Reporting Issues

If you experience problems running TeamCity and believe they are related to the software, please contact us with a detailed description of the issue.

To fix a problem, we may need a wide range of information about your system as well as various logs. The section below explains how to collect such information for different issues.

In this section:

Best Practices When Reporting Issues

Following these guidelines will ensure timely response and effective issue resolution. Check Feedback for appropriate ways to contact us.

- if you have issue with your build logic, make sure it is a TeamCity-related issue (i.e. it does not reproduce in the same environment without TeamCity) and include details of your investigation
- note the TeamCity version in use, including the build number (can be found in the footer and the teamcity-server.log). Consider checking if the issue is still relevant in the most recent TeamCity version;
- update previous postings of yours on the topic instead of creating new ones, if you still create a duplicate make sure to note all previous postings on the same topic you have made or found;
- post one issue per submission, post the most important issue first, or post several issues noting others if they are related;
- note the pattern of issue occurrence (first time, recurring), how it was mitigated before, whether there were any recent environment changes;
- be specific: note exact times, error messages, etc.;
- describe the expected and actual behavior;
- when reporting build procedure issues, try to reproduce them without TeamCity on the agent machine in the same environment and let us know the results;
- detail the related settings configured in TeamCity (include screenshots, settings files, actual values, REST API e ntity representations);
- include related screenshots of the TeamCity UI (always include the entire page and the browser URL in the capture);
- include related text messages as text (not as image), include the messages with all the details;
- do not include large portions (above 10Kb) of the textual data into the email text, rather attach it in a file;
- use common multi-platform file formats like .png, .txt, .zip, do not use combining formats (.pdf) until really necessary, do not use proprietary formats like .doc, .eml, rar, etc.;
- when sending files greater than 500Kb in size or more than three files, package them into a single archive;
- attach/upload archive with TeamCity server logs (see details, ideally, the entire <server home>\logs directory with all the directories inside. If impractical, all the files updated around or later than the issue time); if related

to the build-time or agent behavior, attach the entire <agent home>\logs directory
- if a specific build is affected, include the build logs for the related builds (downloaded via dedicated link from the build results' Build Log tab);
- for performance/slowness/delays issues, take a set of (10+ spread across the issue time) server or agent thread dumps during the issue occurrence and make sure to send us the dumps with the related data;
- when replacing/masking data in logs, note the replacements patterns used;
- note if there is an anti-virus installed and if there is a network proxy or reverse proxy in front of the TeamCity server;
- when relevant, note the OS, versions of any manually installed components like Java, the TeamCity distribution used (.exe, .tar.gz)
- note any customizations/not standard environment settings;
- list any non-bundled plugins installed;
- if any, note the previous manual modifications of the TeamCity Data Directory or the database;
- when suggesting an improvement or feature or asking for settings advice, detail why you need the feature and what the original goal you want to achieve is. Suggestions as to how you would like to see the feature are welcome too;
- check the sections below for common cases and specific information to collect and send to us.

Slowness, Hangings and Low Performance

If TeamCity is running slower than you would expect, please use the notes below to locate the slow process and send us all the relevant details if the process is a TeamCity one.

Determine Which Process Is Slow

If you experience a slow TeamCity web UI response, checking for changes process, server-side sources checkout, long cleanup times or other slow server activity, your target should be the machine where the TeamCity server is installed.
If the issue is related only to a single build, you will need to investigate the TeamCity agent machine which is running the build and also the server.

Investigate the system resources (CPU, memory, IO) load. If there is a high load, determine the process causing it. If it is not a TeamCity-related process, that might need addressing outside of the TeamCity scope. Also check for generic slow-down reasons like anti-virus software, maintenance times, etc.

If it is the TeamCity server that is loading the CPU/IO or there is no substantial CPU/IO load and everything runs just fine except for TeamCity, then this is to be investigated further.
Check if you have any Conflicting Software like anti-virus running on the machine and disable/uninstall it.
Check that the database used by TeamCity and the file storage of the TeamCity Data Directory do not have performance issues.

If you have a substantial TeamCity installation, check your memory settings as the first step.

