Build Configuration Template

On this page:

Overview

Build Configuration Templates allow you to eliminate duplication of build configuration settings. If you want to have several similar (not necessarily identical) build configurations and be able to modify their common settings in one place without having to edit each configuration, create a build configuration template with those settings. Modifying template settings affects all buil d configurations associated with this template. Since TeamCity 2017.2, it is possible to define a default template in a project for all build configurations in this project and its subprojects. See the section below for details.

Build configuration templates support project hierarchy: once created, available templates include the ones from the current project and its parents. On copying a project or a build configuration, the templates that do not belong to the target project or one of its parents are automatically copied. You can associate a build configuration a template only if the template belongs to the current project or one of its parents.

If you upgraded from versions earlier than TeamCity 8.0, your configurations from one project might be associated with templates from an unrelated project. After upgrading to TeamCity 8+, such templates may become inaccessible. To reuse build configuration templates from an unrelated project, manually move the templates into the common parent project (or the Root project if you want them to be globally available).

Creating build configuration template

There are several ways to create a build configuration template:

- Manually, like a regular build configuration.
- Extract from an existing build configuration: there is the Extract template option available from the Actions button at the top right corner of the screen. Note that if you extract a template from a build configuration, the original configuration automatically becomes associated with the newly created template.

Defining default template for project

Default templates, available since TeamCity 2017.2, allow affecting all build configurations in this project and its subprojects: you can easilty add a specific build feature to all build configurations of a project, or switch all build configurations to some specific checkout mode, or provide a default failure condition.

You can associate all the build configurations of the project with a default template using the General Settings page of the project administration, by selecting a template from the Default template drop-down. The option is available if at least one template is defined in the project or its parent. All new build configurations will inherit the default template settings.

The settings of the existing configurations will be preserved.

If defined, it affects all build configurations and subprojects of this project unless other default templates are defined in subprojects. With default templates, you can easily modify all project's build configurations, for example:

- add a specific build feature to all build configurations of a project,
- switch all build configurations to some specific checkout mode,

- provide a default failure condition.

Related settings changes

Since 2017.2 the project configuration schema was changed to accommodate for default templates.

- XML: The project-config.xml file now containts the `<default-template ref="...." />` element.
- DSL: Project configuration now contains the defaultTemplate = "..." method. See a sample project configuration with a default template configured:

```
object Project : Project({
   uuid = "2b241ffb-9019-4e60-9a3a-d5475ab1f312"
   extId = "ExampleProject"
   parentId = "_Root"
   name = "Example Project"
   defaultTemplate = "ExampleProject_MyDefaultTemplate"
   ...
   features {
      ...
   }
 ...
})
```

Associating build configurations with templates

- You can create new build configurations based on a template
- You can associate/attach any number of existing build configurations with/to a template: there's the Associate with Template option (Attach to template... since TeamCity 2017.2) available from the Actions button at the top right corner of the screen.

 When you associate an existing build configuration with a template, the build configuration inherits all the settings defined in the template, and if there's a conflict, the template settings supersede the settings of the build configuration (except dependencies, parameters, and requirements). The settings inherited from a template can be overridden.

You can associate a build configuration to a template only if the template belongs to the current project or one of its parents. A template which has at least one associated build configuration cannot be deleted, the associated build configurations need to be detached first. Since TeamCity 2017.2 you can associate a build configuration with multiple templates.

Associating build configuration with multiple templates

Since TeamCity 2017.2 a build configuration can be attached to multiple templates using the "Attach to template..." action menu. In the Actions menu, the Manage templates action allows users to:

- change the order of templates, which affects the overlapping settings priority and the build step order: the priority is given to the settings from the template higher in the list. It affects such entities as parameter names, setting ids (for build steps, triggers, features, artifact dependencies and requirements), VCS roots or snapshot dependency source build configurations ids if they overlap between templates attached to a build configuration.
- detach the build configuration from some of the templates (the user marks those to be detached and then has to apply their changes)
- detach the build configuration from all templates using correspondingly named button on the same dialog window

You can view all the templates attached to a build configuration on the Build Configuration Settings page.

The settings from all templates the build configuration is attached to are inherited and you can view where they are inherited from in the view/edit Build Configuration Settings pages.

When a build configuration is detached from some of its templates, all the effective (i.e. not overridden in the config and not overlapped by some higher priority template) settings inherited from them are copied to the configuration. The copying build configuration logic is the same as extracting a template. On moving a build configuration/project, the logic checks all templates to which the build configuration is attached.

