

Teamscale

Administration and Configuration Guide

Version 2.0

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München

CQSE
Continuous Quality in Software Engineering

Contents

1 System Requirements **3**
1.1 Analysis Server . . . 3
1.2 Clients . . . 3

2 Installation **4**
2.1 Configuration . . . 4
2.2 License . . . 4
2.3 Starting Teamscale . . . 4
2.3.1 JVM Memory Settings . . . 6
2.4 Installing Teamscale as a service . . . 6
2.4.1 Windows . . . 6
2.4.2 Linux . . . 7
2.5 Configuring HTTPS . . . 7
2.6 Integration with Apache Webserver . . . 8
2.7 Using Cassandra as Alternative Storage System . . . 9
2.8 Integration with SAP NetWeaver Application Server ABAP . . . 10
2.8.1 Deploy Teamscale Adapter to SAP NetWeaver Application Server ABAP . . . 11
2.8.2 Configure Execution Variant . . . 12
2.8.3 Schedule Background Job for Teamscale adapter . . . 13
2.8.4 Schedule Periodic Job for ABAP import . . . 14
2.8.5 Configure ABAP Projects in Teamscale . . . 15
2.9 System Properties Supported by Teamscale . . . 15
2.10 HTTP(S) Proxy . . . 15
2.11 Self-signed Certificates . . . 16

3 Permissions **17**
3.1 Users & Groups . . . 17
3.2 LDAP Server Integration . . . 17
3.2.1 Global Catalog . . . 18
3.2.2 Fallback Servers . . . 18

4 Projects **20**
4.1 Accounts for external connectors . . . 20
4.2 Creating Analysis Profiles . . . 20
4.3 Creating Projects . . . 21
4.3.1 Connecting to TFS over HTTPS . . . 21
4.3.2 Post-Commit Hooks . . . 21
4.4 Git Connector . . . 23

5 Integration of External Analysis Tools **24**
5.1 General Approach . . . 24

5.2 Clang Static Analyzer . . . 25
5.3 Findbugs . . . 25
5.4 Goanna . . . 25
5.5 Test coverage tools . . . 26
5.5.1 JaCoCo . . . 26
5.5.2 Cobertura . . . 26
5.5.3 gcov . . . 27
5.6 ESLint . . . 27
5.7 PC-lint . . . 27
5.8 Pylint . . . 28
5.9 StyleCop . . . 28
5.10 SAP Code Inspector . . . 28

6 Updates **29**
6.1 Upgrading from previous versions of Teamsale . . . 29
6.1.1 Specific steps for migrating from Teamscale 1.0 . . . 29
6.1.2 Specific steps for migrating to Teamscale 1.3 . . . 30
6.1.3 Specific steps for migrating to Teamscale 1.3.2 . . . 30

7 Support **31**

A SAPLink Installation and Usage **32**
A.1 Install SAPLink . . . 32
A.2 Import ABAP Code Nuggets using SAPLink . . . 33

1 System Requirements

1.1 Analysis Server

Category	Requirement
Operating System	Windows or Linux
Java	Oracle Java Development Kit (JDK) with Server VM, 64-bit, Version 1.8 or later; a JRE is **not** sufficient
RAM	4 GB minimum
CPU	2 GHz, 2 or more cores recommended
Disk Space	Installation: 250 MB, Data: 10 GB recommended

Table 1.1: System requirements for analysis server

Note: The disk space requirements for data highly depend on

- the amount of source code of the analyzed software
- the time range of the history being analyzed
- the commit behavior (frequency and number of committed files) within the time range analyzed

As an example, a system of about 500 kLOC and a history of 1.5 years requires about 1 GB of disk space.

1.2 Clients

Category	Requirement
Web Browser	Internet Explorer version 9 or later, Firefox version 20 or later, Chrome version 25 or later
Eclipse	Eclipse SDK version 3.7 or later
Visual Studio	Windows Vista or later, .NET Framework 4.5 or later, Visual Studio Version 2010 or later

Table 1.2: System requirements for clients

2 Installation

This chapter gives a brief overview of the installation procedure. Before continuing unzip the archive into the preferred target directory. For the remainder of this document, `<Teamscale-Dir>` is the absolute path to this directory.

2.1 Configuration

Basic configuration of Teamscale can be done by editing the file `teamscale-config.properties`. Table 2.1 shows the configuration options.

2.2 License

Teamscale needs a valid license to run. Teamscale automatically searches several locations for a license file named `teamscale.license` (in this order):

1. The directory specified in the Java system property `teamscale.license.path` (if it is set)
2. The Teamscale installation directory (given by TEAMSCALE_HOME environment variable; this variable is set automatically, if using the start script)
3. The current working directory
4. The home directory of the user running Teamscale

Before starting Teamscale for the first time, it has to be ensured that a valid license exists in one of these locations.

2.3 Starting Teamscale

As a prerequisite for starting Teamscale it has to be ensured that the Java executable of the JDK (a JRE will not suffice) is on the path. Teamscale can be started with Teamscale startup shell script by executing following command:

```
<Teamscale-Dir>/teamscale.sh
```

In a Windows environment please use the corresponding `.bat` file in the same directory.

By default, Teamscale uses the configuration file `<Teamscale-Dir>/teamscale-config.properties`. To provide a different configuration file, use the `-c` option of the start script:

```
<Teamscale-Dir>/teamscale.sh -c <path-to-teamscale-config.properties>
```

In the default configuration, Teamscale starts a web server on your machine, using port 8080. This web server will be also available from other machines in your network. If you do not want this, remove the comment before the line `server.bind-hostname=localhost` in the file `teamscale-config.properties`.

Check if Teamscale has started successfully by opening `http://localhost:8080` in a supported web browser. You can log in using the default user »admin« and the password »admin«. Teamscale writes a log-file named `teamscale.log` in the execution directory.

2.3.1 JVM Memory Settings

By default, the Teamscale start script will launch a JVM with a maximum Java heap size of 2.500 MB. You can change this by setting the environment variable `TEAMSCALE_MEMORY`.

2.4 Installing Teamscale as a service

You might want to install Teamscale as a service to allow for more consistent management with other services on the same machine and to ensure that Teamscale is automatically started after a reboot. Additionally, this allows Teamscale to automatically restart in case of a crash.

Before installing Teamscale as a service, ensure that the configuration as described in the previous sections was completed and both Cassandra and Teamscale can be started manually. Then follow the steps in one of the following subsections depending on your operating system.

2.4.1 Windows

The required files are located in the folder `windows` in the Teamscale installation directory.

1. Edit the file `teamscale-service.xml`.

   a) Enter or edit the correct path to the Teamscale root directory, for example

   ```
   <env name="TEAMSCALE_HOME" value="c:\opt\teamscale" />
   ```

   b) Enter or edit the correct path to the JDK installation directory, version 1.8 or higher, for example

   ```
   <env name="JAVA_HOME" value="c:\program files\java\jdk1.8.0_45" />
   ```

   c) Optionally uncomment and fill the section `<serviceaccount>` to run the services as a different user. Alternatively you can set this information after installing using the service console, which avoids storing the password in clear text.
   In both cases the user needs to have the *Logon as a service* right. This right can be granted in the *Local Security Policy* management console by navigating to *Local Policies > User Rights Assignment* and adding the user to *Logon as a service*.

2. Install the Teamscale service by running the following command as an administrator:

```
teamscale-service.exe install
```

3. Check that the service is running. Directly after the installation, you might have to start the service manually or alternatively reboot the server.

Uninstall If you later have to uninstall the service run the following command:

```
teamscale-service.exe uninstall
```

2.4.2 Linux

The required files are located in the folder `linux` in the Teamscale installation directory. The installation is based on `init.d`, which is supported on most Linux distributions (this guide was tested with *Ubuntu 12.04*).