Collect Data

During the slow operation, take several thread dumps of the slow process (see below for thread dump taking approaches) with 5-10 seconds interval. If the slowness continues, take several more thread dumps (e.g. 3-5 within several minutes) and then repeat after some time (e.g. 10 minutes) while the process is still being slow.

Then send us a detailed description of the issue accompanied with the thread dumps and full server (or agent) logs covering the issue. Unless it is undesirable for some reason, the preferred way is to file an issue into our issue tracker and let us know via support email. Please include all the relevant details of investigation, including the CPU/IO load information, what specifically is slow and what is not, note affected URLs, visible effects, etc.
For large amounts of data, use our file upload service to share the archives with us.

Server Thread Dump

When an operation on the server is slow, take a set of the server thread dumps (10+) spread over the time of the slowness. TeamCity automatically saves thread dumps on super slow operations, so there might already be some saved in `logs/threadDumps-<date>` directories.
It is recommended to send us an archive of the entire content of server's `<TeamCity Home>/logs/threadDumps-<date>` directories for all the recent dates.

It is recommended that you take a thread dump of the TeamCity server from the Web UI if the hanging is local and you can still open the TeamCity Administration pages: go to the Administration | Server Administration | Diagnostics page and click the Save Thread Dump button to save a dump under the `<TeamCity Home>/logs/threadDumps-<date>` directory (where you can later download the files from "Server Logs").
If the server is fully started but the web UI is not responsive, try the direct URL using the actual URL of your TeamCity server.

If the UI is not accessible (or the server is not yet fully started), you can take a server thread dump manually using the approaches described below.

You can also adjust the `teamcity.diagnostics.requestTime.threshold.ms=30000` internal property to change the timeout after which a thread dump is automatically created in the `threadDumps-<date>` directory under TeamCity logs whenever there is a user-originated web request taking longer than timeout.

Collecting CPU Profiling Data on Server

If you experience degraded server performance and the TeamCity server process is producing a large CPU load, take a CPU profiling snapshot and send it to us accompanied with the detailed description of what you were doing and what your system setup is.
You can take CPU profiling and memory snapshots by installing the server profiling plugin and following the instructions on the plugin page.

Here are some hints to get the best results from CPU profiling:

- after starting the server, wait for some time to allow it to "warm up". This can take from 5 to 20 minutes depending on the data volume that TeamCity stores.
- when a CPU usage increase is found on the server, please try to indicate what actions cause the load.
- start CPU profiling and repeat the action several times (5 - 10).
- capture a snapshot.
- archive the snapshot and send it to us including the description of the actions that cause the CPU load.

Agent Thread Dump

It is recommended that you take an agent thread dump from the Web UI: go to the Agent page, Agent Summary tab, and use the Dump threads on agent action.

If the UI is not accessible, you can take the dump thread manually using the approaches described below. Note that the TeamCity agent consists of two `java` processes: the launcher and agent itself. The agent is triggered by the launcher. You will usually be interested in the agent (nested) process and not the launcher one.

Taking Thread Dump

These can help if you are unable to take a thread dump from the TeamCity web UI.

To take a thread dump:

Under Windows

You have several options:

- To take a server thread dump if the server is run from the console, press Ctrl+Break (Ctrl+Pause on some keyboards) in the console window (this will not work for an agent, since its console belongs to the launcher process). If the server is run as a service, try running it from console by logging under the same use as configured in the service and executing "`<TeamCity server home>\bin\teamcity-server.bat run`" command.

Another approach is to figure out the process id of the TeamCity server process (it's the top-most "java" proces with "org.apache.catalina.startup.Bootstrap start" at the end og the command line and use of of the following approaches:

- run `jstack <pid_of_java_process>` in the bin directory of the Java installation used to by the process (the Java home can be looked up in the process command line. If the installation does not have jstack utility, you might need to get the java version via java -version command, download full JDK of the same version and use "jstack" utility form there). You might also need to supply "-F" flag to the command.
- use TeamCity-bundled agent thread dump tool (can be found in the agent's plugins). Run the command:

  ```
  <TeamCity agent>\plugins\stacktracesPlugin\bin\x86\JetBrains.TeamCity.Injector.exe
  <pid_of_java_process>
  ```

  Note that if the hanging process is run as a service, the thread dumping tool must be run from a console with elevated permissions (using Run as Administrator). If the service is run under System account, you might also need to launch

the thread dumping tools via "PsExec.exe -s <path to the tool>\<tool> <options>". When the service is run under a regular user, wrapping the tool invocation in `PsExec.exe -u <user> -p <password> <path to the tool>\<tool> <options>` might also help.

