many parameters:

    - c:S107

    - cpp:S107

    - csharpsquid:S107

    - csharpsquid:S2436

    - flex:S107

- go:S107

- java:S107

- javascript:S107

- kotlin:S107

- objc:S107

- php:S107

- python:S107

- ruby:S107

- scala:S107

- swift:S107

- tsql:S107

- typescript:S107

- vbnet:S107

## 7.4.40 Merge requests from Azure DevOps Server

*Azure DevOps Server* can be used to measure *merge requests*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Merge request states**. Limit which merge request states to count. This parameter is multiple choice. Possible states are: `abandoned`, `active`, `completed`, `not set`. The default value is: *all states*.

- **Minimum number of upvotes**. Only count merge requests with fewer than the minimum number of upvotes. The default value is: `0`.

- **Private token**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

- **Repository (name or id)**.

- **Target branches to include (regular expressions or branch names)**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/repos/git/manage-your-branches?view=azure-devops

## 7.4.41 Merge requests from GitLab

*GitLab* can be used to measure *merge requests*.

### Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

### Optional parameters

- **Approval states to include (requires GitLab Premium)**. This parameter is multiple choice. Possible approval states are: `approved`, `not approved`, `unknown`. The default value is: *all approval states*.

  See also:

  https://docs.gitlab.com/ee/user/project/merge_requests/approvals/

- **Merge request states**. Limit which merge request states to count. This parameter is multiple choice. Possible states are: `closed`, `locked`, `merged`, `opened`. The default value is: *all states*.

- **Minimum number of upvotes**. Only count merge requests with fewer than the minimum number of upvotes. The default value is: `0`.

- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

- **Target branches to include (regular expressions or branch names)**.

  See also:

  https://docs.gitlab.com/ee/user/project/repository/branches/

## 7.4.42 Merge requests from Manual number

*Manual number* can be used to measure *merge requests*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.43 Metrics from Manual number

*Manual number* can be used to measure *metrics*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.44 Metrics from Quality-time

*Quality-time* can be used to measure *metrics*.

### Mandatory parameters

- **Quality-time URL**. URL of the Quality-time instance, with port if necessary, but without path. For example, 'https://quality-time.example.org'.

### Optional parameters

- **Metric statuses**. This parameter is multiple choice. Possible metric statuses are: `informative (blue)`, `near target met (yellow)`, `target met (green)`, `target not met (red)`, `technical debt target met (grey)`, `unknown (white)`. The default value is: *all metric statuses*.

- **Metric types**. If provided, only count metrics with the selected metric types. This parameter is multiple choice. Possible metric types are: `Average issue lead time`, `Change failure rate`, `Commented out code`, `Complex units`, `Dependencies`, `Duplicated lines`, `Failed CI-jobs`, `Issues`, `Job runs within time period`, `Long units`, `Manual test duration`, `Manual test execution`, `Many parameters`, `Merge requests`, `Metrics`, `Missing metrics`, `Performancetest duration`, `Performancetest stability`, `Scalability`, `Security warnings`, `Sentiment`, `Size (LOC)`, `Slow transactions`, `Software version`, `Source up-to-dateness`, `Source version`, `Suppressed violations`, `Test branch coverage`, `Test cases`, `Test line coverage`, `Tests`, `Time remaining`, `Todo and fixme comments`, `Unmerged branches`, `Unused CI-jobs`, `User story points`, `Velocity`, `Violation remediation effort`, `Violations`. The default value is: *all metric types*.

- **Minimum metric status duration**. Only count metrics whose status has not changed for the given number of days. The default value is: `0`.

- **Report names or identifiers**.

- **Source types**. If provided, only count metrics with one or more sources of the selected source types. This parameter is multiple choice. Possible source types are: `Anchore Jenkins plugin`, `Anchore`, `Axe CSV`, `Axe HTML reporter`, `Axe-core`, `Azure DevOps Server`, `Bandit`, `Calendar date`, `Cargo Audit`, `Checkmarx CxSAST`, `Cobertura Jenkins plugin`, `Cobertura`, `Composer`, `Gatling`, `GitLab`, `Harbor JSON`, `Harbor`, `JMeter CSV`, `JMeter JSON`, `JSON file with security warnings`, `JUnit XML report`, `JaCoCo Jenkins plugin`, `JaCoCo`, `Jenkins test report`, `Jenkins`, `Jira`, `Manual number`, `NCover`, `OJAudit`, `OWASP Dependency-Check`, `OWASP ZAP`, `OpenVAS`, `Performancetest-runner`, `Pyupio Safety`, `Quality-time`, `Robot Framework Jenkins plugin`, `Robot Framework`, `SARIF`, `Snyk`, `SonarQube`, `TestNG`, `Trello`, `Trivy JSON`, `cloc`, `npm`, `pip`. The default value is: *all source types*.

- **Tags**. If provided, only count metrics with one ore more of the given tags.

## 7.4.45 Missing metrics from Manual number

*Manual number* can be used to measure *missing metrics*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.46 Missing metrics from Quality-time

*Quality-time* can be used to measure *missing metrics*.

### Mandatory parameters

- **Quality-time URL**. URL of the Quality-time instance, with port if necessary, but without path. For example, 'https://quality-time.example.org'.

### Optional parameters

- **Report names or identifiers**.
- **Subjects to ignore (subject names or identifiers)**. The Quality-time missing metrics collector will ignore metrics that are missing in the list of subjects to ignore.

## 7.4.47 Performancetest duration from Gatling

*Gatling* can be used to measure *performancetest duration*.

### Mandatory parameters

- **URL to a Gatling report in HTML format or to a zip with Gatling reports in HTML format**.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.48 Performancetest duration from JMeter CSV

*JMeter CSV* can be used to measure *performancetest duration*.

#### Mandatory parameters

- **URL to a JMeter report in CSV format or to a zip with JMeter reports in CSV format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in CSV format.
- **Username for basic authentication**.

### 7.4.49 Performancetest duration from Manual number

*Manual number* can be used to measure *performancetest duration*.

#### Mandatory parameters

- **Number**. The default value is: `0`.

### 7.4.50 Performancetest duration from Performancetest-runner

*Performancetest-runner* can be used to measure *performancetest duration*.

#### Mandatory parameters

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.51 Performancetest stability from Manual number

*Manual number* can be used to measure *performancetest stability*.

#### Mandatory parameters

- **Number**. The default value is: `0`.

### 7.4.52 Performancetest stability from Performancetest-runner

*Performancetest-runner* can be used to measure *performancetest stability*.

#### Mandatory parameters

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.53 Violation remediation effort from Manual number

*Manual number* can be used to measure *violation remediation effort*.

#### Mandatory parameters

- **Number**. The default value is: `0`.

### 7.4.54 Violation remediation effort from SonarQube

*SonarQube* can be used to measure *violation remediation effort*.

#### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

- **Types of effort**. This parameter is multiple choice. Possible effort types are: `effort to fix all bug issues`, `effort to fix all code smells`, `effort to fix all vulnerabilities`. The default value is: *all effort types*.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/metric-definitions/

### 7.4.55 Scalability from Manual number

*Manual number* can be used to measure *scalability*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

### 7.4.56 Scalability from Performancetest-runner

*Performancetest-runner* can be used to measure *scalability*.

**Mandatory parameters**

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

## 7.4.57 Security warnings from Anchore

*Anchore* can be used to measure *security warnings*.

**Mandatory parameters**

- **URL to an Anchore vulnerability report in JSON format or to a zip with Anchore vulnerability reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `Critical`, `High`, `Low`, `Medium`, `Negligible`, `Unknown`. The default value is: *all severities*.

- **URL to an Anchore vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Anchore vulnerability report in JSON format.

- **Username for basic authentication**.

## 7.4.58 Security warnings from Anchore Jenkins plugin

*Anchore Jenkins plugin* can be used to measure *security warnings*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with an Anchore report generated by the Anchore plugin. For example, 'https://jenkins.example.org/job/anchore' or https://jenkins.example.org/job/anchore/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  See also:

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `Critical`, `High`, `Low`, `Medium`, `Negligible`, `Unknown`. The default value is: *all severities*.

- **Username for basic authentication**.

### 7.4.59 Security warnings from Bandit

*Bandit* can be used to measure *security warnings*.

#### Mandatory parameters

- **URL to a Bandit report in JSON format or to a zip with Bandit reports in JSON format**.

#### Optional parameters

- **Confidence levels**. If provided, only count security warnings with the selected confidence levels. This parameter is multiple choice. Possible confidence levels are: `high`, `low`, `medium`. The default value is: *all confidence levels*.

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `high`, `low`, `medium`. The default value is: *all severities*.

- **URL to a Bandit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Bandit report in JSON format.

- **Username for basic authentication**.

### 7.4.60 Security warnings from Cargo Audit

*Cargo Audit* can be used to measure *security warnings*.

#### Mandatory parameters

- **URL to a Cargo Audit report in JSON format or to a zip with Cargo Audit reports in JSON format**.

#### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a Cargo Audit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Cargo Audit report in JSON format.

- **Username for basic authentication**.

- **Warning types**. If provided, only count security warnings with the selected warning types. This parameter is multiple choice. Possible warning types are: `unsound`, `vulnerability`, `yanked`. The default value is: *all warning types*.

## 7.4.61 Security warnings from Checkmarx CxSAST

*Checkmarx CxSAST* can be used to measure *security warnings*.

### Mandatory parameters

- **Password for basic authentication**.

- **Project (name or id)**.

- **URL**. URL of the Checkmarx instance, with port if necessary, but without path. For example 'https://checkmarx.example.org'.

- **Username for basic authentication**.

### Optional parameters

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `high`, `info`, `low`, `medium`. The default value is: *all severities*.

## 7.4.62 Security warnings from JSON file with security warnings

*JSON file with security warnings* can be used to measure *security warnings*.

### Mandatory parameters

- **URL to a generic vulnerability report in JSON format or to a zip with generic vulnerability reports in JSON format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `high`, `low`, `medium`. The default value is: *all severities*.

- **URL to a generic vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the generic vulnerability report in JSON format.

- **Username for basic authentication**.

## 7.4.63 Security warnings from Harbor

*Harbor* can be used to measure *security warnings*.

**Mandatory parameters**

- **URL**. URL of the Harbor instance, with port if necessary, but without path. For example 'https://demo.goharbor.io'.

**Optional parameters**

- **Password for basic authentication**.

- **Projects to ignore (regular expressions or project names)**. Projects to ignore can be specified by project name or by regular expression.

- **Projects to include (regular expressions or project names)**. Projects to include can be specified by project name or by regular expression.

- **Repositories to ignore (regular expressions or repository names)**. Repositories to ignore can be specified by repository name or by regular expression.

- **Repositories to include (regular expressions or repository names)**. Repositories to include can be specified by repository name or by regular expression.

- **Username for basic authentication**.

### 7.4.64  Security warnings from Harbor JSON

*Harbor JSON* can be used to measure *security warnings*.

**Mandatory parameters**

- **URL to a Harbor vulnerability report in JSON format or to a zip with Harbor vulnerability reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `critical`, `high`, `low`, `medium`. The default value is: *all severities*.

- **URL to a Harbor vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Harbor vulnerability report in JSON format.

- **Username for basic authentication**.

## 7.4.65 Security warnings from Manual number

*Manual number* can be used to measure *security warnings*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.66 Security warnings from OpenVAS

*OpenVAS* can be used to measure *security warnings*.

### Mandatory parameters

- **URL to an OpenVAS report in XML format or to a zip with OpenVAS reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `high`, `log`, `low`, `medium`. The default value is: *all severities*.

- **URL to an OpenVAS report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OpenVAS report in XML format.

- **Username for basic authentication**.

## 7.4.67 Security warnings from OWASP Dependency-Check

*OWASP Dependency-Check* can be used to measure *security warnings*.

### Mandatory parameters

- **URL to an OWASP Dependency-Check report in XML format or to a zip with OWASP Dependency-Check reports in XML format**.

### Optional parameters

- **Parts of file paths to ignore (regular expressions)**. Parts of file paths to ignore can be specified by regular expression. The parts of file paths that match one or more of the regular expressions are removed. If, after applying the regular expressions, multiple warnings are the same only one is reported.

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `critical`, `high`, `low`, `medium`, `moderate`. The default value is: *all severities*.

- **URL to an OWASP Dependency-Check report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP Dependency-Check report in XML format.

- **Username for basic authentication**.

### 7.4.68  Security warnings from OWASP ZAP

*OWASP ZAP* can be used to measure *security warnings*.

#### Mandatory parameters

- **URL to an OWASP ZAP report in XML format or to a zip with OWASP ZAP reports in XML format**.

#### Optional parameters

- **Count alert types or alert instances**. Determine whether to count each alert type in the OWASP ZAP report as a security warning or to count each alert instance (URL). This parameter is single choice. Possible count alert types or instances setting are: `alert instances`, `alert types`. The default value is: `alert instances`.

- **Parts of URLs to ignore (regular expressions)**. Parts of URLs to ignore can be specified by regular expression. The parts of URLs that match one or more of the regular expressions are removed. If, after applying the regular expressions, multiple warnings are the same only one is reported.

- **Password for basic authentication**.

- **Private token**.

- **Risks**. If provided, only count security warnings with the selected risks. This parameter is multiple choice. Possible risks are: `high`, `informational`, `low`, `medium`. The default value is: *all risks*.

- **URL to an OWASP ZAP report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP ZAP report in XML format.

- **Username for basic authentication**.

### 7.4.69  Security warnings from Pyupio Safety

*Pyupio Safety* can be used to measure *security warnings*.

#### Mandatory parameters

- **URL to a Safety report in JSON format or to a zip with Safety reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Safety report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Safety report in JSON format.

- **Username for basic authentication**.

## 7.4.70 Security warnings from SARIF

*SARIF* can be used to measure *security warnings*.

**Mandatory parameters**

- **URL to a SARIF vulnerability report in JSON format or to a zip with SARIF vulnerability reports in JSON format**.

**Optional parameters**

- **Levels**. If provided, only count entities with the selected levels. This parameter is multiple choice. Possible levels are: `error`, `none`, `note`, `warning`. The default value is: *all levels*.
- **Password for basic authentication**.
- **Private token**.
- **URL to a SARIF vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the SARIF vulnerability report in JSON format.
- **Username for basic authentication**.

## 7.4.71 Security warnings from Snyk

*Snyk* can be used to measure *security warnings*.

**Mandatory parameters**

- **URL to a Snyk vulnerability report in JSON format or to a zip with Snyk vulnerability reports in JSON format**.

## Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Severities**. If provided, only count security warnings with the selected severities. This parameter is multiple choice. Possible severities are: `high`, `low`, `medium`. The default value is: *all severities*.

- **URL to a Snyk vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Snyk vulnerability report in JSON format.

- **Username for basic authentication**.

## 7.4.72 Security warnings from SonarQube

*SonarQube* can be used to measure *security warnings*.

## Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

## Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  See also:

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  See also:

  https://docs.sonarqube.org/latest/user-guide/user-token/

- **Security hotspot review priorities**. This parameter is multiple choice. Possible review priorities are: `high`, `low`, `medium`. The default value is: *all review priorities*.

  See also:

  https://docs.sonarqube.org/latest/user-guide/security-hotspots/

- **Security hotspot statuses**. This parameter is multiple choice. Possible hotspot statuses are: `acknowledged`, `fixed`, `safe`, `to review`. The default value is: `acknowledged`, `to review`.

  See also:

  https://docs.sonarqube.org/latest/user-guide/security-hotspots/

- **Security issue types (measuring security hotspots requires SonarQube 8.2 or newer)**. This parameter is multiple choice. Possible types are: `security_hotspot`, `vulnerability`. The default value is: `vulnerability`.

  See also:

  https://docs.sonarqube.org/latest/user-guide/rules/

- **Severities**. This parameter is multiple choice. Possible severities are: `blocker`, `critical`, `info`, `major`, `minor`. The default value is: *all severities*.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/issues/

- **Tags to include**.

  **See also:**

  https://docs.sonarsource.com/sonarqube/latest/user-guide/issues/

## 7.4.73 Security warnings from Trivy JSON

*Trivy JSON* can be used to measure *security warnings*.

### Mandatory parameters

- **URL to a Trivy vulnerability report in JSON format or to a zip with Trivy vulnerability reports in JSON format**.

### Optional parameters

- **Levels**. If provided, only count security warnings with the selected levels. This parameter is multiple choice. Possible levels are: `critical`, `high`, `low`, `medium`, `unknown`. The default value is: *all levels*.
- **Password for basic authentication**.
- **Private token**.
- **URL to a Trivy vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Trivy vulnerability report in JSON format.
- **Username for basic authentication**.

## 7.4.74 Sentiment from Manual number

*Manual number* can be used to measure *sentiment*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.75 Slow transactions from Gatling

*Gatling* can be used to measure *slow transactions*.

**Mandatory parameters**

- **URL to a Gatling report in HTML format or to a zip with Gatling reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Response time type to evaluate against the target response time**. Which response time type to compare with the target response time to determine slow transactions. This parameter is single choice. Possible response time types to evaluate are: `50th percentile`, `75th percentile`, `95th percentile`, `99th percentile`, `maximum`, `mean`, `minimum`. The default value is: `95th percentile`.

- **Target response time**. The response times of the transactions should be less than or equal to the target response time. The default value is: `1000`.

- **Transaction-specific target response times (regular expressions or transaction names:target response time)**. Transactions-specific target responses times (in milliseconds) can be specified by transaction name or by regular expression, separated from the target response time by a colon, e.g.: '/api/v?/search/.*:1500'.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **Username for basic authentication**.

### 7.4.76 Slow transactions from Manual number

*Manual number* can be used to measure *slow transactions*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

### 7.4.77 Slow transactions from JMeter CSV

*JMeter CSV* can be used to measure *slow transactions*.

**Mandatory parameters**

- **URL to a JMeter report in CSV format or to a zip with JMeter reports in CSV format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Response time type to evaluate against the target response time**. Which response time type to compare with the target response time to determine slow transactions. This parameter is single choice. Possible response time types to evaluate are: `90th percentile`, `95th percentile`, `99th percentile`, `maximum`, `mean`, `median`, `minimum`. The default value is: `90th percentile`.

- **Target response time**. The response times of the transactions should be less than or equal to the target response time. The default value is: `1000`.

- **Transaction-specific target response times (regular expressions or transaction names:target response time)**. Transactions-specific target responses times (in milliseconds) can be specified by transaction name or by regular expression, separated from the target response time by a colon, e.g.: '/api/v?/search/.*:1500'.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in CSV format.

- **Username for basic authentication**.

## 7.4.78 Slow transactions from JMeter JSON

*JMeter JSON* can be used to measure *slow transactions*.

**Mandatory parameters**

- **URL to a JMeter report in JSON format or to a zip with JMeter reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Response time type to evaluate against the target response time**. Which response time type to compare with the target response time to determine slow transactions. This parameter is single choice. Possible response time types to evaluate are: `90th percentile`, `95th percentile`, `99th percentile`, `maximum`, `mean`, `median`, `minimum`. The default value is: `90th percentile`.

- **Target response time**. The response times of the transactions should be less than or equal to the target response time. The default value is: `1000`.

- **Transaction-specific target response times (regular expressions or transaction names:target response time)**. Transactions-specific target responses times (in milliseconds) can be specified by transaction name or by regular expression, separated from the target response time by a colon, e.g.: '/api/v?/search/.*:1500'.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in JSON format.

- **Username for basic authentication**.

## 7.4.79 Slow transactions from Performancetest-runner

*Performancetest-runner* can be used to measure *slow transactions*.

### Mandatory parameters

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Thresholds**. If provided, only count transactions that surpass the selected thresholds. This parameter is multiple choice. Possible thresholds are: `high`, `warning`. The default value is: *all thresholds*.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **Username for basic authentication**.

## 7.4.80 Software version from Performancetest-runner

*Performancetest-runner* can be used to measure *software version*.

### Mandatory parameters

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Username for basic authentication**.

## 7.4.81  Software version from SonarQube

*SonarQube* can be used to measure *software version*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path.  For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  See also:

  https://docs.sonarqube.org/latest/branches/overview/
- **Private token**.

  See also:

  https://docs.sonarqube.org/latest/user-guide/user-token/

## 7.4.82  Source up-to-dateness from Anchore

*Anchore* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **URL to an Anchore details report in JSON format or to a zip with Anchore details reports in JSON format**.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to an Anchore vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Anchore vulnerability report in JSON format.
- **Username for basic authentication**.

### 7.4.83  Source up-to-dateness from Anchore Jenkins plugin

*Anchore Jenkins plugin* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

#### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with an Anchore report generated by the Anchore plugin. For example, 'https://jenkins.example.org/job/anchore' or https://jenkins.example.org/job/anchore/job/main' in case of a pipeline job.

#### Optional parameters

- **Password or API token for basic authentication**.

  See also:

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.84  Source up-to-dateness from Axe-core

*Axe-core* can be used to measure *source up-to-dateness*.

When running Axe-core on a webpage, the run function returns a results object. The results objects may be stored in separate JSON files and served to *Quality-time* in a zipfile, or the results objects can be combined in one JSON file that contains a list of results objects.

Axe-core adds a `timestamp` field to each results object. That field is used by *Quality-time* to determine the up-to-dateness of the report. If there is more than one results object in the JSON file, *Quality-time* uses the first one it encounters, assuming that all timestamps in one JSON file will be roughly equal.

**Tip:** When combining results objects, make sure the `timestamp` field is retained in the JSON.

#### Mandatory parameters

- **URL to an Axe-core report in JSON format or to a zip with Axe-core reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to an Axe-core report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Axe-core report in JSON format.

- **Username for basic authentication**.

### 7.4.85 Source up-to-dateness from Azure DevOps Server

*Azure DevOps Server* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

**Optional parameters**

- **Branch**. The default value is: `master`.

- **File or folder path**. Use the date and time the path was last changed to determine the up-to-dateness. If no path is specified, the pipeline is used to determine the up-to-dateness.

- **Pipelines to ignore (regular expressions or pipeline names)**. Pipelines to ignore can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Pipelines to include (regular expressions or pipeline names)**. Pipelines to include can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.

- **Private token**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

- **Repository (name or id)**.

### 7.4.86 Source up-to-dateness from Bandit

*Bandit* can be used to measure *source up-to-dateness*.

---

**Mandatory parameters**

- **URL to a Bandit report in JSON format or to a zip with Bandit reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Bandit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Bandit report in JSON format.

- **Username for basic authentication**.

### 7.4.87 Source up-to-dateness from Calendar date

*Calendar date* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **Date**. The default value is: `2024-01-31`.

### 7.4.88 Source up-to-dateness from Cobertura

*Cobertura* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a Cobertura report in XML format or to a zip with Cobertura reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Cobertura report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Cobertura report in XML format.

- **Username for basic authentication**.

## 7.4.89 Source up-to-dateness from Cobertura Jenkins plugin

*Cobertura Jenkins plugin* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the Cobertura plugin. For example, 'https://jenkins.example.org/job/cobertura' or https://jenkins.example.org/job/cobertura/job/main' in case of a pipeline job.

### Optional parameters

- **Password or API token for basic authentication**.

   **See also:**

   https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.90 Source up-to-dateness from Checkmarx CxSAST

*Checkmarx CxSAST* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **Password for basic authentication**.
- **Project (name or id)**.
- **URL**. URL of the Checkmarx instance, with port if necessary, but without path. For example 'https://checkmarx.example.org'.
- **Username for basic authentication**.

## 7.4.91 Source up-to-dateness from Gatling

*Gatling* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a Gatling report in HTML format or to a zip with Gatling reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

## 7.4.92 Source up-to-dateness from GitLab

*GitLab* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.
- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

**Optional parameters**

- **Branch**. The default value is: `master`.

  See also:

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **File or folder path**. Use the date and time the path was last changed to determine the up-to-dateness. If no path is specified, the pipeline is used to determine the up-to-dateness.
- **Number of days to look back for selecting pipeline jobs**. The default value is: `90`.
- **Pipeline statuses to include**. This parameter is multiple choice. Possible pipeline statuses are: `canceled`, `created`, `failed`, `manual`, `pending`, `preparing`, `running`, `scheduled`, `skipped`, `success`, `waiting for resource`. The default value is: *all pipeline statuses*.
- **Pipeline triggers to include**. This parameter is multiple choice. Possible pipeline triggers are: `api`, `chat`, `external pull request event`, `external`, `merge request event`, `ondemand DAST scan`, `ondemand DAST validation`, `parent pipeline`, `pipeline`, `push`, `schedule`, `trigger`, `web-IDE`, `web`. The default value is: *all pipeline triggers*.
- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

## 7.4.93 Source up-to-dateness from JaCoCo

*JaCoCo* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a JaCoCo report in XML format or to a zip with JaCoCo reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **URL to a JaCoCo report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JaCoCo report in XML format.
- **Username for basic authentication**.

## 7.4.94 Source up-to-dateness from JaCoCo Jenkins plugin

*JaCoCo Jenkins plugin* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the JaCoCo plugin. For example, 'https://jenkins.example.org/job/jacoco' or https://jenkins.example.org/job/jacoco/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.95 Source up-to-dateness from Jenkins

*Jenkins* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL**. URL of the Jenkins instance, with port if necessary, but without path. For example, 'https://jenkins.example.org'.

**Optional parameters**

- **Build result types**. Limit which build result types to include. This parameter is multiple choice. Possible result types are: `Aborted`, `Failure`, `Not built`, `Success`, `Unstable`. The default value is: *all result types*.

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Jobs to include (regular expressions or job names)**. Jobs to include can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.96 Source up-to-dateness from Jenkins test report

*Jenkins test report* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with a test report generated by the JUnit plugin. For example, 'https://jenkins.example.org/job/test' or https://jenkins.example.org/job/test/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.97 Source up-to-dateness from JMeter CSV

*JMeter CSV* can be used to measure *source up-to-dateness*.

#### Mandatory parameters

- **URL to a JMeter report in CSV format or to a zip with JMeter reports in CSV format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in CSV format.
- **Username for basic authentication**.

### 7.4.98 Source up-to-dateness from JUnit XML report

*JUnit XML report* can be used to measure *source up-to-dateness*.

#### Mandatory parameters

- **URL to a JUnit report in XML format or to a zip with JUnit reports in XML format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a JUnit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JUnit report in XML format.
- **Username for basic authentication**.

### 7.4.99 Source up-to-dateness from NCover

*NCover* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a NCover report in HTML format or to a zip with NCover reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.100  Source up-to-dateness from Robot Framework

*Robot Framework* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a Robot Framework report in XML format or to a zip with Robot Framework reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **URL to a Robot Framework report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Robot Framework report in XML format.
- **Username for basic authentication**.

### 7.4.101  Source up-to-dateness from OpenVAS

*OpenVAS* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to an OpenVAS report in XML format or to a zip with OpenVAS reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **URL to an OpenVAS report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OpenVAS report in XML format.
- **Username for basic authentication**.

## 7.4.102 Source up-to-dateness from OWASP Dependency-Check

*OWASP Dependency-Check* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **URL to an OWASP Dependency-Check report in XML format or to a zip with OWASP Dependency-Check reports in XML format**.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to an OWASP Dependency-Check report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP Dependency-Check report in XML format.
- **Username for basic authentication**.

## 7.4.103 Source up-to-dateness from OWASP ZAP

*OWASP ZAP* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **URL to an OWASP ZAP report in XML format or to a zip with OWASP ZAP reports in XML format**.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to an OWASP ZAP report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP ZAP report in XML format.
- **Username for basic authentication**.

## 7.4.104 Source up-to-dateness from Performancetest-runner

*Performancetest-runner* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.105 Source up-to-dateness from Quality-time

*Quality-time* can be used to measure *source up-to-dateness*.

**Mandatory parameters**

- **Quality-time URL**. URL of the Quality-time instance, with port if necessary, but without path. For example, 'https://quality-time.example.org'.

**Optional parameters**

- **Report names or identifiers**.