1. Edit the file `teamscale.init`.
   a) Enter or edit the correct path to the Teamscale root directory, for example

   ```
   TEAMSCALE_HOME=/opt/teamscale
   ```

   b) Enter or edit the correct path to the JDK or JRE installation directory, version 1.8 or higher, for example

   ```
   export JAVA_HOME=/opt/jdk
   ```

   c) Optionally uncomment and fill the `EXTRA_START_ARGS` line to run the services as a different user.
2. Install the service by running from the command line:

   ```
   sudo ./install-service.sh teamscale
   ```

3. Start the service by running the following command:

   ```
   sudo service teamscale start
   ```

Uninstall If you later have to uninstall the service run the following command:

```
sudo ./uninstall-service.sh teamscale
```

2.5 Configuring HTTPS

Teamscale can optionally provide HTTPS communication, either in addition to HTTP or exclusively. The enablement of both can be controlled via the settings `server.port` and `https.port` in the server configuration (see also Section 2.1).

To set up HTTPS communication for Teamscale, a pair of private key and certificate is required. Your company may already have a certificate and key available or a new pair has to be generated. Please consult your IT operations team for potential regulations in your company. Technically, you also have the option of generating a self-signed certificate (not recommended for security reasons).

Teamscale requires the private key and certificate to be stored in the *Java Keystore* format. Prior to importing a certificate in a Java keystore, the certificate has to be converted to the PKCS 12 format. In case you are in a Microsoft Windows environment and have the certificate stored in the Certificate Management of the operating system, you can directly export the certificate and private key in the PKCS 12 format using the Certificate Export Wizard. You can then skip the conversion in the next paragraph and continue with the creation of the Java keystore.

In case the certificate and key was created with OpenSSL, the conversion can be done using the `openssl` command line tool[1]. Assuming your certificate is available in a file `myhost.crt` and your private key in a file `myhost.key`, the following command will combine them and save them a file `myhost.p12` converting them to the PKCS12 format which is compatible with the Java keystore (you will be asked for an export password):

```
openssl pkcs12 -export -in myhost.crt -inkey myhost.key -out myhost.p12
```

After this, you can create a new Java keystore and import the certificate/key pair into the newly created store. This can be done with the `keytool` command line tool that is part of the JDK (located in the `bin` folder of the JDK) The following command will create a new file `myhost.jks` containing a Java keystore with both the certificate and private key (you will be asked for import and export passwords):

```
keytool -importkeystore -srckeystore myhost.p12 -srcstoretype pkcs12 -destkeystore my-
host.jks
```

To enable HTTPS for Teamscale, all configuration settings starting with *https.* have to be properly configured (see also Section 2.1). Make sure to properly configure the path to the newly generated Java keystore, its password as well as the certificate alias. If you do not know the alias of your certificate, you can look it up with a keytool command:

```
keytool -list -keystore myhost.jks
```

If everything was properly configured, Teamscale will accept HTTPS connections on the HTTPS port specified in the Teamscale settings (`https.port`). All connections to the configured HTTP port (i. e. the value for `server.port` or the default of 8080) will be forwarded to the HTTPS port. If the HTTP port is set to 0, HTTP is disabled and only HTTPS connections are accepted.

[1] Please refer to the OpenSSL website (https://www.openssl.org) for download and installation instructions.

2.6 Integration with Apache Webserver

To fulfill certain requirements, such as different authentication mechanisms or HTTPS support, Teamscale can be operated behind a webserver, such as Apache. One possible setup has Teamscale running on the same machine as the Apache server. For this, the `teamscale-config.properties` should contain the following two lines.

```
server.urlprefix=teamscale
server.bind-hostname=localhost
```

This sets an explicit URL prefix to use and ensures that the instance is reachable only from the local machine (otherwise Apache could be circumvented). The Apache configuration then contains the following lines:

```
ProxyPass         /teamscale  http://localhost:8080/teamscale
ProxyPassReverse  /teamscale  http://localhost:8080/teamscale
ProxyRequests     Off

# This is important, as otherwise some URLs are blocked by Apache
AllowEncodedSlashes On
```

Depending on your configuration and goals, slightly different and additional directives are required. Please refer to the Apache manual for more information.

2.7 Using Cassandra as Alternative Storage System

If you want to support a distributed setup or run into incompatibilities with LevelDB (the default storage system), you can also use Cassandra as a storage system. For this change the *database.type* setting in the *teamscale-config.properties* file. Note that for using Cassandra with Teamscale, at least 6 GB of RAM are required.

If used with the storage system Cassandra, a Cassandra server must be running on the same machine. To start Cassandra, change into the `apache-cassandra` sub-directory within the Teamscale installation and run `bin/cassandra` on Linux or `bin\cassandra.ps1` (PowerShell scripts have to be enabled, see Enabling Powershell on Windows) on Windows. Cassandra should be stopped only if Teamscale is not running. Stop Cassandra by sending a signal (i.e., `kill`) to the Cassandra process.

Enabling Powershell on Windows

1. Start *Windows PowerShell* **as Administrator**
2. Execute `Set-ExecutionPolicy unrestricted` via the PowerShell
3. PowerShell scripts should now be enabled

Configuring an external storage directory for Cassandra (optional) If you want to separate the user- and project-specific data created by Teamscale from the installation files, you can instruct Cassandra to store its files to a different directory. To do so, open the file `apache-cassandra/conf/cassandra.yaml`.[2] Starting in line 95, change the entries `data_file_directories`, `commitlog_directory`, and `saved_caches_directory` to the desired path.

Configuring Cassandra's Log File To change Cassandra's default logging behaviour (log everything to cassandra.log in the teamscale main dir), open the file `apache-cassandra/conf/logback.xml` and change the main logfile's <file> directive in line *23* and the <fileNamePattern> directive used for log rotation in line 25.

Configuring Cassandra's Memory Amount By default Cassandra runs with 1500M of RAM. To configure the amount used, edit the configuration in the `conf/` directory:

1. On Windows: cassandra-env.ps1 in line *124* (MAX_HEAP_SIZE = "1500M")
2. On Linux: cassandra-env.sh in line *143* (MAX_HEAP_SIZE="1500M")

Installing Cassandra as a Service If you installed Teamscale as a service and want to use Cassandra, you should also install Cassandra as a service. The description is contained in the README file in the *windows* and *linux* directories of the Teamscale installation directory.

Troubleshooting Memory Issues with Cassandra

Cassandra terminates with error *java.io.IOException: Map failed* If you encounter this error, it is recommended to set the memlock limit of the operating system to *unlimited*. Moreover, the value for *vm.max_map_count* should be increased. For more information please consult the Cassandra FAQ: https://wiki.apache.org/cassandra/FAQ#memlock

2.8 Integration with SAP NetWeaver Application Server ABAP

To enable the analysis of ABAP source code, the Teamscale ABAP exporter is required to run inside the SAP NetWeaver Application Server ABAP where the ABAP source code is located. The exporter is shipped with the Teamscale enterprise distribution and reads the source code of ABAP development objects from the ABAP repository (within the database of the SAP AS) and writes ZIP files containing the exported ABAP source code to the file system. On the Teamscale server, the Teamscale ABAP Importer reads these ZIP files and commits the ABAP source code to an internal file-based *Git*[3] repository. Teamscale obtains the source code by connecting to this internal Git repository. Figure 2.1 illustrates this export/import architecture.

