Configuring Schedule Triggers

The Schedule Trigger allows you to set the time when a build of the configuration will be run. The Builds Schedule page of the current project settings displays the configured build times. More than one Schedule trigger can be added to a build configuration.

On this page:

Triggering Conditions

The settings in this section define time and other conditions for automatic build triggering. You can schedule a recurring build or set a specific date and time for it.

Date and Time

In addition to triggering builds daily or weekly at a specified time for a particular time zone, you can specify advanced time settings using cron-like expressions. This format provides more flexible scheduling options.

TeamCity uses Quartz for working with cron expressions. See the examples below or consider using the CronMaker utility to generate expressions based on Quartz cron format.

Examples

	Each 2 hours at :30	Every day at 11:45PM	Every Sunday at 1:00AM	Every last day of month at 10:00AM and 10:00PM
Seconds	0	0	0	0
Minutes	30	45	0	0
Hours	0/2	23	1	10,22
Day-of-month	*	*	?	L
Month	*	*	*	*
Day-of-week	?	?	1	?
Year(Optional)	*	*	*	*

See also other examples.

Brief description of the cron format used

Cron expressions are comprised of six fields and one optional field separated with a white space. The fields are respectively described as follows:

Field Name	Values	Special Characters
Seconds	0-59	, - * /

Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	1-12 of JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * ? / L #
Year(Optional)	empty, 1970-2099	, - * /

For the description of the special characters, please refer to Quartz CronTrigger Tutorial.

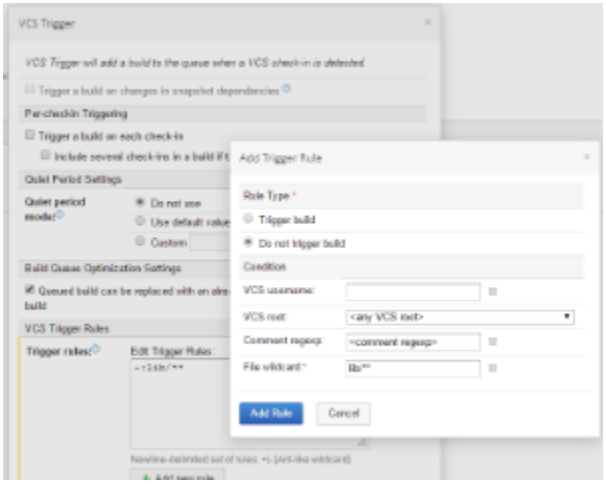
VCS Changes

You can restrict schedule trigger to start builds only if there are pending changes in your version control by selecting the corresponding option. The Trigger only if there are pending changes option considers newly detected pending changes only: if there were pending changes before the trigger was created, the build is not triggered.

VCS Trigger Rules

If no trigger rules specified, a build is triggered upon any detected change displayed for the build configuration. You can affect the changes detected by changing the VCS root settings and specifying Checkout Rules.

To limit the changes that trigger the build, use the VCS trigger rules. You can add these rules manually in the text area (one per line), or use the Add new rule option to generate them.



Each rule is ether an "include" (starts with "+") or an "exclude" (starts with "-").

General Syntax

The general syntax for a single rule is:

```
+|-[:[user=VCS_username;][root=VCS_root_id;][comment=VCS_comment_regexp]]:Ant_like_wildcard
```

Where:

- Ant_like_wildcard - A wildcard to match the changed file path. Only "*" and "**" patterns are supported,the "?" pattern is not supported. The file paths in the rule can be relative (not started with '/' or '\') to match resulting paths on the agent or absolute (started with '/') to match VCS paths relative to a VCS root. For each file in a change the most specific rule is found (the rule matching the longest file path). The build is triggered if there is at least one file with a matching "include" rule or a file with no matching "exclude" rules.
- VCS_username- if specified, limits the rule only to the changes made by a user with the corresponding VCS username .
- VCS_root_id - if specified, limits the rule only to the changes from the corresponding VCS root.

- VCS_comment_regexp - if specified, limits the rule only to the changes that contain specified text in the VCS comment. Use the Java Regular Expression pattern for matching the text in a comment (see examples below). The rule matches if the comment text contains a matched text portion; to match the entire text, include the ^ and $ special characters.