### 7.4.106 Source up-to-dateness from Robot Framework Jenkins plugin

*Robot Framework Jenkins plugin* can be used to measure *source up-to-dateness*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with a test report generated by the Robot Framework plugin. For example, 'https://jenkins.example.org/job/robot' or https://jenkins.example.org/job/robot/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.107 Source up-to-dateness from SonarQube

*SonarQube* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

## 7.4.108 Source up-to-dateness from TestNG

*TestNG* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **URL to a TestNG report in XML format or to a zip with TestNG reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a TestNG report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the TestNG report in XML format.

- **Username for basic authentication**.

## 7.4.109 Source up-to-dateness from Trello

*Trello* can be used to measure *source up-to-dateness*.

### Mandatory parameters

- **Board (title or id)**.

  **See also:**

  https://trello.com/1/members/me/boards?fields=name

- **URL**. The default value is: `https://trello.com`.

### Optional parameters

- **API key**.

  **See also:**

  https://trello.com/app-key

- **Lists to ignore (title or id)**.

- **Token**.

  **See also:**

  https://trello.com/app-key

## 7.4.110 Source version from Axe-core

*Axe-core* can be used to measure *source version*.

When running Axe-core on a webpage, the run function returns a results object. The results objects may be stored in separate JSON files and served to *Quality-time* in a zipfile, or the results objects can be combined in one JSON file that contains a list of results objects.

Axe-core adds a `testEngine` field to each results object. That field is used by *Quality-time* to determine the version of Axe-core used to generate the report. If there is more than one results object in the JSON file, *Quality-time* uses the first one it encounters, assuming that all test engines used in one JSON file will be equal.

**Tip:** When combining results objects, make sure the `testEngine` field is retained in the JSON.

### Mandatory parameters

- **URL to an Axe-core report in JSON format or to a zip with Axe-core reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to an Axe-core report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Axe-core report in JSON format.

- **Username for basic authentication**.

## 7.4.111 Source version from cloc

*cloc* can be used to measure *source version*.

**Mandatory parameters**

- **URL to a cloc report in JSON format or to a zip with cloc reports in JSON format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a cloc report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the cloc report in JSON format.

- **Username for basic authentication**.

## 7.4.112 Source version from Cobertura

*Cobertura* can be used to measure *source version*.

**Mandatory parameters**

- **URL to a Cobertura report in XML format or to a zip with Cobertura reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Cobertura report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Cobertura report in XML format.

- **Username for basic authentication**.

### 7.4.113 Source version from Checkmarx CxSAST

*Checkmarx CxSAST* can be used to measure *source version*.

#### Mandatory parameters

- **Password for basic authentication**.
- **URL**. URL of the Checkmarx instance, with port if necessary, but without path. For example 'https://checkmarx.example.org'.
- **Username for basic authentication**.

### 7.4.114 Source version from Gatling

*Gatling* can be used to measure *source version*.

#### Mandatory parameters

- **URL to a Gatling report in HTML format or to a zip with Gatling reports in HTML format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

### 7.4.115 Source version from GitLab

*GitLab* can be used to measure *source version*.

#### Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

#### Optional parameters

- **Private token (with read_api scope)**.

  See also:

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

## 7.4.116  Source version from Jenkins

*Jenkins* can be used to measure *source version*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

### Mandatory parameters

- **URL**. URL of the Jenkins instance, with port if necessary, but without path.  For example, 'https://jenkins.example.org'.

### Optional parameters

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.117  Source version from Jira

*Jira* can be used to measure *source version*.

### Mandatory parameters

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

### Optional parameters

- **Password for basic authentication**.
- **Private token**.

  **See also:**

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Username for basic authentication**.

## 7.4.118  Source version from OpenVAS

*OpenVAS* can be used to measure *source version*.

**Mandatory parameters**

- **URL to an OpenVAS report in XML format or to a zip with OpenVAS reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **URL to an OpenVAS report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OpenVAS report in XML format.
- **Username for basic authentication**.

### 7.4.119 Source version from OWASP Dependency-Check

*OWASP Dependency-Check* can be used to measure *source version*.

**Mandatory parameters**

- **URL to an OWASP Dependency-Check report in XML format or to a zip with OWASP Dependency-Check reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **URL to an OWASP Dependency-Check report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP Dependency-Check report in XML format.
- **Username for basic authentication**.

### 7.4.120 Source version from OWASP ZAP

*OWASP ZAP* can be used to measure *source version*.

**Mandatory parameters**

- **URL to an OWASP ZAP report in XML format or to a zip with OWASP ZAP reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to an OWASP ZAP report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OWASP ZAP report in XML format.

- **Username for basic authentication**.

### 7.4.121 Source version from Quality-time

*Quality-time* can be used to measure *source version*.

**Mandatory parameters**

- **Quality-time URL**. URL of the Quality-time instance, with port if necessary, but without path. For example, 'https://quality-time.example.org'.

### 7.4.122 Source version from Robot Framework

*Robot Framework* can be used to measure *source version*.

**Mandatory parameters**

- **URL to a Robot Framework report in XML format or to a zip with Robot Framework reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Robot Framework report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Robot Framework report in XML format.

- **Username for basic authentication**.

### 7.4.123 Source version from SonarQube

*SonarQube* can be used to measure *source version*.

---

**Mandatory parameters**

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Private token**.

  See also:

  https://docs.sonarqube.org/latest/user-guide/user-token/

### 7.4.124 Suppressed violations from Manual number

*Manual number* can be used to measure *suppressed violations*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

### 7.4.125 Suppressed violations from SonarQube

*SonarQube* can be used to measure *suppressed violations*.

**Mandatory parameters**

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  See also:

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  See also:

  https://docs.sonarqube.org/latest/user-guide/user-token/

- **Severities**. This parameter is multiple choice. Possible severities are: `blocker`, `critical`, `info`, `major`, `minor`. The default value is: *all severities*.

  See also:

  https://docs.sonarqube.org/latest/user-guide/issues/

- **Types**. This parameter is multiple choice. Possible types are: `bug`, `code_smell`, `vulnerability`. The default value is: *all types*.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/rules/

Configurations:

- Rules used to detect suppressed violations:

  - c:S1291

  - cpp:S1291

  - csharpsquid:S1309

  - java:NoSonar

  - java:S1309

  - java:S1310

  - java:S1315

  - objc:S1291

  - php:NoSonar

  - plsql:NoSonarCheck

  - python:NoSonar

  - tsql:NoSonar

## 7.4.126 Test cases from Jenkins test report

*Jenkins test report* can be used to measure *test cases*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a test report generated by the JUnit plugin. For example, 'https://jenkins.example.org/job/test' or https://jenkins.example.org/job/test/job/main' in case of a pipeline job.

### Optional parameters

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.127 Test cases from Jira

*Jira* can be used to measure *test cases*.

### Mandatory parameters

- **Issue query in JQL (Jira Query Language)**.

  **See also:**

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

  **See also:**

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `errored`, `failed`, `passed`, `skipped`, `untested`. The default value is: *all test results*.

- **Username for basic authentication**.

## 7.4.128 Test cases from JUnit XML report

*JUnit XML report* can be used to measure *test cases*.

### Mandatory parameters

- **URL to a JUnit report in XML format or to a zip with JUnit reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a JUnit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JUnit report in XML format.

- **Username for basic authentication**.

## 7.4.129 Test cases from Manual number

*Manual number* can be used to measure *test cases*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.130 Test cases from Robot Framework

*Robot Framework* can be used to measure *test cases*.

### Mandatory parameters

- **URL to a Robot Framework report in XML format or to a zip with Robot Framework reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a Robot Framework report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Robot Framework report in XML format.

- **Username for basic authentication**.

## 7.4.131 Test cases from TestNG

*TestNG* can be used to measure *test cases*.

### Mandatory parameters

- **URL to a TestNG report in XML format or to a zip with TestNG reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a TestNG report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the TestNG report in XML format.

- **Username for basic authentication**.

## 7.4.132 Tests from Azure DevOps Server

*Azure DevOps Server* can be used to measure *tests*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Names of test runs to include (regular expressions or test run names)**. Limit which test runs to include by test run name.

- **Private token**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

- **States of the test runs to include**. Limit which test runs to include by test run state. This parameter is multiple choice. Possible test run states are: `aborted`, `completed`, `in progress`, `not started`. The default value is: *all test run states*.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `incomplete`, `not applicable`, `passed`. The default value is: *all test results*.

## 7.4.133 Tests from Gatling

*Gatling* can be used to measure *tests*.

### Mandatory parameters

- **URL to a Gatling report in HTML format or to a zip with Gatling reports in HTML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `success`. The default value is: *all test results*.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **Username for basic authentication**.

## 7.4.134  Tests from Jenkins test report

*Jenkins test report* can be used to measure *tests*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a test report generated by the JUnit plugin. For example, 'https://jenkins.example.org/job/test' or https://jenkins.example.org/job/test/job/main' in case of a pipeline job.

### Optional parameters

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `passed`, `skipped`. The default value is: *all test results*.

- **Username for basic authentication**.

## 7.4.135  Tests from JMeter CSV

*JMeter CSV* can be used to measure *tests*.

### Mandatory parameters

- **URL to a JMeter report in CSV format or to a zip with JMeter reports in CSV format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `success`. The default value is: *all test results*.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in CSV format.

- **Username for basic authentication**.

## 7.4.136 Tests from JMeter JSON

*JMeter JSON* can be used to measure *tests*.

### Mandatory parameters

- **URL to a JMeter report in JSON format or to a zip with JMeter reports in JSON format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `success`. The default value is: *all test results*.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **URL to a JMeter report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JMeter report in JSON format.

- **Username for basic authentication**.

## 7.4.137 Tests from JUnit XML report

*JUnit XML report* can be used to measure *tests*.

### Mandatory parameters

- **URL to a JUnit report in XML format or to a zip with JUnit reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `errored`, `failed`, `passed`, `skipped`. The default value is: *all test results*.

- **URL to a JUnit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JUnit report in XML format.

- **Username for basic authentication**.

### 7.4.138 Tests from Manual number

*Manual number* can be used to measure *tests*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

### 7.4.139 Tests from Performancetest-runner

*Performancetest-runner* can be used to measure *tests*.

**Mandatory parameters**

- **URL to a Performancetest-runner report in HTML format or to a zip with Performancetest-runner reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `success`. The default value is: *all test results*.

- **Transactions to ignore (regular expressions or transaction names)**. Transactions to ignore can be specified by transaction name or by regular expression.

- **Transactions to include (regular expressions or transaction names)**. Transactions to include can be specified by transaction name or by regular expression.

- **Username for basic authentication**.

## 7.4.140  Tests from Robot Framework

*Robot Framework* can be used to measure *tests*.

### Mandatory parameters

- **URL to a Robot Framework report in XML format or to a zip with Robot Framework reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `fail`, `pass`, `skip`. The default value is: *all test results*.

- **URL to a Robot Framework report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Robot Framework report in XML format.

- **Username for basic authentication**.

## 7.4.141  Tests from Robot Framework Jenkins plugin

*Robot Framework Jenkins plugin* can be used to measure *tests*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a test report generated by the Robot Framework plugin. For example, 'https://jenkins.example.org/job/robot' or https://jenkins.example.org/job/robot/job/main' in case of a pipeline job.

### Optional parameters

- **Password or API token for basic authentication**.

  See also:

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `fail`, `pass`. The default value is: *all test results*.

- **Username for basic authentication**.

## 7.4.142 Tests from SonarQube

*SonarQube* can be used to measure *tests*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `errored`, `failed`, `passed`, `skipped`. The default value is: *all test results*.

## 7.4.143 Tests from TestNG

*TestNG* can be used to measure *tests*.

### Mandatory parameters

- **URL to a TestNG report in XML format or to a zip with TestNG reports in XML format**.

### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **Test results**. Limit which test results to count. Note: depending on which results are selected, the direction of the metric may need to be adapted. For example, when counting passed tests, more is better, but when counting failed tests, fewer is better. This parameter is multiple choice. Possible test results are: `failed`, `ignored`, `passed`, `skipped`. The default value is: *all test results*.

- **URL to a TestNG report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the TestNG report in XML format.

- **Username for basic authentication**.

## 7.4.144 Time remaining from Calendar date

*Calendar date* can be used to measure *time remaining*.

### Mandatory parameters

- **Date**. The default value is: `2024-01-31`.

## 7.4.145 Todo and fixme comments from Manual number

*Manual number* can be used to measure *todo and fixme comments*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.146 Todo and fixme comments from SonarQube

*SonarQube* can be used to measure *todo and fixme comments*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  See also:

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  See also:

  https://docs.sonarqube.org/latest/user-guide/user-token/

Configurations:

- Rules used to detect todo and fixme comments:

  - c:S1134
  - c:S1135
  - cloudformation:S1135
  - cpp:S1134
  - cpp:S1135

- csharpsquid:S1134
- csharpsquid:S1135
- docker:S1135
- go:S1134
- go:S1135
- java:S1134
- java:S1135
- javascript:S1134
- javascript:S1135
- kotlin:S1134
- kotlin:S1135
- kubernetes:S1135
- objc:S1134
- objc:S1135
- php:S1134
- php:S1135
- plsql:S1134
- plsql:S1135
- python:S1134
- python:S1135
- ruby:S1134
- ruby:S1135
- scala:S1134
- scala:S1135
- swift:S1134
- swift:S1135
- terraform:S1135
- tsql:S1134
- tsql:S1135
- typescript:S1134
- typescript:S1135
- vbnet:S1134
- vbnet:S1135
- Web:S1134
- Web:S1135
- xml:S1134

– xml:S1135

### 7.4.147 Test branch coverage from Cobertura

*Cobertura* can be used to measure *test branch coverage*.

#### Mandatory parameters

- **URL to a Cobertura report in XML format or to a zip with Cobertura reports in XML format**.

#### Optional parameters

- **Password for basic authentication**.

- **Private token**.

- **URL to a Cobertura report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Cobertura report in XML format.

- **Username for basic authentication**.

### 7.4.148 Test branch coverage from Cobertura Jenkins plugin

*Cobertura Jenkins plugin* can be used to measure *test branch coverage*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

#### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the Cobertura plugin. For example, 'https://jenkins.example.org/job/cobertura' or https://jenkins.example.org/job/cobertura/job/main' in case of a pipeline job.

#### Optional parameters

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.149 Test branch coverage from JaCoCo

*JaCoCo* can be used to measure *test branch coverage*.

#### Mandatory parameters

- **URL to a JaCoCo report in XML format or to a zip with JaCoCo reports in XML format**.

#### Optional parameters

- **Password for basic authentication**.
- **Private token**.
- **URL to a JaCoCo report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JaCoCo report in XML format.
- **Username for basic authentication**.

### 7.4.150 Test branch coverage from JaCoCo Jenkins plugin

*JaCoCo Jenkins plugin* can be used to measure *test branch coverage*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

#### Mandatory parameters

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the JaCoCo plugin. For example, 'https://jenkins.example.org/job/jacoco' or https://jenkins.example.org/job/jacoco/job/main' in case of a pipeline job.

#### Optional parameters

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/
- **Username for basic authentication**.

### 7.4.151 Test branch coverage from Manual number

*Manual number* can be used to measure *test branch coverage*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

## 7.4.152  Test branch coverage from NCover

*NCover* can be used to measure *test branch coverage*.

**Mandatory parameters**

- **URL to a NCover report in HTML format or to a zip with NCover reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

## 7.4.153  Test branch coverage from SonarQube

*SonarQube* can be used to measure *test branch coverage*.

**Mandatory parameters**

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path.  For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

## 7.4.154  Test line coverage from Cobertura

*Cobertura* can be used to measure *test line coverage*.

**Mandatory parameters**

- **URL to a Cobertura report in XML format or to a zip with Cobertura reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a Cobertura report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Cobertura report in XML format.

- **Username for basic authentication**.

## 7.4.155  Test line coverage from Cobertura Jenkins plugin

*Cobertura Jenkins plugin* can be used to measure *test line coverage*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the Cobertura plugin. For example, 'https://jenkins.example.org/job/cobertura' or https://jenkins.example.org/job/cobertura/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.156  Test line coverage from JaCoCo

*JaCoCo* can be used to measure *test line coverage*.

**Mandatory parameters**

- **URL to a JaCoCo report in XML format or to a zip with JaCoCo reports in XML format**.

**Optional parameters**

- **Password for basic authentication**.

- **Private token**.

- **URL to a JaCoCo report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the JaCoCo report in XML format.

- **Username for basic authentication**.

### 7.4.157 Test line coverage from JaCoCo Jenkins plugin

*JaCoCo Jenkins plugin* can be used to measure *test line coverage*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL to Jenkins job**. URL to a Jenkins job with a coverage report generated by the JaCoCo plugin. For example, 'https://jenkins.example.org/job/jacoco' or https://jenkins.example.org/job/jacoco/job/main' in case of a pipeline job.

**Optional parameters**

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

### 7.4.158 Test line coverage from Manual number

*Manual number* can be used to measure *test line coverage*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

## 7.4.159 Test line coverage from NCover

*NCover* can be used to measure *test line coverage*.

**Mandatory parameters**

- **URL to a NCover report in HTML format or to a zip with NCover reports in HTML format**.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.
- **Username for basic authentication**.

## 7.4.160 Test line coverage from SonarQube

*SonarQube* can be used to measure *test line coverage*.

**Mandatory parameters**

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.
- **URL**. URL of the SonarQube instance, with port if necessary, but without path. For example, 'https://sonarcloud.io'.

**Optional parameters**

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

## 7.4.161 Unmerged branches from Azure DevOps Server

*Azure DevOps Server* can be used to measure *unmerged branches*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Branches to ignore (regular expressions or branch names)**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/repos/git/manage-your-branches?view=azure-devops

- **Number of days since last commit after which to consider branches inactive**. The default value is: 7.

- **Private token**.

  **See also:**

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

- **Repository (name or id)**.

## 7.4.162 Unmerged branches from GitLab

*GitLab* can be used to measure *unmerged branches*.

### Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.

- **Project (name with namespace or id)**.

  **See also:**

  https://docs.gitlab.com/ee/user/project/

### Optional parameters

- **Branches to ignore (regular expressions or branch names)**.

  **See also:**

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **Number of days since last commit after which to consider branches inactive**. The default value is: 7.

- **Private token (with read_api scope)**.

  **See also:**

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

## 7.4.163 Unmerged branches from Manual number

*Manual number* can be used to measure *unmerged branches*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.164 Unused CI-jobs from Azure DevOps Server

*Azure DevOps Server* can be used to measure *unused ci-jobs*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Number of days since last build after which to consider pipelines inactive**. The default value is: `21`.
- **Pipelines to ignore (regular expressions or pipeline names)**. Pipelines to ignore can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.
- **Pipelines to include (regular expressions or pipeline names)**. Pipelines to include can be specified by pipeline name or by regular expression. Use {folder name}/{pipeline name} for the names of pipelines in folders.
- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.165 Unused CI-jobs from GitLab

*GitLab* can be used to measure *unused ci-jobs*.

### Mandatory parameters

- **GitLab instance URL**. URL of the GitLab instance, with port if necessary, but without path. For example, 'https://gitlab.com'.
- **Project (name with namespace or id)**.

  See also:

  https://docs.gitlab.com/ee/user/project/

**Optional parameters**

- **Branches and tags to ignore (regular expressions, branch names or tag names)**.

  **See also:**

  https://docs.gitlab.com/ee/user/project/repository/branches/

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression.

- **Number of days to look back for selecting pipeline jobs**. The default value is: 90.

- **Number of days without builds after which to consider CI-jobs unused**. The default value is: 90.

- **Private token (with read_api scope)**.

  **See also:**

  https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

### 7.4.166 Unused CI-jobs from Jenkins

*Jenkins* can be used to measure *unused ci-jobs*.

To authorize *Quality-time* for (non-public resources in) Jenkins, you can either use a username and password or a username and API token. Note that, unlike other sources, when using the API token Jenkins also requires the username to which the token belongs.

**Mandatory parameters**

- **URL**. URL of the Jenkins instance, with port if necessary, but without path. For example, 'https://jenkins.example.org'.

**Optional parameters**

- **Jobs to ignore (regular expressions or job names)**. Jobs to ignore can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Jobs to include (regular expressions or job names)**. Jobs to include can be specified by job name or by regular expression. Use {parent job name}/{child job name} for the names of nested jobs.

- **Number of days without builds after which to consider CI-jobs unused**. The default value is: 90.

- **Password or API token for basic authentication**.

  **See also:**

  https://www.jenkins.io/doc/book/system-administration/authenticating-scripted-clients/

- **Username for basic authentication**.

## 7.4.167 Unused CI-jobs from Manual number

*Manual number* can be used to measure *unused ci-jobs*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.168 User story points from Azure DevOps Server

*Azure DevOps Server* can be used to measure *user story points*.

### Mandatory parameters

- **URL including organization and project**. URL of the Azure DevOps instance, with port if necessary, and with organization and project. For example: 'https://dev.azure.com/{organization}/{project}'.

### Optional parameters

- **Issue query in WIQL (Work Item Query Language)**. This should only contain the WHERE clause of a WIQL query, as the selected fields are static. For example, use the following clause to hide issues marked as done: "[System.State] <> 'Done'". See https://docs.microsoft.com/en-us/azure/devops/boards/queries/wiql-syntax?view=azure-devops.

- **Private token**.

  See also:

  https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops

## 7.4.169 User story points from Jira

*Jira* can be used to measure *user story points*.

### Mandatory parameters

- **Issue query in JQL (Jira Query Language)**.

  See also:

  https://support.atlassian.com/jira-software-cloud/docs/use-advanced-search-with-jira-query-language-jql/

- **Story points field (name or id)**. The default value is: `Story Points`.

  See also:

  https://confluence.atlassian.com/jirakb/how-to-find-id-for-custom-field-s-744522503.html

- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html
- **Username for basic authentication**.

### 7.4.170 User story points from Manual number

*Manual number* can be used to measure *user story points*.

**Mandatory parameters**

- **Number**. The default value is: `0`.

### 7.4.171 Velocity from Jira

*Jira* can be used to measure *velocity*.

**Mandatory parameters**

- **Board (name or id)**.

  See also:

  https://support.atlassian.com/jira-software-cloud/docs/what-is-a-jira-software-board/
- **Number of sprints to base velocity on**. Velocity is defined as the number of user story points realized per sprint. By default, velocity is calculated by averaging the number of user story points realized in the three most recently completed sprints. The number of sprints to use can be changed via this parameter. The default value is: `3`.
- **Type of velocity**. Whether to report the number of story points committed to, the number of story points actually completed, or the difference between the two. This parameter is single choice. Possible velocity type are: `committed points`, `completed points minus committed points`, `completed points`. The default value is: `completed points`.
- **URL**. URL of the Jira instance, with port if necessary. For example, 'https://jira.example.org'.

**Optional parameters**

- **Password for basic authentication**.
- **Private token**.

  See also:

  https://confluence.atlassian.com/enterprise/using-personal-access-tokens-1026032365.html
- **Username for basic authentication**.

## 7.4.172 Velocity from Manual number

*Manual number* can be used to measure *velocity*.

### Mandatory parameters

- **Number**. The default value is: `0`.

## 7.4.173 Violations from Axe CSV

*Axe CSV* can be used to measure *violations*.

### Mandatory parameters

- **URL to an Axe report in CSV format or to a zip with Axe reports in CSV format**.

### Optional parameters

- **Elements to exclude (regular expressions)**. Elements to exclude can be specified by regular expression.
- **Elements to include (regular expressions)**. Elements to include can be specified by regular expression.
- **Impact levels**. If provided, only count accessibility violations with the selected impact levels. This parameter is multiple choice. Possible impact levels are: `critical`, `minor`, `moderate`, `serious`. The default value is: *all impact levels*.
- **Password for basic authentication**.
- **Private token**.
- **URL to an Axe report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Axe report in CSV format.
- **Username for basic authentication**.

## 7.4.174 Violations from Axe-core

*Axe-core* can be used to measure *violations*.

When running Axe-core on a webpage, the run function returns a results object. The results objects may be stored in separate JSON files and served to *Quality-time* in a zipfile, or the results objects can be combined in one JSON file that contains a list of results objects.

**Mandatory parameters**

- **URL to an Axe-core report in JSON format or to a zip with Axe-core reports in JSON format**.

**Optional parameters**

- **Elements to exclude (regular expressions)**. Elements to exclude can be specified by regular expression.

- **Elements to include (regular expressions)**. Elements to include can be specified by regular expression.

- **Impact levels**. If provided, only count accessibility violations with the selected impact levels. This parameter is multiple choice. Possible impact levels are: `critical`, `minor`, `moderate`, `serious`. The default value is: *all impact levels*.

- **Password for basic authentication**.

- **Private token**.

- **Result types**. Limit which result types to count. This parameter is multiple choice. Possible result types are: `inapplicable`, `incomplete`, `passes`, `violations`. The default value is: `violations`.

- **Tags to ignore (regular expressions or tags)**. Tags to ignore can be specified by tag or by regular expression.

- **Tags to include (regular expressions or tags)**. Tags to include can be specified by tag or by regular expression.

- **URL to an Axe-core report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the Axe-core report in JSON format.

- **Username for basic authentication**.

### 7.4.175 Violations from Axe HTML reporter

*Axe HTML reporter* can be used to measure *violations*.

**Mandatory parameters**

- **URL to an Axe report in HTML format or to a zip with Axe reports in HTML format**.

**Optional parameters**

- **Elements to exclude (regular expressions)**. Elements to exclude can be specified by regular expression.

- **Elements to include (regular expressions)**. Elements to include can be specified by regular expression.

- **Impact levels**. If provided, only count accessibility violations with the selected impact levels. This parameter is multiple choice. Possible impact levels are: `critical`, `minor`, `moderate`, `serious`. The default value is: *all impact levels*.

- **Password for basic authentication**.

- **Private token**.

- **Result types**. Limit which result types to count. This parameter is multiple choice. Possible result types are: `inapplicable`, `incomplete`, `passes`, `violations`. The default value is: `violations`.

- **Tags to ignore (regular expressions or tags)**. Tags to ignore can be specified by tag or by regular expression.

- **Tags to include (regular expressions or tags)**. Tags to include can be specified by tag or by regular expression.

• **Username for basic authentication**.

## 7.4.176 Violations from Manual number

*Manual number* can be used to measure *violations*.

### Mandatory parameters

• **Number**. The default value is: `0`.

## 7.4.177 Violations from OJAudit

*OJAudit* can be used to measure *violations*.

### Mandatory parameters

• **URL to an OJAudit report in XML format or to a zip with OJAudit reports in XML format**.

### Optional parameters

• **Password for basic authentication**.

• **Private token**.

• **Severities**. If provided, only count violations with the selected severities. This parameter is multiple choice. Possible severities are: `advisory`, `error`, `exception`, `incomplete`, `warning`. The default value is: *all severities*.

• **URL to an OJAudit report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the OJAudit report in XML format.

• **Username for basic authentication**.

## 7.4.178 Violations from SARIF

*SARIF* can be used to measure *violations*.

### Mandatory parameters

• **URL to a SARIF vulnerability report in JSON format or to a zip with SARIF vulnerability reports in JSON format**.

**Optional parameters**

- **Levels**. If provided, only count entities with the selected levels. This parameter is multiple choice. Possible levels are: `error`, `none`, `note`, `warning`. The default value is: *all levels*.

- **Password for basic authentication**.

- **Private token**.

- **URL to a SARIF vulnerability report in a human readable format**. If provided, users clicking the source URL will visit this URL instead of the SARIF vulnerability report in JSON format.

- **Username for basic authentication**.

## 7.4.179  Violations from SonarQube

*SonarQube* can be used to measure *violations*.

### Mandatory parameters

- **Project key**. The project key can be found by opening the project in SonarQube and looking at the bottom of the grey column on the right.

- **URL**. URL of the SonarQube instance, with port if necessary, but without path.  For example, 'https://sonarcloud.io'.

### Optional parameters

- **Branch (only supported by commercial SonarQube editions)**. The default value is: `master`.

  **See also:**

  https://docs.sonarqube.org/latest/branches/overview/

- **Private token**.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/user-token/

- **Severities**. This parameter is multiple choice. Possible severities are: `blocker`, `critical`, `info`, `major`, `minor`. The default value is: *all severities*.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/issues/

- **Tags to include**.