The exporter performs an incremental export, i. e. only objects which were modified (updated, moved or deleted) since the last export are contained in a single export. The following outlines the steps required to integrate Teamscale with SAP NetWeaver Application Server ABAP (details of each step are described in the subsequent dedicated sections):

[2] Cassandra configuration parameter documentation: http://www.datastax.com/documentation/cassandra/2.1/cassandra/configuration/configCassandra_yaml_r.html
[3] http://git-scm.com/

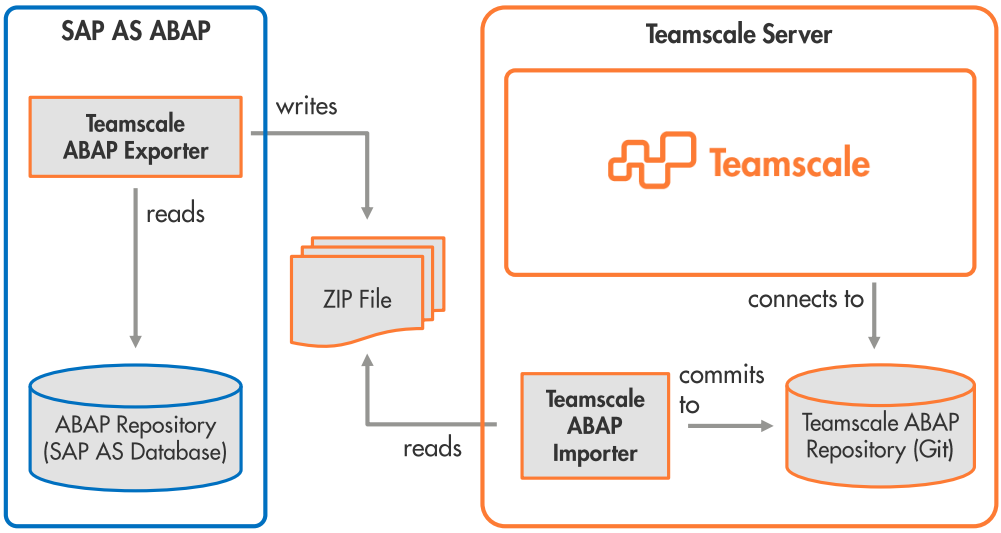


Figure 2.1: Outline of ABAP source code export/import architecture.

- Within SAP NetWeaver Application Server ABAP
  1. Deploy the Teamscale adapter to the SAP NetWeaver Application Server ABAP.
  2. Configure an execution parameters variant for the Teamscale adapter.
  3. Schedule a periodic background job to execute the Teamscale adapter.
- On Teamscale server
  4. Schedule periodic execution of the ABAP import for Teamscale.
- Within Teamscale
  5. Configure an ABAP project with a Git Connector in Teamscale.

2.8.1 Deploy Teamscale Adapter to SAP NetWeaver Application Server ABAP

The Teamscale adapter for SAP NetWeaver Application Server ABAP is distributed as transport files as well as SAPLink[4] *nugget* which has to be imported using SAPLink. The standard way of deployment is by using the transport. A deployment using SAPLink is usefull if the ABAP exporter should only be imported as local objects.

Deployment by Transport

The transport files can be found in the sub-directories `cofiles` and `data` of `<Teamscale-Dir>/abap/exporter/transport`, the files there must be copied to the transport directories `/usr/sap/trans/cofiles` or `/usr/sap/trans/data` respectively. The transport can be performed using SAP transaction `STMS`. If an older version of the ABAP Exporter is already deployed to the system, import option *Overwrite Originals* (unconditional mode 2) should be enabled (to be set in *Import Request* (**Ctrl+F11**) dialog).

[4] http://wiki.scn.sap.com/wiki/display/ABAP/SAPlink

Deployment by SAPLink

A version of SAPLink is shipped with the Teamscale enterprise distribution. See appendix A on how to install SAPLink.

It is required to install two SAPLink *nuggets*:

- `ZCQSE_ABAP_EXPORTER.nugg`
- `ZCQSE_TEAMSCALE_ADAPTER.nugg`

These can be found in the `<Teamscale-Dir>/abap/exporter/saplink_nuggets` directory of the Teamscale distribution. Import both nuggets following the instructions in appendix A.2.

After installing, the Teamscale ABAP exporter needs to be activated:

- Go to transaction `SE38` and enter `ZCQSE_INCREMENTAL_ABAP_EXPORT` as program name.
- Select *Activate* (**Shift+F9**)
- In the dialog, select *Whole Worklist* (**Shift+F6**), *Select All* (**F9**) and confirm with *Continue* (**Enter**).
- The database tables

  `ZCQSEABAPEXP` which stores the time of the last export of an export configuration,

  `ZCQSEEXPOBJ` which stores the list of already exported objects (used to identify deleted objects) and

  `ZCQSEPCKPTH` which stores the package hierarchy of the last export (used to detect moves of objects)

  need to be activated, too:

  - Go to transaction `SE11` (ABAP Dictionary) and enter `ZCQSEABAPEXP` in the field *Database Table*.
  - Hit *Activate* (**Ctrl+F3**).
  - Repeat the same for the tables `ZCQSEEXPOBJ` and `ZCQSEPCKPTH`.

2.8.2 Configure Execution Variant

Some parameters have to be defined to run the incremental ABAP export for Teamscale. These parameters are saved as an execution *variant* which is used when planning a periodic job. To configure and save the parameters the following steps are required:

- Go to transaction `SE38` and enter `ZCQSE_INCREMENTAL_ABAP_EXPORT` as program name.
- *Execute* (**F8**) the program.
- Fill in the parameter fields:

Destination Export File The full path of the destination file where the exported source code should be saved to. If *Save to Application Server* is enabled, a path on the application server must be used - usually there exist some export/import directories. It is recommended to use a file name in the form of `abap_export_%sysid%_%timestamp%.zip`. `%sysid%` and `%timestamp%` are variables which will be replaced with the system identifier and a timestamp information of every export. This will avoid to accidentally overwrite files.

Save to Application Server If enabled, the exported source code will be saved to the file system of the application server (default). Otherwise it will be saved to the user's local machine (e. g. for theing purposes).

Export Configuration ID An identifier for this configuration. The configuration identifier is used to differentiate exports which will result in different code histories. For every configuration identifier, the first export executed with this identifier will do an initial export of the entire selected code base, from the second export on only the modified code since the previous export with this identifier is exported.

Include Y and Z Namespace If enabled, code in the default custom Y and Z namespaces will be included in the export. Usually this option should be enabled.

Custom Namespace to Include If code from a special custom namespace, e.g. something like `/XYZ/`, should be included in the export, enter the custom namespace (including the '/') here. Only one namespace can be entered here. If multiple are required, use multiple export jobs (with different export configurations identifiers). Leave this field empty, if only Y and Z namespaces should be exported.

Export Full Code Base If this option is enabled, the entire code base will be exported, regardless if the exporter did already run with this export configuration. This option is used to ensure the repository used by Teamscale is in sync with the actual ABAP repository in SAP NetWeaver Application Server ABAP. Such a synchornisation export can be exported manually if regular ABAP imports to Teamscale fail due to inconsistencies. Also a separate background job could be scheduled (e.g. once at night) with this option enabled to automatically fix inconsistencies. Levae this option disabled for a usual incremental export.

Use Temp File Name for ZIP If this option is enabled, the ZIP file with the exported source code will initially get a temporary name and will be renamed to `*.zip` only after the file is fully written. Depending on how the ZIP files are moved to the Teamscale server this may be required to avoid copying/moving of an unfinished file. Usually this parameter is disabled.

- Hit *Save*  (**Ctrl+S**) to save the parameters as *Variant*. In the opening dialog enter some variant name and a description and hit *Save*  (**Ctrl+S**) again.
- You may leave `SE38` now (by pressing *Back*  (**F3**) twice).

2.8.3 Schedule Background Job for Teamscale adapter

The Teamscale source code export has to be performed periodically on the SAP NetWeaver Application Server ABAP. To plan such a periodic job, the following steps are required:

- Start transaction `SM36` (Define Background Job).
- Enter a Job Name.
- *Specify start condition*  (**F5**):

- Either select *Immediate* or specify Date/Time.
- Enable check-box for *Periodic Job*.
- Specify *Period Values*, usually the interval should be within 5 to 30 minutes, recommended is 10 minutes. (Unless the initial export, the execution of one job lasts a few seconds only.)
- Leave start condition dialog by conforming with *Save* (**Ctrl+S**).

