REST API

This page describes using TeamCity REST API in TeamCity versions 5.x-7.x. For information on REST API in the latest TeamCity version, refer to TeamCity 10.x and 2017.1, for other versions see the corresponding documentation version .

On this page:

General information

To use a REST API, an application makes an HTTP request to the TeamCity server and parses the response.

The TeamCity REST API can be used for integrating applications with TeamCity and for those who want to script interactions with the TeamCity server. TeamCity's REST API allows accessing resources (entities) via URL paths.

If your server web UI is accessible via the `http://teamcity:8111/` URL, use:
`http://teamcity:8111/httpAuth/app/rest/application.wadl` - to get the full list of supported requests and names of parameters (for basic HTTP authentication).

REST Authentication

You can authenticate yourself for the REST API in the following ways:

- Using basic HTTP authentication. Provide a valid TeamCity username and password with the request: `http://teamcity:8111/httpAuth/app/rest/application.wadl`
- Using access to the server as a guest user (if enabled): `http://teamcity:8111/guestAuth/app/rest/application.wadl`

If you perform a request from within a TeamCity build, consider using teamcity.auth.userId/teamcity.auth.password system properties as credentials (within TeamCity settings you can reference them like %system.teamcity.auth.userId% and %system.teamcity.auth.password%).

Superuser access

If you add the "rest.use.authToken=true" internal property, any user can perform superuser operation if the authToken is passed in the URL parameter. The authToken will be logged into `logs/teamcity-rest.log` log. You will still need to supply valid user credentials to use this approach.

REST API Versions

Under the `http://teamcity:8111/app/rest/` URL the latest version is available.
Under `http://teamcity:8111/app/rest/<version>` URL, earlier versions CAN be available. Our general policy is to supply TeamCity with at least ONE previous version.
In TeamCity 8.0.x you can use "6.0" or "7.0" instead of <version> to get earlier versions of the protocol.

Examples

`http://teamcity:8111/httpAuth/app/rest/version` - to get the plugin version
`http://teamcity:8111/httpAuth/app/rest/projects` - to get the list of projects, then follow hrefs
`http://teamcity:8111/httpAuth/app/rest/buildTypes/id:bt284/builds?status=SUCCESS&tag=EAP` - (example ids are used) to get builds
`http://teamcity:8111/httpAuth/app/rest/builds/?locator=<buildLocator>` - to get builds by "build locator".
`http://teamcity:8111/httpAuth/app/rest/changes?buildType=id:bt133&sinceChange=id:24234` - (example ids are used) to get all the changes in the build configuration since the change identified by the id.
`http://teamcity:8111/httpAuth/app/rest/users` - to get the TeamCity users list

As a rule, single value responses are "text/plain" and complex value responses support both "application/xml" and "application/json". Supply appropriate "Accept" header in the request to get the necessary response type.

If you get an error in response to your request and want to investigate the reason, please look into rest-related server logs.

General Notes

When posting XML, be sure to specify the "Content-Type: application/xml" HTTP header.
Requests that respond with collections (.../projects, .../buildTypes, .../builds, .../changes) serve partial items with only the most important item fields. Use URL constructed with the value of the `"href"` item attribute to get the full item data.

Build Requests

List builds: GET `http://teamcity:8111/httpAuth/app/rest/builds/?locator=<buildLocator>`
Get details of a specific build: GET `http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>` (also supports DELETE to delete a build)

Build Locator

In a number of places, a string might be specified that defines what builds to filter/affect (referred to as "<buildLocator>"). This is called "build locator" in the scope of REST API.

Examples of supported locators:

- `id:<internal build id>` - internal build id when you need to specify a specific build
- `number:<build number>` - to find build by build number, provided build configuration is already specified
- `<dimension1>:<value1>,<dimension2>:<value2>` - to find builds by multiple criteria

The list of supported build dimensions:
buildType:(<buildTypeLocator>) - only the builds of the specified build configuration
tags:<tags> - ","(comma) - a delimited list of build tags (only builds containing all the specified tags are returned)
status:<SUCCESS/FAILURE/ERROR> - list builds with the specified status only
user:(<userLocator>) - limit builds to only those triggered by the user specified
personal:<true/false/any> - limit builds by a personal flag.
canceled:<true/false/any> - limit builds by a canceled flag.
running:<true/false/any> - limit builds by a running flag.
pinned:<true/false/any> - limit builds by a pinned flag.