  **See also:**

  https://docs.sonarsource.com/sonarqube/latest/user-guide/issues/

- **Types**. This parameter is multiple choice. Possible types are: `bug`, `code_smell`, `vulnerability`. The default value is: *all types*.

  **See also:**

  https://docs.sonarqube.org/latest/user-guide/rules/

# DEPLOYMENT INSTRUCTIONS

This document describes how to deploy, and if needed move, the *Quality-time* application. It is aimed at *Quality-time* operators.

*Quality-time* consists of a set of Docker containers that together form the application. See the *software documentation* for an overview of the different containers. It is assumed the containers are deployed using a docker-composition. An alternative deployment based on a Helm chart and intended for an OpenShift (Kubernetes) cluster is described in the Helm for OpenShift README.

*Quality-time* furthermore assumes an LDAP service is available to authenticate users or that forwarded authentication is used.

## 8.1 Docker-composition

This document assumes docker-compose is used to deploy the containers. The docker folder of the *Quality-time* repository contains different compose files for running *Quality-time* in development and continuous integration mode. You can use these compose files as basis for your own deployment configuration.

To deploy *Quality-time* locally, follow these steps:

1. Make a directory, e.g. `quality_time` and change directory to it.

2. Download the docker compose files from the docker folder](https://github.com/ICTU/quality-time/tree/master/docker) and place them in the folder created in the previous step.

3. Set the version number: `export QUALITY_TIME_VERSION=v5.0.0`.

4. Run `docker compose up`.

5. Go to http://localhost to access the application.

6. To log in, use one of the default users available in the *test LDAP server*.

---

**Note:** By default, the application listens on port 80. To change this, set the `PROXY_PORT` environment variable to a different port before starting the application. For example: `export PROXY_PORT=1080`.

---

# 8.2 Configuring authentication (mandatory)

You need to either configure an LDAP server to authenticate users with or configure forwarded authentication.

## 8.2.1 LDAP

To configure an LDAP server to authenticate users with, set the `LDAP_URL`, `LDAP_ROOT_DN`, `LDAP_LOOKUP_USER_DN`, `LDAP_LOOKUP_USER_PASSWORD`, and `LDAP_SEARCH_FILTER` environment variables. Note that `LDAP_URL` may be a comma-separated list of LDAP connection URL(s). Add the LDAP environment variables to the API-server service in the compose file:

```
api_server:
  environment:
    - LDAP_URL=ldap://ldap:389
    - LDAP_ROOT_DN=dc=example,dc=org
    - LDAP_LOOKUP_USER_DN=cn=admin,dc=example,dc=org
    - LDAP_LOOKUP_USER_PASSWORD=admin
    - LDAP_SEARCH_FILTER=(|(uid=$username)(cn=$username))
```

When using the `LDAP_SEARCH_FILTER` as shown above, users can use either their LDAP canonical name (`cn`) or their LDAP user id to login. The `$username` variable is filled by *Quality-time* at run time with the username that the user enters in the login dialog box.

**See also:**

See https://ldap.com/ldap-filters/ for more information on LDAP filters.

## 8.2.2 Forwarded authentication

To configure Forwarded Authentication, set the `FORWARD_AUTH_ENABLED` and `FORWARD_AUTH_HEADER` environment variables. Add the environment variables to the API-server service in the compose file:

```
api_server:
  environment:
    - FORWARD_AUTH_ENABLED=True
    - FORWARD_AUTH_HEADER=X-Forwarded-User
```

> **Danger:** Only enable Forwarded Authentication if *Quality-time* is setup behind a reverse proxy that is responsible for authentication and direct access to *Quality-time* is not possible.

# 8.3 Configuring hostnames and ports (optional)

The hostnames and ports of the different containers can be configured via environment variables. See the *software documentation* for an overview of the available hostname and port environment variables per component.

## 8.4 Configuring example reports (optional)

By default, the server will check for the presence of example reports in the database on startup. If none are present, three example reports will be added to the database. To prevent this behavior, set the `LOAD_EXAMPLE_REPORTS` environment variable to false for the API-server:

```
api_server:
  environment:
    - LOAD_EXAMPLE_REPORTS=False
```

## 8.5 Configuring measurement frequency (optional)

The collector component is responsible for collecting measurement data from sources. It wakes up periodically and gets a list of all metrics from the database. For each metric, the collector gets the measurement data from each of its sources and stores a new measurement in the database.

If a metric has been recently measured and its parameters haven't been changed, the collector skips the metric.

By default, the collector measures metrics whose configuration hasn't been changed every 15 minutes, sleeps 60 seconds in between measurements, and measures at most 30 metrics every time it wakes up. The defaults can be changed as follows:

```
collector:
  environment:
    - COLLECTOR_SLEEP_DURATION=10  # Wake up every 10 seconds
    - COLLECTOR_MEASUREMENT_LIMIT=25  # Measure at most 25 metrics on every wake up
    - COLLECTOR_MEASUREMENT_FREQUENCY=600  # Measure metrics at least every 10
↪minutes
```

> **Warning:** Note that the frontend warns users when metrics have not been measured for a long period, currently hardcoded to one hour. That means that if you set the collector measurement frequency to more than one hour, users will see warnings that the measurement data is old.

## 8.6 Configuring notification frequency (optional)

The notifier component is responsible for notifying users via MS Teams about changed metric statuses. It wakes up periodically and gets a list of all metrics from the database. For each metric, the notifier decides whether a notification is possible and needed.

By default, the notifier wakes up every minute to check for changed metric statuses. This frequency can be changed as follows:

```
notifier:
  environment:
    - NOTIFIER_SLEEP_DURATION=120  # Check for notifications every two minutes
```

## 8.7 Configuring MongoDB credentials (optional)

The default MongoDB credentials can be changed as follows:

```
database:
  environment:
    - MONGO_INITDB_ROOT_USERNAME=admin
    - MONGO_INITDB_ROOT_PASSWORD=secret
```

See the documentation on the MongoDB image for more information.

## 8.8 Configuring renderer localisation (optional)

The date/time format and timezone of the reports that user sees are determined by the user's browser. To configure the date/time format and timezone of exported PDFs, the renderer can be configured as follows:

```
renderer:
  environment:
    - LC_ALL=en_GB.UTF-8  # To get European dates (DD-MM-YYYY)
    - TZ=Europe/Amsterdam  # To get Central European Time
```

## 8.9 Configuring renderer proxy (optional)

```
renderer:
  environment:
    - PROXY_HOST=www  # Hostname of service
    - PROXY_PORT=80  # Port of service
    - PROXY_PROTOCOL=http  # http/https
```

## 8.10 Configuring logging (optional)

The options for configuring logging are limited at the moment. The MongoDB daemon can be told to produce less logging by passing the `--quiet` flag:

```
database:
  command: --quiet
```

The collector, notifier, and API-server all have log level `WARNING` as default. This can be overridden by setting an environment variable to `DEBUG`, `INFO`, `WARNING`, `ERROR` or `CRITICAL`.

| Component | Log level environment variable |
| --- | --- |
| Collector | COLLECTOR_LOG_LEVEL |
| API-server | API_SERVER_LOG_LEVEL |
| Notifier | NOTIFIER_LOG_LEVEL |

The proxy access log is turned off. Please submit an issue if you need this and possibly other logging settings to be configurable.

## 8.11 Using a custom base image for the www container (optional)

The base image of the www container can be configured as follows in the compose file:

```
www:
  build:
    context: ../components/proxy
    args:
      IMAGE_NAME: nginx
      IMAGE_VERSION: stable-perl
```

## 8.12 Moving *Quality-time*

The easiest way to move a *Quality-time* instance is to deploy a new *Quality-time* instance at the new location and then copy the database contents from the old instance to the new instance. All *Quality-time* data is contained in the Mongo database, so that is the only data that needs to be copied.

Start a new mongo container and use that to run the mongodump and mongorestore commands:

```
docker run -dP --rm --name mongo mongo
docker exec -ti mongo mongodump --uri "mongodb://root:<pwd>@<hostname or ip>:27017" --
→out /tmp/dump/qt_dump
docker exec -ti mongo mongorestore --uri "mongodb://root:<pwd>@<hostname or ip>:27017
→" /tmp/dump/qt_dump
```

The <hostname or ip> is the hostname or IP address of the Swarm manager in case of Docker Swarm.

As the dump is stored in a temporary container, the dump will disappear as soon as the container is removed. To keep the dump around, map a folder (-v) in the mongo container.

**See also:**

See Back Up and Restore with MongoDB Tools for more information about the mongodump and mongorestore commands.

# DEVELOPER MANUAL

This document is aimed at *Quality-time* developers and maintainers and describes how to develop, test, document, release, and maintain *Quality-time*. To read more about how *Quality-time* is structured, see the *software documentation*.

## 9.1 Developing

### 9.1.1 Running *Quality-time* locally

When developing *Quality-time*, there are two ways to run *Quality-time* locally: in Docker completely (scenario 1 below) or partly in Docker and partly from shells (scenario 2 below).

If you want to get *Quality-time* up and running quickly, for example for a demo, we recommend scenario 1. For software development, we recommend scenario 2.

#### Install prerequisites

Prerequisites are Docker and Git for both scenario's. For scenario 2 you also need Python 3.12 and a recent version of Node.js (we currently use Node.js v20).

Clone this repository:

```
git clone git@github.com:ICTU/quality-time.git
cd quality-time
```

If you don't have a public key in your GitHub account, use:

```
git clone https://github.com/ICTU/quality-time.git
cd quality-time
```

#### Scenario 1: run all components in Docker

To run *Quality-time* in Docker completely, open a terminal and start all containers with docker compose:

```
docker compose up
```

The advantage of this scenario is that Python and Node.js don't need to be installed. However, as building the containers can be time-consuming we don't recommend this for working on the Quality-time source code.

### Scenario 2: run bespoke component from shells and other components in Docker

In this scenario, we run the *bespoke components* from shells and the *standard components* and *test components* as Docker containers.

The advantage of this scenario is that you don't need to rebuild the bespoke container images while developing. Also, the server component and the frontend component have auto-reload, meaning that when you edit the code, they will restart and run the new code automatically. The collector and notifier components don't have auto-reload, and need to stopped and started by hand to activate new code.

### Start standard and test components in Docker

Open a terminal and start the *standard containers* and *test components* with docker compose:

```
docker compose up database ldap phpldapadmin mongo-express testdata
```

PHP-LDAP-admin is served at http://localhost:3890 and can be used to inspect and edit the LDAP database. Click login, check the "Anonymous" box and click "Authenticate" to login.

Mongo-express is served at http://localhost:8081 and can be used to inspect and edit the database contents.

The test data is served at http://localhost:8080.

There are two users defined in the LDAP database:

- User `Jane Doe` has user id `jadoe` and password `secret`.
- User `John Doe` has user id `jodoe` and password `secret`.

### Start the API-server

Open another terminal and run the API-server:

```
cd components/api_server
python3 -m venv venv
. venv/bin/activate  # on Windows: venv\Scripts\activate
ci/pip-install.sh
python src/quality_time_server.py
```

The API of the API-server is served at http://localhost:5001, e.g. access http://localhost:5001/api/v3/report to get the available reports combined with their recent measurements.

---

**Note:** If you're new to Python virtual environments, note that:

- Creating a virtual environment (`python3 -m venv venv`) has to be done once. Only when the Python version changes, you want to recreate the virtual environment.
- Activating the virtual environment (`. venv/bin/activate`) has to be done every time you open a new shell and want to use the Python installed in the virtual environment.
- Installing the requirements (`ci/pip-install.sh`) has to be repeated when the dependencies, specified in the requirements files, change.

---

**See also:**

- See the Python docs for more information on creating virtual environments.

---

- See this Gist on how to automatically activate and deactivate Python virtual environments when changing directories.

## Start the collector

Open another terminal and run the collector:

```
cd components/collector
python3 -m venv venv
. venv/bin/activate   # on Windows: venv\Scripts\activate
ci/pip-install.sh
python src/quality_time_collector.py
```

## Start the frontend

Open another terminal and run the frontend:

```
cd components/frontend
npm install --ignore-scripts
npm run start
```

The frontend is served at http://localhost:3000.

## Start the notifier

Optionally, open yet another terminal and run the notifier:

```
cd components/notifier
python3 -m venv venv
. venv/bin/activate   # on Windows: venv\Scripts\activate
ci/pip-install.sh
python src/quality_time_notifier.py
```

## Preparing the shared component

*Quality-time* has one component that only contains shared code. The shared code is used by all Python components.

To create a virtual environment for the shared component and install the dependencies run the following:

```
cd components/shared_code
python3 -m venv venv
. venv/bin/activate   # on Windows: venv\Scripts\activate
ci/pip-install.sh
```

## 9.1.2 Coding style

This section contains some notes on coding style used in this project. It's far from complete, however.

### Python

Most of the coding standard are enforced by the *quality checks*.

Methods that can or should be overridden in subclasses have a name with one leading underscore, e.g. `_api_url(self) -> URL`. Methods that should only be used by a class instance itself have a name with two leading underscores, e.g. `__fields(self) -> List[str]`.

Production code and unit tests are organized in parallel hierarchies. Each Python component has a `src` with the production code and a `tests` folder with the unit tests. The folder layout of the `tests` follows the layout of the `src` hierarchy.

### JavaScript

Functional React components are preferred over class-based components.

Production code and unit tests are organized together in one `src` folder hierarchy.

## 9.1.3 Adding metrics and sources

*Quality-time* has been designed with the goal of making it easy to add new metrics and sources. The *data model* specifies all the details about metrics and sources, like the scale and unit of metrics, and the parameters needed for sources. In general, to add a new metric or source, only the data model and the *collector* need to be changed.

### Adding a new metric

To add a new metric you need to make two changes to the data model:

1. Add a specification of the new metric to the data model. See the documentation of the *shared data model* component for a description of the data model and the different metric fields.

2. Update the `metric_type` parameter of the `quality_time` source in the data model. You need to add the human readable name of the new metric to the `values` list of the `metric_type` parameter and you need to add a key-value pair to the `api_values` mapping of the `metric_type` parameter, where the key is the human readable name of the metric and the value is the metric key.

Be sure to run the unit tests of the shared data model component after adding a metric to the data model, to check the integrity of the data model. If you forget to do step 2 above, one of the tests will fail. Other than changing the data model, no code changes are needed to support new metrics.

Suppose we want to add a lines of code metric to the data model, to measure the size of software. We would add the metric to the `METRICS` model in `src/shared_data_model/metrics.py`:

```python
"""Data model metrics."""

from .meta.metric import ..., Metric, Tag, Unit

...

METRICS = {
```

```
    ...
    "loc": Metric(
        name="Size (LOC)",
        description="The size of the software in lines of code.",
        rationale="The size of software is correlated with the effort it takes to␣
→maintain it. Lines of code is "
        "one of the most widely used metrics to measure size of software.",
        unit=Unit.LINES,
        target="30000",
        near_target="35000",
        sources=["manual_number"],
        tags=[Tag.MAINTAINABILITY],
    ),
    ...
}
```

Since we have no (automated) source for the size metric yet, we have added manual number to the list of sources. We also need to add the size metric to the list of metrics that the manual number source supports:

```
"""Manual number source."""

from ..meta.source import Source
...
from ..parameters import IntegerParameter


MANUAL_NUMBER = Source(
    name="Manual number",
    description="A number entered manually by a Quality-time user.",
    parameters=dict(
        number=IntegerParameter(
            ...
            metrics=[
                ...
                "loc",  # Add the size metric here
                ...
            ],
        )
    ),
)
```

After restart of the API-server, you should be able to add the new metric to a quality report and select manual number as a source for the new metric.

## Adding a new source

To add support for a new source, the source (including a logo) needs to be added to the data model. In addition, code to retrieve and parse the source data needs to be added to the collector component, including unit tests of course.

## Adding a new source to the data model

To add a new source you need to make three changes to the data model:

1. Add a specification of the source to the data model. See the documentation of the *shared data model* component for a description of the data model and the different source fields.

2. Update the `source_type` parameter of the `quality_time` source in the data model. You need to add the human readable name of the new source to the `values` list of the `source_type` parameter and you need to add a key-value pair to the `api_values` mapping of the `source_type` parameter, where the key is the human readable name of the source and the value is the metric source key (`cloc` in the example below).

3. Add a small PNG file of the logo in `components/shared_code/src/shared_data_model/logos`. Make sure the filename of the logo is `<source_type>.png`. The frontend will use the `api/v3/logo/<source_type>` endpoint to retrieve the logo.

Be sure to run the unit tests of the shared data model component after adding a source to the data model, to check the integrity of the data model. If you forget to do step 2 above, one of the tests will fail.

Suppose we want to add cloc as source for the LOC (size) metric and read the size of source code from the JSON file that cloc can produce. We would add a `cloc.py` to `src/shared_data_model/sources/`:

```python
"""cloc source."""

from ..meta.source import Source
from ..parameters import access_parameters


CLOC = Source(
    name="cloc",
    description="cloc is an open-source tool for counting blank lines, comment lines,
    and physical lines of source "
    "code in many programming languages.",
    url="https://github.com/AlDanial/cloc",
    parameters=dict(
        **access_parameters(["loc"], source_type="cloc report", source_type_format=
    "JSON")
    ),
)
```

Because cloc can be used to measure the lines of code metric, we need to add the cloc source to the list of sources that can measure lines of code:

```python
METRICS = {
    ...
    "loc": Metric(
        name="Size (LOC)",
        description="The size of the software in lines of code.",
        rationale="The size of software is correlated with the effort it takes to
    maintain it. Lines of code is "
        "one of the most widely used metrics to measure size of software.",
        unit=Unit.LINES,
```

(continues on next page)

```
        target="30000",
        near_target="35000",
        sources=["cloc", "manual_number"],  # Add cloc here
        tags=[Tag.MAINTAINABILITY],
    ),
    ...
}
```

### Adding a new source to the collector

To specify how *Quality-time* can collect data from the source, a new subclass of `SourceCollector` needs to be created.

Add a new Python package to the `source_collectors` folder with the same name as the source type in the data model. For example, if the new source type is `cloc`, the folder name of the collectors is also `cloc`. Next, create a module for each metric that the new source supports. For example, if the new source `cloc` supports the metric LOC (size) and the metric source-up-to-dateness, you would create two modules, each containing a subclass of `SourceCollector`: a `ClocLOC` class in `cloc/loc.py` and a `ClocSourceUpToDateness` class if `cloc/source_up_to_dateness.py`. If code can be shared between these classes, add a `cloc/base.py` file with a `ClocBaseClass`.

To reduce duplication, `SourceCollector` has several abstract subclasses. The class hierarchy is currently as follows:

- `SourceCollector`

  - `UnmergedBranchesSourceCollector`: for sources that collect data for the number of unmerged branches metric

  - `TimeCollector`: for sources that collect time since or until a certain moment in time

    * `TimePassedCollector`: for source-up-to-dateness

      · `JenkinsPluginSourceUpToDatenessCollector`: for getting the source-up-to-dateness from Jenkins plugins

    * `TimeRemainingCollector`: for sources that time remaining until a future date

  - `SourceVersionCollector`: for sources that report version numbers

  - `SlowTransactionsCollector`: for sources that report slow performance transactions

  - `JenkinsPluginCollector`: for sources that collect their data from Jenkins plugins

  - `FileSourceCollector`: for sources that parse files

    * `CSVFileSourceCollector`: for sources that parse CSV files

    * `HTMLFileSourceCollector`: for sources that parse HTML files

    * `JSONFileSourceCollector`: for sources that parse JSON files

    * `XMLFileSourceCollector`: for sources that parse XML files

To support cloc as source for the LOC (size) metric we need to read the size of source code from the JSON file that cloc can produce. We add a `cloc/loc.py` file and in `loc.py` we create a `ClocLOC` class with `JSONFileSourceCollector` as super class. The only method that needs to be implemented is `_parse_source_responses()` to get the amount of lines from the cloc JSON file. This could be as simple as:

```
"""cloc lines of code collector."""

from base_collectors import JSONFileSourceCollector
from model import SourceMeasurement, SourceResponses


class ClocLOC(JSONFileSourceCollector):
    """cloc collector for size/lines of code."""

    async def _parse_source_responses(self, responses: SourceResponses) ->␣
↪SourceMeasurement:
        loc = 0
        for response in responses:
            for key, value in (await response.json()).items():
                if key not in ("header", "SUM"):
                    loc += value["code"]
        return SourceMeasurement(value=str(loc))
```

Most collector classes are a bit more complex than that, because to retrieve the data they have to deal with APIs and while parsing the data they have to take parameters into account. See the collector source code for more examples.

## Writing and running unit tests

To test the `ClocLOC` collector class, we add unit tests to the collector tests package, for example:

```
"""Unit tests for the cloc source."""

from ...source_collector_test_case import SourceCollectorTestCase


class ClocLOCTest(SourceCollectorTestCase):
    """Unit tests for the cloc loc collector."""

    SOURCE_TYPE = "cloc"
    METRIC_TYPE = "loc"

    async def test_loc(self):
        """Test that the number of lines is returned."""
        cloc_json = {
            "header": {}, "SUM": {},  # header and SUM are not used
            "Python": {"nFiles": 1, "blank": 5, "comment": 10, "code": 60},
            "JavaScript": {"nFiles": 1, "blank": 2, "comment": 0, "code": 30}}
        response = await self.collect(get_request_json_return_value=cloc_json)
        self.assert_measurement(response, value="90", total="100")
```

Note that the `ClocTest` class is a subclass of `SourceCollectorTestCase` which creates a source and metric for us, specified using `SOURCE_TYPE` and `METRIC_TYPE`, and provides us with helper methods to make it easier to mock sources (`SourceCollectorTestCase.collect()`) and test results (`SourceCollectorTestCase.assert_measurement()`).

In the case of collectors that use files as source, also add an example file to the *test data component*.

To run the unit tests:

```
cd components/collector
ci/unittest.sh
```

You should get 100% line and branch coverage.

**Running quality checks**

To run the quality checks:

```
cd components/collector
ci/quality.sh
```

Because the source collector classes register themselves (see `SourceCollector.__init_subclass__()`), Vulture will think the new source collector subclass is unused:

```
ci/quality.sh
src/source_collectors/file_source_collectors/cloc.py:26: unused class 'ClocLOC' (60%
→confidence)
```

Add `Cloc*` to the `NAMES_TO_IGNORE` in `components/collector/ci/quality.sh` to suppress Vulture's warning.

# 9.2 Testing

This section assumes you have created a Python virtual environment, activated it, and installed the requirements for each Python component and that you installed the requirements for the frontend component, as described *above*.

## 9.2.1 Unit tests

To run the unit tests and measure unit test coverage of the backend components (this assumes you have created a Python virtual environment, activated it, and installed the requirements as described *above*):

```
cd components/api_server  # or components/collector, components/notifier, components/
→shared_code
ci/unittest.sh
```

To run the frontend unit tests:

```
cd compontents/frontend
npm run test
```

## 9.2.2 Quality checks

To run ruff, mypy, and some other security and quality checks on the backend components:

```
cd components/api_server  # or components/collector, components/notifier, components/
→shared_code
ci/quality.sh
```

### 9.2.3 Feature tests

The feature tests currently test all features through the API served by the API-server. They touch all components except the frontend, the collector, and the notifier. To run the feature tests, invoke this script, it will build and start all the necessary components, run the tests, and gather coverage information:

```
tests/feature_tests/ci/test.sh
```

The `test.sh` shell script will start a server under coverage and then run the feature tests.

It's also possible to run a subset of the feature tests by passing the feature file as argument:

```
tests/feature_tests/ci/test.sh tests/feature_tests/features/metric.feature
```

### 9.2.4 Application tests

The application tests in theory test all components through the frontend, but unfortunately the number of tests is too small to meet that goal. To run the application tests, start all components and then start the tests:

```
docker-compose up -d
docker run -it -w `pwd` -v `pwd`:`pwd` --network=container:qualitytime_www_1 python:3.
↪12.1-bookworm tests/application_tests/ci/test.sh
```

# 9.3 Documentation and changelog

The documentation is written in Markdown files and published on Read the Docs.

To generate the documentation locally:

```
cd docs
python3 -m venv venv
. venv/bin/activate   # on Windows: venv\Scripts\activate
ci/pip-install.sh
make html
open build/html/index.html
```

`make html` also generates the `docs/src/reference.md` reference manual, containing an overview of all subjects, metrics, and sources.

To check the correctness of the links:

```
make linkcheck
```

## 9.4 Releasing

### 9.4.1 Preparation

Make sure the release folder is the current directory, and you have the dependencies for the release script installed:

```
cd release
python3 -m venv venv
. venv/bin/activate
ci/pip-install.sh
```

Run the release script with `--help` to show help information, including the current release.

```
python release.py --help
```

### 9.4.2 Decide the release type

*Quality-time* adheres to Semantic Versioning, so first you need to decide on the type of release you want to create:

- Create a **major** release if the next release contains backwards incompatible changes, and optionally other changes and bug fixes.

- Create a **minor** release if the next release contains non-breaking changes, and optionally bug fixes.

- Create a **patch** release if the next release contains bug fixes only.

If you want to test the release (for example, deploy it to a test environment, or roll out a release to early adopters), it's possible to create a **release candidate** for a major, minor, or patch release.

---

**Important:** To determine whether a release is major, minor, or patch, compare the changes to the *previous most recent release*, excluding release candidates.

---

### 9.4.3 Determine the version bump

Having decided on the release type, there are the following possibilities for the version bump argument that you will be passing to the release script:

- If the current release is a release candidate,

  - and you want to create another release candidate, use: `rc`. If the current release is e.g. v3.6.1-rc.0, this will bump the version to v3.6.1-rc.1.

  - and the next release will not be, use: `drop-rc`. If the current release is e.g. v3.6.1-rc.0, this will bump the version to v3.6.1.

  - and changes have been made since the previous release candidate that impact the release type, use: `rc-major`, `rc-minor`, or `rc-patch`. If the current release is e.g. v3.6.1-rc.0, using `rc-minor` will bump the version to v3.7.0-rc.0.

- If the current release is not a release candidate:

  - and you want to create a release candidate, use: `rc-major`, `rc-minor`, or `rc-patch`. If the current release is e.g. v3.6.1, using `rc-minor` will bump the version to v3.7.0-rc.0.

– and you don't want to create a release candidate, use: `major`, `minor`, or `patch`. If the current release is e.g. v3.6.1, using `minor` will bump the version to v3.7.0.

### 9.4.4 Check the preconditions

The release script will check a number of preconditions before actually creating the release. To check the preconditions without releasing, invoke the release script with the version bump as determined:

```
python release.py --check-preconditions-only <bump>  # Where bump is major, minor,␣
→patch, rc-major, rc-minor, rc-patch, rc, or drop-rc
```

If everything is ok, there is no output, and you can proceed creating the release. Otherwise, the release script will list the preconditions that have not been met and need fixing before you can create the release.

### 9.4.5 Create the release

To release *Quality-time*, issue the release command (in the release folder) using the type of release you picked:

```
python release.py <bump>  # Where bump is major, minor, patch, rc-major, rc-minor, rc-
→patch, rc, or drop-rc
```

If all preconditions are met, the release script will bump the version numbers, update the change history, commit the changes, push the commit, tag the commit, and push the tag to GitHub. The GitHub Actions release workflow will then build the Docker images and push them to Docker Hub. It will also create an Software Bill of Materials (SBOM) for the release, which can be found under the "Artifacts" header of the workflow run.

The Docker images are `quality-time_database`, `quality-time_renderer`, `quality-time_api_server`, `quality-time_collector`, `quality-time_notifier`, `quality-time_proxy`, `quality-time_testldap`, and `quality-time_frontend`. The images are tagged with the version number. We don't use the `latest` tag.

## 9.5 Maintenance

### 9.5.1 Python and JavaScript dependencies

Keeping dependencies up-to-date is an important aspect of software maintenance. Python (pip) and JavaScript (npm) dependencies are kept up-to-date via the Dependabot GitHub action.

For Python, we follow the dependency management practice described by James Bennett, to a large extent.

