# Common Problems



## Most frequently used documentation sections

Configuring server memory settings
Reporting server slowness issues

Back to top

## Build works locally but fails or misbehaves in TeamCity

If a build fails or otherwise misbehaves in TeamCity but you believe it should not, follow this procedure:
Find a way to run the task from a command prompt. Make sure it works on the TeamCity agent machine, under the same user as the TeamCity agent runs under, with the same environment the agent receives. If necessary, run the TeamCity agent under a different user or tweak its environment. When the command runs OK, configure the same command in a TeamCity build using the command-line runner with the custom script setting. If that works, try other runner if that feels applicable.

Here are details on the approach:
Check that the build runs fine from the command prompt when run on the same machine as the TeamCity agent and under the same user that the agent is running, with the same environment variables and the same working directory, same architecture (32/64 bit) command line.

If the TeamCity build agent is run as a service (e.g. it is installed as a Windows service), try running the TeamCity agent under a regular user with administrative permissions from the command line. See also Windows Service limitations.

If this fixes the issue, you can try to figure out why running under the service is a problem for the build. Most often this is service-specific and is not related to TeamCity directly. Also, you can set up the TeamCity agent to be run from the console all the time (e.g. configure an automatic user logon and run the agent on the user logon).

Here are the detailed steps you can use to run a build from the command line:
Assuming you have a configured build in TeamCity which is failing, do the following:

- run the build in TeamCity and see it misbehaving
- disable the agent so that no other builds run on it. This can be done while the build is still in progress
- log in to the agent machine using the same user as the one running the TeamCity agent (check the right user in the machine processes list)
- stop the agent
- in a command line console, "cd" to the checkout directory of the build in question (the directory can be looked up in the

beginning of the build log in TeamCity)
- run the build with a command line as you would do on a developer machine. This is runner-dependent. (For some runners you can look up the command line used by TeamCity in the build log, see also the `logs\teamcity-agent.log` a gent log file for the command line used by TeamCity)
- if the build fails - investigate the reason as the issue is probably not TeamCity-related and should be investigated on the machine.
- if it runs OK, continue
- in the same console window "cd" to <TeamCity agent home>\bin and start TeamCity agent from there with the `agent start` command
- ensure the runner settings in TeamCity are appropriate and should generate the same command line as you used manually. e.g. use "Command Line" build step with "Custom script" option and the same command which can be saved in a .sh or .bat file and run from the command prompt
- run the build in TeamCity selecting the agent in the Run custom build dialog
- when finished, enable the agent

If the build succeeds from the console but still fails in TeamCity, please use a command line runner in TeamCity to launch the same command as in the console. If it still behaves differently in TeamCity, most probably this is an environment or a tool issue.
If the command line runner works but the dedicated runner does not while the options are all the same, please create a new issue in our tracker detailing the case. Please attach all the build step settings, the build log, all agent logs covering the build, the command you used in the console to run the build and the full console output of the build.

Back to top

# Build is slow under TeamCity

If you experience slow builds, the first thing to do is to check the build log to see if there are some long operations or the time is just spread over the entire process.
You can compare build logs of slower and faster builds to figure out what the difference is.
You can also run the build from the console on the same machine as detailed above to see if there is any difference between the build run from the console and the build in TeamCity.

If the slowness is spread over all the operations, the agent machine resources (CPU, disk, memory, network) are to be analyzed during the build to see if there is a bottleneck in any of those. If there is, the process loading the resource is to be found and investigated (e.g. with the help of the thread dump taken via "View thread dump" link on the running build results).

If there is some long operation and it is a TeamCity-related one (before start or after end of the actual build process), the TeamCity agent and server are to be analyzed (logs and thread dumps).

If you want to turn to us with the issue, please describe the visible effects, detail the process of investigation and attach the build log, full agent logs and other data collected.

# Started Build Agent is not available on the server to run builds

First start of agent after installation or TeamCity server upgrade/plugin installation can take time as agent downloads updates form the server and auto-upgrades.
Regularly, agent should become connected in 1 to 10 minutes, depending on the agent/server network connection speed.