branch:<branch locator> - since TeamCity 7.1 limit the builds by branch. <branch locator> can be the branch name (displayed in the UI, or "(name:<name>,default:<true/false/any>,unspecified:<true/false/any>,branched:<true/false/any>)". If not specified, only builds from the default branch are returned.

agentName:<name> - agent name to return only builds ran on the agent with name specified

sinceBuild:(<buildLocator>) - limit the list of builds only to those after the one specified
sinceDate:<date> - limit the list of builds only to those started after the date specified. The date should in the same format as

dates returned by REST API (e.g. "20130305T170030+0400").

project:<project locator> - since TeamCity 8.0 limit the list to the builds of the specified project (belonging to any build type directly or indirectly under the project)

count:<number> - serve only the specified number of builds
start:<number> - list the builds from the list starting from the position specified (zero-based)
lookupLimit:<number> - since TeamCity 7.0 limit processing to the latest N builds only. If none of the latest N builds match the other specified criteria of the build locator, 404 response is returned.

If the value is to contain the "," symbol, it should be enclosed into parentheses: "(<value>)".

Build Tags

Get tags: GET `http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/tags/`
Replace tags: PUT `http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/tags/` (should put the same XML of JSON as returned by GET)
Add tags: POST `http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/tags/` (should post the same XML of JSON as returned by GET or just a plain-text tag name)
(<buildLocator> here should match a single build only)

Build Pinning

Get current pin status: GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/pin/ (returns "true" or "false" text)
Pin: PUT http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/pin/ (the text in the request data is added as a comment for the action)
Unpin: DELETE http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/pin/ (the text in the request data is added as a comment for the action)
(<buildLocator> here should match a single build only)

Artifacts

Since TeamCity 7.0 (and deprecated in 8.0):
GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/artifacts/files/<artifact relative name> (returns the content of a build artifact)

Since TeamCity 8.0
GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/artifacts/content/<artifact relative name> (returns the content of a build artifact)
Media-Type: application/octet-stream or a more specific media type (determined from artifact name)
Possible error: 400 if the specified path references a directory

GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/artifacts/metadata/<artifact relative name> (returns information about a build artifact)
Media-Type: application/xml or application/json

GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/artifacts/children/<artifact relative name> (returns the list of artifact children for directories and archives)
Media-Type: application/xml or application/json
Possible error: 400 if the artifact is neither a directory nor an archive

<artifact relative name> supports referencing files under archives using "!/" delimiter after the archive name.

Examples:
GET http://teamcity:8111/httpAuth/app/rest/builds/id:100/artifacts/children/my-great-tool-0.1.jar!/META-INF
GET http://teamcity:8111/httpAuth/app/rest/builds/id:100/artifacts/metadata/my-great-tool-0.1.jar!/META-INF/MANIFEST.MF
GET http://teamcity:8111/httpAuth/app/rest/builds/id:100/artifacts/metadata/my-great-tool-0.1.jar!/lib/commons-logging-1.1.1.jar!/META-INF/MANIFEST.MF
GET http://teamcity:8111/httpAuth/app/rest/builds/id:100/artifacts/content/my-great-tool-0.1.jar!/lib/commons-logging-1.1.1.jar!/META-INF/MANIFEST.MF

Authentication

If you download the artifacts from within a TeamCity build, consider using `teamcity.auth.userId/teamcity.auth.password` system properties as credentials for the download artifacts request: this way TeamCity will have a way to record that one build used artifacts of another and will display it on the build's Dependencies tab.

Other Build Requests

Build fields:
Get single build's field: GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/<field_name> (accepts/produces text/plain) where <field_name> is one of "number", "status", "id", "branchName" and other build's bean attributes

Statistics:
Get build statistics: GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/statistics/ only standard/bundled statistic values are listed. See also Custom Chart#listOfDefaultStatisticValues
Get single build statistics value: GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/statistics/<value_name>

Projects and Build Configuration Lists

List of projects: GET http://teamcity:8111/httpAuth/app/rest/projects
Project details: GET http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>
<projectLocator> can be `id:<internal_project_id>` or `name:<project%20name>`

List of Build Configurations: GET http://teamcity:8111/httpAuth/app/rest/buildTypes
List of Build Configurations of a project: GET http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/buildTypes
Build Configuration details: GET http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>
<buildTypeLocator> can be `id:<btXXX_internal_buildConfiguration_id>` or `name:<Build%20Configuration%20name>`

Build Configuration And Template Settings

Get build configuration details: GET http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>

Please note that there is no transaction, etc. support for settings editing in TeamCity, so all the settings modified via REST API are taken into account at once. This can result in half-configured builds triggered, etc. Please make sure you pause a build configuration before changing its settings if this aspect is important for your case.