### 9.5.2 Docker images

Base images used in the Docker containers, and additionally installed software, need to be upgraded by hand from time to time. These are:

- API-server: the Python base image.

- Collector: the Python base image.

- Notifier: the Python base image.

- Frontend: the Node base image, the curl version, the npm version, and the serve version.

- Database: the MongoDB base image.

- Proxy: the Nginx base image.

- Renderer: the Node base image, the curl version, the Chromium version, and the npm version.

- Test data: the Python base image.

- Container images directly specified in compose files used for development and continuous integration: `mongo-express`, `ldap`, `phpldapadmin`, and `selenium`.

# SOFTWARE DOCUMENTATION

This document describes the *Quality-time* software. It is aimed at *Quality-time* developers and maintainers.

## 10.1 Component overview

*Quality-time* consists of eight Docker components, as depicted above.

### 10.1.1 Bespoke components

There are four bespoke components:

- A *frontend*, serving the user interface. The frontend is written in JavaScript using ReactJS and Semantic UI React.

- An *API-server* serving the API for the user interface. The API-server is written in Python using Bottle as web framework.

- A *collector* to collect the measurements from the sources. The collector is written in Python using `aiohttp` as HTTP client library.

- A *notifier* to notify users about events such as metrics turning red. The notifier is written in Python.

Source code that is shared between the Python components lives in the *shared data model* and *shared code* components. These are not run-time components. The code of these components is shared at build time, when the Docker images are created. The data model is used by all Python components, i.e. the API-server, the collector, and the notifier. The shared code is used by the API-server.

## 10.1.2 Standard components

*Quality-time* uses three standard components:

- A *proxy*, routing traffic from and to the user's browser. The proxy is based on Nginx.

- A *database*, for storing reports and measurements. The database is based on MongoDB.

- A *renderer*, to export reports to PDF. The renderer is based on Puppeteer.

In addition, unless forward authentication is used, an LDAP server is expected to be available to authenticate users.

## 10.1.3 Test components

For testing purposes there a few additional components:

- A web server serving *test data*.

- A *test LDAP server*.

- A tool to administer users in the LDAP server (phpldapadmin).

- A tool to view and edit the database contents (mongo-express).

# 10.2 Frontend

The frontend contains the React frontend code. This component was bootstrapped using Create React App.

## 10.2.1 Health check

As a health check, the favicon is downloaded.

## 10.2.2 Environment variables

The frontend uses the following environment variables:

| Name | Default value | Description |
|---|---|---|
| FRONTEND_PORT | 5000 | The port the frontend listens on. |

# 10.3 API-server

## 10.3.1 API

The API of the API-server is versioned. The version is not changed when backwards compatible changes are made, such as the addition of new endpoints. The public *API* is documented separately.

## 10.3.2 Health check

The Dockerfile contains a health check that uses curl to retrieve an end-point (api/health) from the API-server. Officially, this end-point does not exist, but since the server returns an empty JSON file for non-existing endpoints it works for checking the health of the API-server.

## 10.3.3 Environment variables

The API-server uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| API_SERVE | 5001 | Port of the API-server. |
| API_SERVE | WARNING | Log level. Allowed values are `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`. |
| DATABASE_ | `mongodb://root:root@dat` | Mongo database connection URL. |
| LDAP_URL | `ldap://ldap:389` | Comma-separated list of LDAP connection URL(s). |
| LDAP_ROOT | `dc=example,dc=org` | LDAP root distinguished name. |
| LDAP_LOOK | `cn=admin,dc=example,dc=org` | LDAP lookup user distinguished name. |
| LDAP_LOOK | `admin` | LDAP lookup user password. |
| LDAP_SEAF | `(&#124;(uid=$$userna` | LDAP search filter. With this default search filter, users can use either their LDAP canonical name (`cn`) or their LDAP user id to login. The `$username` variable is filled by *Quality-time* at run time with the username that the user enters in the login dialog box. |
| LOAD_EXAM | True | Whether or not to import example reports in the database on start up. |
| FOR-WARD_AUTH | False | Whether or not to enable forward authentication. |
| FOR-WARD_AUTH | `X-Forwarded-U` | Header to use for getting the username if forward authentication is turned on. |

## 10.4 Collector

The collector is responsible for collecting measurement data from sources. It wakes up periodically and retrieves a list of all metrics from the database. For each metric, the collector gets the measurement data from each of its sources and stores a new measurement to the database.

If a metric has been recently measured and its parameters haven't been changed, the collector skips the metric.

By default, the collector measures metrics whose configuration hasn't been changed every 15 minutes and sleeps 60 seconds in between measurements. This can be changed using the environment variables listed below.

### 10.4.1 Health check

Every time the collector wakes up, it writes the current date and time in ISO format to the health check file. This date and time is read by the Docker health check (see the Dockerfile). If the written date and time are too long ago, the collector container is considered to be unhealthy.

### 10.4.2 Environment variables

The collector uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| COLLEC-TOR_LOG_LEVEL | WARNING | Log level. Allowed values are DEBUG, INFO, WARNING, CRITICAL, and ERROR. |
| COLLEC-TOR_SLEEP_DUR. | 20 | The maximum amount of time (in seconds) that the collector sleeps between collecting measurements. |
| COLLEC-TOR_MEASUREMEI | 30 | The maximum number of metrics that the collector measures each time it wakes up. If more metrics need to be measured, they will be measured the next time the collector wakes up. |
| COLLEC-TOR_MEASUREMEI | 900 | The amount of time (in seconds) after which a metric should be measured again. |
| DATABASE_URL | mongodb://root:root@datal | Mongo database connection URL. |
| HEALTH_CHECK_ | /home/collector/health_check.txt | Path to the file used for health check. |
| HTTP(S)_PROXY | | Proxy to use by the collector. |

**See also:**

See the aiohttp documentation for more information on proxy support.

## 10.5 Notifier

The notifier is responsible for notifying users about significant events, such as metrics turning red. It wakes up periodically and asks the server for all reports. For each report, the notifier determines whether notification destinations have been configured, and whether events happened that need to be notified.

### 10.5.1 Health check

Every time the notifier wakes up, it writes the current date and time in ISO format to the health check file. This date and time is read by the Docker health check (see the Dockerfile). If the written date and time are too long ago, the notifier container is considered to be unhealthy.

## 10.5.2 Environment variables

The notifier uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| DATABASE_URL | mongodb://<br>root:root@database:2701 | Mongo database connection URL. |
| HEALTH_CHECK_FII | /home/notifier/<br>health_check.txt | Path to the file used for health check. |
| NOTI-<br>FIER_LOG_LEVEL | WARNING | Log level. Allowed values are DEBUG, INFO, WARN-<br>ING, ERROR, and CRITICAL. |
| NOTI-<br>FIER_SLEEP_DURAT | 60 | The amount of time (in seconds) that the notifier sleeps<br>between sending notifications. |

# 10.6 Shared code

The shared code component contains code and resources shared between the servers and the collector and notifier components. This includes the *example reports*, a *shared data model*, and code to initialize the servers, access the database, and provide endpoints.

## 10.6.1 Example reports

The example reports are imported when a server is started and the database doesn't contain any sample reports yet. Turn off the loading of example report by setting LOAD_EXAMPLE_REPORTS to False. See the sections on configuration of the servers below.

# 10.7 Shared data model

The shared data model contains the data model that is shared between all Python components.

## 10.7.1 Data model

The data model package describes the domain model used by the application. It allows for a frontend that doesn't need to know about specific metrics and sources. When a server component starts up, it checks whether the data model has changed, and if so, imports it into the database.

The data model package consists of a meta model and the data model itself. The data model consists of four major parts:

- Scales
- Metrics
- Sources
- Subjects

The meta model uses Pydantic to specify the components and attributes of the data model.

## Scales

The `scales` part of the data model defines the scales that can be used to measure metrics. At the time of writing these include an absolute count scale, a percentage scale, and a version number scale.

Each metric defines the scales it supports.

## Metrics

The `metrics` part of the data model is a dictionary with all supported metric types. The keys are the metric type names. The values are objects describing the metric type. All metric types have the following meta model:

```python
class Metric(DescribedModel):
    """Base model for metrics."""

    scales: list[str] = Field(["count"], min_length=1)
    default_scale: str = "count"
    unit: Unit = Unit.NONE
    addition: Addition = Addition.SUM
    direction: Direction = Direction.FEWER_IS_BETTER
    target: str = "0"
    near_target: str = "10"
    sources: list[str] = Field(..., min_items=1)
    tags: list[Tag] = []
    rationale: str = ""  # Answers the question "Why measure this metric?", included
→in documentation and UI
    rationale_urls: list[str] = []
    explanation: str | None = ""  # Optional explanation of concepts in text format,
→included in documentation and UI
    explanation_urls: list[str] = []
    documentation: str | None = ""  # Optional documentation in Markdown format, only
→included in the documentation
```

The `name` is the default name of metrics of this type. The `description` describes what the metric measures. These are part of the `DescribedModel`.

The `scales` list shows which scales the metric supports and the `default_scale` specifies which scale is the default scale.

The `unit` is the default unit of the metric, e.g. lines of code, security warnings, or in the above example, complex units.

The `addition` determines how values from multiple sources are combined: possible values are `max`, `min`, and `sum`.

The `direction` specifies whether smaller measurement values are better or worse.

The `target` is the default target value for the metric. The `near_target` is when the metric becomes red. Values between `target` and `near_target` are yellow.

The list of `sources` contains the keys of source types that support this metric type.

Finally, `tags` are strings used to group related metrics.

Users with sufficient rights can override the default name, unit, and target of metrics via the user interface.

## Sources

The `sources` part of the data model is a dictionary that describes all supported source types. The keys are the source type names. The values are objects describing the source type. All source types have the following meta model:

```python
class Source(DescribedModel):
    """The source model extends the base model with source parameters and measurement
→entities."""

    url: HttpsUrl | None = None
    documentation: dict[str, str] | None = None   # Documentation in Markdown format
    configuration: dict[str, Configuration] = {}
    parameters: dict[str, Parameter]
    entities: dict[str, Entity] = {}
    issue_tracker: bool | None = False
```

The `name` is the default name of sources of this type. The `description` gives some background information on the source type. These are part of the `DescribedModel`.

The `url` links to a landing page describing the source type.

## Configuration

In cases where *Quality-time* needs information about sources that doesn't need to be parameterizable, `Configurations` can be added to the source. A configuration consists of a name (via `NamedModel`), a list of metrics to which the configuration applies, and a value:

```python
class Configuration(NamedModel):
    """Configuration for specific metrics."""

    metrics: list[str] = Field(..., min_items=1)
    value: list[str] = Field(..., min_items=1)
```

## Parameters

The `parameters` describe the parameters that need to be entered by the user to configure the source:

```python
class Parameter(NamedModel):
    """Source parameter model."""

    short_name: str | None = None
    help: str | None = None
    help_url: HttpsUrl | None = None
    type: ParameterType
    placeholder: str | None = None
    mandatory: bool = False
    default_value: str | list[str] = ""
    unit: str | None = None
    metrics: list[str] = Field(..., min_items=1)
    values: list[str] | None = None
    api_values: dict[str, str] | None = None
    validate_on: list[str] | None = None
```

Each parameter has a `name` (via `NamedModel`) and a `short_name` used as label in the user interface. The parameter can have a `help` string or a `help_url` (but not both).

The `type` specifies the type of the parameter and the widget used to get user input. Possible values are amongst others `string`, `password`, `integer`, and `multiple_choice`.

The `placeholder` contains text to display in case of multiple choice parameters. For example, in the case of a multiple choice 'severities' parameter with possible values of 'low', 'medium', 'high', the placeholder can be set to 'all severities' to indicate that by default all severities will be measured.

The `mandatory` field indicates whether the parameter is mandatory.

The `default_value` specifies the default value. In case of multiple choice parameters this should be a, possibly empty, list of values.

The `unit` indicates the unit of the parameter. If the `type` is `integer` the `unit` and `min_value` need to be specified.

For each parameter, a list of `metrics` must be given for which the parameter is applicable. This is needed because not every metric needs the same parameters.

If the `type` is `multiple_choice` the possible `values` need to be specified. Also, an `api_values` mapping can specify how the values map to the values used in the API of the source.

The `validate_on` field specifies that the parameter needs to be validated when the parameters in the list change. This can be used to specify that e.g. a URL parameter must be validated when the user changes the password parameter.

### Entities

Measurement entities are the things that are counted or measured to get the measurement value. For example, the measurement entities of the 'violations' metric are the individual violations. Sometimes, the measurement entities are not interesting enough to show, e.g. when measuring the size in terms of lines of code. In other cases, groups of entities are shown, for example test runs as entities for the 'tests' metric.

The `Entity` and `EntityAttribute` meta models look as follows:

```python
class Entity(BaseModel):
    """Measurement entity (violation, warning, etc.)."""

    name: str = Field(..., regex=r"[a-z]+")
    name_plural: str | None = None
    attributes: list[EntityAttribute]
    measured_attribute: str | None = None


class EntityAttribute(NamedModel):
    """Attributes of measurement entities."""

    key: str | None = None
    help: str | None = None
    url: str | None = None  # Which key to use to get the URL for this attribute
    color: dict[str, Color] | None = None
    type: EntityAttributeType | None = None
    alignment: EntityAttributeAligment | None = None  # If not given, the aligment is
    →based on the attribute type
    pre: bool | None = None  # Should the attribute be formatted using <pre></pre>?
    →Defaults to False
    visible: bool | None = None  # Should this attribute be visible in the UI?
    →Defaults to True
```

Each entity contains the name (both singular and plural) of the entities and a list of `attributes`.

The attributes are shown as columns in the front end. Each attribute/column consists of a `name` (via `NamedModel`), which is used as column header, and a `key`, used to get the data from the database.

An attribute/column can have a key `url` to specify which field contains the URL to be used in the column. In theory, each column can link to a different URL this way.

To specify the data type of the attribute/column, use the `type` field. If no type is specified, `string` is assumed and no special formatting is applied. Other types supported at the moment are `date`, `datetime`, `float`, `integer`, and `status`. When using `date` or `datetime`, the column should be an ISO-formatted date or date-time string and `Date.toLocaleDateString()` or `Date.toLocaleString()` is used to format the date or date-time.

Values can be mapped to colors using the optional `color` field with a column-value-to-color mapping as value. Possible colors are `positive` (green), `negative` (red), `warning` (yellow) and `active` (grey). These correspond to the possible states of table rows in Semantic UI React.

Users can mark entities as false positive to ignore them. By default, *Quality-time* subtracts one from the metric value for each ignored entity. However, this would be incorrect if an entity represents a value greater than one, for example when the metric is the amount of ready user story points and each entity is a user story. In that case *Quality-time* can use an attribute of the entity to subtract from the value. The entity field `measured_attribute` determines which attribute to use.

In most cases, the measured attribute is one of the attributes. In other cases, the measured attribute may depend on the parameters selected by the user. For example, when measuring 'tests' using Azure DevOps as source, the test results (failed/passed/etc.) selected by the user influence how many tests *Quality-time* has to subtract from the total if the user decides to ignore a test run. To accommodate this, it is possible to add an attribute that is not shown by the front end, but is used as measured attribute, by marking the attribute as not visible.

Of course, the collector needs to compute the extra attribute and add it to the measurement entities.

### Subjects

The `subjects` part of the data model is an object where the keys are the subject types and the values are objects describing the subject. The `Subject` meta model looks as follows:

```python
class Subject(DescribedModel):
    """Base model for subjects."""

    metrics: list[str] = Field(..., min_items=1)
```

The `name` is the default name of the subject. The `description` describes the subject type. Both fields are part of `DescribedModel`.

The list of `metrics` contains the metrics that make the most sense for the subject type, and is used for filtering the list of metrics in the dropdown menu of the buttons for moving and copying metrics.

## 10.8 Proxy

The proxy routes traffic from and to the user's browser. *Quality-time* uses Nginx, but this can be replaced by another proxy if so desired.

The proxy Dockerfile adds the *Quality-time* configuration to the Nginx image.

### 10.8.1 Environment variables

The proxy uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| FRONTEND_HOST | frontend | The host name of the frontend. |
| FRONTEND_PORT | 5000 | The port the frontend listens on. |
| API_SERVER_HOST | api_server | The hostname of the API-server. |
| API_SERVER_PORT | 5001 | The port the API-server listens on. |

## 10.9 Database

The database component consists of a Mongo database to store reports and measurements.

The proxy Dockerfile wraps the MongoDB image in a *Quality-time* image so the MongoDB version number can be changed when needed.

*Quality-time* stores its data in a Mongo database using the following collections: `datamodels`, `measurements`, `reports`, `reports_overviews`, and `sessions`.

The two server components are the only components that directly interacts with the database.

Data models, reports, and reports overviews are temporal objects. Every time a new version of the data model is loaded or the user edits a report or the reports overview, an updated copy of the object (a "document" in Mongo-parlance) is added to the collection. Since each copy has a timestamp, this enables the API-server to retrieve the documents as they were at a specific moment in time and provide time-travel functionality.

### 10.9.1 Environment variables

The database uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| MONGO_INITDB_ROOT_USERNAME | root | The MongoDB root username. |
| MONGO_INITDB_ROOT_PASSWORD | root | The MongoDB root password. |

## 10.10 Renderer

The renderer component is used to export quality reports to PDF. *Quality-time* uses puppeteer.

The renderer Dockerfile wraps puppeteer with a small API that uses puppeteer to convert a report URL into a PDF file.

### 10.10.1 Health check

The Dockerfile contains a health check that uses curl to retrieve an API (api/health) from the renderer API server.

### 10.10.2 Environment variables

The renderer uses the following environment variables:

| Name | Default value | Description |
| --- | --- | --- |
| PROXY_HOST | www | Hostname of the proxy. The renderer uses this to access the reports that need to be exported to PDF. |
| PROXY_PORT | 80 | Port of the proxy. The renderer uses this to access the reports that need to be exported to PDF. |
| PROXY_PROTC | http | Protocol of the proxy. The renderer uses this to access the reports that need to be exported to PDF. |
| LC_ALL | | Set the date format in the PDF export. For example, to get DD-MM-YYYY use: en_GB.UTF-8. |
| TZ | | Make the PDF export use the correct timezone. For example, to get Central European Time use: Europe/Amsterdam. |

## 10.11 Test data

This component contains test data for the example reports. The Docker image is published as ictu/quality-time_testdata on Docker Hub.

### 10.11.1 Running the test data component

The test data component is started as part of the docker composition for development, see the *developer manual*.

To serve the test data locally, you can also start a web server from a console, for example:

```
python3 -m http.server
```

### 10.11.2 Adding test data

Add the example file(s) to the test data reports and update one or more of the example reports in the shared code component.

### 10.11.3 Acknowledgements

- `cobertura.xml` was copied from https://github.com/Bachmann1234/diff_cover/blob/main/diff_cover/tests/fixtures/dotnet_covera

- `testng-results.xml` was copied from https://github.com/richie-b/AtnApiTest/blob/master/test-output/testng-results.xml.

## 10.12 Test LDAP server

A test LDAP server with test users is included for development and testing purposes. An admin interface (phpldapadmin) is included to administer users in this LDAP server.

### 10.12.1 LDAP users

The LDAP database has two users:

| User | Email address | Username | Password |
|------|---------------|----------|----------|
| Jane Doe | janedoe@example.org | jadoe | secret |
| John Doe | johndoe@example.org | jodoe | secret |

# **API**

The API of *Quality-time* is used by the frontend, but part of it can also be used for integration purposes. Below the public endpoints are documented.

API documentation can be retrieved via http://www.quality-time.example.org/api (all versions, all routes), http://www.quality-time.example.org/api/v3 (all routes for a specific version, in this case version 3), and http://www.quality-time.example.org/api/v3/<route_fragment> (all routes matching a specific text fragment).

## 11.1 Search

For each of the four domain object types a search POST-endpoint is available:

- `api/v3/metric/search`
- `api/v3/report/search`
- `api/v3/source/search`
- `api/v3/subject/search`

Each endpoint expects a JSON with an attribute name and value to search for:

```
{
    "<attribute_or_parameter_name>": "<value>"
}
```

For example:

```
{
    "title": "My application"
}
```

The search endpoints can search for exactly one attribute at a time. The matching is done on the complete value and is case sensitive.

The endpoints return a JSON of the following form if no errors occur:

```
{
    "domain_object_type": "domain object type passed to the endpoint",
    "ok": true,
    "search_query": "query passed to the endpoint",
    "uuids": ["list of uuids"]
}
```

For example:

```
{
    "domain_object_type": "metric",
    "ok": true,
    "search_query": {"name": "Metric 1"},
    "uuids": ["e887047a-9ae5-41c2-8fc2-bd9a767420dc"]
}
```

If something goes wrong, an error response is returned:

```
{
    "domain_object_type": "metric",
    "error": "error message",
    "search_query": {"name": "Metric 1"},
    "ok": false
}
```

The `search_query` is not included in the error response if parsing the search query parameters fails.

## 11.2 Export to PDF

If the PDF report needs to be downloaded programmatically, e.g. for inclusion in a release package, use the endpoint: `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf`. No authorization is needed for this endpoint.

The `report_uuid` is the unique identifier that *Quality-time* assigns to a report. It can be found by navigating to a report in the browser and looking for the `report_uuid` in the address bar. For example, when the URL in the browser's address bar is `https://www.quality-time.example.org/f1d0e056-2440-43bd-b640-f6753ccf4496?hidden_columns=comment`, the part between the last slash and the question mark is the `report_uuid`.

To hide metrics that do not need any action, set the `metrics_to_hide` parameter to `no_action_needed`, i.e. `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?metrics_to_hide=no_action_needed`. To hide all metrics and only export the report dashboard to PDF, set the `metrics_to_hide` parameter to `all`.

To hide columns from the report, set the `hidden_columns` parameter, for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?hidden_columns=target,comment`. Possible options are `trend`, `status`, `measurement`, `target`, `source`, `comment`, `issues`, and `tags`.

To hide tags from the report, set the `hidden_tags` parameter, for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?hidden_tags=security,usability`.

To expand metrics and set the active tab of the metric detail information, add the `tabs` parameter, i.e. `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?tabs=<metric_uuid>:<tab_index>,<metric_uuid>:<tab_index>,...`. The metric UUID can be found by navigating to a report in the browser, expanding the metric, and looking for the `tabs` parameter in the address bar. For example, when the URL in the browser's address bar is `https://www.quality-time.example.org/1d0e056-2440-43bd-b640-f6753ccf4496?tabs=d4c0dea1-b072-417f-804e-6045544748db:0`, the part between the equal sign and the colon is the metric UUID of the expanded metric. The number after the colon is the number of the active tab, e.g. 0 is the metrics configuration tab, 1 is the source configuration tab, 2 is the trend graph, etc.

To show the measurement trend, add the `nr_dates` parameter and set it to more than 1, for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?nr_dates=5`. The maximum supported value is 7 dates.

To change the time between dates shown, use the `date_interval` parameter. The interval should be an integer and can have the value 1, 7, 14, 21, or 28. For example, for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?date_interval=7`.

To add issue attributes to the exported report, set the `show_issue_summary`, `show_issue_creation_date`, and/or `show_issue_update_date` parameters to `true`. For example, `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?show_issue_summary=true`.

To export an older version of a report, add the `report_date` parameter with a date value in ISO-format (YYYY-MM-DD), for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?report_date=2020-09-01`.

Reports contain the report URL in the footer of the report. When exporting PDFs manually, the *Quality-time* frontend supplies the report URL to the endpoint. When using the endpoint directly to export a report to PDF, the report URL needs to be supplied as parameter. Add the `report_url` parameter with the URL of the report, for example `https://www.quality-time.example.org/api/v3/report/<report_uuid>/pdf?report_url=https://www.quality-time.example.org/<report_uuid>`.

---

**Tip:** It is also possible to download a PDF version of the reports overview via the API. Use the endpoint `https://www.quality-time.example.org/api/v3/reports_overview/pdf`. No authorization is needed for this endpoint. The parameters for exporting a report, listed above, can also be used when exporting the reports overview.

---

## 11.3 Export and import reports as JSON

*Quality-time* provides functionality for importing and exporting reports in JSON format. This functionality can be used for backing up reports or for transferring reports from one *Quality-time* instance to another one. Currently, this functionality is only available via the API, with one endpoint for importing and one for exporting the JSON reports.

A *Quality-time* report in JSON format contains the latest configuration of the report, with all its subjects, metrics and sources. It does not contain any actual measurements. The credentials of configured sources are encrypted on export to protect sensitive data.

To use the import and export endpoints you need to be authenticated. For example, using curl:

```
curl --cookie-jar cookie.txt --request POST --header "Content-Type: application/json"
↪--data '{"username": "jadoe", "password": "secret"}' https://quality-time.example.
↪org/api/v3/login
```

### 11.3.1 Exporting reports

The exporting endpoint is available via `https://quality-time.source.org/api/v3/report/<report-uuid>/json?public_key=<public-key>`. The exporting endpoint returns JSON content only.

For example, using curl, and assuming you have logged in as shown above:

```
curl --cookie cookie.txt --output report.json https://quality-time.source.org/api/v3/
↪report/97b2f341-45ce-4f2b-9a71-3675f2f54cf7/json
```

The `report_uuid` is the unique identifier that *Quality-time* assigns to a report. It can be found by navigating to a report in the browser and looking for the `report_uuid` in the address bar. For example, when the URL in the browser's address bar is `https://quality-time.source.org/f1d0e056-2440-43bd-b640-f6753ccf4496?hidden_columns=comment`, the part between the last slash and the question mark is the `report_uuid`.

The public key argument is optional. If no public key is provided, the public key of the exporting *Quality-time* instance is used for encrypting the source credentials. If the report needs to be imported in a different *Quality-time* instance, the public key of that instance should be provided. It can be obtained at `https://quality-time.destination.org/api/v3/public_key`. The exported JSON report can only be imported into the *Quality-time* whose public key has been used for the encryption of credentials during the export.

The public key endpoint returns a JSON like this:

```
{
  "public_key": "-----BEGIN PUBLIC KEY-----\
↪nMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAtXvAeTqsgXIb98rGfZDk\
↪n4ZUssjjrMOCOL7xuZh6lYwN41UP0Day78tbzMxCx8mLfT76DckK1xkeGkKpS/UYD\
↪n2ooDXluplRDGxfebZg+qy54toW878rnYe4HJu6YoDaBnthr1Muy9ttHOVB+6ucXY\
↪nX23uzOF6FD4rZZTn4uGpEF9qfpzaVZrSpqWy9YAfZEsNNjqmbPYR+H0WjdihIpgY\
↪n3AabLHdw02VN8cIzgh1ILLPFcBo2CqNWpETNIGdlPORfDiUx6HVxSXt80xwxFpop\
↪n9hXQDuKSDVGlpVl5YKTwRyqEcFvhbTEJ1gJ+FksCRfrZ/hdVlI5mlCyN/gi+k3gO\
↪nErtN0kFlIwCPyLHw5hsi/f8rLGpG1MaXmtI4fBoTbnozwaTcmoO9GO/Ell3ITTBW\
↪nJbS3fSqKDtTU3NhalnJk5h99yQc+tgHIc+y/odKcicTDw5ZvlNIsY/ig6Z1BqYOl\n3FEI9a+I/mhcynSM/
↪30elGsi+j/ZrWyhD6uB3E9+UtL5l7FtDWyIIoE7DaMQJZxg\
↪nDNLCHWKACjjE+Tjr4ExUEgtzcMMmRXL2QZkylxxT9WU0Qe0U3nwJWBj6h+3xJird\npJ3weRfCPwrZ/
↪6SxWE19tmZiynNnvnywxTJKgT15/Qkv1T0QyVCH/UeyxAhAXqYc\nBulld6J57dZwlpfWtf/
↪ua3cCAwEAAQ==\n-----END PUBLIC KEY-----\n"
}
```

To be able to pass the public key as query parameter to the export endpoint, it needs to be encoded. Download the public key as file:

```
curl --output public_key.json https://quality-time.destination.org/api/v3/public_key
```

And then encode the public key as follows:

```
$ python3 -c 'import json; import urllib.parse; key = json.load(open("public_key.json
↪"))["public_key"]; print(urllib.parse.quote_plus(key))'
-----BEGIN+PUBLIC+KEY----- ... encoded public key ... -----END+PUBLIC+KEY-----%0A
```

## 11.3.2 Importing reports

The importing endpoint is available via `https://quality-time.destination.org/api/v3/report/import`. The import endpoint accepts JSON content only. See the example reports for the format.