If the agent is not connected within that time, check the name of the agent (as configured in conf\buildAgent.properties file) and check the tabs under the Agents server UI section:

- the agent is under Connected - the agent is ready to run builds
- the agent is under Disconnected - the agent was connected to the server, but became disconnected. Check the "Inactivity reason" in the table. If the reason is "Agent has unregistered (will upgrade)", then wait for several more minutes
- the agent is under Unauthorized - all the agents connected to the server for the first time should be authorized by a server administrator

If the agent stays in the state for more than 10 minutes and you have a fast network connection between the agent and the server, please:

- check the agent process is running and the serverURL in conf\buildAgent.properties is correct;
- check that all the requirements are met;
- check agent logs (teamcity-agent.log, launcher.log, upgrade.log) for any related messages/errors;
- check server logs (teamcity-server.log) for any messages/errors mentioning agent name or IP.

If you cannot find the cause of the delayed agent upgrade in the logs, contact us and provide the full agent and server logs. Please also check/include the state of the agent processes (java ones) on the agent machine.
Back to top

# Artifacts of a build are not cleaned

If you encounter a case when artifacts are preserved while they should have been removed by the server cleanup process, please check:

- the cleanup rules of the build configuration in question, artifacts cleanup section
- presence of the icon "This build is used by other builds" in the build history line (prior to Pin action/icon on Build History)
- build's Dependencies tab, "Delivered Artifacts" section. For every build configuration, check whether "Prevent dependency artifacts clean-up" is turned ON (this is default value). If it is, then the build's artifacts are not cleaned because of the setting.
  Read more on cleanup settings.

Back to top

# Database-related issues

## "out of memory" error with internal (HSQLDB) database

If during the TeamCity server start-up you encounter errors like:

- `"error in script file line: ... out of memory"`
- `"java.sql.SQLException: out of memory",`
  perform the following:
- try increasing server memory. If this does not help, most probably this means that you have encountered internal database corruption. You can try to deal with this corruption using the notes based on the HSQLDB documentation.

A way to attempt a manual database restore:

- stop the TeamCity server
- backup the <TeamCity Data Directory>/system/buildserver.data file
- remove the <TeamCity Data Directory>/system/buildserver.data file and replace it with zero-size file of the same name
- start the TeamCity server

However, if the database does not recover automatically, chances that it can be fixed manually are minimal.

The internal (HSQL) database is not stable enough for production use and we highly recommend using an external database for TeamCity non-evaluation usage.
If you encountered database corruption, you can restore the last good backup or drop builds history and users, but preserve the settings, see Migrating to an External Database#Switching to Another Database.

## The transaction... log is full

This error can occur with an MS SQL or Sybase database. In this case we recommend increasing the transaction log for the TeamCity database. The log size can be 1 - 16 GB depending on the number of build agents in the system and the number of tests all agents report daily.

## The table 'table_name' is full

This error can occur with a MySQL database. The error indicates that the database has run out of free space either on the disk where the database files are located or in the temp directory.

## Unable to extend ... segment ... in tablespace ...

This error can occur with an Oracle database. The error indicates that Oracle could not obtain more space for a table or an index as the database has run out of space on the disk or the specified quotas are insufficient.

## NOCOUNT is enabled on MS SQL Server

`teamcity-server.log` reports that the unsupported "NOCOUNT" option is enabled on the MS SQL database server. Contact your DBA to disable the setting as described here.

## MySQL JDBC driver error: PacketTooBigException

The ER_NET_PACKET_TOO_LARGE error (PacketTooBigException / Packet for query is too large) is caused by the server-side max_allowed_packet configuration variable set to a low value or left at the default one.

The variable controls the maximum size of MySQL communication buffer with 4MB being the default for Windows builds of MySQL 5.6, whereas in popular Linux distributions (e. g. Debian and Fedora Core), this variable defaults to 16MB, for both i686 and amd64 architectures.

1. Check the value of `max_allowed_packet` configuration variable by either examining `my.cnf` / `my.ini`, or via

```
mysql> show variables like 'max_allowed_packet';
```

2. Increase (or explicitly set) the value of `max_allowed_packet` configuration variable in `my.cnf` or `my.ini`:

```
[mysqld]
max_allowed_packet=16M
```

Setting a client-side value (via `maxAllowedPacket` connection property) in the JDBC URL will have no effect, as this value cannot exceed the server-side limit.

