Configuring VCS Triggers

VCS triggers automatically start a new build each time TeamCity detects new changes in the configured VCS roots and displays the change in the pending changes. Only one VCS trigger can be added to a build configuration.

A new VCS trigger with the default settings triggers a build once there are pending changes in the build configuration: the version control is polled for changes according to the checking for changes interval of a VCS root honoring a VCS commit hook i f configured. Only the changes matched by the checkout rules are displayed as pending and thus are processed by the trigger. If several check-ins are made within short time frame and discovered by TeamCity together, only one build will be triggered.

After the last change is detected, a quiet period can be configured to wait for some time without changes before the build is queued.

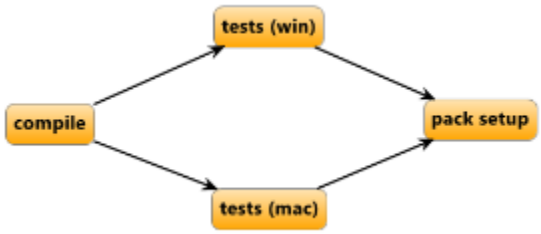
The global default value for both options is 60 seconds and can be configured for the server on the Administration | Global Settings page.

You can adjust a VCS trigger to your needs using the options described below:

Trigger a build on changes in snapshot dependencies

If you have a build chain (i.e. a number of builds interconnected by snapshot dependencies), the triggers are to be configured in the final build in the chain. This is pack setup in the image below.

Let's take a build chain from the example: `pack setup`--depends on--`tests`--depends on--`compile`.



With the VCS Trigger set up in the `pack setup` configuration, the whole build chain is usually triggered when TeamCity detects changes in `pack setup`; changes in `compile` will trigger `compile` only and not the whole chain. If you want the whole chain to be triggered on a VCS change in `compile`, add a VCS trigger with the Trigger on changes in snapshot dependencies op tion enabled to the final build configuration of the chain, `pack setup`.
This will not change the order in which builds are executed, but will only trigger the whole build chain, if there is a change in any of snapshot dependencies. In this setup, no VCS triggers are required for the `compile` or `tests` build configurations.

If triggering rules are specified (described below), they are applied to all the changes (including changes from snapshot dependencies) and only the changes matching the rules trigger the build chain.

See also details at the Build Dependencies page.

Per-check-in Triggering

When this option is not enabled, several check-ins by different committers can be made; and once they are detected, TeamCity will add only one build to the queue with all of these changes.

If you have fast builds and enough build agents, you can make TeamCity launch a new build for each check-in ensuring that no other changes get into the same build. To do that, select the Trigger a build on each check-in option. If you select the Include several check-ins in build if they are from the same committer option, and TeamCity will detect a number of pending changes, it will group them by user and start builds having single user changes only.

This helps to figure out whose change broke a build or caused a new test failure, should such issue arise.

Quiet Period Settings

By specifying the quiet period you can ensure the build is not triggered in the middle of non-atomic check-ins consisting of several VCS check-ins.

A quiet period is a period (in seconds) that TeamCity maintains between the moment the last VCS change is detected and a build is added into the queue. If new VCS change is detected in the Build Configuration within the period, the period starts over from the new change detection time. The build is added into the queue only if there were no new VCS changes detected within the quiet period. Note that the actual quiet period will not be less than the maximum checking for changes interval among the VCS roots of a build configuration, as TeamCity must ensure that changes were collected at least once during the quiet period.

The quiet period can be set to the default value (60 seconds, can be changed globally at the Administration | Global Settings pa ge) or to a custom value for a build configuration.

> ⚠ Note that when a build is triggered by a trigger with the VCS quiet period set, the build is put into the queue with fixed VCS revisions. This ensures the build will be started with only the specific changes included. Under certain circumstances this build can later become a History Build.

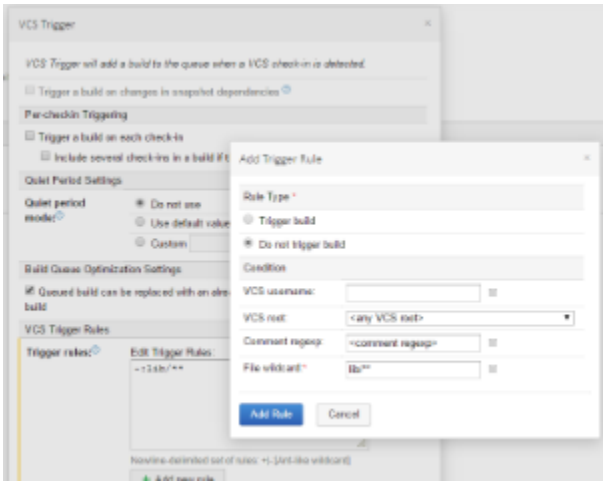
Build Queue Optimization Settings

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.

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 either 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 principle s ^ and $ in pattern stand for string beginning and ending)

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.

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.

Triggering a Build on Branch Merge

The VCS trigger is fully aware of branches and will trigger a build once a check-in is detected in a branch.

When changes are merged / fast-forwarded from one branch to another, strictly speaking there are no actual changes in the code. By default, the VCS trigger behaves in the following way:

- When merging/fast forwarding of two non-default branches: the changes in a build are calculated with regard to previous builds in the same branch, so if there is a build on same commit in a different branch, the trigger will start a build in another branch pointing to the same commit.
- If the default branch is one of the branches in the merging/fast-forwarding, the changes are always calculated against the default branch, if there is a build on same revision in the default branch, TeamCity will not run a new build on the same revision.