For example, using curl, and assuming you have logged in as shown above:

```
$ curl --cookie cookie.txt --request POST --header "Content-Type: application/json" --
↪data @report.json https://quality-time.destination.org/api/v3/report/import
{"ok": true, "new_report_uuid": "97a3e341-44ce-4f2b-4471-36e5f2f34cf6"}
```

On import, all UUIDs contained in the report (UUIDs of the report, subjects, metrics and sources) will be replaced to prevent conflicts if the report already exists.

If the report contains encrypted credentials, the importing *Quality-time* instance will decrypt the credentials using its public key. Note that if the credentials were encrypted using the public key of a different *Quality-time* instance, an error will occur, and the import will fail.

To allow for seeding a *Quality-time* instance with default reports, imported reports may contain unencrypted credentials. These unencrypted credentials will be imported unchanged.

### 11.3.3 Copying reports from one *Quality-time* instance to another

Tying the previous two sections together, these steps export a report from a source *Quality-time* instance and import it into a destination instance:

```
# Get the public key of the destination Quality-time
$ curl --output public_key.json https://quality-time.destination.org/api/v3/public_key
# Encode the public key
$ python3 -c 'import json; import urllib.parse; key = json.load(open("public_key.json
↪"))["public_key"]; print(urllib.parse.quote_plus(key))'
-----BEGIN+PUBLIC+KEY-----encoded-public-key-----END+PUBLIC+KEY-----%0A
# Log in to the source Quality-time
$ curl --cookie-jar cookie.txt --request POST --header "Content-Type: application/json
↪" --data '{"username": "jadoe", "password": "secret"}' https://quality-time.source.
↪org/api/v3/login
# Copy the public key and use it in the next line to export the report
$ curl --cookie cookie.txt --output report.json https://quality-time.source.org/api/
↪v3/report/1352450b-30fa-4a82-aec5-7b5d0017ee13/json?public_key=-----
↪BEGIN+PUBLIC+KEY-----encoded-public-key-----END+PUBLIC+KEY-----%0A
# Log in to the destination Quality-time
$ curl --cookie-jar cookie.txt --request POST --header "Content-Type: application/json
↪" --data '{"username": "jadoe", "password": "secret"}' https://quality-time.
↪destination.org/api/v3/login
# Import the report in the destination Quality-time
$ curl --cookie cookie.txt --request POST --header "Content-Type: application/json" --
↪data @report.json https://quality-time.destination.org/api/v3/report/import
```

## 11.4 Monitoring metric statuses

To enable monitoring metric statuses outside of *Quality-time*, the `api/v3/report/<report_uuid>/metric_status_summary` endpoint can be used. It returns a JSON response for the specified report in the following format:

```
{
    "report_uuid": "xyz",
    "title": "Report title",
    "red": 1,
    "green": 4,
    "other colors": "..."
}
```

The `report_uuid` is the unique identifier that *Quality-time* assigns to a report. It can be found by navigating to a report in the browser and looking for the `report_uuid` in the address bar. For example, when the URL in the browser's address bar is `https://www.quality-time.example.org/f1d0e056-2440-43bd-b640-f6753ccf4496?hidden_columns=comment`, the part between the last slash and the question mark is the `report_uuid`.

# TWELVE

# CHANGELOG

All notable changes to *Quality-time* will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

## 12.1 v5.7.0 - 2024-01-31

### 12.1.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.1.2 Fixed

- When time traveling, show the most recent measurements of the selected date instead of the earliest measurements of the selected date. Fixes #7891.

### 12.1.3 Added

- In MS Teams notifications, have the mentioned metrics link to the respective metrics in the quality report. Closes #7615.

- When measuring security warnings with Cargo audit JSON reports as source, include warnings (unsound and yanked packages) in addition to vulnerabilities. Also add a parameter to select which warning types to count. Closes #7889.

### 12.1.4 Removed

- Don't load the Next-action example report and remove it from the code base; the Next-action project has been discontinued. Note, existing Next-action example reports are not deleted. Closes #7936.

## 12.2  v5.6.0 - 2024-01-12

### 12.2.1  Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.2.2  Fixed

• Links in MS Teams notifications were incorrect. Note: the fix only works for new notification destinations. Please recreate existing notification destinations in *Quality-time* to get correct links in MS Teams notifications. Fixes #7614.

• If a URL with a hashtag was used to navigate to *Quality-time*, shared URLs would get an additional hashtag. Fixes #7761.

### 12.2.3  Added

• When collapsing metrics via the collapse button in the menu bar, also collapse expanded headers. Closes #4790.

• Add the possibility to include SonarQube violations by tag. Closes #6967.

• Allow for hiding report, subject and tag cards from dashboards via the settings panel. Closes #7482.

### 12.2.4  Changed

• The violations metric can now also be used to measure accessibility violations. Existing accessibility violation metrics have been changed to the violations metrics. Closes #562.

• Improve the icon for collapsing all expanded user interface controls and move the reset settings button from the settings panel to the menu bar so that it is easier to use. Closes #4395.

## 12.3  v5.5.0 - 2023-12-15

### 12.3.1  Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.3.2  Added

• Add a metric for tracking todo and fixme comments in source code. Closes #5630.

• Support version 4.0 of the OWASP Dependency Check DTD (OWASP Dependency Check version 9). Closes #7655.

### 12.3.3 Changed

- Update the SonarQube rules used to collect data for the long units, complex units, commented-out code, many parameters, and suppressed violations metrics. Note that this change may cause *Quality-time* to report more violations for these metrics. Closes #7653.

## 12.4 v5.4.0 - 2023-12-11

### 12.4.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.4.2 Fixed

- Newly added subjects would not be visible. Newly added metrics while filtering would not be visible. Fixes #7552.

### 12.4.3 Added

- Add an endpoint `api/v3/report/<report_uuid>/metric_status_summary` that returns a summary of the metric statuses for the specified report in JSON format. See the *API-documentation*. Closes #6146.

- Add endpoints `api/v3/<report|subject|metric|source>/search` that allow for search for objects by attribute value. See the *API-documentation*. Closes #7579.

## 12.5 v5.3.1 - 2023-11-08

### 12.5.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.5.2 Fixed

- The example URL in the popup for the Azure DevOps Server URL parameter would overflow the popup. Fixes #7144.

- When selecting a tag, metrics without tags would not be hidden. Fixes #7432.

- When hiding tags on the homepage, not only hide the metrics with the selected tags from the subject tables, but also from the subject cards in the dashboard. Fixes #7433.

- When expanding a metric, use the latest measurement instead of the oldest to show the measurement entities. Fixes #7435.

## 12.6 v5.3.0 - 2023-11-07

### 12.6.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.6.2 Added

- The visibility of tags can be configured via the Settings panel. When a tag is hidden, its tag card is also hidden in the dashboard. When all tags of metric are hidden, the metric itself is hidden. When all metrics in a subject are hidden, the subject is also hidden. Closes #7086.

- Support downloading Harbor JSON vulnerability reports from a JSON file. Closes #7147.

- In addition to hiding metrics that do not require immediate action, also allow for hiding all metrics so that only the dashboard of a report is visible. Closes #7211.

- Allow for changing and resetting settings per report. Closes #7213.

- Allow for showing all metrics on the reports overview page. Closes #7215.

### 12.6.3 Removed

- Tag reports. Clicking a tag on the reports overview page now filters the metrics by tag instead of opening a dynamically generated, read-only tag report. Makes #1301 obsolete. Closes #7214.

### 12.6.4 Changed

- Upgrade MongoDB from 6.0 to 7.0.

- Move the dark/light mode setting to a separate menu in the menu bar. Store the value of the dark/light mode in the local storage of the user's browser instead of in the URL as query parameter. Closes #7212.

### 12.6.5 Fixed

- When measuring the number of job runs within a time period using Jenkins as source, filters to include and exclude jobs would be ignored. Fixes #7172.

- Speed up the rendering of the dashboard to prevent exported PDFs from containing a partly rendered dashboard. Fixes #7208.

- Speed up the retrieval of measurements from the database. Fixes #7371.

- Don't crash the UI if the technical debt end date is not a valid date.

## 12.7 v5.2.0 - 2023-09-29

### 12.7.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.7.2 Fixed

- The "Reset all settings" button did not reset filtered tags. Fixes #6947.

### 12.7.3 Added

- Support Trivy JSON files as source for the security warnings metric. Closes #6927.

- The renderer component can be configured to use HTTPS, instead of HTTP, to connect to the proxy component.

## 12.8 v5.1.0 - 2023-09-05

### 12.8.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.8.2 Fixed

- Notify the user that editing is not possible when they open a tag report or time travel. Fixes #3380.

- When the frontend cannot reach the server, notify the user of the problem. When the server is reachable, notify the user as well. Fixes #3580.

- In the login dialog, don't tell the user credentials are invalid when in fact the server couldn't be reached. Fixes #4704.

- When changing the metric type, tags in the tags field would not be updated immediately. Fixes #5116.

- Use another date picker widget with better usability. Fixes #5446.

- Show warning message when the user session expires. Fixes #5327.

- When accessing Harbor with invalid credentials, Harbor returns data from public projects only. To prevent the invalid credentials from going unnoticed, explicitly check Harbor credentials before retrieving data. Fixes #6484.

- Fix the landing URL for Harbor artifacts. Fixes #6485.

- When changing the metric type, don't remove tags the user added to the metric. Fixes #6524.

- When time traveling, *Quality-time* would show measurements as being outdated. While technically correct, this doesn't constitute an issue that needs to be fixed and shouldn't be flagged as a problem. Fixes #6555.

- Don't make the vertical axis of trend graphs 200 high when the maximum value is 100 (for example, when displaying metrics with a percentage scale). Fixes #6621.

- Clicking "Reset all settings" in the Settings panel would result in a type error in the user interface. Fixes #6759.

### 12.8.3 Added

- Include the selected tags in the URL query parameter so the selected tags are retained when navigating between reports and when the URL is shared with another user. Closes #4551.

- When changing the status of measurement entities (violations, warnings, issues, etc.) also set the status end date. The default end dates can be changed by expanding the report title and navigating to the 'Desired reaction times' tab. For individual measurement entities, the default can be overridden by first changing the status and then changing or removing the status end date. Closes #5099.

- Add support for Cargo Audit, a linter for Rust Cargo.lock files for crates, as source for the 'security warnings' metric. Closes #6347.

- Show when a measurement entity (violation, warning, issue, etc.) was first seen by *Quality-time* so users can easily identify new entities. *Quality-time* did not keep track of this before, so for existing entities the 'first seen' date is missing. Closes #6351.

- Add SonarQube Swift rules for complex units, methods with too many lines and functions with too many parameters. Closes #6493.

- Add a note to the reference manual about how to configure Jenkins private tokens in *Quality-time*. Closes #6557.

### 12.8.4 Changed

- Rename the 'will be fixed' status of measurement entities (violations, warnings, issues, etc.) to 'fixed'. Closes #6556.

- Set the feature compatibility of the database to MongoDB 6.0 to prepare for the upcoming 7.0 release of MongoDB. Closes #6662.

## 12.9 v5.0.1 - 2023-06-26

### 12.9.1 Deployment notes

If your currently installed *Quality-time* version is v4.10.0 or older, please read the v5.0.0 deployment notes first.

### 12.9.2 Fixed

- When measuring velocity using Jira as source, the number of sprints to base velocity on can be changed via a parameter. Add help text to the parameter to explain how velocity is calculated. Fixes #6349.

- When measuring average issue lead time, users can configure how far back *Quality-time* should look for selecting issues. Add a tool tip to this lookback parameter explaining which issues are selected: "Issues are selected if they are completed and have been updated within the number of days configured". Fixes #6350.

- Jira issue statuses were not collected. Fixes #6435.

- Jira issues created from *Quality-time* would have an incorrect unit in the issue title and description. Fixes #6437.

## 12.10 v5.0.0 - 2023-06-23

### 12.10.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes first.

In this version of *Quality-time* the internal server component is no longer used. The notifier and collector components talk directly to the database, instead of using the internal server. This means that the docker composition **must** be changed:

- Remove the `internal_server` section.
- Rename the `external_server` section to `api_server` and make the following changes in that section:
    - Rename the `image` to `ictu/quality-time_api_server`
    - Rename `EXTERNAL_SERVER_PORT` to `API_SERVER_PORT`.
    - Rename `EXTERNAL_SERVER_LOG_LEVEL` to `API_SERVER_LOG_LEVEL`.
- In the `www` section:
    - Rename `EXTERNAL_SERVER_HOST=external_server` to `API_SERVER_HOST=api_server`.
    - Rename `EXTERNAL_SERVER_PORT` to `API_SERVER_PORT`.
    - Change the `depends_on: external_server` into `depends_on: api_server`.
- In the `collector` section:
    - Add the same `DATABASE_URL` environment variable as the `api_server` section has.
    - Remove the `INTERNAL_SERVER_HOST` and `INTERNAL_SERVER_PORT` environment variables.
    - Change the `depends_on: internal_server` into `depends_on: database`.
- In the `notifier` section:
    - Add the same `DATABASE_URL` environment variable as the `api_server` section has.
    - Remove the `INTERNAL_SERVER_HOST` and `INTERNAL_SERVER_PORT` environment variables.
    - Change the `depends_on: internal_server` into `depends_on: database`.
- In the `frontend` section:
    - Change the `depends_on: external_server` into `depends_on: database`.
- Update the version number of all images to `v5.0.0`.

See the example docker-compose.yml for an overview of all images.

See the deployment instructions for other configuration options.

### 12.10.2 Fixed

- Reports for tags with spaces in them could not be exported to PDF. Fixes #4765.
- Remove useless popup that appears when hovering tags in the dashboard. Fixes #5525.
- Don't attempt to collect all projects and repositories from Harbor when the user has configured filters on projects and/or repositories. Partially fixes #6220.
- Don't query SonarQube for the rule `plsql:PlSql.FunctionAndProcedureExcessiveParameters` when collecting data for the 'many parameters' metric. Sonarcloud.io gives a permission denied error when querying for that rule. Fixes #6277.

- Assume metric value 0 when it is omitted from SonarQube API response. Fixes #6346.

### 12.10.3 Added

- When using SonarQube security hotspots as source for the security warnings metric, allow for filtering hotspots by hotspot status ('to review', 'acknowledged', 'fixed', 'safe'). The default value of this parameter is to show hotspots 'to review' and 'acknowledged'. Closes #5956.

- Add axe analysis to Quality-time pipeline. Closes #5402.

- Add support for using the SARIF format as source for the 'violations' metric. This makes it possible to use Robocop as source for the violations metric. Closes #6314.

### 12.10.4 Changed

- Write a more descriptive issue summary and description for issues created from Quality-time. Closes #4747.

- Change the default sort order of dates in a quality report from descending to ascending. Closes #5998.

- Opt in to the new Chrome headless implementation for PDF-export, see https://developer.chrome.com/articles/new-headless/. Closes #6149.

- Added more SonarQube rules to report on 'suppressed violations' and 'long units' for the Python language. Added a SonarQube rule to report on 'many parameters' for the Kotlin language. Closes #6193.

- Throw a timeout error when collecting measurement data from a source takes longer than the configured time between measurement attempts (this can be changed via the `MAX_SLEEP_DURATION` environment variable which has a default value of 20 seconds). Partially fixes #6220.

- Upgrade MongoDB from 5.0 to 6.0. Closes #6358.

## 12.11 v4.10.0 - 2023-04-26

### 12.11.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.11.2 Fixed

- Fix a regression introduced in *Quality-time* v4.9.0 that causes all SonarQube security hotspots to be shown as part of the security warnings metric, instead of only the hotspots with status "to review". Fixes #5953.

### 12.11.3 Added

- Allow for using Harbor (tested with the Trivy security scanner) as source for the security warnings metric. Closes #3729.

- Add the time-ago or time-to-go to labels of date input fields. Closes #5123.

## 12.12  v4.9.0 - 2023-04-14

### 12.12.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.12.2 Fixed

- GitLab pipeline source up-to-dateness should take filters into account. Fixes #5181.

- In addition to making health check files group writeable for a configurable user to ease use in OpenShift, also make the user writing the files part of the group root. Further fixes #5310.

- The unit name of metrics in MS Teams notifications would not be rendered correctly. Fixes #5347.

- In trend graphs, prevent overlapping x-axis labels and make the y-axis length closer to the maximum measurement value. Closes #5786.

- Bars for subjects or tags with little metrics in the dashboard would get too narrow to be legible. Fixes #5792.

### 12.12.3 Added

- Add the possibility in metrics of type missing metrics to exclude certain subjects from a report. Closes #5119.

- Add SonarQube issue status rationale to entity data of suppressed violations. Closes #4926.

- Allow for filtering out specific Axe measurement entities through regular expressions. Closes #3328.

### 12.12.4 Changed

- Turn off request logging by the frontend container. Closes #5654.

- Get the max results Jira parameter from Jira itself instead of hard coding it. Closes #4789.

### 12.12.5 Removed

- Remove the colored background from trend graphs that was meant to indicate the status of the metric on certain points in time. Closes #5786.

## 12.13  v4.8.0 - 2023-03-13

### 12.13.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.13.2 Fixed

- Prevent negative time remaining in the calendar source. Fixes #5267.

- Tags were not printed correctly in the reference manual. Fixes #5282.

- Prevent users from entering negative desired response times. Fixes #5328.

- Update default value in calendar date source upon restart of external server. Partially fixes #5448.

- In addition to version 3.1, also support version 3.0 of the OWASP Dependency Check XSD. Closes #5586.

### 12.13.3 Added

- Add options to the technical debt dropdown menu to set and clear accepted technical debt including the technical debt target and technical debt end date. Closes #5125 and #5127.

- Add the option to configure a custom base image for the `www` container via the compose file. Closes #5312.

### 12.13.4 Changed

- Make health check files group writeable for a configurable user to ease use in OpenShift. Closes #5310.

- The test LDAP server is now directly using `bitnami/openldap`, instead of the custom built `quality-time_testldap`. Closes #5311.

## 12.14 v4.7.0 - 2023-01-23

### 12.14.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.14.2 Fixed

- Chrome would not show all historic data when displaying multiple dates. Fixes #4832.

- Make the "reset all settings" button also reset the report date. Fixes #4960.

- Break long words and URLs in the comments column, if necessary. Fixes #5118.

- Fixed typo's in description of metrics and sources: changed all instances of 'amount' where it should have been 'number'.

- Add the shared code components to the developer documentation. Fixes #5130.

- Fix link in the deployment documentation to the OpenShift README. Fixes #5235.

### 12.14.3 Added

- Support GitLab CI pipelines as source for the source-up-to-dateness metric. Closes #3927.

- Add a legend to the dashboard. Closes #4927.

- Allow for changing the log level of the backend containers via environment variables. See the *deployment manual*. Closes #4943.

- When using cloc as source for the LOC (Size) metric with the `--by-file` option, *Quality-time* can filter the cloc data by filename. When also setting the scale of the LOC metric to percentage, this enables tracking the percentage of test code. Closes #4958.

- Make the description of subjects, metrics, and sources in the UI of *Quality-time* link to the relevant documentation on Read-The-Docs. Closes #5121.

- Make the web page title reflect the report title. Closes #5129.

- Support version 3.1 of the OWASP Dependency Check XSD. Closes #5251.

### 12.14.4 Changed

- The default log level of the notifier, internal server, and external server components is now WARNING (was INFO). The default log level of the collector is unchanged and still WARNING.

- The metric "Lead time for changes" has been renamed to "Average issue lead time". Existing metric configurations are automatically updated. Closes #3500.

## 12.15 v4.6.1 - 2022-11-07

### 12.15.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.15.2 Fixed

- When showing multiple dates in a quality report and also time traveling, *Quality-time* would not retrieve the most recent measurements for the selected dates. Fixes #4793.

## 12.16 v4.6.0 - 2022-11-03

### 12.16.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

## 12.16.2 Fixed

- When configuring Azure DevOps as source for the source up-to-dateness metric, the tooltip says files get preference, but in reality code pipelines get preference. Updated the tooltip. Fixes #4685.

- When showing multiple dates in a quality report, the *Quality-time* frontend retrieves 28 weeks of measurements so that the maximum period (seven weeks times a four-week interval) can be displayed. However, the frontend would always get the most recent 28 weeks, even if the user was time traveling. Fixes #4705.

- When measuring failed or unused CI-jobs with GitLab as source, *Quality-time* would try to retrieve all jobs to determine which ones have failed or are unused. Retrieving all jobs would fail, however, if the GitLab instance used has a large amount (tens of thousands) of jobs. To fix this, only jobs that ran in the "look-back period" will be retrieved. By default, the look-back period is 90 days. The look-back period can be changed in the GitLab source tab, if so desired. Fixes #4737.

- Show all supported environment variables with their default values in the example docker-compose.yml. Fixes #4769.

## 12.16.3 Added

- Date pickers have a "today" button now. Closes #378.

- New metric "Lead time for changes", to keep track of the average lead time of changes completed in a certain time period in Jira and/or Azure DevOps. Closes #3501.

- When displaying multiple dates, also show the metric deadline overrun. The deadline overrun is the number of days a metric was not addressed within the desired response time. Closes #4538.

- Create an SBOM for every *Quality-time* release. Closes #4584.

- When adding an issue tracker to a report, allow for configuring an epic to use as parent issue for issues created from *Quality-time*. Closes #4614.

- When creating issues in the issue tracker from *Quality-time*, mention the user who created the issue in the issue description for traceability. Closes #4743.

## 12.16.4 Changed

- Reduce the considerable amount of logging that *Quality-time* generates by turning off the proxy access log, turning off the asyncio debug logging in the collector, increasing the log level to warning in the collector, and by passing `--quiet` to MongoDB. Closes #4736.

## 12.17 v4.5.0 - 2022-10-04

### 12.17.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.17.2 Fixed

- When changing the type of a metric with configured sources, retain the sources whether they support the new metric type or not. Flag the sources that do not support the new metric type as a 'configuration error' and explain how the user can fix the situation. Fixes #4443 and closes #4444.

- It looked like removing an issue tracker password or private token did not work: the user interface would still show a series of black dots in the password or private token field when later revisiting the issue tracker tab, even though passwords and private tokens were in fact being removed from the report. Fixes #4564.

- Adding issues did not work unless the selected issue tracker project and issue type supported labels. Fixes #4586.

### 12.17.3 Added

- New metric "Job runs within time period", to keep track of deployment frequency in Jenkins, GitLab and/or Azure DevOps. Closes #3498.

- Retrieve the possible projects and issue types for issue trackers from the configured issue tracker and present them via dropdown menus so the user can't enter non-existing projects or issue types. Closes #4586.

## 12.18 v4.4.0 - 2022-09-16

### 12.18.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.18.2 Fixed

- When changing the type of a metric with configured sources, don't remove the sources that support both the old and the new metric type. Fixes #4443.

- The button in the settings panel to 'Reset all settings' didn't work. Fixes #4527.

- Due to a small change to the loading spinner, *Quality-time* would no longer wait for the spinner to disappear before converting a report to PDF, causing the PDF to be empty for big reports. Fixes #4542.

- Clicking the *Quality-time* logo should navigate to the home page, but this did not work if the user traveled to a point in time where the report did not exist yet. Fixes #4526.

- Measurements of metrics with expired technical debt would not be merged in the database when unchanged, causing the number of measurements to grow much faster than normal, which in turn led to performance issues. To clean up the database, the unmerged measurements in the database are merged by a migration script that runs when the internal server component starts up. Before updating and removing measurements, the affected measurements are copied to two separate backup collections in the database, called `backup_updated_measurements` and `backup_deleted_measurements`. The log of the internal server component shows, per metric and in total, how many measurements will have been updated and deleted. The migration script can safely be run multiple times. Depending on the number of measurements in the database, the migration script takes a few seconds to a few minutes to run. Fixes #4554 and closes #4556.

### 12.18.3 Added

- When creating issues, include the rationale for the metric in the created issue so that people encountering the issue in the issue tracker have a better understanding of why the issue needs addressing. Closes #4232.

- When measuring violations with SonarQube as source, add the selected issue types (bug, vulnerability, and/or code smell) and severities (info, minor, major, critical, and/or blocker) to the landing URL so that the user sees only the relevant violations when navigating to SonarQube. Closes #4449.

- Make Jira issue identifiers uppercase when querying Jira for the status of issues, so that users don't have to enter uppercase Jira identifiers in *Quality-time*. Closes #4450.

- Allow for setting default labels on Jira issues created from *Quality-time*. The labels can be configured in the issue tracker tab under the report header. Closes #4468.

- Add a `target="_blank"` to URLs added in comments so that links open in a new tab by default. Can be prevented by adding an explicit `target="_top"` to URLs, for example: `This is an <a href="https://example.org" target="_top">example</a>`. Closes #4521.

### 12.18.4 Changed

- Move the contents of the "Notes on specific metrics" and "Notes on specific sources" from the user manual to the reference manual. Closes #4446.

## 12.19 v4.3.0 - 2022-08-24

### 12.19.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.19.2 Fixed

- In the development documentation, mention that the parameters of the 'Quality-time' source need to be changed when adding new metrics or sources to the data model. Fixes #4278.

- When time traveling, metrics would be incorrectly flagged as not having been measured recently. Fixes #4279.

- The documentation about exporting and importing reports via the API did not mention that the public key of the destination *Quality-time* instance has to be encoded before being passed as parameter to the export endpoint of the source *Quality-time* instance. Fixes #4313.

- The status start date of metrics would only be set on their first status *change*, not on their first status. Fixes #4327.

- When measuring failed or unused CI-jobs, Jenkins multi-branch pipeline jobs based on branch names containing forward slashes (for example 'feature/432-new-customer') could not be marked as false positive or won't fix. Fixes #4434.

### 12.19.3 Added

- Metrics that require action now have a desired reaction time. *Quality-time* shows the time left in the metric tables. The desired reaction times can be configured via the report header. Closes #4190.

### 12.19.4 Changed

- The metric detail tabs were flattened to simplify the user interface. The tabs that were previously nested tabs under 'Metric' are now top-level tabs. Closes #4390.

## 12.20 v4.2.0 - 2022-07-22

### 12.20.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

### 12.20.2 Fixed

- Don't add items when hitting enter while an add-item button with collapsed dropdown menu has focus. Fixes #4144.

- Don't close the dropdown menu of an add-item button when entering a space into the filter input field. Fixes #4147.

- The target column would show the technical debt target value instead of the target value when technical debt was accepted. Fixed by always displaying the target value and showing the technical debt target value in a popup on hovering the target value. Fixes #4171.

- The header of error messages in popups would be hard to read in dark mode. Fixes #4174.

### 12.20.3 Added

- *Quality-time* now ignores accepted technical debt not only when the technical debt end date has passed, but also when all issues associated with the metric have been done. When a metric has accepted technical debt that is being ignored, the target value is shown with a grey background and has a popup explaining why the accepted technical debt is being ignored. Closes #3935.

- Allow for filtering items in an add-item button dropdown menu without first clicking the filter input field. Closes #4114.

- Allow the configuration of multiple LDAP servers, see the deployment instructions for more information on configuring LDAP. Closes #4141.

- Mention the repository owner in the README.md and add contributing guidelines and a code of conduct to make the *Quality-time* repository comply with the ICTU GitHub policy. Closes #4142.

- Add an explanation of what version numbers are supported for the 'source version' and 'software version' metrics. Closes #4146.

- In the issue popups, and in the issue cards if so configured via the settings, show the due date, the release, and the sprint of issues if they have them. Closes #4186.

## 12.20.4 Changed

- As the "Collapse all metrics" button isn't a setting, move it from the settings panel to the menu bar. Closes #3382.

- Set the feature compatibility of the database to MongoDB 5.0 to prepare for the upcoming 6.0 release of MongoDB. Closes #4041.