If neither of these work for the server running as a service, try running the server from console and not as a service. This way the first (Ctrl+Break) option can be used.

Under Linux

- run `jstack <pid_of_java_process>` (using jstack from the Java installation as used by the process) or `kill -3 <pid_of_java_process>`. In the latter case output will appear in `<TeamCity Home>/logs/catalina.out` or `<TeamCity agent home>/logs/error.log`.

See also Server Performance section above.

Database-related Slowdowns

When the server is slow, check if the problem is caused by database operations.
It is recommended to use database-specific tools.

You can also use the `debug-sql` server logging preset. Upon enabling, all the queries which take longer 1 second will be logged into the teamcity-sql.log file. The time can be changed by setting the `teamcity.sqlLog.slowQuery.threshold` internal property. The value should be set in milliseconds and is 1000 by default.

MySQL

Along with the server thread dump, please attach the output of the "show processlist;" SQL command executed in MySQL console. Like with thread dumps, it makes sense to execute the command several times if slowness occurred and send us the output.
Also, MySQL can be set up to keep a log of long queries executed with the changes in `my.ini`:

```
[mysqld]
...
log-slow-queries
long_query_time=15
```

The log can also be sent to us for analysis.

Back to top

OutOfMemory Problems

If you experience problems with TeamCity "eating" too much memory or OutOfMemoryError/"Java heap space" errors in the log, please do the following:

- Determine what process encounters the error (the actual building process, the TeamCity server, or the TeamCity agent). You can track memory and CPU usage by TeamCity with the charts on the Administration | Server Administration | Diagnostics page of your TeamCity web UI.
- If the server is to blame, please check you have increased memory settings from the default ones for using the server in production (see the section).
- If the build process is to blame, set "JVM Command Line Parameters" settings in the build runner. Increase value for `'-Xmx'` JVM option, like -Xmx1200m, e.g. Java Inspections builds may specifically need to increase `-Xmx` value.
- If the TeamCity server is to blame and increasing the memory size does not help, please report the case for us to investigate. For this, while the server is high on memory consumption, take several server thread dumps as described above, get the memory dump (see below) and all the server logs including threadDumps-* sub-directories, archive the results and send them to us for further analysis. Make sure that Xmx setting is less than 8Gb before getting the dump:
  - if a memory dump (hprof file) is created automatically the `java_xxx.hprof` file is be created in the process startup directory (`<TeamCity Home>/bin` or `<TeamCity Agent home>/bin`);
  - for the server, you can also take memory dump manually when the memory usage is at its peak. Go to the Administration | Server Administration | Diagnostics page of your TeamCity web UI and click Dump Memory Snapshot.
  - another approach to take a memory dump manually is to use the `jmap` standard JVM command line utility of the full JVM installation of the same version as the Java used by the process. Example command line is:

```
jmap -dump:file=<file_on_disk_to_save_dump_into>.hprof <pid_of_the_process>
```

See how to change JVM options for the server and for agents.

Back to top

"Too many open files" Error

1. Determine what computer it occurs on
2. Determine the process which has opened a lot of files and the files list (on Linux use `lsof`, on Windows you can use handle or TCPView for listing sockets)
3. If the number is under thousands, check the OS and the process limits on the file handles (on Linux use `ulimit -n`) and increase them if necessary. Please note that default Linux 1024 handles per process is way too small for a server application like TeamCity. Please increase the number to at least 16000. Please check the actual process limits after the change as there are different settings in the OS for settings global and per-session limits (e.g. see the post)

If the number of files is large and looks suspicious and the locking process is a TeamCity one (the TeamCity agent or server with no other web applications running), then, while the issue is still occurring, grab the list of open handles several times with several minutes interval and send the result to us for investigation together with the relevant details.

Please note that you will most probably need to reboot the machine with the error after the investigation to restore normal functioning of the applications.