- *Define steps* (**F6**):
  - Enter `ZCQSE_INCREMENTAL_ABAP_EXPORT` as name of the ABAP program.
  - Enter the variant name of the parameter variant specified before (see 2.8.2).
  - It is recommended to use `EN` (English) es execution language.
  - Confirm the step definition with *Save* (**Ctrl+S**).
  - Return to `SM36` (Define Background Job) main dialog with *Back* (**F3**).
- Confirm job definition with *Save* (**Ctrl+S**).

In the same way, you may consider to set up a further periodic background job for a full export of the code base to ensure that the repository read by Teamscale is in sync with the actual ABAP repository (option *Export Full Code Base* enabled). Such a synchronization job is usually scheduled to be executed once every night.

2.8.4 Schedule Periodic Job for ABAP import

Once the steps before were completed, a ZIP file holding modified ABAP code is periodically written to the file system. It is required to import these ZIP files to the internal Git repository of the Teamscale server before the code can be analyzed.

Single Import

A single import requires the following steps:

- Move the exported ZIP files from the export location to a working directory for the import. This step is not required, but recommended to avoid conflicts with newer exports from the SAP system.
- Execute the ABAP Teamscale Importer by running the command

  ```
  <Teamscale-Dir>/abap/abap_import.sh
  ```

  In a Windows environment please use the corresponding `.bat` file in the same directory.

  The following **command line parameters** are supported, **-r** and either **-c** or **-z** are required:

  -r Path to the directory of the internal Git repository on the Teamscale server. This should be located besides the data directory of Teamscale. At the initial import, the directory is expected to be not existent or empty – a Git repository must not be initialized manually.

  -c Path to the directory of the ZIP files to import. All ZIP files in this directory will be imported. This option is an alternative to the option **-z**.

-z Path to a single ZIP file to import. This option is an alternative to the option **-c**.

-a Path to an archive directory, where the ZIP file should be archived. This is an optional parameter, but archiving is highly recommended to be able to restore the version history in case of an inconsistent repository. The archive directory will be created if it does not exist. Not that ZIP files which do not contain changes to the repository will be archived in `_unmodified` sub-directory. There, usually full exports for synchronization are stored which do not contain a change in normal operation.

-g Gap time in seconds to separate commits of a user. If the time span of the last update times of two objects which were modified by the same user is more than the specified interval, the modification will be versioned as two separate commits. This is an optional parameter, the default is 180 seconds.

-m E-mail domain of users. Since internally the repository is organized as a Git file repository, the specified mail domain is attached to the SAP system users id for the commit user. This is an optional parameter.

Periodic Execution

The before mentioned single import has to be scheduled for periodic execution. This can be done by any job scheduler available on the Teamscale installation machine. Possible job schedulers include for example *cron*, *Micorosft Task Schedular*, *Jenkins*, *Bamboo* and similar tools.

Ideally the job is scheduled in a way that it runs when a new exported ZIP file arrives, but also a scheduling with a very short time interval (e.g. 1 to 2 minutes) is possible, since the Teamscale ABAP importer terminates immediately if no ZIP File is found (best to use with option **-c** of the importer).

2.8.5 Configure ABAP Projects in Teamscale

ABAP projects are configured in Teamscale using the *Git* connector referring to the ABAP repository directory specified during ABAP import.

2.9 System Properties Supported by Teamscale

The following table lists all system properties supported by Teamscale. These properties are appended to the Java command line using the -D switch and can, for example, be added to the start script in the `TEAMSCALE_VM_ARGS` variable. Most of these properties are for very specific scenarios and not required for a normal Teamscale installation.

2.10 HTTP(S) Proxy

If access to source code repositories, issue trackers etc. has to go through an HTTP proxy, the proxy host and port must be configured via system properties. To do so, add the following to the TEAMSCALE_VM_ARGS variable in the start scripts:

```
-Dhttp.host=<Proxy-Host> -Dhttp.port=<Proxy-Port>
```

If the proxy should also be used for HTTPS connections, use the following arguments instead:

```
-Dhttps.host=<Proxy-Host> -Dhttps.port=<Proxy-Port>
```

Furthermore, for the HTTPS proxy, the proxy's SSL signing certificate needs to be added to Teamscale's certificate store. Follow the procedure in 2.11 for that.

2.11 Self-signed Certificates

If access to source code repositories, issue trackers etc. is through SSL-secured connections and the certificates for those connections are not signed by one of the certificate authorities known to a default Java installation, Teamscale needs to be made aware of these certificates. Start by downloading the certificate in question, e.g. through your web browser, and store it in DER format (`.cer` file extension).

Next, create a keystore that contains that certificate using the `keytool` command line tool (located in the `bin` folder of the Java installation) as follows, where `certificate.cer` is the certificate you downloaded, `keystore.jks` is the keystore to be created and `Alias` is the alias under which to store the certificate in the keystore:

```
keytool -importcert -file certificate.cer -keystore keystore.jks -alias Alias
```

You will be prompted for a password for the keystore. Multiple certificates can be imported into the same keystore under different aliases.

Edit the Teamscale start script (either `teamscale.bat` or `teamscale.sh`) and add the following options to the TEAMSCALE_VM_ARGUMENTS:

```
-Djavax.net.ssl.trustStore=<path-to-keystore-file>
-Djavax.net.ssl.trustStorePassword=<password>
```

Option	Description	Default
`server.port`	Teamscale HTTP server port	`8080`
`server.urlprefix`	Prefix of Teamscale URLs	`<empty>`
`server.bind-hostname`	Bind address of the HTTP server	`<none>`
`database.directory`	Database directory where all data is stored	`<Teamscale-Dir>/storage`
`database.cache-size`	The cache size used by the database in MB	`512`
`engine.workers`	The number of concurrent analysis worker jobs. When working with multiple projects, this can be used to parallelize analyses. Increasing this value requires the Teamscale JVM memory settings to be adapted as well (see Section 2.3.1). Allocate about 2GB per worker. For best performance use as many workers as cpu cores are available.	`1`
`cassandra.port`	The port the Cassandra DB is running on	`9160`
`servicelog.loglevel`	Log level for logging service calls—one of OFF, INFO, WARN, ERROR	`WARN`
`servicelog.loguser`	Whether to log the user of service calls	`false`
`database.type`	This is an expert setting and should not be changed	`leveldb`
`https.port`	Port to be used for HTTPS. If this option is not set, HTTPS is disabled (Please refer to Section 2.5 for details on how to configure HTTPS for Teamscale.)	
`https.keystore-path`	The absolute path to the Java keystore containing the certificate and private key	
`https.keystore-password`	The password for the keystore	
`https.certificate-alias`	The alias of the certificate and private key in the keystore	

Table 2.1: Basic configuration settings

Property	Description
com.teamscale.disable-cpuinfo	If set to true, the CPU info in the system view is disabled. This may be required on systems that do not support CPU querying.

Table 2.2: Overview of supported system properties

3 Permissions

3.1 Users & Groups

Permission management in Teamscale is based on users and groups. Each developer using Teamscale needs to have a user account. Each user can be tied to multiple groups. Each group is tied to a role, which defines the permissions available to the group members. Teamscale provides the following roles:

Role	Description
Admin	Admins may access any project and change global system settings.
Developer	Developers may view a project and blacklist findings.
Project Lead	Project leads may view a project, blacklist findings and create baselines.
Build	Build permissions allow to upload external analysis data, but no other actions.

Table 3.1: Group roles overview

3.2 LDAP Server Integration

Teamscale provides LDAP integration to synchronize your LDAP users and groups with the teamscale server. Teamscale can import groups or individual users from a configured LDAP server.

To configure LDAP go to the *Admin* perspective and open the *Settings* view. Press *Add* next to the *LDAP Server* category and enter a name for the server. Fill in the settings for the LDAP server. The different options are described in Table 3.2.