- Move the metric parameters for technical debt, issues, and comments to a separate technical debt tab so that the metric configuration tab is less crowded with parameters. Closes #4165.

# 12.21 v4.1.0 - 2022-07-05

## 12.21.1 Deployment notes

If your currently installed *Quality-time* version is v4.0.0 or older, please read the v4.0.0 deployment notes.

## 12.21.2 Fixed

- Trend graphs would show yellow background areas as grey for metrics with the "more is better" direction. Fixes #1380.

- If for some reason measurements are not updated, the only way to detect this in the UI was to check the last measurement attempt in the popup of measurement values. Fixed by coloring measurement values red when the last measurement attempt is more than one hour ago. Fixes #4075.

- When using SonarQube as source for the 'suppressed violations' metric, the source URL SonarQube would direct users to a SonarQube page with only part of the information. Unfortunately, SonarQube does not allow for creating a filter that shows a combination of suppressed issues and suppressions in the source code. Partially fixed by sending users to a page with all issues. Fixes #4080.

- When showing multiple dates, the most recent measurement of one of the metrics would sometimes be shown as unknown despite not being unknown at all.

## 12.21.3 Changed

- Updated the link to the SIG-TÜViT "Evaluation Criteria for Trusted Product Maintainability" to the 2022 version. Closes #4065.

## 12.21.4 Added

- Allow for turning off evaluation of metric targets, making the metric "informative". Informative metrics are shown with an i (for information) icon on a blue background in the user interface, regardless of their measurement value (unless the source data could not be read or parsed, then they are shown as status unknown/white like all metrics). Closes #2051.

- Allow for creating issues from *Quality-time*. If an issue tracker is configured (note: including project key and issue type), users can create an issue for a metric by clicking the 'Create new issue' button in the metric configuration tab. *Quality-time* will use the issue tracker's API to create a new issue and will add the new issue's id to the tracked issue ids. The created issue is opened in a new browser tab for further editing. You may have to allow *Quality-time* to open popup windows in your browser. Closes #2931.

- Add a 'software version' metric that can be used to measure the version of the software analysed by sources. Sources currently supporting the 'software version' metric are: Performancetest-runner and SonarQube. Closes #3981.

- Add subjects to the reference documentation. Closes #4043.

- Add popups to the metric target, near target, and technical debt target fields in the metric configuration tab that visualize how measurement values are evaluated against the metric target values. Closes #4064.

## 12.22  v4.0.0 - 2022-06-15

### 12.22.1 Deployment notes

If your currently installed *Quality-time* version is not v3.37.0, please read the v3.37.0 deployment notes first.

This version of *Quality-time* splits the server component into two: an external server component serving the external API and an internal server component serving the collector and notifier components. This means that the docker composition **must** be changed:

- Rename the `server` service to `external_server`.

  - Use the image `ictu/quality-time_external_server`.

  - If used, rename the `SERVER_PORT` environment variable to `EXTERNAL_SERVER_PORT`.

- Add an `internal_server` section.

  - Use the image `ictu/quality-time_internal_server`.

  - If you want to change the default port (5002) of the internal server, add an `INTERNAL_SERVER_PORT` environment variable.

  - Add the same `DATABASE_URL` environment variable as the external server has.

  - Set the `LOAD_EXAMPLE_REPORTS` environment variable to the same value as the external server has (True or False).

  - Add a `depends_on: database`.

- In the `www` (proxy) section:

  - Change `SERVER_HOST` and `SERVER_PORT` (if used) to `EXTERNAL_SERVER_HOST` and `EXTERNAL_SERVER_PORT`.

  - Change the `depends_on: server` to `depends_on: external_server`.

- In the `frontend` section:

  - Change the `depends_on: server` to `depends_on: external_server`.

- In the `collector` section:

  - Change `SERVER_HOST` and `SERVER_PORT` (if used) to `INTERNAL_SERVER_HOST` and `INTERNAL_SERVER_PORT`.

  - Change the `depends_on: server` to `depends_on: internal_server`.

- In the `notifier` section:

  - Change `SERVER_HOST` and `SERVER_PORT` (if used) to `INTERNAL_SERVER_HOST` and `INTERNAL_SERVER_PORT`.

  - Change the `depends_on: server` to `depends_on: internal_server`.

- Update the version number of all images to `v4.0.0`.

See the example docker-compose.yml for an overview of all images.

See the deployment instructions for other configuration options.

## 12.22.2 Added

- The Performancetest-runner HTML report now reports the breaking point as the absolute number of virtual users as well as percentage of the maximum number of virtual users. This allows the 'scalability' metric to support the count scale in addition to the already supported percentage scale. Closes #3980.

## 12.22.3 Fixed

- *Quality-time* would use the Jira issue picker endpoint to suggest issue ids to the user when typing text in the issue id field. However, the Jira endpoint uses the Jira interaction history of the user contacting the endpoint to make suggestions. In the case of *Quality-time*, this would be the user configured in the issue tracker. If this is a system user without any Jira interaction history, no suggestions would be made. Fixed by using the Jira search endpoint instead of the Jira issue picker endpoint. Fixes #4017.

- The user interface would sometimes crash when navigating to the reports overview right after editing an input field on a report page. Fixes #4034.

# 12.23 v3.37.0 - 2022-06-07

## 12.23.1 Deployment notes

If your currently installed *Quality-time* version is not v3.36.0, please read the v3.36.0 deployment notes.

## 12.23.2 Fixed

- When measuring the 'missing metrics' metric, *Quality-time* was still using an old endpoint to get the data, resulting in a parse error. Fixes #3855.

- Wait for the spinner to disappear before converting a report to PDF. Fixes #3932.

## 12.23.3 Changed

- Verify SSL certificates when checking secure (HTTPS) URLs entered by the user.

- The 'Add subject', 'Add metric', and 'Add source' buttons now have a dropdown menu with the available subject, metric, and source types. If there are more than five options, the dropdown menu can be filtered to reduce the number of options. When adding metrics, the dropdown menu only shows the metric types that can measure the subject. When adding sources, the dropdown only shows the source types that support the metric. Closes #3718.

### 12.23.4 Added

- When a Jira instance has been configured as issue tracker, use the Jira instance to suggest issues ids when the user starts typing in the metric issue id field. Closes #3561.

- Explain in the documentation what information needs to be present in Axe-core JSON reports to support the 'source up-to-dateness' and 'source version' metrics. Closes #3903.

## 12.24 v3.36.0 - 2022-05-16

### 12.24.1 Deployment notes

If your currently installed *Quality-time* version is not v3.35.0, please read the v3.35.0 deployment notes first.

Because of the new renderer component (see below) the following environment variables are obsolete and can be removed from the `docker-compose.yml`:

- `server:`
    - `PROXY_HOST`
    - `PROXY_PORT`
    - `RENDERER_ACCESS_KEY`
- `renderer:`
    - `ALLOW_HTTP`

If the proxy does not have the default name/port `www:80`, the renderer needs to told how to reach the proxy by means of the `PROXY_HOST` and `PROXY_PORT` environment variables. For example:

```
renderer:
  image: ictu/quality-time_renderer:v3.36.0
  environment:
    - PROXY_HOST=qt
    - PROXY_PORT=8080
```

### 12.24.2 Fixed

- Adding values to input fields in the user interface that allow for multiple values didn't work. Fixes #3801.

- The detail information of the 'metrics' metric with *Quality-time* as source would show "NaN" (not a number) as the value of the measurements and targets of the measured metrics. Fixes #3811.

- Optional Jira parameters to specify a Jira field name or id were mandatory in practice due to the bug fix for issue #3714. Fixes #3845.

### 12.24.3 Changed

- Use a custom JavaScript API-server to wrap Puppeteer, instead of the unmaintained URL to PDF Microservice, to support the rendering of PDF reports. Closes #3767. Fixes #3297.

### 12.24.4 Added

- Show the rationale for the status of measurement entities in the measurement entity table. Closes #3788.

## 12.25 v3.35.0 - 2022-05-09

### 12.25.1 Deployment notes

To upgrade to this version of *Quality-time* your currently installed version needs to be at least version 3.32.0. If your currently installed version is older, you need to first install v3.32.0, v3.33.0, or v3.34.0 before installing v3.35.0.

Background information: *Quality-time* uses MongoDB as database component. A MongoDB instance is either backward-compatible with the previous MongoDB version or forward-compatible with a next MongoDB version. To configure this, the MongoDB feature compatibility version has to be set in the database. *Quality-time* has been using MongoDB v4.4 for a while. Up until *Quality-time* v3.32.0 the database was backward-compatible with MongoDB v4.2. Starting from *Quality-time* v3.32.0 the database has been made forward-compatible with MongoDB v5.0. This ensures that if *Quality-time* v3.35.0 is not functioning properly with MongoDB v5.0, a rollback to a previous version of *Quality-time* (but not older than v3.32.0) with MongoDB v4.4 is possible.

### 12.25.2 Fixed

- When using Axe CVS as source for the accessibility violations metric, the "nested-interactive" violation type would be ignored by *Quality-time*. Fixes #3628.

- When using Gatling as source for the source version metric, the source version would not be found, when the version number was not present on the first line of the simulation.log. Fixes #3661.

- The data model for SonarQube contained some rules, which no longer exist. These were kept for backwards compatibility but now cause the SonarQube web interface to "freeze". These rules are removed. Fixes #3706.

- When measuring 'manual test duration', show an error if the manual test duration field name or id does not exist in Jira, instead of silently ignoring the issue and reporting zero minutes. Fixes #3714.

- When showing a historic report in combination with measurements at multiple dates, *Quality-time* would use the latest report to decide which metrics to retrieve the measurements for, instead of the report at the date shown. Fixes #3722.

- When measuring 'performancetest scalability' with the Performancetest-runner as source, don't consider a trend breakpoint of 100% to be an error. Fixes #3787.

## 12.25.3 Changed

- Don't render minutes as hours:minutes in the GUI, but as an integer, to prevent confusion about what each number represents (hours, minutes, seconds?). Closes #3577.

- In the settings panel under the "Sort column" setting, clicking the same column multiple times now alternates between ascending and descending sort order. This makes the setting consistent with how column headers behave. It also removes the need for a separate "Sort direction" setting in the settings panel. Closes #3646.

- The database component was upgraded to MongoDB 5.0.7. **Note that to upgrade to this version of *Quality-time* your previous version needs to be at least version 3.32.0**. Closes #3647.

- The proxy component now uses Nginx 1.21.6 instead of Caddy. Closes #3687.

## 12.25.4 Removed

- The deprecated (since version 3.24.0) API endpoints `/api/v3/reports` and `/api/v3/tag_report` have been removed. Closes #1416.

- The Axe CSV "violation type" parameter that could be used to select which violation types to count has been removed to help fix #3628. The parameter was already impractical to ignore certain violation types because it would require the user to select all violation types except the ones to be ignored. Also, if Axe adds new violation types, *Quality-time* would need to be updated to prevent it from ignoring the new violation types.

## 12.25.5 Added

- Add support for Jira personal access token. Closes #592.

- Add a rationale to each metric explaining why one would want to measure it. The rationale is listed in the metrics overview in the documentation and is available as popup dialog in the user interface. Closes #3578.

- Add a 'minimum status duration' parameter to the 'metrics' metric to allow for counting metrics only when they have had the same status for a minimum number of days. Closes #3681.

- Add support for SARIF JSON files as source for the 'security warnings' metric. Closes #3730.

# 12.26 v3.34.0 - 2022-03-27

## 12.26.1 Fixed

- The visibility settings of the issue attributes summary, creation date, and update date were not applied to tag reports. Fixes #2827.

- Subject subtitles would partly obscure the header of their subject's metrics table. Fixes #3443.

- GitLab job artifacts archives downloaded via the GitLab API would not be recognized as zipped archives. Fixes #3478.

- GitLab uses non-standard version numbers like "14.5.2-ee"; be prepared. Fixes #3519.

- In the settings panel, disable options that are not relevant. E.g. if multiple dates are shown, don't allow for toggling the visibility of columns that are never visible if multiple dates are shown. Fixes #3521.

- Multiple choice fields such as fields for tags and issue identifiers would not save newly entered items on focus loss. Fixes #3560.

## 12.26.2 Changed

- Whether the issue attributes summary, creation date, and update date are visible is no longer stored in the report, but part of the settings. Consequently, the visibility of the issue attributes can be changed via the settings panel. Closes #3485.

## 12.26.3 Added

- Highlight metrics on hover to make it easier to indicate which metric is being discussed in online meetings. Closes #3132.

- Add dark mode. Switch between dark and light mode via the settings panel. Closes #3159.

- Show an error message when the user adds issue ids to metrics in a report that doesn't have an issue tracker configured. Closes ##3228.

- Add the summary of an issue to its popup, and, optionally, to the issue status in the metric table. Closes #3368.

# 12.27 v3.33.0 - 2022-02-13

## 12.27.1 Fixed

- Don't reset settings such as the visible columns and the current report date when navigating between reports. Fixes #3410.

## 12.27.2 Changed

- Use a menu instead of a button in the settings panel to hide metrics that don't require action.

- Use a separate unit column in the metrics tables, regardless of how many dates are shown. This makes the one-date and multiple-dates views more similar. It also saves screen real estate as the unit is no longer repeated in both the measurement and target column. Closes #3268.

- Close the settings panel when it loses focus. Closes #3381.

## 12.27.3 Added

- Add a setting to the settings panel to allow for reversing the order of the date columns in the metric tables. Closes #2928.

- Add a button to the settings panel to reset all settings to their default values. Closes #3183.

- Allow for adding an end date to the status of measurement entities. After the end date passes, the measurement entity is considered to be 'Unconfirmed' again. This makes it possible to mark an entity as e.g. won't fix for a certain period of time. Closes #3332.

- Add a 'time remaining' metric that measures the number of days remaining until a future date. Use the 'calendar date' source to set that date. Closes #3366.

- Add the metric sort column to the settings panel so that the button to reset all settings also resets the sort column. Closes #3412.

## 12.28  v3.32.0 - 2022-01-24

### 12.28.1  Fixed

- If a metric did not have sources (with all mandatory parameters configured), the status of issues would not be collected. Fixes #3221.

- Allow for specifying zip files as Gatling source. Fixes #3226.

- Remove spaces from file paths in OWASP Dependency Check security warnings before applying the regular expressions to remove variable parts from the file paths. Unfortunately, this may change the key of some OWASP Dependency Check security warnings, causing the status (false positive, won't fix, etc.) of the warning in *Quality-time* to be lost. Fixed as part of #3259.

### 12.28.2  Changed

- Use the 'number of dates' menu (now located in the settings panel, see #3248 below) to switch between what used to called 'trend' view (multiple dates) and 'details' view (one date). If you export PDFs via the API, you may need to change the URL parameters: `trend_table_nr_dates` is now called `nr_dates` and `trend_table_interval` is now called `date_interval`. Also, `trend_table_interval` was a number of weeks, `date_interval` is a number of days. See the documentation on PDF-exports via the API. Closes #3206.

- Make the subject title and header row of subject tables 'sticky', meaning that the title and header rows stay visible until the whole table scrolls off-screen. Closes #3219.

- Move the contents of the hamburger menu to a settings panel that can be brought into view via the menu bar. Closes #3248.

### 12.28.3  Added

- Add a button to the settings panel to collapse all expanded metrics at once. Closes #3133.

- Show the key of OWASP Dependency Check security warnings in the measurement entity details to allow for verification of the regular expressions used to remove variable parts from file paths. Closes #3259.

## 12.29  v3.31.0 - 2022-01-13

### 12.29.1  Fixed

- The dropdown menu for determining the scope of parameter changes (Apply change to source/metric/etc.) would not appear when clicking the "Apply change to" part of the label. Fixes #3112.

- OWASP ZAP uses a non-standard versioning scheme (D-year-month-day) for its weekly versions, be prepared. Fixes #3117.

- Show a more informative error message if no merge request information can be retrieved from GitLab for the 'merge requests' metric. Fixes #3166.

- The hamburger submenus were only partially clickable. Also make the hamburger menu popup on hover for better discoverability. Fixes #3181.

- When sorting metrics by status, order by how urgently action is required: 'unknown' (white), 'target not met' (red), 'near target met' (yellow), 'technical debt accepted' (grey), 'target met' (green). Fixes #3184.

- Reset the edit scope of source parameters to 'source' after each edit. Fixes #3198.

## 12.29.2 Changed

- Use users' full name instead of their username in the change log so it's easier to see who changed what. Closes #2930.

- Improved tooltips for the measurement column in the metrics details table. Closes #3171.

## 12.29.3 Added

- Show the source, comment, issues, and tags columns in the metric trend view. Closes #2414, #3203, #3037, and #3202.

- Allow for copying permanent links to metrics, subjects, and reports via the new 'Share' tabs. Closes #2925.

- Allow for adding comments to the reports overview, to reports, and to subjects. Expand the title of the reports overview, report, or subject to enter comments. Entered comments are shown below the title of the reports overview, report, or subject. Basic HTML (headers, bold, italic, links, etc.) is allowed. Closes #2926.

- Explain in the documentation how to include the correct report URL in the footer when exporting reports to PDF via the API. Closes #2954.

- In addition to the 90th percentile, also allow for evaluating the 95th and 99th percentile transaction response time when using JMeter CSV or JSON as source for the 'slow transactions' metric. Closes #3084.

- Add support for Gatling as source for the 'slow transactions', 'tests', 'performancetest duration', 'source up-to-dateness', and 'source version' metrics. Closes #3085, #3086, #3087, #3088, and #3089.

- The dependencies in OWASP Dependency Check reports do not always have unique keys. However, *Quality-time* needs dependencies to be uniquely identifiable to detect whether the dependencies change between measurements. If needed, *Quality-time* generates keys for dependencies itself, based on the dependencies' file paths. If for some reason the file path changes between measurements, however, the key changes as well. To remediate this, allow for ignoring parts of file paths using regular expressions, when measuring 'dependencies' or 'security warnings' with OWASP Dependency Check as source. Closes #3099.

- After changing multiple source parameters at once, show a toaster message with the number of sources updated. Closes #3137.

- In the metric trend view (selectable via the hamburger menu), allow for setting the interval between dates to one day, in addition to one or more weeks. Closes #3182.

- Allow for sorting the metrics when displayed in the trend view. Closes #3207.

## 12.29.4 Removed

- Don't hide the tag pie charts in the dashboard when the user hides the tags column: it's not obvious that hiding the tags column would have that effect. Closes #3202.

## 12.30 v3.30.2 - 2021-12-19

### 12.30.1 Fixed

- A bug in the Quality-time API would cause the sparkline graphs to draw the recent measurements as if they all happened on the current day and cause the notifier to send notifications to MS Teams every minute. Fixes #3071 and #3073.

## 12.31 v3.30.1 - 2021-12-17

### 12.31.1 Fixed

- The notifier would crash. Fixes #3065.

## 12.32 v3.30.0 - 2021-12-16

### 12.32.1 Fixed

- Don't show a cursor in input fields when they are read only, e.g. because the user hasn't logged in, is time traveling, or is viewing a tag report. Fixes #2933.
- When time traveling, *Quality-time* would show deleted reports after their deletion date. Fixes #2997.
- Zero values were not shown in the measurement detail tables. Fixes #3008.

### 12.32.2 Added

- Support JMeter CSV output as source for the 'performancetest duration', 'slow transactions', 'tests', and 'source up-to-dateness' metrics. Closes #2965, #2966, #2967 and #2010.

## 12.33 v3.29.0 - 2021-12-03

### 12.33.1 Fixed

- Time travel was broken: *Quality-time* would show the current measurement value regardless of the date selected. Fixes #2958.

## 12.33.2 Changed

- More flexible parsing of Axe-core JSON files to account for the different ways people aggregate Axe-core output into one JSON file, for the 'source up-to-dateness' and 'version' metrics. Closes #2910.

- Upgrade MongoDB to version 4.4. **Note that to upgrade to this version of *Quality-time* your previous version needs to be at least version 3.24.0**. Closes #2911.

## 12.33.3 Added

- Support JMeter JSON output as source for the 'slow transactions' and 'tests' metrics. Closes #2766, #2936, and #2950.

## 12.34 v3.28.0 - 2021-11-22

### 12.34.1 Fixed

- Use the correct report URL in the footer of reports exported to PDF. Fixes #2750.

- The ARIA (Rich Internet Application Accessibility) label of status pie charts would report the wrong number of red metrics. Fixes #2779.

- The security warnings in OWASP ZAP reports do not have unique keys. However, *Quality-time* needs security warnings to be uniquely identifiable to detect whether the list of warnings changes between measurements. Therefore, *Quality-time* generates keys for OWASP ZAP security warnings itself. Unfortunately, the key that *Quality-time* generated, was not guaranteed to be unique. Fixes #2852.

- Multiple edits with the same description would show up as one entry in the changelog. Fixes #2893.

- *Quality-time* was using a broken and deprecated API endpoint to collect data for the 'metrics' metric. Fixes #2897.

### 12.34.2 Added

- Allow for using HTML reports generated by axe-html-reporter as source for the accessibility violations metric. Closes #2813.

## 12.35 v3.27.0 - 2021-10-25

### 12.35.1 Fixed

- Use ❯ instead of / to create subject breadcrumbs in tag reports, so they are consistent with breadcrumbs in copy and move button dropdowns.

- Prevent "Warning: `Infinity` is an invalid value for the `width` CSS style property." messages in the console log.

- Prevent `Error: Problem parsing d="M-2592.670630208333,NaNL-2592.670630208333,` `...` messages in the console log. These messages were caused by trying to create a sparkline graph for the source version metric. Fixes #2663.

- Use submenus in the hamburger menu to make it shorter and prevent menu items from being drawn off-screen. Fixes #2666.

- Measurement entities marked as false positive or fixed weren't being crossed out. Fixes #2739.

## 12.35.2 Changed

- Performance improvements. Closes #2692 and #2695.

- Make the metric tables use less vertical space when in details view. This allows for more metrics to fit on the screen. It also makes the vertical space used by the details view and the trend view more similar.

- Use a lightning bolt icon for metrics that don't meet their target value, to suggest danger and/or risk. The previously used x-shaped icon is typically associated with closing things, and thus less appropriate.

## 12.35.3 Removed

- Remove the box around dashboards to reduce visual clutter.

- Remove the 'scroll to dashboard' button; it's not really needed (users can use the home button) and an unusual feature.

# 12.36 v3.26.0 - 2021-10-04

## 12.36.1 Changed

- Use tabs to better organize the settings that are accessible via expandable headers. Also add icons to the tabs.

- More flexible parsing of Axe-core JSON files to account for the different ways people aggregate Axe-core output into one JSON file. Closes #2657.

## 12.36.2 Added

- Make metrics expandable in the metric trend table view so metrics and sources can be configured in the trend table view as well as in the metric detail view. Closes #2176

- Allow for adding issues to metrics to e.g. track progress on resolving technical debt. Closes #2215 and #2628.

- Make report title in the footer a URL to the report itself. Closes #2532.

- Sentiment metric. Closes #2533.

- Add documentation on how to move *Quality-time* from location to another. Closes #2538.

# 12.37 v3.25.0 - 2021-09-06

## 12.37.1 Fixed

- Don't use the UUIDs generated by OpenVAS to compare OpenVAS security warnings because they are not stable across OpenVAS scans. Fixes #2544.

- Don't mention the admin user in the "Trying it out" section of the documentation as the admin user currently doesn't exist in the osixia/docker-openldap image. Fixes #2565.

### 12.37.2 Added

- Add pagination support for Jira so queries that result in more than 500 results are retrieved completely. Closes #2386.

- Allow Robot Framework reports as source for the test cases metric. Closes #2534.

- Allow Jenkins test reports as source for the test cases metric. Closes #2543.

### 12.37.3 Removed

- Don't send notifications about metrics having the same status for three weeks: it's not useful enough. Closes #2529.

## 12.38 v3.24.0 - 2021-08-23

### 12.38.1 Fixed

- In addition to "low", "medium", "high", and "critical", the OWASP Dependency Check may report vulnerabilities with severity "moderate". Allow for using this severity for filtering vulnerabilities. Fixes #2337.

- When measuring 'missing metrics', count missing metric types per subject instead of per report. Fixes #2352.

- Add subject name to metrics in MS Teams notifications, so it's clear which metric changed status when different subjects have metrics with the same name. Fixes #2353.

- After copying a metric, subject, or report, the same item could not be copied again. Fixes #2364.

- After reloading a report, edit controls are shown even when the user has no edit permission. Fixes #2373.

- The security warnings in OWASP ZAP reports do not have unique keys. However, *Quality-time* needs security warnings to be uniquely identifiable to detect whether the list of warnings changes between measurements. Therefore, *Quality-time* generates keys for OWASP ZAP security warnings itself. Unfortunately, the key that *Quality-time* generated, was not guaranteed to be unique. Fixes #2429.

- JUnit XML files may have empty test suites, be prepared. Fixes #2507.

### 12.38.2 Added

- When measuring merge requests with GitLab Premium as source, the merge requests can be filtered by approval state. Closes #1979.

- Include the key of Jira issues in the measurement details of the 'issues', 'manual test duration', 'manual test execution', and 'user story points' metrics. Prepares for #2139.

- Add a test cases metric to count the number of test cases that have been executed, possibly limited to passed, failed, and/or skipped test cases. See the *reference manual*. Closes #2139.

- Include the tests from TestNG XML reports in the measurement details of the 'tests' metric. Closes #2388.

- The API has a new endpoint in REST style, `/api/v3/report`, to retrieve all reports.

- Publish *Quality-time* documentation at Read the Docs.

### 12.38.3 Changed

- The API endpoint `/api/v3/report/{report_uuid}` now also supports tag UUIDs.

- Use react-toastify for toast messages instead of the unmaintained react-semantic-toasts. Fixes #2290.

- Use semantic-ui-calendar-react-17 for date pickers instead of the unmaintained semantic-ui-calendar-react. Fixes #2291.

### 12.38.4 Deprecated

- The API endpoint `/api/v3/reports` is deprecated. Use `/api/v3/reports_overview` and `/api/v3/report` instead.

- The API endpoint `/api/v3/tag_report` is deprecated. Use `/api/v3/report/{report_uuid}` instead.

### 12.38.5 Removed

- Remove the search function as its functionality is limited and users indicate they don't use it. Closes #2305.

## 12.39 v3.23.3 - 2021-06-29

### 12.39.1 Fixed

- Work around a bug in `aiohttp` that causes GitLab connections to hang and timeout when the GitLab data is paginated. Fixes #2231.

- The report dashboard layout couldn't be changed. Fixes #2305.

## 12.40 v3.23.2 - 2021-06-17

### 12.40.1 Fixed

- To prevent overloading *Quality-time*, the collector now measures a limited number (30 by default) of metrics each time it wakes up. If there are more than 30 metrics to measure, these get postponed to the next wake-up. To compensate, the collector wakes up more often (every 20 seconds instead of every 60 seconds) to see whether metrics need measuring. Metrics recently edited by users get priority.

- Fix a performance regression in the collector component, introduced in v3.23.0.

- Allow for importing reports with metrics that have no scale or addition attribute. Fixes #2262.

## 12.41 v3.23.1 - 2021-06-13

### 12.41.1 Fixed

- Allow for importing reports with metrics that have no tags. Fixes #2262.

## 12.42 v3.23.0 - 2021-06-09

### 12.42.1 Fixed

- Allow for importing reports with unencrypted credentials. Fixes #2238.
- Axe CSV files may contain duplicate accessibility violations. These duplicated violations couldn't be marked as false positive, won't fix, etc. Fixed by ignoring duplicate violations. Fixes #2232.

### 12.42.2 Added

- Add 'Missing metrics' metric. Closes #1477.

### 12.42.3 Removed

- Drop support for Wekan source for Issue and Source-up-to-dateness metrics. Closes #2229.