Agent does not connect to the server

Please refer to Common Problems

Logging events

The TeamCity server and agent create logs that can be used to investigate issues.

> How to enable DEBUG logging on server?
> Before reproducing the problem it makes sense to enable 'DEBUG' log level for TeamCity classes.
> On the server side, go to the Administration | Server Administration | Diagnostics page and select logging preset ('debug-all', 'debug-vcs', etc).
>
> After that, DEBUG messages will go to `teamcity-*.log` files (read more).

For detailed information, please refer to the corresponding sections:

TeamCity Server Logs
Viewing Build Agent Logs

Back to top

Version Control debug logging

> To enable VCS logging on the server side, switch logging preset to "debug-vcs" in administration web UI and then retrieve `logs/teamcity-vcs.log` log file.

Most VCS operations occur on the TeamCity server, but if you're using the agent-side checkout, VCS checkout occurs on the build agents.

For the agent and the server, you can change the Log4j configuration manually in <TeamCity Home>/conf/teamcity-server-log4j.xml or <BuildAgent home>/conf/teamcity-agent-log4j.xml files to include the following fragment:

```
<category name="jetbrains.buildServer.VCS" additivity="false">
   <appender-ref ref="ROLL.VCS"/>
   <appender-ref ref="CONSOLE-ERROR"/>
   <priority value="DEBUG"/>
</category>

<category name="jetbrains.buildServer.buildTriggers.vcs" additivity="false">
   <appender-ref ref="ROLL.VCS"/>
   <appender-ref ref="CONSOLE-ERROR"/>
   <priority value="DEBUG"/>
</category>
```

Please also update the `<appender name="ROLL.VCS"` node to increase the number of the files to store:

```
<param name="maxBackupIndex" value="30"/>
```

If there are separate logging options for specific version controls, they are described below.

Subversion debug logging

> To enable SVN logging on the server side, switch the logging preset to "debug-SVN" in the administration web UI and then retrieve the `logs/teamcity-vcs.log` and `logs/teamcity-svn.log` files.

An alternative manual approach is also necessary for agent-side logging.

First, please enable the generic VCS debug logging, as described above.

Uncomment the SVN-related parts (the `SVN.LOG` appender and `javasvn.output` category) of the Log4j configuration file on the server and on the agent (if the agent-side checkout is used). The log will be saved to the `logs/teamcity-svn.log` file. Generic VCS log should be also taken from `logs/teamcity-vcs.log`

ClearCase

Uncomment the Clearcase-related lines in the `<TeamCity Home>/conf/teamcity-server-log4j.xml` file.
The log will be saved to `logs/teamcity-clearcase.log` directory.

Patch Application Problems

In case the server-side checkout is used, the "patch" that is passed from the server to the agent can be retrieved by:

- add the property `system.agent.save.patch=true` to the build configuration.
- trigger the build.

the build log and the agent log will contain the line "Patch is saved to file ${file.name}"
Get the file and supply it with the problem description.

Back to top

Logging for .NET Runners

To investigate process launch issues for .Net-related runners, enable debugging as described below. The detailed information will then be printed into the build log. It is recommended not to have the debug logging for a long time and revert the settings after investigation.

Add the `teamcity.agent.dotnet.debug=true` configuration parameter in the build configuration or on the agent and run the build.

Alternative way to enable the logging

1. Open the <agent home>/plugins/dotnetPlugin/bin folder.
2. Make a backup copy of teamcity-log4net.xml
3. Replace teamcity-log4net.xml with the content of teamcity-log4net-debug.xml

> ⚠ After a debug log is created, it is recommended to roll back the change.
> The change in the teamcity-log4net.xml will be removed on the build agent autoupgrade.

Back to top

Remote Run Problems

The changes that are sent from the IDE to the server on a remote run can be retrieved from the server .BuildServer/system/changes directory. Locate the <change_number>.changes file that corresponds to your change (you can pick the latest number available or deduce the URL of the change from the web UI).
The file contains the patch in the binary form. Please send it with the problem description.