Related settings changes

- XML: If a build configuration is attached to a single template, the resulting config XML format stays the same as it was before ("ref" attribute of the "settings" element). If it is attached to a number of templates, then references to them are stored in a separate element under "settings" node, as follows:

```
<inherits>
<ref id="Template1_ExternalId" />
<ref id="Template2_ExternalId />
.....
</inherits>
```

- DSL: Kotlin DSL is extended, so within a build type definition users can use the `templates(vararg)` method accepting either external ids or DSL template instances (but not a mix of them, so if both templates defined inside and outside DSL are used in the same configuration, the external ids of both must be used). The older `template(...)` method and property cannot be used multiple times within the same build type definition to indicate that it is inherited from multiple templates - following earlier implementation, each time this method is used, it overrides the previous template external id. It is preserved for backward compatibility.

Detaching build configurations from template

When you detach a build configuration from a template using the Detach from template option available from the Actions butto n at the top right corner of the build configuration settings screen, all settings from the template will be copied to the build configuration and enabled for editing.

Redefining settings inherited from template

Although a build configuration associated with a template inherits all its settings (marked as inherited in the UI), it is still possible to redefine a number settings in an associated build configuration. Modifying settings in the template will influence all configurations associated with this template.

Inherited settings cannot be deleted from an associated build configuration.

The following settings can be overridden in a build configuration inherited from a template:

- build number format
- artifact paths
- build options (hanging builds detection, triggering personal builds, status widget, number of simultaneously running builds)
- VCS checkout mode
- checkout directory
- clean all files before build
- show changes from snapshot dependencies
- execution timeout
- all common build failure conditions, including execution timeout
- parameters
- agent requirements
- artifact dependencies (since TeamCity 10.0)
- build features (since TeamCity 2017.1)
- build failure conditions (since TeamCity 2017.1)
- build triggers (since TeamCity 2017.1)

Note that if you rename an inherited parameter or an agent requirement in a build configuration, this parameter/ agent requirement is actually disconnected from the template and is treated as a new parameter created in the build configuration. For example, if you rename a parameter in an associated build configuration, and then rename the same parameter in the template, you will end up with two parameters in the build configuration: the new one inherited from the template, and the old one which was redefined in the build configuration.

Besides, you can redefine settings configured via parameter references or add new items to lists.

Modifying Settings

Text field settings
When you specify some fixed value in a text field of a template, it is inherited as is and cannot be changed in an associated build configuration.
However, in most of the text fields of your template settings (except names (build configuration, parameter, build step), descriptions, agent requirements, typed parameters definitions), you can use a reference to a build parameter instead of the actual value. Thus you can define the actual value of this parameter in each particular associated build configuration separately.
See below for an example of configuration parameters usage.

Other settings: drop-downs, lists, check boxes, password fields, etc.
These settings are inherited from a template as is and cannot be redefined in an associated build configuration.

Collections
A collection of settings, such as parameters, build steps, VCS roots, or build triggers can be extended in an inherited configuration.

- You can add a new element to a collection, for example, one more VCS root.
- In some cases, you can reorder the collection elements in the inherited configuration, for example, build steps.

Modified settings are highlighted with a yellow border and the Reset button appears on the right of the modified settings enabling you to revert the changes to the original settings of the template.

Using parameter reference

To introduce a configuration parameter reference, use the `%ParameterName%` syntax in the template text field. Once introduced, this parameter appears on the Parameters page of the build configuration template marked as requiring a value.
You can either specify the parameter's default value or leave it without any value. You can then define the actual value for the parameter in a build configuration associated with the template.

See also Configuring Build Parameters and Defining and Using Build Parameters in Build Configuration#Using Build Parameters in Build Configuration Settings.

Example of configuration parameters usage

Assume that you have two similar build configurations that differ only by checkout rules. For instance, checkout rules for the first configuration should contain `'+:release_1_0 => .'` and for the second one `'+:trunk => .'`. All other settings are equal. It would be useful to have one template to associate with both build configurations, but this means changing the checkout rules in each build configuration separately.

To do so, perform the following steps:

1. Extract a template from one of those configurations.
2. In the template settings, navigate to Version Control Settings, open the Checkout rules dialog for the VCS root and enter there: `%checkout.rules%`
3. For the inherited build configuration, open the configuration settings page and on the Parameters page specify the actual value for the `checkout.rules` configuration parameter.
4. For the second build configuration, use the Associate with template option from Actions and choose the template. Specify an appropriate value for the `checkout.rules` parameter right in the "Associate with Template" dialog. Click "Associate".

As a result, you'll have two build configurations with different checkout rules, but associated with one template.

This way you can create a configuration parameter and then reference it from any build configuration, which has a text field.

See also:

Administrator's Guide: Creating and Editing Build Configurations | Configuring Build Parameters