## 12.43 v3.22.0 - 2021-05-26

### 12.43.1 Fixed

- When the collector posts new measurements to the server, the server looks up previous measurements in the database to see if the measurement value has changed. This lookup was slow due to a missing index on the measurements collection. Fixes #2155.
- Edit controls were not hidden or made read-only in the UI when time traveling and after logout. Fixes #2170.
- Measuring security warnings with Anchore as source would throw a parse error if the source was an unzipped Anchore JSON file. Fixes #2177.
- When all users are allowed to edit reports, no users would be able to edit measurement entities to mark them as false positive, fixed, etc. Fixes #2179.

### 12.43.2 Removed

- Since subjects have a trend table view, there is little added value in also having a trend table per metric. Remove the trend table view for individual metrics. Closes #2174.

### 12.43.3 Added

- Add support for Robot Framework v4 XML files. Closes #2136.

## 12.44 v3.21.0 - 2021-04-25

### 12.44.1 Fixed

- Prevent (rare) crashes of the UI when switching tabs in the metric details. Fixes #1873.

- Don't assume that because GitLab returns jobs sorted by ID, they are also sorted by date. Fixes #2036.

- Prevent timeouts when collecting failed CI-jobs or unused CI-jobs from GitLab by removing the limit on open connections. Fixes #2037.

- *Quality-time* wouldn't recognize zip files in URLs with query strings (e.g. `https://git.example.org/foo.zip?job=bar`). Fixes #2057.

### 12.44.2 Added

- Separate permissions for editing measured entities. Closes 1842.

- When measuring accessibility violations with Axe-core as source, allow for counting incomplete, inapplicable, and passed rule checks, in addition to violations. Closes 2026.

### 12.44.3 Removed

- Remove the 'random number' source. Closes #2038.

- It's no longer possible to make *Quality-time* wait before sending a notification. Closes #2039.

## 12.45 v3.20.0 - 2021-04-07

### 12.45.1 Fixed

- When a measurement detail, such as a violation or failed CI-job, is no longer reported by the source, remember its status (confirmed, false positive, won't fix, etc.) for three weeks so that if the detail reappears, its status is reapplied as well. Fixes #1867.

### 12.45.2 Added

- Reports can be exported and imported via API. Partially fixes 1693.

- When measuring security warnings with OWASP ZAP as source, allow for counting alert types as security warnings as opposed to alert instances. Closes #1902.

- Added a new metric 'source version' that can be used to measure the version of a source and compare it with a minimum or maximum version number. See the *metrics and sources overview* for a list of sources that support this metric. Closes #1904.

- Added support for the Anchore Jenkins plugin as source for the 'security warnings' and 'source up-to-dateness' metrics. Closes #1980.

- Added support for Axe-core JSON files (or zips with Axe-core JSON files) as source for measuring accessibility violations. Closes #1981.

### 12.45.3 Removed

- The axe-selenium-python source type has been removed. To read JSON files produced by axe-selenium-python, use the new Axe-core source type. This works, as the JSON format produced by axe-selenium-python is basically the same as the format produced by Axe-core. Sources of type axe-selenium-python in existing reports are automatically changed into Axe-core. No user action is needed.

## 12.46 v3.19.1 - 2021-02-28

### 12.46.1 Fixed

- The *Quality-time* API would return an internal server error (status code 500) if the database contains a source of a type that is no longer supported. Fixes #1732.

- When opening *Quality-time*, don't show the user as logged in when their session has expired. Fixes #1927.

- When measuring the up-to-dateness of a folder in GitLab with more than 100 files or subfolders, *Quality-time* uses the GitLab pagination API to retrieve the files and subfolders in batches of 100 each. However, due to a bug, the collector component would get stuck in a loop, retrieving the same files over and over again. Fixes #1938.

- Notifier would not work for recently set notification destinations. Fixes #1946.

## 12.47 v3.19.0 - 2021-02-21

### 12.47.1 Fixed

- When measuring size (lines of code) with SonarQube as source, some languages couldn't be ignored. Fixes #1818.

- The trend table view would format durations incorrectly, e.g. 5 minutes would be displayed as '5 hours'. Fixes #1900.

- Anchore security warnings have no hash. *Quality-time* would create a hash based on the security warning's CVE and affected package. However, if the Anchore source consists of a zip file with multiple reports, multiple combinations of the same CVE and package may be present. Add the report filename to the hash to make it unique. Fixes #1907.

- When measuring security warnings with SonarQube as source, allow for filtering security hotspots by review priority. Fixes #1910.

- The trend table view would sometimes, erroneously, show recent data as missing. Fixes #1924.

### 12.47.2 Added

- Add a 'merge requests' metric and add Azure DevOps and GitLab as possible sources. Closes #1644.

## 12.48 v3.18.0 - 2021-02-03

### 12.48.1 Fixed

- Prevent notifier crashes. Fixes #1830 and #1878.
- Logical sorting for entity status options. Fixes #1841.

### 12.48.2 Added

- Subjects can now show metrics in a trend table view. Use the hamburger menu to switch between the default detail view and the trend table view. Closes #1649.
- When measuring failed pipelines or unused pipelines with Azure DevOps as source, allow for including pipelines by name or by regular expression. Prepares for #1804.
- Allow using Azure DevOps pipelines as source for the 'source up-to-dateness' metric. Closes #1804.

## 12.49 v3.17.1 - 2021-01-24

### 12.49.1 Fixed

- When measuring failed jobs with GitLab as source, *Quality-time* would get the 100 most recent jobs instead of the 100 most recent *failed* jobs. Fixes #1813.
- Make sure the notifier component does not crash when a metric has no recent measurements. Fixes #1831.
- Adding or removing the OWASP ZAP "Parts of URLs to ignore" parameter would fail. Fixes #1846.

## 12.50 v3.17.0 - 2021-01-17

### 12.50.1 Changed

- Wrap the database (MongoDB), proxy (Caddy) and renderer (url-to-pdf-api) in *Quality-time* images, so these components have the same version number as the other components and don't need to be updated by downstream maintainers separately. Note that your Docker composition needs to be changed once to use these new *Quality-time* images. See the example docker-compose.yml. Closes #1770.
- Increase render timeout so that larger reports can be exported to PDF. Closes #1771.
- Add no-cache option for /api/v3/logo to the Caddy configuration.

## 12.50.2 Fixed

- The frontend would reload the change log every time the server notified the frontend about the number of measurements, causing unnecessary updates of the UI. Fixes #1555.

- When the user opens a report in the frontend, don't send unneeded data to the frontend. Fixes #1764.

- Don't crash the notifier when a metric has an unknown (white) status. Fixes #1802.

- Some dependencies in the OWASP Dependency Check report have no hash. In those cases *Quality-time* would create a hash based on the file path of the dependency. However, file paths aren't necessarily unique across dependencies. Add the file name to the hash to make it unique. Fixes #1819.

## 12.50.3 Added

- Add the tags of accessibility rules to the detail information of axe-selenium-python sources. Closes #1751.

- Allow for filtering axe-selenium-python accessibility violations by tag. Closes #1752.

# 12.51 v3.16.0 - 2020-12-12

## 12.51.1 Added

- Show since when a metric has its current status via a popup over the status icon. Closes #1091.

- When adding a notification destination, it's possible to specify how long *Quality-time* should wait before sending a notification. If more notifications happen during the wait period, they will be bundled. Also see the *user manual*. Partially implements #1223.

- Notifications will now be sent for all status changes. Previously, notifications were only sent when a metric either turned red or white. Partially implements #1223.

- Allow using Jenkins jobs as source for the 'source up-to-dateness' metric. Closes #1680.

## 12.51.2 Fixed

- Accepting the technical debt of a metric with one or more unreachable sources would set the metric status to 'technical debt target met' instead of 'unknown'. Fixes #1636.

- Collapsing an expanded metric would sometimes result in a crash of the frontend. Fixes #1717.

## 12.51.3 Changed

- MongoDB was upgraded to 4.2.11. No migration steps are needed. Update the MongoDB version number in your composition configuration.

## 12.52  v3.15.0 - 2020-11-29

### 12.52.1  Added

- When using Jira as source for the 'issues' and the 'user story points' metric, show the issue type in the metric details. Closes #1674.

- Allow for limiting editing rights to specific people. Grant editing rights to people by adding their username or email address to the editors field on the homepage. Expand the overview title to access the editors field. Also see the *user manual*. Closes #294.

### 12.52.2  Fixed

- When invoking the reports API endpoint (api/v3/reports) without a `report_date` parameter, the server would sometimes return a deleted report. Fixes #1683.

- Make sure that the collector does not crash when a metric has a source that is no longer supported. Fixes #1699.

- When measuring test branch coverage with a JaCoCo XML file that doesn't contain any branches, don't complain but report 100% coverage. Fixes #1700.

## 12.53  v3.14.1 - 2020-11-17

### 12.53.1  Fixed

- Undo the fix for #1656 as it causes timeouts. This fix was meant to prevent 403 responses (access forbidden) from GitLab when using HEAD requests. If they do still happen (can't reproduce at the moment) we'll need to find another solution. Fixes #1675.

- Turn on processing of all DTDs (despite the fact that security tools complain that this is insecure) because otherwise XML reports referring to a DTD can't be read. Fixes #1676.

## 12.54  v3.14.0 - 2020-11-15

### 12.54.1  Added

- When hiding the tags column, also hide the tag pie charts in the report dashboard. Closes #1595.

- Allow for adding more than one Microsoft Teams webhook to a report so notifications can be sent to more than one channel.

## 12.54.2 Changed

- Group Snyk security warnings by top-level dependency. Closes #1616. Contributed by @greckko.

## 12.54.3 Removed

- Remove support for the source "OWASP Dependency Check Jenkins plugin". Fixes #1666.

## 12.54.4 Fixed

- In Microsoft Teams notifications, show missing values as "?" rather than "None". Fixes #1637.

- Turn on processing of DTDs (despite the fact that security tools complain that this is insecure) because otherwise some XML reports (notably OJAudit) can't be read. Fixes #1655.

- When using folders and/or files in GitLab as source for the 'source up-to-dateness' metric, *Quality-time* would use HEAD requests to get the ids of commits from GitLab. For issue #1638, it was necessary to pass the private token as header instead of URL parameter. Unfortunately, this results in 403 (access forbidden) responses for HEAD requests. It's unclear why. Using GET requests instead does work, so we use that as a work-around. Fixes #1656.

# 12.55 v3.13.0 - 2020-11-08

## 12.55.1 Added

- When using the Performancetest-runner as source for the 'slow transactions' or 'tests' metric, add a transactions-to-include parameter in addition to the transactions-to-ignore parameter to make it easier to select the relevant transactions. Closes #1647.

## 12.55.2 Removed

- The SonarQube rules that *Quality-time* uses to query SonarQube for the 'commented out code', 'complex units', 'long units', 'many parameters', and 'suppressed violations' metrics are no longer a parameter that the user can change. The reason is that it's rarely necessary to change these parameters and at the same time it's easy to accidentally remove a rule and get incorrect results as a consequence. The used rules are documented in the *metrics and sources overview*. Closes #1648.

## 12.55.3 Changed

- When using GitLab as source with a private token, pass the token to GitLab as header instead of URL parameter to prevent redirection. Closes #1638.

## 12.55.4 Fixed

- Introduce separate namespace for internal APIs. Fixes #1632.

- When using the same Microsoft Teams webhook in multiple reports, notifications for one report could also contain metrics of other reports.

# 12.56  v3.12.0 - 2020-10-31

## 12.56.1 Added

- When a report has a Microsoft Teams webhook configured, in addition to sending notifications for metrics turned red (target not met), also send notifications when metrics turn white (error parsing source data or troubles connecting to source). Partially implements #1223.

- Add a trend table to each metric to see the trend of a metric in tabular form. The number of dates shown and the time between dates can be adjusted through the 'hamburger' menu in the table header. Closes #1536.

- When measuring with SonarQube as source, include the creation date and last update date of issues such as violations and security warnings in the metric details. Closes #1564.

- In addition to ignoring Jenkins jobs by name or regular expression, also allow for including Jenkins jobs by name or regular expression. Closes #1596.

## 12.56.2 Fixed

- When a source zip file doesn't contain any files with the expected extension, report an error instead of continuing with an empty list of files, because that may result in incorrect measurements. Fixes #1618.

# 12.57  v3.11.0 - 2020-10-25

## 12.57.1 Added

- Added a generic JSON file format that can be used as source for the 'security warnings' metric. See the *reference manual* for details on the exact format. Closes #1479. Contributed by @greckko.

- Include the expanded/collapsed state of metrics, including which tab is active, in the URL so that the renderer uses that state when exporting the report to PDF. Closes #1594.

- In the Microsoft Teams notifications, include which metric(s) turned red. Partially implements #1223.

# 12.58  v3.10.0 - 2020-10-18

## 12.58.1 Added

- Support for Forwarded Authentication in a situation where *Quality-time* is behind a reverse proxy that is responsible for authentication. See the *deployment instructions*.

- Notifications of new red metrics to Microsoft Teams, using webhooks. See the *user manual*. Note that your Docker composition needs to be changed to include the new notifier component. See the example docker-compose.yml and the *deployment instructions*. Partially implements #1223.

## 12.59 v3.9.0 - 2020-10-11

### 12.59.1 Added

- When measuring 'tests' with the Performancetest-runner as source, allow for ignoring transactions by name or regular expression. Closes #1550.

### 12.59.2 Fixed

- Show how to set the time zone of the renderer in the docker-compose.yml so that PDF exports contain the correct local time. Fixes #1529.

- The date picker for the end date of technical debt has a minimum date set to today. Apparently, if the current value of the technical debt end date is far enough in the past so that the whole month popup consists of disabled dates, the date picker will crash. Worked around by removing the minimum date. Fixes #1534.

- Running `docker-compose up` in the project root folder wouldn't work on Windows. Fixes #1543.

## 12.60 v3.8.0 - 2020-10-04

### 12.60.1 Changed

- Use local time instead of UTC in the filenames of PDF exports. Closes #1505.

### 12.60.2 Added

- The 'slow transactions' metric with the Performancetest-runner as source allows for ignoring transactions by name or by regular expression. Closes #1493.

- The 'tests' metric now also supports the percentage scale so it's possible to e.g. report the percentage of tests failed. Closes #1494.

- Added Cobertura Jenkins plugin as possible source for the 'test line coverage', 'test branch coverage', and 'source up-to-dateness' metrics. Closes #1520.

- Added Robot Framework Jenkins plugin as possible source for the 'tests', and 'source up-to-dateness' metrics. Closes #1521.

## 12.60.3 Fixed

- Some exceptions thrown by the `aiohttp` library have no explicit error message. This would cause *Quality-time* to try and parse the non-existing source response, erroneously complaining about a parse error. Although in these cases the connection error would be logged, without an error message the logging would not be informative. Fixed by having the collector log the class of the `aiohttp` exception if the error message is empty. Fixes #1422.

- The PDF export would always export the most recent report, even when the user picked another date. Fixes #1498.

- The 'commented-out code' metric claimed to measure the number of lines of commented-out code, but SonarQube actually reports the number of *blocks* of commented-out lines of code. Changed the metric description and unit to conform to the SonarQube data. Fixes #1507.

- Trend graphs showing metrics with minutes as unit would have their y-axis labeled 'hours'. Fixes #1522.

- Tokens with an underscore would not be completely redacted from the collector log. Fixes #1523.

# 12.61 v3.7.0 - 2020-09-27

## 12.61.1 Added

- When exporting quality reports to PDF, hide the same metric table rows and columns in the PDF as hidden by the user in the user interface. Closes #1466.

- In addition to the trend, target, source, comment, and tags columns, also allow for hiding the status and measurement columns in metric tables. Closes #1474.

## 12.61.2 Fixed

- The measurement value and target of metrics with unit minutes and their scale set to percentage were formatted incorrectly (e.g. "0:50%" instead of "50%"). Fixes #1480.

- The measurement value and target of metrics with unit minutes and their scale set to count were displayed as 'hours:minutes minutes'. This would be confusing: e.g. '3:10 minutes' looks like 3 minutes and 10 seconds instead of 3 hours and 10 minutes. Fixed by changing 'minutes' to 'hours'. Fixes #1484.

- The security warnings in OWASP ZAP reports do not have unique keys. However, *Quality-time* needs security warnings to be uniquely identifiable to detect whether the list of warnings changes between measurements. Therefore, *Quality-time* generates keys for OWASP ZAP security warnings itself. Unfortunately, the key that *Quality-time* generated, was not guaranteed to be unique. Fixes #1492.

- Time travel was broken. Fixes #1497.

# 12.62 v3.6.0 - 2020-09-19

## 12.62.1 Changed

- Moved the button to hide metrics that don't require action to the new 'hamburger' menu on the top left side of each metric table. The menu is needed to allow for menu items to hide columns. See #1464.

## 12.62.2 Added

- Support version 2.4 and 2.5 of the OWASP Dependency Check XSD. Closes #1460.

- Allow for hiding the trend, target, source, comment, and tags columns in the metric tables. This can be done through the 'hamburger' menu on the top left side of each metric table. Closes #1464.

## 12.62.3 Fixed

- Retrieving measurements for the trend graph of a metric with many measurements and details (violations, user stories, security warnings, etc.) was slow because *Quality-time* would retrieve all details for all measurements even though it only needs the details for the most recent measurement. Fixes #1468.

# 12.63 v3.5.0 - 2020-09-12

## 12.63.1 Fixed

- Open source detail URLs (e.g. links to individual user stories or violations) in a separate window or tab. Fixes #1434.

- When measuring 'velocity' with Jira as source, *Quality-time* would only retrieve the first 50 boards from Jira, making the metric fail for some boards. Fixes #1445.

## 12.63.2 Changed

- Right align columns with numbers in the metric details. Closes #1384.

## 12.63.3 Added

- Added TestNG XML reports as possible source for the 'tests' and 'source up-to-dateness' metrics. Closes #1204.

- When measuring 'velocity' with Jira as source, the metric can also report the number of points committed to. Closes #1406.

- When measuring 'velocity' with Jira as source, the metric can also report the number of points completed minus the number of points committed to. Closes #1408.

# 12.64 v3.4.0 - 2020-09-05

## 12.64.1 Fixed

- Use the default value of a source parameter if the user sets it to an empty string. Fixes #1417.

- SonarQube plugins for Java and JavaScript had some of their rules renamed. Add the renamed rules to the 'commented out code', 'complex units', 'many parameters', 'long units', and 'suppressed violations' metrics. Fixes #1423.

### 12.64.2 Changed

- Rename the 'ready user story points' to 'user story points' as it can not just be used to count ready user stories, but rather any selection of user stories. Closes #1415.

### 12.64.3 Added

- Add 'velocity' metric. Currently, the only source supported for this metric is Jira. Closes #1407.

- Add axe-selenium-python JSON reports as possible source for the 'accessibility violations' metric. Closes #1424.

## 12.65 v3.3.0 - 2020-08-29

### 12.65.1 Fixed

- The 'source up-to-dateness' metric combined with the Calendar would report a parse error instead of returning the number of days since the specified date. Fixes #1399.

- After an attempt to login with invalid credentials and closing/reopening the login dialog, it would still show the error message. Fixes #1401.

- When measuring 'tests' with Azure DevOps, test runs could not be marked as false positive or won't fix. Fixes #1402.

- Specifying the Jira user story point field only worked if done by id, not by name. Fixes #1409.

### 12.65.2 Added

- When measuring ready user story points with Jira as source, show the sprint(s) of the user stories in the details. When measuring issues with Jira as source, show issue status, priority, sprint(s), creation date, and date of last update in the details. Closes #1411.

## 12.66 v3.2.0 - 2020-08-22

### 12.66.1 Fixed

- When measuring the 'manual test duration' metric *Quality-time* would subtract one minute for each ignored test case, instead of the duration of the ignored test case. Fixes #1361.

- *Quality-time* would not measure the 'manual test execution' metric until the user changed the 'default expected manual test execution frequency' field. Fixes #1363.

- SonarQube doesn't consider security hotspots to be violations anymore since version 8.2. Therefore, when using SonarQube as source for the 'violations' or the 'suppressed violations' metric, you can no longer select security hotspots as a violation type to be included or excluded. If you have a 'violations' or 'suppressed violations' metric that was configured to measure only the number of security hotspots you will need to remove it, as it will now measure all violation types. To count security hotspots, use the 'security warnings' metric with SonarQube as source instead, and configure it to only count security hotspots. Fixes #1381.

- When measuring 'tests' with Azure DevOps, only the latest build of all matching test runs was reported instead of the latest build of each matching test run. Fixes #1382.

- Jenkins jobs with slashes in their names couldn't be marked as false positive or won't fix. Fixes #1390.

- Typing an invalid date in the report date picker would crash the front end. Fixes #1394.

### 12.66.2 Added

- When measuring the number of test cases using Jenkins test report as source, for each failing test case show for how many builds it has been failing. Closes #1373.

- Add a new metric 'violation remediation effort' that reports the effort needed to remediate violations. Currently, the only source supporting this metric is SonarQube. Closes #1374.

- Allow for filtering Azure DevOps test runs by test run state. Closes #1383.

- Added Snyk JSON reports as source for the 'security warnings' metric. Contributed by @greckko.

## 12.67 v3.1.0 - 2020-08-07

### 12.67.1 Fixed

- Trend graph didn't take technical debt end date into account. Fixes #1272.

- Don't store the age of last commits, last builds, and last execution of manual tests, but only the date. This prevents the measurements from being updated daily. Fixes #1341.

- Don't show tokens in the log of the collector when retrieving the URL fails. Fixes #1354.

### 12.67.2 Added

- When measuring the 'size' metric with SonarQube, show the non-commented lines of code per programming language as measurement details and allow for ignoring specific languages. Due to a SonarQube limitation this is only possible when measuring size using non-commented lines of code (the default) and not when measuring size using all lines. Closes #1216.

- When measuring 'issues' with Jira, show status, priority, creation date and last update date for each issue in the details tab. Closes #1351.

### 12.67.3 Changed

- Increase the number of ticks on the x-axis of trend graphs. Closes #1126.

## 12.68 v3.0.0 - 2020-07-31

### 12.68.1 Fixed

- When opening a tag report containing a subject without a name, the UI would complain that the server is unavailable. Fixes #1309.

- Parse JSON files even if the source web server doesn't set the content-type header to application/json. Fixes #1325. Contributed by @walterdeboer.

## 12.68.2 Added

- When using GitLab as source for the 'unused jobs' or 'failed jobs' metrics, allow for ignoring jobs/pipelines by name, by branch, and by tag. Closes #1288.

## 12.68.3 Changed

- *Quality-time* now uses version 3 of the *Quality-time* API to read data from other *Quality-time* instances for the 'metrics' metric. This means that the source-*Quality-time* needs to be at least version 2.4.0. Closes #1208.
- Upgrade Python to version 3.8.5. Closes #1314.

## 12.68.4 Removed

- Version 2 of the *Quality-time* API, which was deprecated since version 2.4.0, has been removed.
- Remove code to fix the database structure. If you are migrating from a *Quality-time* version < 2.0.0 you need to install at least one *Quality-time* version >= 2.0.0 and < 3.0.0.

# 12.69 v2.5.2 - 2020-07-10

## 12.69.1 Fixed

- Removing the end date of accepted technical debt would not correctly update the metric status. Fixes #1284.
- The 'tests' metric with the Performancetest-runner as source would fail to read the number of performance tests due to changes in the report format. Fixes #1291.

# 12.70 v2.5.1 - 2020-07-06

## 12.70.1 Fixed

- Older GitLab versions don't return a `web_url` as part of the `repository/branches` API, be prepared. Fixes #1280.

# 12.71 v2.5.0 - 2020-07-05

## 12.71.1 Fixed

- Sorting the unmerged branches by date of last commit would crash the UI. Fixes #1270.

## 12.71.2 Added

- *Quality-time* can be used as source for the 'source up-to-dateness' metric. Closes #1227.

- Have unmerged branches in *Quality-time* link to the branches in GitLab or Azure DevOps. Closes #1268.

- Documentation on how to add metrics and sources to *Quality-time*. Closes #1273.

- Add Cobertura coverage reports as source for the coverage metrics. Closes #1275.

# 12.72 v2.4.1 - 2020-06-25

## 12.72.1 Fixed

- Don't disable the 'scroll to dashboard' button when the dashboard is visible. This requires listening for scroll events which makes scrolling of big reports slow and causes other user interface issues like dropdown menu's not working. Fixes #1260 and #1261.

# 12.73 v2.4.0 - 2020-06-23

## 12.73.1 Added

- Add cloc as source for the 'size' metric. As opposed to SonarQube, cloc makes it easy to exclude certain programming languages from the size measurement. Closes #460.

- When using Azure DevOps as source for the 'tests' metric, allow for filtering tests by test run name and show the test runs as detail information. Closes #1215.

- Add a button to the menu bar to scroll to the dashboard. Closes #1231.

- Add OWASP Dependency Check as source for the 'dependencies' metric. Closes #1239.

## 12.73.2 Changed

- Moved the Copy and Move buttons next to the Add buttons, making the UI more consistent. This also allows the user to copy an existing item to the right position in one go, instead of having to copy and then move it. To support adding items by copying an existing item, the API has been updated to version 3. Version 2 of the API is deprecated. See http://quality-time.example.org/api/, http://quality-time.example.org/api/v2, and http://quality-time.example.org/api/v3. Note that your Docker composition may need to be changed to use the new API version. See the Caddy proxy configuration in the example docker-compose.yml. Closes #1197.

## 12.73.3 Fixed

- Reordering items or changing the type of a metric would sometimes fail because *Quality-time* would also try to save the layout of the dashboard. Fixed by only saving the dashboard layout when the user deliberately changes the layout by dragging a card. Fixes #1160.

- Columns with numbers in the source entity views were incorrectly sorted as text. Fixes #1196.

- Collecting unmerged branches using Azure DevOps as source would fail if the project name contained spaces and the user did not specify a repository. *Quality-time* would fail to find the default repository because it would use the URL-quoted project name to look for it, instead of the unquoted project name. Fixes #1224.

- When using Jira as source for the 'ready user story points' metric, changing the status of a user story in the details tab didn't work. Fixes #1230.

- When using Jira as source for the 'ready user story points' metric, changing the status of a user story in the details tab to won't fix, false positive or fixed would reduce the total number of story points with one instead of the number of story points of the ignored user story. Fixes #1233.

- The `git clone` URL in the README.md required people to have a public SSH key added to their GitHub account. Replaced with a HTTPS URL which doesn't have this issue. Fixes #1235.

- When using the OWASP Dependency Check as source for the 'security warnings' metric, changing the status of a warning in the details tab didn't work. Fixes #1238.

- The trend sparkline graphs, showing the trend over the last week, would always use the full width, even when there was less than a week of data. Fixes #1241.

## 12.74 v2.3.2 - 2020-06-10

### 12.74.1 Changed

- Open the source links in separate window. Closes #1203.

### 12.74.2 Fixed

- The 'source up-to-dateness' metric could report a negative number of days ago due to differences in timezone or system clock between *Quality-time* and the source. Fixes #1217.

- Re-enable environment variables to set a proxy to be used by the collector. See the *deployment documentation*. Fixes #1217.

## 12.75 v2.3.1 - 2020-06-02

### 12.75.1 Fixed

- Don't strip hyphens from usernames when authenticating them with LDAP. Fixes #1198.

## 12.76 v2.3.0 - 2020-06-01

### 12.76.1 Added

- SonarQube can be used as source for the 'security warnings' metric. *Quality-time* collects the vulnerabilities and/or security hotspots. Closes #1136.

- npm-outdated and pip-outdated JSON reports can be used as source for the 'dependencies' metric. Partially implements #1065.

## 12.76.2 Fixed

- *Quality-time* was using the 5.1 version of the Azure DevOps API to get the number of tests for the 'tests' metric causing *Quality-time* to not work with Azure DevOps Server 2019. Fixed by using the 5.0 version of the API that also returns the required data. Fixes #1182.

- Don't log a traceback the first time the collector component attempts to download the data model from the server component and fails. As the collector typically starts up faster than the server, one failed attempt is to be expected. Fixes #1187.

## 12.77 v2.2.4 - 2020-05-11

### 12.77.1 Fixed

- Adding a metric or changing the type of a metric would sometimes fail due to a conflict with saving the dashboard layout. Fixed by only saving the dashboard layout when the user manually rearranges cards and not when a card gets added or removed. Fixes #1160.

