Configuring General Settings

When creating a build configuration, specify the following settings:

Setting	Description
Name	The build configuration name
Build Configuration ID	The ID of the build configuration (must be unique across all build configurations and templates in the system).
Description	An optional description for the build configuration.
Build Number Format	This value is assigned to the build number. For more information, refer to the Build Number Format section below.
Build Counter	Specify the counter to be used in the build numbering. Each build increases the build counter by 1. Use the Reset counter link to reset counter value to 1.
Artifact Paths	Patterns to define artifacts of a build. For more information, refer to the Artifact Paths section below.
Build Options	Additional options for this build configuration. For more information, refer to the following sections below: • Hanging Build Detection • Allow Triggering Personal Builds • Enable Status Widget • HTML Status Widget • Limit Number of Simultaneously Running Builds

Build Number Format

In the Build number format field you can specify a pattern which is resolved and assigned to the Build Number on the build start.

The license could not be verified: License Certificate has expired!

The following substitutions are supported in the pattern:

Pattern	Description
`%build.counter%`	a build counter unique for each build configuration. It is maintained by TeamCity and will resolve to a next integer value on each new build start. The current value of the counter can be edited in the Build counter field.
`%build.vcs.number.<VCS_root_name>%`	the revision used for the build of the VCS root with `<VCS_root_name>` name. Read more on the property.
`%property.name%`	a value of the build property with the corresponding name. All the Predefined Build Parameters are supported (including Reference-only server properties).

A build number format example:
`1.0.%build.counter%.%build.vcs.number.My_Project_svn%`

Though not required, it is still highly recommended to ensure the build numbers are unique. Please include the build counter in the build number and do not reset the build counter to lesser values.
It is also possible to change the build number from within your build script. For details, refer to Build Script Interaction with TeamCity.

Artifact Paths

Build artifacts are files produced by the build which are stored on TeamCity server. Read more about build artifacts.
On the General Settings page of the build configuration, you can specify explicit paths to build artifacts or patterns to define

artifacts of a build.

Explicit Paths

If you know the names of your build artifacts and their exact paths, you can specify them here.

If you have a build finished on an agent, you can use the checkout directory browser  and select artifacts from the tree. TeamCity will place the paths to them into the input field.

Paths Patterns

The Artifact Paths field also supports newline- or comma-delimited patterns of the following format:

```
file_name|directory_name|wildcard [ => target_directory|target_archive ]
```

The format contains:

- `file_name` — to publish the file. The name should be relative to the Build Checkout Directory.
- `directory_name` — to publish all the files and subdirectories within the directory specified. The directory name should be a path relative to the Build Checkout Directory. The files will be published preserving the directories structure under the directory specified (the directory itself will not be included).
- `wildcard` — to publish files matching Ant-like wildcard pattern (only "*" and "**" wildcards are supported). The wildcard should represent a path relative to the build checkout directory. The files will be published preserving the structure of the directories matched by the wildcard (directories matched by "static" text will not be created). That is, TeamCity will create directories starting from the first occurrence of the wildcard in the pattern.
- `target_directory` — the directory in the resulting build's artifacts that will contain the files determined by the left part of the pattern. This path is a relative one with the root being the root of the build artifacts.
- `target_archive` — the path to the archive to be created by TeamCity by packing build artifacts determined in the left part of the pattern. TeamCity treats the right part of the pattern as `target_archive` whenever it ends with a supported archive extension, i.e. `.zip`, `.7z`, `.jar`, `.tar.gz`, or `.tgz`.

Although absolute paths are supported, it is recommended to use paths relative to the build checkout directory.

The optional part starting with the `=>` symbols and followed by the target directory name can be used to publish the files into the specified target directory. If the target directory is omitted, the files are published in the root of the build artifacts. You can use "." (dot) as a reference to the build checkout directory.

The target paths must not be absolute. Non-relative paths will produce errors during the build.

You can use build parameters in the artifacts specification. For example, use "`mylib-%system.build.number%.zip`" to refer to a file with the build number in the name.