Add the server by pressing the *Add* button. You can now import entire groups of users using the *Import* button in the *Groups* view. If you wish to import single users, use the *Import* button in the *Users* view. If you want to update user or group information use the synchronize buttons found in both views.

Typical settings for Active Directory When configuring an *Active Directory* server, typically, the following values can be used for the general settings.

Typical settings for Open LDAP The following table shows typical settings for configuring an Open LDAP server.

Option	Description
Hostname	The server's hostname (URL).
Port	The server's port.
SSL	Select this checkbox if the server is using SSL for connections.
Base DN for users	The DN under which the users are stored on the LDAP tree. For example: `ou=users,dc=example,dc=com`.
Base DN for groups	The DN under which the groups are stored on the LDAP tree. For example: `ou=groups,dc=example,dc=com`.
Group attribute	The attribute groups are saved with. For example: `cn`, if the group DN looks like: `cn=GROUPNAME,ou=groups,dc=example,dc=com`.
DN for the initial bind	Full DN of the user used for the connection. This user should have read access to all users and groups. For example: `uid=readadmin,ou=users,dc=example,dc=com`.
Password for initial bind	The password used by the bind user.
Login attribute	The attribute users are saved with. For example: `uid`, if the user DN looks like: `uid=USERNAME,ou=users,dc=example,dc=com`.
Group member attribute	The attribute under which each group stores its members.
First name attribute	The attribute used to store the first name of users.
Last name attribute	The attribute used to store the last name of users.
Email attribute	The attribute used to store the user's email address.

Table 3.2: LDAP server options

Option	Value
Group attribute	cn
Login attribute	sAMAccountName
Group member attribute	member
First name attribute	givenName
Last name attribute	sn
Email attribute	mail

Table 3.3: Typical settings for Active Directory

3.2.1 Global Catalog

If users are stored in different domains and are thus not available from a single LDAP server it is recommended to query the global catalog of one of the directory servers using port 3268 (or 3269 with SSL). The global catalog offers read-only access to most data of the whole domain tree and thus reduces the overhead to follow LDAP redirects to other servers. Not using the global catalog and following referrals will create a new connection for each LDAP request which may lead to *refused connection* errors.

3.2.2 Fallback Servers

If the LDAP server has fallback servers which are listed in the DNS record, it is recommended to configure the server host using the domain name instead of a single IP address. Teamscale will then use the fallback servers if the primary server is not available.

Option	Value
Group attribute	cn
Login attribute	uid
Group member attribute	memberUid
First name attribute	givenName
Last name attribute	sn
Email attribute	mail

Table 3.4: Typical settings for Open LDAP

4 Projects

A *project* in Teamscale describes a collection of code for which the same analysis settings and permissions are used. Typically, a Teamscale project corresponds to a software project developed by a team. However, Teamscale projects may also aggregate multiple development projects or a large development project may be divided into multiple Teamscale projects. Teamscale analyzes a project with one or more *external connectors*, including version control systems and issue trackers.

4.1 Accounts for external connectors

Before creating a project using external connectors such as version control systems or issue trackers, a corresponding account for authenticating with the external service has to be created.

This can be done in the *Settings* tab of the *Admin* perspective. To create a new account click on *Add* in the *Account Credentials* section and enter the URL of the external service as well as the username and password.

4.2 Creating Analysis Profiles

An *analysis profile* describes the analysis settings used for analyzing one or more projects. Analysis profiles are created and managed in the *Analysis Profiles* view of the *Project* perspective.

The first step when creating an analysis profile is to select the programming languages of the code to be analyzed and the external analysis tools to be used together with Teamscale. Note that this selection affects only the further configuration of the project profile. Even if, for example, *Java* is not selected during creation of the profile, it can later be used to analyze *Java* code. Omitting a language or a tool only hides all configuration settings specific for the language or tool to simplify the configuration.

Once the languages and tools are selected, the default analysis settings are shown and can be edited. Besides global analysis settings, most settings are organized by *quality indicators*, which themselves consist of quality analysis groups. Each such analysis group combines settings, metrics, and findings related to a specific quality aspect. A detailed description of each configuration element is available as a tool tip. For findings, the setting can be *OFF* to disable the finding, or *YELLOW* or *RED* to enable the finding and denote its criticality. For some findings the additional setting *AUTO* is available, which lets the analysis decide the criticality. For example, the method length analysis would mark long methods as YELLOW and very long methods as RED (depending on the configured thresholds). As an example, Figure 4.1 shows how to configure the File Size analysis group.

Most of the quality indicators are fixed, but there are also custom quality indicators. These custom quality indicators can be added and renamed as needed. Analysis groups be moved between them using drag & drop. The organization of these quality indicators is also reflected in the representation of the findings found during the analysis of a project.

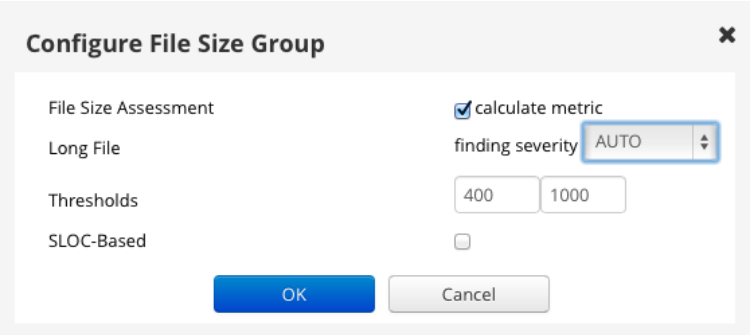


Figure 4.1: Configuring file size

4.3 Creating Projects

Projects are created and managed in the *Projects* view of the *Project* perspective. Besides a name and an analysis profile, a project consists of one or more connectors. These connectors integrate Teamscale with source code management systems and issue trackers. Connectors can be freely combined to, for example, read code from a Subversion repository and additional files from a Git repository. Multiple connectors of the same kind may also be used to reflect more complex development scenarios.

4.3.1 Connecting to TFS over HTTPS

To enable HTTPS communication with TFS, you have to import the certificate used by TFS into a Java keystore and configure Teamscale with this keystore.

The TFS certificate can be imported using the `keytool` command line as described in 2.11.

4.3.2 Post-Commit Hooks

By default, Teamscale uses polling to regularly check for new revisions on the server. Depending on the polling interval you configured, this can lead to a large delay between the actual commit and the update of the data in Teamscale, or to a very high load on your version control system server (especially with many projects).

One solution to this problem are post-commit hooks, which are supported by many version control systems. For this, configure your Teamscale project with a large polling interval. Polling will still happen from time to time as a fallback if the post-commit hook failed for some reason. Then you have to configure your version control system to call the following URL after each commit:

```
http://teamscale.company.com/post-commit-hook?
... user=USERNAME&access-token=TOKEN&repository=REPOSITORY
```

Here the part `http://teamscale.company.com/` is the URL of your teamscale server. The *USERNAME* is the name of a user that has build permissions, i. e. is in a group with role *Build*. The *TOKEN* is the access token for this user, which is accessible via the user administration page. Finally, REPOSITORY is the URL of the repository as configured in Teamscale. For example, if you have a project reading from the Subversion repository `http://svn.company.com/project1/trunk/foo`, the URL may be