Since TeamCity 7.0
Get/set paused build configuration state: GET/PUT http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/paused (put "true" or "false" text as text/plain)
Build configuration settings: GET/DELETE/PUT http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/settings/<setting_name>
Build configuration parameters: GET/DELETE/PUT http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/parameters/<parameter_name> (accepts/produces text/plain)
Build configuration steps: GET/DELETE http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/steps/<step_id>
Create build configuration step: POST http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/steps
The XML posted is the same as retrieved by GET request to `.../steps/<step_id>`

Features, triggers, agent requirements, artifact and snapshot dependencies follow the same pattern as steps with URLs like:
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/features/<id>
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/triggers/<id>
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/agent-requirements/<id>
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/artifact-dependencies/<id>
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/snapshot-dependencies/<id>

Build configuration VCS roots: GET/DELETE http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/vcs-root-entries/<id>
Attach VCS root to a build configuration: POST http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/vcs-root-entries The XML posted is the same as retrieved by GET request to http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/vcs-root-entries/<id>

Create a new empty build configuration: POST plain text (name) to http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/buildTypes
Copy a build configuration: POST XML `<newBuildTypeDescription name='Conf Name' sourceBuildTypeLocator='id:bt42' copyAllAssociatedSettings='true' shareVCSRoots='false'/>` to http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/buildTypes

Read, detach and attach a build configuration from/to a template: GET/DELETE/PUT http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/template (PUT accepts template locator with "text/plain" Content-Type)

VCS Roots

Since TeamCity 7.0

List all VCS roots: GET `http://teamcity:8111/httpAuth/app/rest/vcs-roots`
Get details of a VCS root/delete a VCS root: GET/DELETE `http://teamcity:8111/httpAuth/app/rest/vcs-roots/<vcsRootLocator>`
Where <vcsRootLocator> is "id:<internal VCS root id>"
Create a new VCS root: POST VCS rot XML (the one like retrieved for a GET request for VCS root details) to `http://teamcity:8111/httpAuth/app/rest/vcs-roots`

Also supported:
GET/PUT `http://teamcity:8111/httpAuth/app/rest/vcs-roots/<vcsRootLocator>/properties/<property_name>`
GET/PUT `http://teamcity:8111/httpAuth/app/rest/vcs-roots/<vcsRootLocator>/<field_name>` where <field_name> is one of the following: name, shared, project (post project locator to "project" to associate a VCS root with a specific project). Before TeamCity 8.0 project used to be a "projectId".

Project Settings

Get project details: GET `http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>`

Since TeamCity 7.0
Delete a project: DELETE `http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>`
Create a new empty project: POST plain text (name) to `http://teamcity:8111/httpAuth/app/rest/projects/`

Since TeamCity 8.0: Create (or copy) a project: POST XML `<newProjectDescription name='New Project Name' id='newProjectId' copyAllAssociatedSettings='true'><parentProject locator='id:project1'/><sourceProject locator='id:project2'/></newProjectDescription>` to `http://teamcity:8111/httpAuth/app/rest/projects`. See also an example.
TeamCity 7.0-7.1.x: Copy a project: POST XML `<newProjectDescription name='Project Name' sourceProjectLocator='id:project2' copyAllAssociatedSettings='true' shareVCSRoots='false'/>` to `http://teamcity:8111/httpAuth/app/rest/projects`

Edit project parameters: GET/DELETE/PUT `http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/parameters/<parameter_name>` (accepts/produces text/plain)
Project name/description/archived status: GET/PUT `http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/<field_name>` (accepts/produces text/plain) where <field_name> is one of "name", "description", "archived".

Since TeamCity 8.0
Project's parent project: GET/PUT XML `http://teamcity:8111/httpAuth/app/rest/projects/<projectLocator>/parentProject`

Data Backup

Start backup: POST `http://teamcity:8111/httpAuth/app/rest/server/backup?includeConfigs=true&includeDatabase=true&includeBuildLogs=true&fileName=<fileName>` where <fileName> is the prefix of the file to save backup to. The file will be created in the default backup directory (see more).
Get current backup status (idle/running): GET `http://teamcity:8111/httpAuth/app/rest/server/backup`

Users

List of users: GET `http://teamcity:8111/httpAuth/app/rest/users`
Get specific user details: GET `http://teamcity:8111/httpAuth/app/rest/users/<userLocator>`
Create a user: POST `http://teamcity:8111/httpAuth/app/rest/users`
Update specific user: PUT `http://teamcity:8111/httpAuth/app/rest/users/<userLocator>`
For POST and PUT requests for a user, post data in the form retrieved by corresponding GET request. Only the following attributes/elements are supported: name, username, email, password, roles, groups, properties.
Work with user roles: `http://teamcity:8111/httpAuth/app/rest/users/<userLocator>/roles`