# Database character set/collation-related problems

## Character set/collation mismatch

TeamCity reports character set/collation mismatch error: database tables/columns have a character set or collation that is not the same as the default character set or collation in your database schema. You may see this message if you are using a non-unicode character set as default for your database as TeamCity enforces unicode charset for some of the `varchar` fields.

## TeamCity displays ???? instead of national symbols

If you want to allow your local characters in texts in TeamCity (e.g. VCS messages, test names, user names, etc.), you need to migrate to a database with the appropriate character set.

## "Unique key violations" or "Duplicates found" error on restore from backup

TeamCity server restoration from a backup may fail with errors like `"Unique key violation"` or `"Duplicates found"` if the character sets (or collations) of the source and destination databases are not the same, and the cardinality of the destination character set is less than that of the source database.

To resolve the problem, select and set up the proper character set (and collation) for the destination database.

As for case sensitivity, the possible transitions are:
CS CS
CI CS
CI CI
However, it is recommended to always use CS.

If the source character set is Unicode or UTF, the destination one must also be Unicode or UTF.
If the source character set is 8-bit non-UTF, the destination one can be the same or Unicode/UTF.

This applies to TeamCity 6.0 and above.

## Resolve character set/collation-related problems

To fix a problem, perform the following steps:

1. Create a new database with the appropriate character set and collation. We recommend using a unicode case-sensitive collation: see instructions for PostgreSQL and MySQL. For MySQL, **`utf8_bin`** or **`utf8mb4_bin`** is preferred.

   > ⚠ See also PostgreSQL, MySQL, MS SQL documentation for details on character set.

2. Copy the current `<TeamCity Data Directory>/config/database.properties` file, and change the database references in the copy to the newly created database.
3. Stop the TeamCity server.
4. Use the `maintainDB` tool to migrate to the new database:

   ```
   maintainDB migrate [-A <path-to-data-dir>] -T <new-database-properties-file>
   ```

   Depending on the size of your database, the migration may take from several minutes to several hours. For more information on the `maintainDB tool`, see this section.
5. Upon the successful completion of the database migration, the `maintainDB` tool should update the `<TeamCity Data Directory>/config/database.properties` file with references to the new database. Ensure that the file has been updated. Edit the file manually if the tool fails to do it automatically.
6. Start the TeamCity server.

Back to top

## MySQL exception: Specified key was too long; max key length is 767 bytes

## Index column size too large. The maximum column size is 767 bytes

Check if the charater set of your MySQL database. It is recommended to use the `utf8` character set.

If your database uses the `utf8mb4` character set (available since MySQL 5.5), set the the following InnoDB configuration options (under `[mysqld]` section in `my.cnf` or `my.ini`) for TeamCity 2017.2.1+ to run:

- `innodb_large_prefix=1` for index key prefixes longer than 767 bytes (up to 3072 bytes) to be allowed for InnoDB tables that use DYNAMIC row format (deprecated in MySQL 5.7.7).
- `innodb_file_format=Barracuda` to enable the DYNAMIC row format.
- `innodb_file_per_table=1` to enable the Barracuda file format

The parameters above have the following default values:

| | MySQL 5.5 | MySQL 5.6 | MySQL 5.7 | MariaDB 5.5 | MariaDB 10.0 | MariaDB 10.1 | MariaDB 10.2 |
|---|---|---|---|---|---|---|---|
| innodb_large_prefix | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| innodb_file_format | Antelope | Antelope | Barracuda | Antelope | Antelope | Antelope | Barracuda |
| innodb_file_per_table | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

Depending on the database server version, the following configuration changes need to be made:

MySQL 5.5:

- `innodb_large_prefix=1`
- `innodb_file_format=Barracuda`
- `innodb_file_per_table=1`

MySQL 5.6 and MariaDB from 5.5 to 10.1:

- `innodb_file_format=Barracuda`
- `innodb_file_per_table=1`

For MySQL 5.7+ and MariaDB 10.2+ use the defaults, no changes are required.

Back to top

## 'This driver is not configured for integrated authentication' error with MS SQL database

During TeamCity installation, the following error might occur when connecting and creating the MS SQL database with Windows integrated security: "SQL error when doing: Taking a connection from the data source: This driver is not configured for integrated authentication."