Back to top

Logging in IntelliJ IDEA/Platform-based IDEs

To enable debug logging for the IntelliJ Platform-based IDE plugin, include the following fragment into the Log4j configuration of the <IDE home>/bin/log.xml file:

```
<appender name="TC-FILE" class="org.apache.log4j.RollingFileAppender">
  <param name="MaxFileSize" value="10Mb"/>
  <param name="MaxBackupIndex" value="10"/>
  <param name="file" value="$LOG_DIR$/idea-teamcity.log"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d [%7r] %6p - %30.30c - %m \n"/>
  </layout>
</appender>

<appender name="TC-XMLRPC-FILE" class="org.apache.log4j.RollingFileAppender">
  <param name="MaxFileSize" value="10Mb"/>
  <param name="MaxBackupIndex" value="10"/>
  <param name="file" value="$LOG_DIR$/idea-teamcity-xmlrpc.log"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d [%7r] %6p - %30.30c - %m \n"/>
  </layout>
</appender>

<category name="jetbrains.buildServer.XMLRPC" additivity="false">
  <priority value="DEBUG"/>
  <appender-ref ref="TC-XMLRPC-FILE"/>
</category>

<category name="jetbrains.buildServer" additivity="false">
  <priority value="DEBUG"/>
  <appender-ref ref="TC-FILE"/>
</category>
```

After changing this file, restart the IDE. The TeamCity plugin debug logs are saved into idea-teamcity* files and will appear in the logs directory of the IDE settings ([<IDE settings/data directory>/system/log directory.

Open in IDE Functionality Logging

(Applicable to IntelliJ IDEA and Eclipse)

Add the following JVM option before starting IDE:
`-Dteamcity.activation.debug=true`
the logging related to the open in IDE functionality will appear in the IDE console.

No Suitable Build Configurations Found for Remote Run

First of all, check that your VCS settings in IDEA correspond to the VCS settings in TeamCity.
If they do not, change them and it should fix the problem.

Secondly, check that the build configurations you expect to be suitable with your IDEA project has either server-side VCS checkout mode or agent-side checkout and NOT manual VCS checkout mode (it is not possible to apply a personal patch for a build with the manual checkout mode because TeamCity must apply that patch after the VCS checkout is done, but it does not know or manage the time when it is performed).

If the settings are the same and you do not use the manual checkout mode but the problem is there, do the following:

- Provide us with your IDEA VCS settings and TeamCity VCS settings (for the build configurations you expect to be suitable with your IDEA project)
- Enable debug logs for the TeamCity IntelliJ plugin (see above)
- Enable the TeamCity server debug logs (see above)
- In the TeamCity IntelliJ plugin, try to start a remote run build
- Provide us with the debug logs from the TeamCity IntelliJ plugin and from the TeamCity server.

Back to top

Logging in TeamCity Eclipse plugin

To enable tracing for the plugin, run Eclipse IDE with the `-debug <filename>` command line parameter. The *<filename>* portion of the argument should be a properties file containing key-value pairs. The name of each property corresponds to the plugin module and the value is either '`true`' (to enable debug) or '`false`'. Here is an example of enabling most common tracing options:

```
jetbrains.teamcity.core/debug = true
jetbrains.teamcity.core/debug/communications = false
jetbrains.teamcity.core/debug/ui = true
jetbrains.teamcity.core/debug/vcs = true
jetbrains.teamcity.core/debug/vcs/detail = true
jetbrains.teamcity.core/debug/parser = true
jetbrains.teamcity.core/debug/platform = true
jetbrains.teamcity.core/debug/teamcity = true
jetbrains.teamcity.core/perfomance/vcs = true
jetbrains.teamcity.core/perfomance/teamcity = true
```

Read more about Eclipse Debug mode Gathering Information About Your Plug-in and built-in Eclipse help.