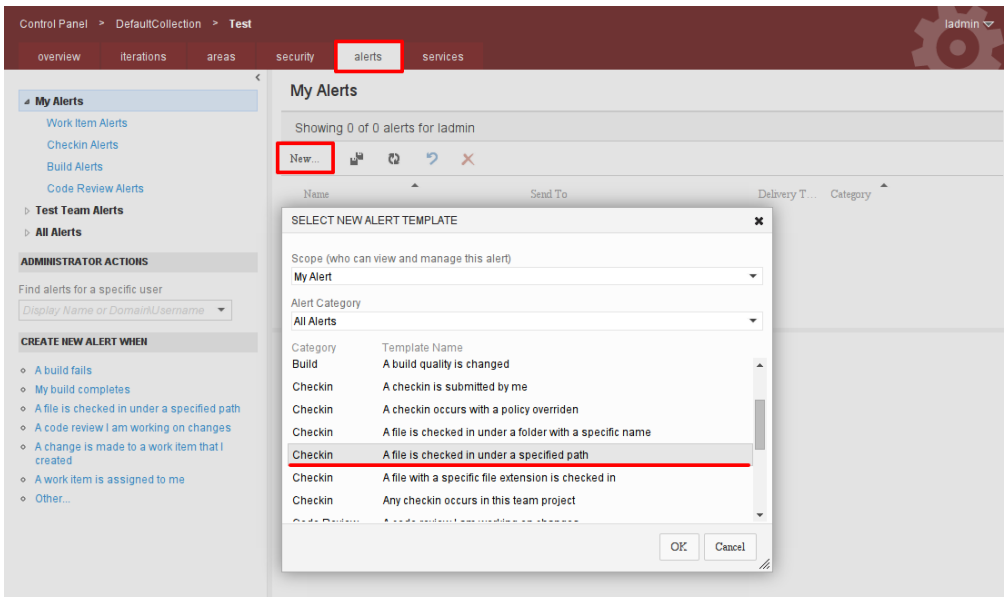


Figure 4.2: Create a new TFS commit alert.

`http://svn.company.com/project1/trunk/foo` or any prefix, such as `http://svn.company.com/pr` or even `http://svn.company.com/`. Note that all projects for which the prefix matches will be checked for updates, so a more specific prefix can help to reduce the server load. However, being too specific does not work. If you pass `http://svn.company.com/project1/trunk/foo/some/file` as repository parameter, the prefix match will fail and no update is triggered.

You can use both GET and POST requests to call the URL.

Subversion Subversion supports the execution of arbitrary (shell) scripts for various hooks. The hook that should be used is called *post-commit*. To access the URL you can use the tools *wget* or *curl*. See the SVN book for more details on configuration of commit hooks.

Team Foundation Server In Team Foundation Server every user can configure custom alerts. These are available via the admin settings for a team project (gear icon in the top loft corner). Switch to the *alerts* tab and create a new alert of type *A file is checked in under a specified path* (see figure 4.2). Then alter the following fields as shown in figure 4.3:

- *Name*: Alter the name to something that reminds you this is the commit hook for Teamscale.
- *Send To*: The URL to the Teamscale commit hook service as described above, whereas `REPOSITORY` is `TFS_URL/TEAM_COLLECTION/FOLDER_PATH`, e.g. `http://yourcompany.com/tfs/DefaultCollection/$MyProject/Dev/Source`.
- *Format*: Change to SOAP.
- *Filter Value*: Change from `$/` to the folder path used in the commit hook, e.g. `$MyTeamProject/Dev/Source`.

If you receive an error on save saying that the user needs the right *Create a SOAP subscription*, an administrator has to issue the following command (e.g. from VisualStudio Command Promt):

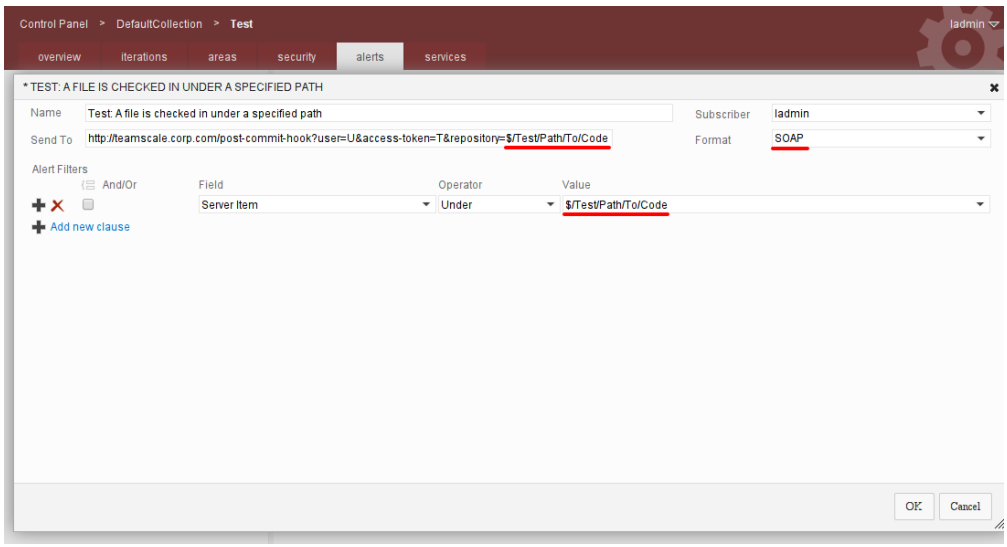


Figure 4.3: Configure the TFS commit alert.

```
tfssecurity.exe /a+ EventSubscription $SUBSCRIPTION: CREATE_SOAP_SUBSCRIPTION
n:DOMAIN\USERNAME ALLOW /collection:TFS_URL/TEAM_COLLECTION
```

Ensure to substitute `DOMAIN` and `USERNAME` with the login of the account you want to create the alert for and `TFS_URL/TEAM_COLLECTION` with the url to your TFS Team Collection.

4.4 Git Connector

If you configure a Git connector (as opposed to a Git Remote connector), you will have to regularly update the configured Git clone externally (`git pull`, e.g. via a Jenkins job). Please beware that when updating this git clone, you must ensure that after the update, the same branch is still checked out. Otherwise, Teamscale will not be able to proceed with the analysis. If this happens you will see errors like this in the worker log in the System perspective:

```
No such revision: 6e48a22dde618e99d86b463a774a28fc70ba527a
```

Changing the Git clone back to the correct branch will fix this problem.

5 Integration of External Analysis Tools

Teamscale also allows to integrate external analysis tools that are not inherently incremental. For this, the analysis tool is executed (for example in a nightly build) and the results are uploaded using Teamscale's service layer. The upload is performed using ConQAT and described in detail in the following sections.

When integrating external tools, it is important that these tools are also activated in the analysis profile, as otherwise findings are imported but not mapped to a quality indicator and, hence, not displayed. Enablement of rules can either be configured in each tool or via the analysis profile.

5.1 General Approach

This section assumes familarity with ConQAT. If required, more information on ConQAT can be found at http://www.conqat.org/. Also note that the Teamscale distribution contains a full ConQAT installation as well. For starting ConQAT, you can use the `conqat.sh` or `conqat.bat` script in the `bin` folder.

The upload of external findings is performed using the processor *ExternalAnalysisResultsPublisher*, which accepts a scope of elements annotated with findings. Additionally, you have to configure the URL of the Teamscale server to use (including username and password), as well as the Teamscale project to upload the findings into. Note that the user provided in the parameters must have upload permissions for the project, i.e., must be in a group with the *build* role.[1]

The processor also requires an *analysis identifier*. This is very important, as it allows Teamscale to differentiate input from different analysis tools. If missing, the findings from one tool are overwritten with findings from another tool later on. The name must be the same for all runs of the same tool. A good practice is to use the name of the analysis tool as analysis identifier.

The final piece of information is the date and time of the code used to import the findings. As Teamscale uses the time information to match the findings to the correct code version and, hence, code locations, this should be as close as possible to the date and time of the analyzed revision. If not explicitely specified, the current date and time are used. Alternatively, the date and time can be directly set using the parameter *import-date* or indirectly specified via *import-revision* using the revision. In the latter case, ConQAT queries the Teamscale server for the timestamp of the commit with the corresponding revision. This will, of course, work only if the revision has already been processed by Teamscale.

In the best case, the layout of this scope (and your files on disk) are the same as in Teamscale. If not, there are also options to adjust the paths prior to the upload, but this functionality is beyond this guide. Please refer to the processor documentation for more information.

[1] Of course an *admin* account works as well, but this is not recommended for production use.