## 12.78 v2.2.3 - 2020-05-10

### 12.78.1 Fixed

- When using *Quality-time* as source for the Metrics metric, a timeout could occur due to *Quality-time* unnecessarily retrieving all measurements (it only needs the most recent ones). Fixes #1154.

## 12.79 v2.2.2 - 2020-04-22

### 12.79.1 Fixed

- Don't include SonarQube security hotspots in the complex units, long units, many parameters, commented out code, and suppressed violations metrics. Fixes #1138 introduced in v2.2.1.

## 12.80 v2.2.1 - 2020-04-22

### 12.80.1 Fixed

- When requesting issues with a severity from SonarQube, SonarQube will not return security hotspots because security hotspots don't have a severity. *Quality-time* incorrectly assumed security hotspots would always be returned regardless of the specified severities. Fixed by making a separate call to the SonarQube issues API if necessary to retrieve the security hotspots. Fixes #1135.

## 12.81 v2.2.0 - 2020-04-14

### 12.81.1 Added

- Trend graphs show the target, near target, and technical debt target (if technical debt is accepted) as background colors. Note that the background colors only become visible after the measurement value of a metric changes or one of its target values is edited. Closes #1087.

### 12.81.2 Changed

- Make it clear in the user interface and the documentation that *Quality-time* can be authenticated with Jenkins using a username and API token, in addition to a username and password. Closes #1125.

### 12.81.3 Fixed

- Remove private tokens from source error messages and collector logging. Fixes #1127.

## 12.82 v2.1.1 - 2020-04-03

### 12.82.1 Fixed

- When the collector fails to collect a measurement, a traceback would be included in the measurement. Unfortunately, tracebacks with the new asynchronous collector are long. To prevent performance issues, the collector now only logs the error in the case of connection errors. Other types of errors still include a traceback. Fixes #1122.

## 12.83 v2.1.0 - 2020-03-29

### 12.83.1 Changed

- The collector wakes up every minute, collects measurement data if necessary, and then would pause for a minute. Changed to have the length of the pause depend on how long the data collection took so that the user does not have to wait too long for a new measurement after changing the configuration of a metric. Closes #1100.
- Made the collector collect measurements in parallel to speed it up.

### 12.83.2 Fixed

- The metrics "violations" and "suppressed violations" show zero violations (green status) even though the component has no SonarQube analysis. Fixes #1090.

## 12.84  v2.0.0 - 2020-03-08

### 12.84.1 Added

- Add a private token parameter to all sources that consists of JSON, XML, or HTML reports so that they can be retrieved from GitLab job artifacts. Closes #1067.

- Add a column to show the status (unconfirmed, confirmed, false positive, etc.) of security warnings, violations, etc. so that the user doesn't have to expand them to see the status. Closes #1070.

- Show end date of technical debt in the measurement target column. Closes #1072.

- Allow for accepting technical debt for a metric that has no sources or failing sources. Closes #1076.

- Make date fields clearable. Closes #1088.

### 12.84.2 Fixed

- Don't store server-side generated report summaries in the database. The previously generated report summaries are removed from the database when the server starts, so starting may take longer than normal. Fixes #1082.

### 12.84.3 Removed

- Version 1 of the API has been removed. API version 1 was deprecated since *Quality-time* v1.3.0. Closes #1051.

- Remove the Docker environment files and move the environment variables to the docker-compose files to simplify the compositions. Closes #1063.

## 12.85  v1.8.1 - 2020-03-03

### 12.85.1 Fixed

- When loading changes to show in the changelog, an internal server error could occur due to Mongo hitting its maximum buffer size for sorting. Add an index to the reports collection to prevent out of memory errors during sorting. Fixes #1077.

## 12.86  v1.8.0 - 2020-03-01

### 12.86.1 Changed

- Cache data model and other performance improvements. Note: the proxy settings for the data model API have been updated. See the Caddy configuration in the docker-compose.yml. The order of the metric tabs has been changed. From left to right: first tab is the metric configuration, second the source(s) configuration, third the trend graph, and finally the tab(s) with details per source, if applicable. This makes it possible to lazily load the data for the trend graph and the details per source and show the tabs as soon as the data becomes available. Fixes #1026.

## 12.86.2 Added

- Allow for specifying variable parts of URLs in OWASP ZAP reports. This makes it possible to mark warnings as false positive even when parts of URLs change between runs of OWASP ZAP. Note: because the way *Quality-time* keeps track of the warnings has been changed, some OWASP ZAP warnings may need to be marked as false positive again. Closes #1045.

- A new metric for measuring the number of (outdated) dependencies and a new source (Composer for PHP) that supports this metric were added. Closes #1056.

# 12.87 v1.7.1 - 2020-02-26

## 12.87.1 Fixed

- When using SonarQube as source for duplication, uncovered lines, or uncovered branches, the landing URL would be incorrect. Fixes #1044.

- The docker-compose YAML file now specifies that the proxy container should wait for the server and frontend containers to start. Fixes #1046.

- The collector would fail if it could not write a timestamp to the health_check.txt file, e.g. due to a permission error. Fixed by writing the health_check.txt file to /tmp instead of the home directory of the default user and by catching and logging any OS errors that may occur. Fixes #1057.

# 12.88 v1.7.0 - 2020-02-22

## 12.88.1 Changed

- As the previous rendering component used by *Quality-time* is no longer maintained, it is replaced with a new component: URL to PDF Microservice. *Quality-time* uses the ICTU fork that packages the service as a Docker container.

## 12.88.2 Fixed

- The new rendering component (see "Changed") shows the date in the correct format. Fixes #1010.

- Show tags added by users in the tag dropdown so they don't have to keep typing added tags. Fixes #1041.

## 12.88.3 Removed

- Because the new rendering component (see "Changed") waits for network activity to stop before converting a report into PDF, the delay parameter is no longer needed.

## 12.89  v1.6.2 - 2020-02-19

### 12.89.1  Fixed

- Use environment variables for both proxy host and port so the renderer uses the right URL to get the report. Fixes #1031.

- OWASP ZAP warning keys were not always unique, causing trouble with marking them as false positive. Fixes #1032.

- The Jenkins test report source would not correctly get the number of passed tests from aggregated test reports. Fixes #1033.

## 12.90  v1.6.1 - 2020-02-18

### 12.90.1  Fixed

- Don't refresh the change log when clicking the "Download report as PDF" button. Fixes #1015.

- Make proxy port configurable. Fixes #1018.

- Changes made to violations, issues, warnings, etc., such as marking them as false positive, were only visible in the metric change log and not in the change logs of the report, subject, and source. Note: because a change needed to be made to the database format to fix this, changes made to violations, issues, warnings, etc. before this release are not visible in the change log. Fixes #1019.

- Anchore vulnerability keys are not always valid as JSON key, causing exceptions when the user tries to make changes to vulnerabilities. Hashing the keys prevents this issue. Fixes #1023.

- The too many parameters, complex units, and long unit metrics with SonarQube as source would always report the percentage as zero. Fixes #1027.

## 12.91  v1.6.0 - 2020-02-12

### 12.91.1  Changed

- Several accessibility related changes, such as improved background colors, larger icons, more contrast, and labels for images. Closes #1005.

### 12.91.2  Fixed

- Changes made to violations, issues, warnings, etc., such as marking them as false positive, would not carry over after a failed measurement, forcing the user to make the same changes again. Fixes #1007.

## 12.92 v1.5.0 - 2020-02-09

### 12.92.1 Added

- Jenkins jobs with the JaCoCo Jenkins plugin can be used as source for the line and branch coverage metrics. Closes #984.

- Add Anchore Docker image vulnerability scan reports in JSON format as possible source for the security warnings metric. Closes #1000.

### 12.92.2 Changed

- Center the status column so that the trend graphs and the status icons have a bit more space between them. Closes #985.

### 12.92.3 Fixed

- Sorting of metrics by measurement value, target value, and status did not work. Fixes #981.

- Exporting tag reports to PDF did not work. Fixes #990.

- When using Jira as source for the issues metric, the URL to Jira in the metrics table would not work properly. Fixes #991.

## 12.93 v1.4.0 - 2020-01-31

### 12.93.1 Added

- Source parameter (URLs, user names, passwords, etc.) changes can be applied to different scopes: to just the source being edited or to multiple sources that have the same type and value as the one being edited. When applying the change to multiple sources, the user can change all sources (that have the same type and value) of a single metric, of a single subject, of a single report, or of all reports. Closes #927.

- Change logs show the users' avatars. Note that email addresses were not recorded in the change log until now, so avatars can only be shown for new changes. Closes #948.

- The delay for generating PDFs can be changed in the report title and can be passed to the API as parameter. Closes #958.

### 12.93.2 Fixed

- The front end was still using one version 1 API. Fixes #947.

- Retrieving the change log would fail if not all recent changes had a change log entry. Fixes #949.

- After changing a value in the UI, *Quality-time* would briefly show the old value while it was updating the database. Fixes #954.

- Documentation API was not reachable. Fixes #966.

## 12.94  v1.3.4 - 2020-01-15

### 12.94.1  Fixed

- Metrics with status unknown in the details view of the 'Metrics' metric did not show a question mark. Fixes #935.

- Check connection for all URL and credential parameters. Fixes #944.

## 12.95  v1.3.3 - 2020-01-15

### 12.95.1  Fixed

- Subject cards in the report dashboard would not have a default subject title when the subject had no title.  Fixes #942.

## 12.96  v1.3.2 - 2020-01-15

### 12.96.1  Fixed

- Metrics with status unknown in the details view of the 'Metrics' metric did not show a question mark. Fixes #935.

## 12.97  v1.3.1 - 2020-01-15

### 12.97.1  Fixed

- Adding sources did not work. Fixes #939.

## 12.98  v1.3.0 - 2020-01-15

### 12.98.1  Added

- Added an option to exclude branches from being reported as unmerged when using GitLab or Azure DevOps as source for the unmerged branches metric. Closes #879.

- Sources can be reordered.

- Sources, metrics, subjects, and reports can be copied. Sources, metrics, and subjects can be moved across metrics, subjects, and reports. Implements #881.

## 12.98.2 Changed

- The "hide metrics not requiring action" buttons now hide metrics in all subjects of a report at once. Closes #907.

- A new, simpler version of the API was introduced, version 2. Version 1 of the API is deprecated. See http://quality-time.example.org/api/, http://quality-time.example.org/api/v1, and http://quality-time.example.org/api/v2.

## 12.98.3 Fixed

- Typo in metric pie chart tooltip ("Uknown"). Fixes #857.

- User documentation incorrectly said that the dashboard layout is persisted in the browser. It is kept in the database since version 1.0.0. Fixes #860.

- Add report title to subject names in tag reports so it is clear from which report each subject comes. Fixes #880.

- Tag reports could not be exported to PDF. Fixes #885.

- Prevent users from entering invalid percentages. Fixes #888.

- Fix Checkmarx landing URL. Fixes #919.

- Remove plain text passwords from HTML. Fixes #921.

- Marking OWASP ZAP warnings as false positives did not work. Fixes #922.

- Remove private tokens from URLs logged by the collector. Fixes #934.

# 12.99 v1.2.0 - 2019-12-10

## 12.99.1 Added

- If users have a Gravatar, it will be shown next to their username after they log in.

- REST API added for importing a complete report. Closes #818.

- Allow GitLab as source for the unused CI-jobs metric. Closes #851.

## 12.99.2 Changed

- Open help URLs in a new window or tab. Closes #842.

# 12.100 v1.1.0 - 2019-12-03

## 12.100.1 Fixed

- Ignore Jira fields that have no number value when summing Jira issues for the ready user story points and manual test duration metrics. Fixes #834.

## 12.100.2 Added

- Added a button (expand a report title to access it) to download a PDF version of a report. The PDF report can also be downloaded via the API: `http://www.quality-time.example.org/api/v1/report/<report_uuid>/pdf`. Closes #828.

- Metric summary cards now have tooltips showing the number of metrics per status (target met, target not met, etc.). Closes #838.

## 12.100.3 Changed

- Show the five most recent changes in the change log table initially so that the buttons below the change log table don't disappear off screen. Each click on the "load more changes" button still loads ten more changes. Closes #836.

# 12.101 v1.0.0 - 2019-11-28

## 12.101.1 Fixed

- Users would not be notified of an expired session when trying to delete something while their session was expired. Fixes #813.

- Prevent double slashes in URLs to Jira issues. Fixes #817.

- Hiding metrics that do not require action did not work. Fixes #824.

## 12.101.2 Added

- Store dashboard layouts on the server instead of in the local storage of the user's browser. Closes #379.

## 12.101.3 Removed

- Removed deprecated metrics from metric types options. Closes #826.

# 12.102 v0.20.0 - 2019-11-23

## 12.102.1 Fixed

- The *Quality-time* source still used port 5001 to access the *Quality-time* API. Fixes #806.

## 12.102.2 Added

- Allow for filtering metrics by metric type and source type in the *Quality-time* source. Closes #805.

## 12.103 v0.19.1 - 2019-11-19

### 12.103.1 Fixed

- Determining the encoding of large OWASP Dependency Check XML reports was slow. Fixes #803.

## 12.104 v0.19.0 - 2019-11-17

### 12.104.1 Fixed

- Use correct API URL when accessing *Quality-time* as source. Fixes #791.

### 12.104.2 Added

- Metric for manual test execution added. Closes #556.
- Azure DevOps can now be a source for the failing jobs metric and the unused jobs metric. Closes #638.

## 12.105 v0.18.0 - 2019-11-12

### 12.105.1 Fixed

- Add keep-alive messages to the server-sent events stream so it does not time out when there are no new measurements for a while. Fixes #787.

### 12.105.2 Added

- In addition to a changelog per report, also keep a changelog for the reports overview. Closes #746.

## 12.106 v0.17.0 - 2019-11-10

### 12.106.1 Fixed

- Make string input fields with suggestions clearable. Fixes #772.
- Open Axe CSV files in universal newline mode. Fixes #777.
- Prevent browser console traceback when switching to the sources tab of a metric. Fixes #779.

## 12.106.2 Added

- More flexibility in configuring LDAP by introducing a `LDAP_SEARCH_FILTER` environment variable and replacing the `LDAP_LOOKUP_USER` variable by `LDAP_LOOKUP_USER_DN`. See the *deployment instructions*. Closes #774.

- Logo for Axe. Closes #778.

## 12.107 v0.16.1 - 2019-11-07

### 12.107.1 Fixed

- Ignoring Jenkins child jobs (jobs within pipelines) did not work. Fixes #763.

- Notifications from the server to the frontend about new measurements were broken after the introduction of the reverse proxy. Fixes #765.

## 12.108 v0.16.0 - 2019-11-02

### 12.108.1 Added

- Allow for ignoring Jenkins jobs by name or regular expression. Closes #747.

- For sources that are comprised of static reports, it is now possible to specify a zip file with reports as URL. *Quality-time* will unzip the file before processing its contents as normal. So far, this has been implemented for Axe CSV reports, Bandit JSON reports, JaCoCo XML reports, JUnit XML reports, NCover HTML reports, OJAudit XML reports, OpenVAS XML reports, OWASP Dependency Check XML reports, OWASP ZAP XML reports, Performancetest-runner HTML reports, Pyup.io Safety JSON reports, and Robot Framework XML reports. Closes #748.

## 12.109 v0.15.0 - 2019-10-30

### 12.109.1 Fixed

- The collector component would crash if an Azure DevOps source was unreachable. Fixes #738.

- Add a changelog entry when a user creates a report. Fixes #742.

### 12.109.2 Changed

- Introduce a reverse proxy (Caddy) so that web frontend and API can both be accessed through the same port. Which is now port 80 by default. Fixes #727.

## 12.110  v0.14.1 - 2019-10-24

### 12.110.1 Fixed

- Login problem solved for LDAP servers where a user bind must be done. Fixes #734.

## 12.111  v0.14.0 - 2019-10-23

### 12.111.1 Fixed

- Toaster messages didn't disappear when clicked. Fixes #717.
- Don't crash the frontend after changing the type of a metric. Fixes #718.

### 12.111.2 Added

- Immediate check of URLs accessibility added. Closes #478.
- When measuring unmerged branches, have the metric landing URL point to the list of branches in GitLab or Azure DevOps. When measuring the source up-to-dateness of a folder or file in GitLab or Azure DevOps, have the metric landing URL point to the folder or file. Closes #711.
- When SonarQube is the source for a metric, users can now select the branch to use. Note that only the commercial editions of SonarQube support branch analysis. Closes #712.
- Subjects can be reordered. Expand a subject title to show the reordering buttons on the lower left-hand side of the subject title panel. The buttons allow one to move a subject to the top of the page, to the previous position, to the next position, and to the bottom of the page. Closes #716.
- Allow for filtering accessibility violations from Axe CSV files by impact level. Closes #730.

### 12.111.3 Changed

- Use the `ldap3` library instead of `python_ldap`. Closes #679.

### 12.111.4 Removed

- The ability to use HQ quality reports as source was removed. Closes #715.

## 12.112  v0.13.0 - 2019-10-20

### 12.112.1 Added

- *Quality-time* now counts the unmerged branches against the default branch in GitLab or Azure DevOps instead of assuming that the master branch is the default branch. Closes #699.

### 12.112.2 Changed

- The "Size (LOC)" metric can now either count all lines of code or all non-commented lines of code. That means the "Size (Non-commented LOC)" metric is now deprecated. Closes #644.

### 12.112.3 Fixed

- Allow for specifying an Azure DevOps repository by name. Fixes #683.

## 12.113  v0.12.2 - 2019-10-16

### 12.113.1 Fixed

- Remove white space at the top of the page in printouts. Fixes #685.
- Don't crash when the user refreshes a report in the browser. Fixes #692.

### 12.113.2 Changed

- *Quality-time* now uses Python 3.8 for the collector and server components. Closes #684.

## 12.114  v0.12.1 - 2019-10-14

### 12.114.1 Fixed

- Allow for specifying the repository when using Azure DevOps as source for the "Source up-to-dateness" metric instead of assuming that the repository has the same name as the project. Fixes #663.
- Do not repeat the top menu bar in printouts, it overlaps with content. Fixes #680.

## 12.115  v0.12.0 - 2019-10-11

### 12.115.1 Fixed

- Because Checkmarx does not immediately return detail information, Checkmarx measurements would alternate between measurements with and without detail information, resulting in a lot of measurements in the database. Fixed by not collecting detail information from Checkmarx anymore. Fixes #670.
- Sources that don't need to access the network ("Calendar", "Manual number", "Random number") would throw an exception. Fixes #672.
- NCover reports weren't parsed correctly. Fixes #675.

## 12.115.2 Removed

- *Quality-time* no longer collects detail information about security warnings from Checkmarx; the Checkmarx API is too complex, resulting in fragile interaction between *Quality-time* and Checkmarx. See #670.

# 12.116 v0.11.0 - 2019-10-07

## 12.116.1 Fixed

- Allow for specifying which test results (skipped, failed, errored, and/or passed) to count when using SonarQube as source for the "Tests" metric. Fixes #634.

- Store measurement values for each scale that a metric supports so that the graphs show correct information when the user changes the metric scale. Fixes #637.

- Do not stop contacting sources after receiving a 401 (Unauthorized) or 403 (Forbidden). Fixes #652.

## 12.116.2 Added

- Add NCover coverage reports as source for the test coverage metrics. Closes #636.

- Add Azure DevOps as source for the "Tests" metric. Requires Azure DevOps Server or Service 2019. Closes #639.

- Add Azure DevOps as source for the "Source up-to-dateness" metric. Closes #640.

- Add Azure DevOps as source for the "Unmerged branches" metric. Closes #641.

- Add percentage scale to the "Complex units", "Many parameters", "Long units", and "Suppressed violations" metrics. Closes #645.

# 12.117 v0.10.2 - 2019-09-26

## 12.117.1 Fixed

- Measuring the source up-to-dateness of folders in GitLab did not work. Fixes #626.

# 12.118 v0.10.1 - 2019-09-25

## 12.118.1 Fixed

- Measuring size (LOC), size (non-commented LOC), tests, and failed tests using SonarQube as source would fail with a parse error. Fixes #623.

## 12.119 v0.10.0 - 2019-09-22

### 12.119.1 Added

- All metrics now have an explicit scale that's either fixed to "Count" or "Percentage", or that can be changed from "Count to "Percentage" and vice versa. Metrics whose scale can be changed: "Duplicated lines", "Metrics", "Test branch coverage", and "Test line coverage". Closes #504.

- Added a 'landing URL' parameter to some sources so *Quality-time* can refer users to a human readable version of a machine readable report. For example, you can add an HTML version of a JaCoCo report to a JaCoCo XML report source. Closes #554.

## 12.120 v0.9.1 - 2019-09-10

### 12.120.1 Fixed

- To prevent reporting Checkmarx internal server errors to users when reports are unexpectedly unavailable, don't immediately remove a Checkmarx report after reading it, but silently ignore a removed report and create a new one. Fixes #468.

- Prevent locked accounts by not contacting a source again after receiving a 401 (unauthorized) or 403 (forbidden) HTTP status, until the configuration of the metric changes. Fixes #604.

## 12.121 v0.9.0 - 2019-09-06

### 12.121.1 Added

- The direction of metrics is now configurable. The direction of a metric determines whether smaller measurements are better, or bigger measurements are better. This means that the "number of tests" metric, with its direction reversed, can now also be used to measure the number of failing tests. The "failing tests" metric is deprecated. Closes #552.

- Metrics can be reordered. Expand a metric to show the reordering buttons on the lower left-hand side of the metric details. The buttons allow one to move a metric to the top of the table, to the previous row, to the next row, and to the bottom of the table. Closes #585.

### 12.121.2 Fixed

- Checkmarx internal server error solved. Fixes #468.
- Use a consistent style for labels of input fields. Fixes #579.
- Added *Quality-time* logo to the *Quality-time* source. Fixes #580.
- When adding HQ as source for the accessibility metric, show the URL and metric id parameters. Fixes #587.
- The layout of the reports overview dashboard would be reset after visiting a tag report. Fixes #588.
- Tag report donut charts were always white. Fixes #589.

## 12.122  v0.8.2 - 2019-08-28

### 12.122.1  Fixed

- Prevent web browsers from automatically filling in username and password in the source configuration tab. Fixes #574.

## 12.123  v0.8.1 - 2019-08-28

### 12.123.1  Fixed

- Changing the subject type now changes the subject name if the default subject name has not been overridden. Fixes #553.
- When a user changes a password field, don't show the old password in the change log unmasked. Fixes #565.
- Use <= and >= for the metric direction in the metric tables instead of < and >. Fixes #567.

## 12.124  v0.8.0 - 2019-08-23

### 12.124.1  Added

- Add meta metrics and the ability to add *Quality-time* itself as source for the meta metrics. Closes #337.
- Accessibility metric for Axe report source added. Closes #338.

### 12.124.2  Fixed

- Don't use the unicode characters for <= and >= in the source code; it caused problems on Windows. Fixes #558.

## 12.125  v0.7.1 - 2019-08-18

### 12.125.1  Fixed

- When generating keys for OWASP ZAP security warnings, strip any hashes from the application URLs to ensure the keys are stable. Fixes #541.
- In addition to version 2.0 also support version 2.1 and 2.2 of the OWASP Dependency Check XML format. Fixes #543.

## 12.126 v0.7.0 - 2019-08-14

### 12.126.1 Added

- Users can now select a suggestion and edit it in input fields with suggestions. Closes #197.

- Users can now login with both their canonical LDAP name as well as with their LDAP user id. Closes #492.

- Allow for using (a safe subset of) HTML and URLs in metric comment fields. Closes #511.

- Added OWASP Dependency Check Jenkins plugin as possible source for the security warnings metric. Closes #535.

### 12.126.2 Fixed

- Break long lines in OpenVAS security warning description to keep the metrics table from becoming too wide. Fixes #452.

- Break long URLs in source error messages to keep the metrics table from becoming too wide. Fixes #531.

- Don't try to retrieve more work items from Azure DevOps than allowed. Fixes #532.

- Return a parse error if OWASP dependency report XML reports don't contain the expected root tag instead of reporting zero issues. Fixes #536.

## 12.127 v0.6.0 - 2019-08-11

### 12.127.1 Added

- Keep track of changes made by users in a change log. The change log for a report can be viewed by expanding the report title. The change log for a subject can be viewed by expanding the subject title. The change logs for metric and their sources be viewed by expanding the metric. Closes #285.

- When the user session is expired (after 24 hours) log out the user and notify them of the expired session. Closes #373.

- Added a metric for measuring the duration of manual tests. Added Jira as default source for the metric. Closes #481.

### 12.127.2 Fixed

- When using OWASP ZAP reports as source for the security warnings metric, report on the number of "instances" instead of "alert items". Fixes #467.

- Don't wait 15 minutes before trying to access a requested Checkmarx SAST XML report, but try again after one minute. Partial fix for #468.

- The Performancetest-runner now uses "scalability" instead of "ramp-up" as name for the scalability measurement. Closes #480.

- OJAudit XML files may contain duplicate violations (i.e. same message, same severity, same model, same location, same everything) which led to problems in the user interface. Fixed by merging multiple duplication violations and adding a count field to the violations. Fixes #515.

- Use Jenkins job timestamp for the source up-to-dateness metric if the Jenkins test report doesn't contain timestamps in the test report itself. Fixes #517.

- Stop sorting metrics when the user adds a new metric to prevent it from jumping around due to the sorting. Fixes #518.

## 12.128 v0.5.1 - 2019-07-18

### 12.128.1 Fixed

- The Trello parameter "lists to ignore" was not displayed properly. Fixes #471.

- The number of issues would not be measured if the source was Jira. Fixes #475.

## 12.129 v0.5.0 - 2019-07-16

### 12.129.1 Added

- Added Pyup.io Safety JSON reports as possible source for the security warnings metric. Closes #450.

- Added Bandit JSON reports as possible source for the security warnings metric and the source up-to-dateness metric. Closes #454.

- Only measure metrics that have all mandatory parameters supplied. Closes #462.

### 12.129.2 Fixed

- Add performance test stability and scalability metrics to the example report. Fixes #447.

- Set up a new LDAP connection for each authentication in an attempt to prevent a "Broken pipe" between *Quality-time* and the LDAP server. Fixes #469.

## 12.130 v0.4.1 - 2019-07-08

### 12.130.1 Fixed

- Frontend can't reach server.

## 12.131 v0.4.0 - 2019-07-07

### 12.131.1 Changed

- Run server on port 5001 instead of 8080 to reduce chances of interfering with other applications.

- Allow for deployments where the different components all have the same hostname, e.g. quality-time.example.org, and only the ports differ.

## 12.132 v0.3.0 - 2019-07-05

### 12.132.1 Added

- Metric for performance test duration added. Closes #401.

- Metric for performance test stability added. Closes #433.

- Metric for performance scalability added. Closes #434.

- Performancetest-runner reports can now be used as metric source for the tests and failed tests metrics. Closes #402.

## 12.133 v0.2.3 - 2019-07-01

### 12.133.1 Fixed

- Time travelling to a date before any report existed would throw an exception on the server. Fixes #416.

- Trend graphs would be too tall and overlap with the next metric. Fixes #420.

- When clicking the report date field in the menu bar, the calendar popup would be displayed at the wrong location before popping up at the right location. Fixes #424.

## 12.134 v0.2.2 - 2019-06-28

### 12.134.1 Fixed

- Version number was missing in the footer of the frontend. Fixes #410.

## 12.135 v0.2.1 - 2019-06-26

### 12.135.1 Fixed

- Work around a limitation of the Travis configuration file. The deploy script does not allow sequences, which is surprising since scripts in other parts of the Travis configuration file do allow sequences. See https://github.com/travis-ci/dpl/issues/673.

## 12.136 v0.2.0 - 2019-06-26

### 12.136.1 Added

- Release Docker containers from Travis CI to Docker Hub.

## 12.137  v0.1.0 - 2019-06-24

### 12.137.1 Added

- Initial release consisting of a metric collector, a web server, a frontend, and a database component.

# THIRTEEN

# INDEX