The most common reason for the problem is the different bitness of the `sqljdbc_auth.dll` MS SQL shared library and the JRE used by TeamCity.

To solve the problem, do the following:

a. Make sure you use the MS SQL native driver (downloadable from the Microsoft Download Center).
b. Use the right JRE bitness — ensure that you are running TeamCity using Java with the same bitness as your `sqljdbc_auth.dll` MS SQL shared library.

By default, TeamCity uses the 32-bit Java. However, both 32-bit and 64-bit Java versions can be used.

To run TeamCity with the required JRE, do one of the following:

- either set the TEAMCITY_JRE environment variable
- or remove the JRE bundled with TeamCity from `<TeamCity home>\jre` and set JAVA_HOME.

> ⚠ Note that on upgrade, TeamCity will overwrite the existing JRE with the default 32-bit version, so you'll have to update to the 64-bit JRE again after upgrade.

See also this related external posting.

Back to top

## Protocol violation error (Oracle only)

This error can occur when the Oracle JDBC driver is not compatible with the Oracle server. For example, Oracle JDBC driver version 11.1 is not compatible with Oracle server version 10.2.

In order to resolve the problem, use the Oracle JDBC driver from your Oracle server installation, or download the driver of the same version as the Oracle server.

# Common Maven issues

There are two kinds of Maven-related issues commonly seen in the TeamCity build configurations:

- Error message on "Maven" tab of build configuration: "An error occurred during collecting Maven project information ... "
- Error message in build configuration with Maven dependencies trigger activated: "Unable to check for Maven dependency Update ..."

If the build configuration produces successful builds despite displaying such error messages, these errors are likely to be caused by the server-side Maven misconfiguration.

To collect information for the Maven tab, or to perform Maven dependencies check (for the trigger), TeamCity runs the embedded Maven. The execution is performed on the server machine, and any agent-side maven settings are not accessible. TeamCity resolves the `settings.xml` files on the server-side separately, as described on this documentation page.

It makes sense to check if the `server-side settings.xml` files contain correct information about remote repositories, proxies, mirrors, profiles, credentials etc.

Back to top

# "Critical error in configuration file" errors

If you encounter the error, it means the settings stored in the TeamCity Data Directory are in inconsistent state. This can occur after manual change of the files or if newer version of TeamCity starts to report the inconsistencies.
To resolve the issue, you can edit the file noted in the message on the server file system. (make sure to create backup copy of the file before any manual edits). Usually server restart is not necessary for the changes to take effect.

VCS root with id "XXX" does not exist
The build configuration or template reference a VCS root which is not defined in the system.
Remedy actions: Restore the VCS root or create a new VCS root with the id noted or edit the file noted in the message to remove the reference to the VCS root.

Back to top

# TeamCity installation problems

If the TeamCity Web UI cannot be accessed after installation, you might be running TeamCity on a port that is already in use by another program. Check and configure your TeamCity installation.

Back to top

# Problems with TeamCity NuGet Feed

If you are experiencing issues with partial TeamCity NuGet Feed, i.e. missing NuGet packages etc., you might have to reindex the TeamCity NuGet Feed.

To force TeamCity to reindex all available packages and reset the NuGet package list, navigate to the server Administratio n| Diagnostics | Caches and use the buildsMetadata Reset link.

For earlier versions, refer to this section.

Back to top

# Problems with .Net-related TeamCity Tools

## Startup performance issues

After upgrade to TeamCity 9.0 or later, .NET Framework below version 4.0 installed on TeamCity agents may cause performance issues of .Net-related TeamCity tools due to Code access security (CAS) policy imposed by Microsoft.

To solve the issue, use one of the options:

a. Add the following setting described in the Microsoft documentation to the `machine.config` file on all agents:

```
<configuration>
 <runtime>
  <generatePublisherEvidence enabled="false"/>
 </runtime>
</configuration>
```

   You can modify the machine.config file as described in this external blog post and pass this config file to all agents, e.g. using a custom script.

b. Alternatively, upgrade .Net Framework on the TeamCity agents to version 4.0 and above. Details are available in the Microsoft documentation.

Back to top