5.2 Clang Static Analyzer

The Open Source LLVM clang compiler includes a static analyzer[2] that performs dataflow analysis on C/C++ code. Analysis is started using the `scan-build` script, which wraps the make process. To enable output in *plist* format (which we require), the option `-plist` must be passed to the `scan-build` command. Additionally, you might want to enable additional checkers using the `-enable-checker` command. To obtain a list of all available checkers, run the following command:

```
clang -cc1 -analyzer-checker-help
```

Note that it is sufficient to provide a package/category to the analyzer to enable more checks. A typical invocation of the static analyzer might look like this:

```
scan-build -plist -enable-checker alpha -enable-checker core   \
           -enable-checker cplusplus -enable-checker deadcode  \
           -enable-checker security -enable-checker unix       \
           make
```

To upload analysis results, you can use the block *ClangFindingsUploader* to upload an existing report consisting of *plist* files. The block is meant to be directly executed via a run config file.

5.3 Findbugs

Findbugs[3] is an Open Source bug pattern search tool working on the byte code of Java programs. This means that in order to use Findbugs, you have to first compile the code.

To upload Findbugs results, you can use the block *FindbugsFindingsUploader* to upload an existing Findbugs report. As Findbugs is bundled with ConQAT, you can also perform the entire analysis (execution and upload) using the block *FindbugsFindingsRunner*. Both blocks are meant to be directly executed via a run config file.

5.4 Goanna

Goanna[4] is a static analysis tool working on C/C++ programs. Teamscale can read the XML report written by Goanna. To generate this output, run Goanna with the `--output-xml=<file>` parameter. For more information on running Goanna, please consult Goanna's manual.

To upload Goanna results, you can use the block *GoannaFindingsUploader* to upload an existing Goanna XML report. The block is meant to be directly executed via a run config file.

[2] http://clang-analyzer.llvm.org/
[3] http://findbugs.sourceforge.net/
[4] http://redlizards.com/

5.5 Test coverage tools

5.5.1 JaCoCo

JaCoCo[5] is an Open Source code coverage tool for Java. Teamscale allows to upload XML coverage reports produced by JaCoCo. To create an XML report with the JaCoCo Ant Tasks use the *xml* sub element in the *jacoco:report* task.

```
<jacoco:report>

    <executiondata>
        <file file="jacoco.exec"/>
    </executiondata>

    <structure name="Example Project">
        <classfiles>
            <fileset dir="classes"/>
        </classfiles>
        <sourcefiles encoding="UTF-8">
            <fileset dir="src"/>
        </sourcefiles>
    </structure>

    <xml destdir="report"/>

</jacoco:report>
```

For detailed information on the usage of the JaCoCo Ant tasks, please refer to the JaCoCo user guide to find out how to generate an XML coverage report.

To upload JaCoCo coverage data, you can use the block *JaCoCoCoverageUploader*.

5.5.2 Cobertura

Cobertura[6] is an Open Source code coverage tool for Java. Teamscale allows to upload XML coverage reports produced by Cobertura. You can generate an XML report with the Cobertura Ant task *cobertura-report* by setting the parameter *format* to *xml* as follows:

```
<cobertura-report format="xml" destdir="${coveragereport.dir}"
   srcdir="${src.dir}" />
```

For further details on the usage of the Cobertura Ant tasks, please refer to the Cobertura user manual.

To upload Cobertura coverage data, you can use the block *CoberturaCoverageUploader*.

[5] http://www.eclemma.org/jacoco/
[6] http://cobertura.github.io/cobertura/

5.5.3 gcov

The tool *gcov*[7] measures line coverage for C/C++ code compiled with the GCC compiler. To determine the code coverage, the code must be compiled using the switch `--coverage`. This creates files with the extension *gcno* during compilation. During execution of the code, the coverage data is stored into files with ending *gcda*. The program *gcov* is then used to convert the coverage data into readable files with the extension *gcov* as follows:

```
gcov -o <OBJ_DIR> -l -p <SOURCE_FILES>
```

Here the `<OBJ_DIR>` is the directory containing the object files. For projects with many subdirectories it should be noted that the parameter must point directly to the directory containing the matching object. The relative path of the source files is not used. The paramters `-l` and `-p` are used to ensure that files with the same name but different directories do not collide.

Alternatively, the python script *gcovr*[8] can be used to execute gcov for complex project layouts. To preserve the gcov files, the parameter `--keep` must be used.

To upload coverage results to Teamscale, you can use the block *GcovCoverageUploader* to upload an existing report consisting of one or more *gcov* files. The block is meant to be directly executed via a run config file.

5.6 ESLint

While ESLint is an internal analysis tool, the ESLint binaries are not shipped with Teamscale. Instead, you will need to install NodeJS[9] on your machine. Once NodeJS is installed, you can run the following command to install the correct ESLint version:

```
npm i -g eslint@0.24.1
```

You will need to have both the NodeJS and ESLint binaries on the PATH and restart Teamscale before you can enable ESLint for your project.

5.7 PC-lint

PC-lint[10] is a static analysis tool for C/C++ that covers a wide range of checks. While the tool is delivered as a Windows executable, it can also be used with Linux using wine and a wrapper script[11]. PC-lint is fully integrated into Teamscale as an incremental analysis tool, but the tool has to be on the path or the explicit path to the tool must be configured in the analysis profile.

[7] http://gcc.gnu.org/onlinedocs/gcc/Gcov.html
[8] http://gcovr.com/
[9] https://nodejs.org/
[10] http://www.gimpel.com/html/pcl.htm
[11] http://www.approxion.com/?p=135

Note for running Teamscale on Windows: If PC-lint crashes during the execution, the Windows crash dialog will open and block Teamscale until the dialog is closed manually. To prevent this behavior, log in as the user running Teamscale and set the following key in the registry to 1:

```
HKEY_CURRENT_USER\Software\ Microsoft\Windows\Windows Error Reporting\DontShowUI
```

5.8 Pylint

Pylint[12] is an Open Source code analysis tool for Python working directly on the source code.

To import Pylint analysis results into Teamscale, you need to create the generated report using a special format. This can be achieved by using the following command line call:

```
pylint -r n --msg-template='{{line: {line}, messageId: "{msg_id}", message: "{msg}",
    absolutePath: "{abspath}", messageName: "{symbol}"}}' > report.json
```

This command generates a JSON file which can be uploaded to Teamscale with the block *PylintFindingsUploader* (the block also supports to upload multiple JSON files at once). The block is meant to be directly executed via a run config file.

5.9 StyleCop

StyleCop[13] is an Open Source style violation checker for C# working directly on the source code. StyleCop is full integrated into Teamscale as an incremental analysis step, so to run StyleCop, you only have to enable it when configuring the analysis profile. When running on Linux, mono has to be installed to execute StyleCop.

5.10 SAP Code Inspector

SAP Code Inspector is an analysis tool for ABAP shipped together with SAP NetWeaver Application Server ABAP, and can be executed with transaction code `SCI`. Teamscale is able to import findings of SAP Code Inspector. To enable Code Inspector findings in Teamscale, the following steps are required:

- In SAP NetWeaver Application Server ABAP, configure an execution variant for program `ZCQSE_ABAP_EXPORT` for a complete export of ABAP source code, there provide information on the Code Inspector variant and variant user (if it is a user-specific Code Inspector variant, leave blank in case of an global Code Inspector variant). See section 2.8.2 for details on defining an execution variant.
- Schedule a (nightly) background job to which executes `ZCQSE_ABAP_EXPORT` with the defined variant, this is similar as described in section 2.8.4.
- Upload the findings to Teamscale using *CodeInspectorFindingsUploader*. The block is meant to be directly executed via a run config file.

[12] http://www.pylint.org/
[13] http://stylecop.codeplex.com/

6 Updates

This chapter contains instructions on how to migrate user- and project-specific data from an existing installation to newer versions of Teamscale.