Examples:

The same `target_archive` name can be used multiple times, for example:

```
*/*.html => report.zip
*/*.css => report.zip!/css/
```

Relative paths inside a zip archive can be used, if needed:

```
results\result1\Dir1\Dir2 => archive.zip!results/result1/Dir1
```

- install.zip — publish file named `install.zip` in the build artifacts
- dist — publish the content of the `dist` directory
- target/*.jar — publish all jar files in the `target` directory
- target/**/*.txt => docs — publish all the txt files found in the `target` directory and its subdirectories. The files will be available in the build artifacts under the `docs` directory.
- reports => reports, distrib/idea*.zip — publish reports directory as `reports` and files matching `idea*.zip` from the `distrib` directory into the artifacts root.

Build Options

The following options are available for build configurations:

Hanging Build Detection

Select the Enable hanging build detection option to detect probably "hanging" builds. A build is considered to be "hanging" if its run time significantly exceeds estimated average run time and the build has not send any messages since the estimation was exceeded. To properly detect hanging builds, TeamCity has to estimate the average time builds run based on several builds. Thus, if you have a new build configuration, it may make sense to enable this feature after a couple of builds have run, so that TeamCity would have enough information to estimate the average run time.

Allow Triggering Personal Builds

Since TeamCity 9.1, you can restrict running personal builds by unchecking the allow triggering personal builds option (on by default).

Enable Status Widget

This option enables retrieving the status and basic details of the last build in the build configuration without requiring any user authentication.
Please note that this also allows getting the status of any specific build in the build configuration (however, builds cannot be listed and no other information except the build status (success/failure/internal error/cancelled) is available).

The status can be retrieved via the HTML status widget described below, or via a single icon with the help of REST API.

HTML Status Widget

This feature allows you to get an overview of the current project status on your company's website, wiki, Confluence or any other web page.
When the Enable status widget option is enabled, an HTML snippet can be included into an external web page and will display the current build configuration status.
For build status icon as a single image, check REST build status icon.

The following build process information is provided by the status widget:

- The latest build results,
- Build number,
- Build status,
- Link to the latest build artifacts.
  The status widget doesn't require users log in to TeamCity.

When the feature is enabled, you need to include the following snippets of code in the web page source:

- Add this code sample in the `<head>` section (or alternatively, add the withCss=true parameter to `externalStatus.html`):

```
<style type="text/css">
 @import "<TeamCity_server_URL>/css/status/externalStatus.css";
</style>
```

- Insert this code sample where you want to display the build configuration status:

```
<script type="text/javascript" src="<TeamCity_server_URL>/externalStatus.html?js=1">
</script>
```

- If you prefer to use plain HTML instead of javascript, omit the js=1 parameter and use iframe instead of the script:

```
<iframe src="<TeamCity_server_URL>/externalStatus.html"/>
```

- If you want to include default CSS styles without modifying the `<head>` section, add the withCss=true parameter

To provide up-to-date status information on specific build configurations, use the following parameter in the URL as many times as needed:

```
&buildTypeId=<external build configuration ID>
```

It is also possible to show the status of all projects build configurations by replacing "`&buildTypeId=<external build configuration ID>`" with "`&projectId=<external project ID>`". You can select a combination of these parameters to display the needed projects and build configurations on your web page.

You can also download and customize the `externalStatus.css` file (for example, you can disable some columns by using `display: none`; See comments in `externalStatus.css`). But in this case, you must not include the withCss=true parameter, but provide the CSS styles explicitly, preferably in the `<head>` section, instead.

Enabling the status widget also allows non-logged in users to get the RSS feed for the build configuration.

Limit Number of Simultaneously Running Builds

Specify the number of builds of the same configuration that can run simultaneously on all agents. This option helps avoid the situation, when all of the agents are busy with the builds of a single project. Enter 0 to allow an unlimited number of builds to run simultaneously.

See also:

- Concepts: Build Configuration