<userLocator> can be of a form:

- `id:<internal user id>` - to reference the user by internal ID
- `username:<user's username>` - to reference the user by username/login name

Since TeamCity 7.0
User's single field: GET/PUT `http://teamcity:8111/httpAuth/app/rest/users/<userLocator>/<field name>`
User's single property: GET/DELETE/PUT `http://teamcity:8111/httpAuth/app/rest/users/<userLocator>/properties/<property name>`

Agents

List of agents: GET `http://teamcity:8111/httpAuth/app/rest/agents`
List of connected agents: GET `http://teamcity:8111/httpAuth/app/rest/agents?includeDisconnected=false`
List of authorized agents: GET `http://teamcity:8111/httpAuth/app/rest/agents?includeUnauthorized=false`

Since TeamCity 7.0
Agent's single field: GET/PUT `http://teamcity:8111/httpAuth/app/rest/agents/<agentLocator>/<field name>`
See also an example for agent enabling/disabling

Build Status Icon

since TeamCity 7.1
Icon that represents build status: GET `http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/statusIcon`
This allows embedding a build status icon into any HTML page with a simple `img` tag:

> For build configuration with internal id "btXXX":
> Status of the last build: <img src="http://teamcity:8111/app/rest/builds/buildType:(id:btXXX)/statusIcon"/>
> Status of the last build tagged with tag "myTag": <img src="http://teamcity:8111/app/rest/builds/buildType:(id:btXXX),tag:myTag/statusIcon"/>

All other `<buildLocator>` options are supported.
If the returned image contains "no permission" text, please ensure that one of the following is true:

- the server has the guest user access enabled and the guest user has permissions to access the build configuration referenced, or
- the build configuration referenced has the "enable status widget" option ON
- you are logged in to the TeamCity server in the same browser and you have permissions to view the build configuration referenced

CCTray

Since TeamCity 7.0
CCTray-compatible XML is available via `http://teamcity:8111/httpAuth/app/rest/cctray/projects.xml`.

Without authentication (only build configurations available for guest user): `http://teamcity:8111/guestAuth/app/rest/cctray/projects.xml`.

CORS Support

Since TeamCity 7.1, REST can be configured to allow Cross-origin requests.
If you want to allow requests from a page loaded from a specific domain, add the domain to comma-separated internal property `rest.cors.origins`.
e.g.

> rest.cors.origins=google.com,myinternalwebpage.org.com

If that does not work, please enable debug logging and investigate the log lines, which are usually descriptive.

Request Sending Tool

One may use curl command line tool to try REST API from a command line.
Example command:

```
curl -v --basic --user USERNAME:PASSWORD --request POST "http://teamcity:8111/httpAuth/app/rest/users/"
--data @data.xml --header "Content-Type: application/xml"
```

Where USERNAME, PASSWORD, "teamcity:8111" should be substituted with real values and data.xml file contains the data to send to the server.

Examples

Making user a system administrator

1. Enable superuser in REST
create a file `<TeamCity Data Directory>\config\internal.properties` with the content:

> rest.use.authToken=true

(add the line if the file already exists)

2. Get authToken
restart the TeamCity server and look into `<TeamCity home>\logs\teamcity-rest.log` for a line:

> Authentication token for superuser generated: 'XXX-YYY-...-ZZZ'.

Copy this "XXX-YYY-...-ZZZ" string. The string is unique for each server restart

3.Issue the request
Get curl command line tool and use a command line:

> curl -v --request PUT http://USER:PASSWORD@teamcity:8111/httpAuth/app/rest/users/username:USERNAME/roles/SYSTEM_ADMIN/g/?authToken=XXX-YYY-...-ZZZ

where
"USER" and "PASSWORD" - the credentials of a valid TeamCity user (that you can log in with)
"teamcity:8111" - the TeamCity server URL
"USERNAME" - the username of the user to be made the system administrator
"XXX-YYY-...-ZZZ" - the authentication token retrieved earlier

Creating a new project

Using curl tool

> curl -v -u USER:PASSWORD http://teamcity:8111/app/rest/projects --header "Content-Type: application/xml" --data-binary "<newProjectDescription name='New Project Name' id='newProjectId'><parentProject locator='id:project1'/></newProjectDescription>"