> When specifying the rules, please note that as soon as you enter any "+" rule, TeamCity will change the implicit default from "include all" to "exclude all". To include all the files, use "+:." rule.
>
> Also, rules are sorted according to path specificity. I.e. if you have an explicit inclusion rule for `/some/path`, and exclusion rule `-:user=some_user:.` for all paths, commits to the `/some/path` from `some_user` will be included unless you add a specific exclusion rule for this user and this path at once, like `-:user=some_user:/some/path/**`

Trigger Rules Examples

```
+:.
-:**.html
-:user=techwriter;root=InternalSVN:/misc/doc/*.xml
-:lib/**
-:comment=minor:**
-:comment=^oops$:**
```

- +:. includes all files
- "`-:**.html`" excludes all `.html` files from triggering a build.
- "`-:user=techwriter;root=InternalSVN:/misc/doc/*.xml`" excludes builds being triggered by `.xml` files checked in by the VCS user "tech writer" to the `misc/doc` directory of the VCS root named Internal SVN (as defined in the VCS Settings). Note that the path is absolute (starts with "/"), thus the file path is matched from the VCS root.
- "`-:lib/**`" prevents the build from triggering by updates to the "lib" directory of the build sources (as it appears on the agent). Note that the path is relative, so all files placed into the directory (by processing VCS root checkout rules) will not cause the build to be triggered.
- "`-:comment=minor:**`" prevents the build from triggering, if the changes check in comment contains word "minor".
- "`-:comment=^oops$:**`" no triggering if the comment consists of the only word "oops" (according to Java Regular Expression principles ^ and $ in pattern stand for string beginning and ending)

Build Changes

The Schedule Trigger can watch a build in a different build configuration and trigger a build if the watched build changes.

In the Build Changes section, select the corresponding box and specify the build configuration and the type of build to watch: last successful build, last pinned build, last finished build, or the last finished build with a specified tag.

TeamCity can promote the watched build if there is a dependency (snapshot or artifact) on its build configuration.

Additional Options

Enforce Clean Checkout

It is possible to force TeamCity to clean all files in the checkout directory before a build. This option can also be applied to snapshot dependencies. In this case, all the builds of the build chain will be forced to use clean checkout. The option also enables rebuilding all dependencies (unless custom dependencies are provided via the custom build dialog or the schedule trigger promotes a build).

Trigger Build on All Enabled and Compatible Agents

Use this option to run a build simultaneously on all agents that are enabled and compatible with the build configuration. This option may be useful in the following cases:

- run a build for agent maintenance purposes (e.g. you can create a configuration to check whether agents function properly after an environment upgrade/update).

- run a build on different platforms (for example, you can set up a configuration, and specify for it a number of compatible build agents with different environments installed).

Build Queue Optimization Settings

By default, TeamCity optimizes the build queue: already queued build can be replaced with an already started build or a more recent queued build. To disable the default behavior, uncheck the box.

Branch Filter

When a VCS Root has branches configured, the Branch filter setting appears in the trigger options.

The Branch filter setting limits a set of logical branch names (the branch names as they appear in the TeamCity UI) which trigger should apply to. The branch filter uses format and precedence similar to the branch specification (but it matches logical branch name and uses no parenthesis).

Details of the format:

```
+:logical branch name
-:logical branch name
```

where `logical branch name` is the part of the branch name matched by the branch specification (i.e. displayed for a build in TeamCity UI), see Working With Feature Branches.

It is possible to use the wildcard placeholder ('*') which matches one or more characters: `+|-:name*` will match the branch '`name1`' but will not match the branch '`name`', which will need to be added explicitly.

Other examples:

Only the default branch is accepted:

```
+:<default>
```

All branches except the default one are accepted:

```
+:*
-:<default>
```

Only branches with the `feature-` prefix are accepted:

```
+:feature-*
```

By default, the branch filter is set to `+:*`, which is the equivalent of an empty branch filter. In this case that a build will be triggered on every branch tracked by the branch specification.

If several rules match a single branch, the most specific (least characters matched by pattern) last rule apply.

The branch filter in the Schedule Trigger works as follows:

- if the option trigger build only if there are pending changes is turned ON, then the trigger will add a build to the queue for all branches matched by the trigger branch filter where pending changes exist
- if trigger build only if there are pending changes is turned OFF, then the trigger will add a build to the queue for all branches matched by the trigger branch filter regardless of presence of pending changes there

Trigger Rules and Branch Filter Combined

Trigger rules and branch filter are combined by AND, which means that the build is triggered only when both conditions are satisfied.

For example, if you specify a comment text in the trigger rules field and provide the branch specification, the build will be triggered only if a commit has the special text and is also in a branch matched by branch filter.