6.1 Upgrading from previous versions of Teamsale

This section describes the general procedure for migrating from previous versions of Teamscale. If you are migrating from Teamscale 1.0 or to Teamscale 1.3, please refer to the subsequent sections and follow the steps described there instead.

1. Export a complete backup via the Backup tab of the Admin perspective.
2. In case you have configured an external data directory (not within the Teamscale installation directory), make sure to delete this directory before starting the new Teamscale version.
3. If you use the integration with SAP NetWeaver Application Server ABAP (see 2.8), make sure to backup the internal Git repository holding the ABAP code (the directory specified with the **-r** option of ABAP Teamscale Importer, see section 2.8.4).
4. Import the backup ZIP file into the new Teamscale instance via the Backup tab of the Admin perspective.
5. You may have to delete the default administrative user (*admin*) after importing the backup as it is always created when starting Teamscale on an empty storage directory.

6.1.1 Specific steps for migrating from Teamscale 1.0

To migrate from Teamscale 1.0, perform the following steps.

1. Export all project configurations to disk by clicking the download icon for all projects in the Projects tab of the Projects perspective.
2. Export a complete backup via the Backup tab of the Admin perspective.
3. The backup file must be modified as follows. Each project configuration must be renamed to *Project-Config* and placed in the corresponding project sub folder in the backup ZIP file. As an example using the *zip* command line tool on Linux you can update the ZIP as follows.

   ```
   zip teamscale-backup.zip myproject/Project-Config
   ```

   Of course, you may also use a graphical ZIP tool to add the project configuration files.
4. In case you have configured an external data directory (not within the Teamscale installation directory), make sure to delete this directory before starting the new Teamscale version.
5. Import the updated ZIP file into the new Teamscale instance via the Backup tab of the Admin perspective.

6.1.2 Specific steps for migrating to Teamscale 1.3

Note: These migration steps are only required for installations that use the architecture assessment capabilities of Teamscale.

Teamscale 1.3 introduced the possibility to model architecture specifications also in a namespace-oriented manner, in addition to the previous file-based paradigm. This means that the include and exclude patterns defined for the architecture components are matched against the full qualified type names of the types in the system, instead of the file path as originating from the version control system.

A setting in the properties of the architecture in the architecture editor is used to control how an architecture is interpreted. The default setting is *type-based*, i. e. old architectures that do not have the flag set are interpreted as type-based. In case you have old architecture definitions that are modeled in a file-oriented manner, these architectures have to be migrated by setting the flag accordingly in the architecture editor. This will cause subsequent architecture assessment runs to interpret the architecture properly. However, the architecture still resides in in the version control system in historic revisions *without* the flag set. When re-analyzing a project, the architecture will be interpreted as type-based for these revision, resulting in spurious findings regarding unmatched types. To tell Teamsale to interpret such architectures as *file-based*, set the environment variable TEAMSCALE_VM_ARGS as follows:

For Linux

```
export TEAMSCALE_VM_ARGS=-Darchitecture.default.file-based=true
```

For Windows

```
set TEAMSCALE_VM_ARGS=-Darchitecture.default.file-based=true
```

6.1.3 Specific steps for migrating to Teamscale 1.3.2

Note: The following steps only apply when using Teamscale behind an Apache Webserver as described in section 2.6.

The option `AllowEncodedSlashes` must now be set to `On` instead of `NoDecode`:

```
1 AllowEncodedSlashes On
```

7 Support

If you need support configuring Teamscale please contact us using support@teamscale.com.

A SAPLink Installation and Usage

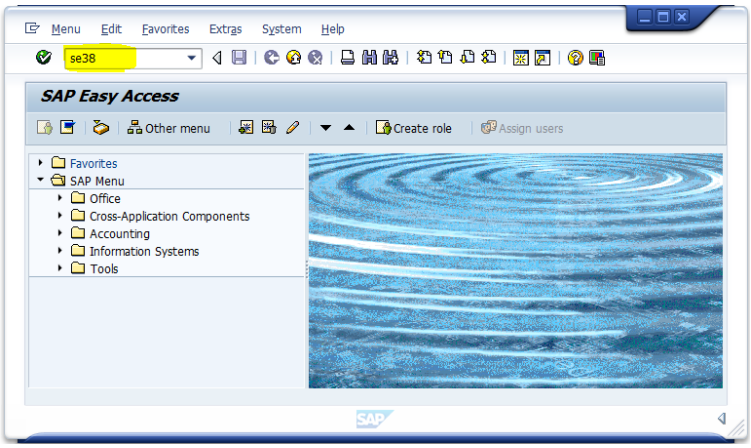


Figure A.1: Start a SAP transaction

To start a SAP transaction enter the transaction code (e.g. *SE38*) in the command field of the icon bar, see figure A.1 (The command field may be hidden. To show it, press the triangular arrow button.)

A.1 Install SAPLink

If SAPLink is already installed on the SAP NetWeaver Application Server ABAP this section can be skipped. Make sure that also all required SAPLink plug-ins are installed (for programs, classes, interfaces, function groups, text elements, DDIC types and database tables).

To install SAPLink, the installation files can be found at `<Teamscale-Dir>/abap/tools/saplink` directory of the Teamscale enterprise distribution.

- Log on to the SAP NetWeaver Application Server ABAP. Make sure the user has modification rights in the ABAP Workbench (SE38, SE80) and a developer key.
- Run transaction SE38 and create `ZSAPLINK_INSTALLER` as executable program in local objects.
  - Enter program name `ZSAPLINK_INSTALLER` *Create* (**F5**)
  - In the next dialog enter a title (e.g. *SAPLink installer*) and select type *Executable program* from the attributes section, *Save* (**Enter**).
  - In the next dialog select *Local Object* (**F7**)
- Copy & paste the code in file `zsaplink_installer.txt` into the editor window of the new created program.
- *Activate* (**Shift+F9**) the installer program.

- Execute the installer
  - Run by *Direct Processing* **(F8)**
  - Enter the path to `SAPlink_Daily.nugg` as installation nugget in the dialog.
  - *Execute* **(F8)** the import. Confirm to allow file system access if requested.
  - If you are asked to confirm overwrite originals, you will usually want to use the version of SAPLink which comes with Teamscale. But confirm with a user/administrator of the SAP system.
- Activate SAPLink:
  - Run transaction `SE38` again (e.g. by pressing *Back* **(F3)** twice) and enter `ZSAPLINK` as program name.
  - Select *Activate* **(Shift+F9)**
  - In the dialog, select *Whole Worklist* **(Shift+F6)**, *Select All* **(F9)** and confirm with *Continue* **(Enter)**.
- Install SAPLink plug-ins form nugget file `SAPlink-plugins_Daily.nugg`, see the following description. For activation, e.g. `ZSAPLINK_BSP` may be selected.

A.2 Import ABAP Code Nuggets using SAPLink

Every ABAP source code object needs to be activated before it can be used. Since SAPLink does not perform the activation, all imported source code must be activated manually. (See the following descriptions.)

- Run transaction `SE38`.
- Enter `ZSAPLINK` as program name, *Execute* **(F8)**.
- In the dialog, select the *Nugget* tab and make sure that option *Import Nugget* is selected.
- The package name should be *$TMP* to import into local objects.
- Enter the path to nugget file as *Nugget File Name*
- It is recommended to enable *Overwrite Originals*.
- *Execute* **(F8)**. Confirm to allow file system access if requested.
- If you are asked to confirm to overwrite originals, you will usually want to use the version which comes with Teamscale. But confirm with a user/administrator of the SAP system.
- Remember one of the imported objects (starting with `Z...`) for the following activation.

Imported programs must be activated:

- Run transaction `SE38` (e.g. by pressing *Back* **(F3)** twice) and enter the name of program which was imported, or navigate to one of the programs using transaction `SE80`.
- Select *Activate* **(Shift+F9)**

- In the dialog, select *Whole Worklist* (**Shift+F6**), *Select All* (**F9**) and confirm with *Continue* (**Enter**).