Back to top

TeamCity Visual Studio Addin issues

TeamCity Addin logging

To capture logs from the TeamCity Visual Studio Addin:

1. Locate the Visual Studio installation directory (in the example below `c:\Program Files (x86)\Microsoft Visual Studio\2017\Common7\IDE`)

2. Run Microsoft Visual Studio executable (`INSTALLATION_DIRECTORY\Common7\IDE\devenv.exe`) from the command line with the ReSharper-related command line arguments:

- For TeamCity VS Add-in as a part of ReSharper Ultimate use **`/ReSharper.LogFile <PATH_TO_FILE>`** and **`/ReSharper.LogLevel <Normal|Verbose|Trace>`** switches

```
c:\Program Files (x86)\Microsoft Visual Studio\2017\Common7\IDE>devenv.exe
/ReSharper.LogFile C:\Users\jetbrains\Desktop\vs.log /ReSharper.LogLevel Verbose
```

- For Legacy version of TeamCity VS Add-in, use **`/TeamCity.LogFile <PATH_TO_FILE>`** and **`/TeamCity.LogLevel <Normal|Verbose|Trace>`** switches

Visual Studio logging

To troubleshoot common Visual Studio, run Microsoft Visual Studio executable (`INSTALLATION_DIRECTORY\Common7\IDE\devenv.exe`) with the **`/Log`** command Line switch and send us resulting log file.

Back to top

dotCover Issues

To collect additional logs generated by JetBrains dotCover, add the `teamcity.agent.dotCover.log` configuration parameter to the build configuration with a path to an empty directory on the agent.
All dotCover log files will be placed there and TeamCity will publish zipped logs as hidden build artifact `.teamcity/.NETCoverage/dotCoverLogs.zip`.

JVM Crashes

On a rare occasion of the TeamCity server or agent process terminating unexpectedly with no apparent reason, it can happen that this is caused by a Java runtime crash.
If this happens, the JVM regularly creates a file named `hs_err_pid*.log` in the working directory of the process. The working directory is usually `<TeamCity server or agent home>/bin` Under WIndows when running as a service it can be other like `C:\Windows\SysWOW64`. You can also search the disk for the recent files with "hs_err_pid" in the name.
See also related the Fatal Error Log section in the document.

Please send this file to us for investigation and consider updating the JVM for the server (or for agents) to the latest version available.

If you get the "There is insufficient memory for the Java Runtime Environment to continue. Native memory allocation (malloc) failed to allocate ..." message with the crash or in the crash report file, make sure to switch to 64 bits JVM or reduce -Xmx setting not to increase 1024m, see details in the memory configuration section.

Build Log Issues

While investigating issues related to a build log, we might need the raw binary build log as stored by TeamCity.
It can be downloaded via the Web UI from the Build Log build's tab: select "Verbose" log detail and use the "raw messages file" link at the top-right.

IntelliJ IDEA Inspections

The inspections result from a TeamCity build may not match inspections from a local run in IntelliJ IDEA.

To help us investigate issues with inspections, do the following:

1. Add "`system.teamcity.dont.delete.temp.result.dir=true`" to the configuration parameters
2. Add "`%system.teamcity.build.tempDir%/inspection*result/** => inspections-reports-data-%build.number%.zip`" rule to Artifact paths
3. Add "`%system.teamcity.build.tempDir%/idea-logs/** => inspections-reports-idea-logs-%build.number%.zip`" rule to Artifact paths
4. Add "`-Didea.log.path=%system.teamcity.build.tempDir%/idea-logs/`" to the runner's 'JVM command line parameters' field.
5. Run a new build.
6. Send us '`inspections-reports-*.zip`' files.

Uploading Large Data Archives

Files under 10 MB in size can be attached right into the tracker issue (if you do not want the attachments to be publicly accessible, limit the attachment visibility to "teamcity-developers" user group only).
You can also send small files (up to 2 MB) via email: teamcity-support@jetbrains.com or via online form (up to 20 MB). Please do not forget to mention your TeamCity version and environment and archive the files before attaching.

FTP
If the file is over 10 MB, you can upload the archived files to ftp://ftp.intellij.net/.uploads and let us know the exact file name.
If you receive the permission denied error on an upload attempt, please rename the file. It's OK that you do not see the file listing on the FTP.
The FTP accepts standard anonymous credentials: username: "anonymous", password: "<your e-mail>".
In addition to usual, unencrypted connections, TLS ones are also supported.

In case of access issues, time-out errors, etc. please try using passive FTP mode.

HTTP

You can upload a file via https://uploads.services.jetbrains.com/ form and let us know the exact file name.

If you cannot upload a large file in one go, try splitting the file into parts and upload them separately.

Back to top