REST API Plugin

Added by Yegor Yarko, last edited by Yegor Yarko on Feb 07, 2012

General Info

Vendor	JetBrains
License	Apache 2.0
Type	free, open-source

Plugin Description

Exposes TeamCity API via REST.

The plugin is bundled since TeamCity 5.0

Usage

If your server is accessible via `http://teamcity:8111/` URL, use:
`http://teamcity:8111/httpAuth/app/rest/application.wadl` - to the get list of supported requests and names of parameters. Provide valid TeamCity username and password with the request as BASIC HTTP authentication.

For example:
`http://teamcity:8111/httpAuth/app/rest/version` - to get plugin version
`http://teamcity:8111/httpAuth/app/rest/projects` - to get projects list, then follow href's
`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 build configuration since the change identified by id.
`http://teamcity:8111/httpAuth/app/rest/users` - to get 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 necessary response type.

If you get error in response to your request and want to investigate the reason, enable debug logging in the server logs for "jetbrains.buildServer.server.rest" category and then see `logs/teamcity-rest.log` file.

Feel free to ask questions and provide feedback in our plugins forum.

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>` - use it 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) -delimited list of build tags (only builds containing all the specified tags are returned)
status:<SUCCESS/FAILURE/ERROR> - list the builds with the specified status only
user:(<userLocator>) - limit the builds to only those triggered by user specified
personal:<true/false/any> - limit the builds by personal flag.
canceled:<true/false/any> - limit the builds by canceled flag.
running:<true/false/any> - limit the builds by running flag.
pinned:<true/false/any> - limit the builds by pinned flag.
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.

count:<number> - serve only the specified number of builds
start:<number> - list the builds fromt he list starting from the position specified (zero-based)

If the value is to contain "," 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)

Build Statistics

Get build statistics value: GET http://teamcity:8111/httpAuth/app/rest/builds/<buildLocator>/statistics/
(<buildLocator> here should match a single build only)

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>

Since TeamCity 7.0
Build configuration settings: GET/DELETE/PUT
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/settings/<s
Build configuration parameters: GET/DELETE/PUT
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/parameters/
(accepts/produces text/plain)
Build configuration steps: GET/DELETE
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/steps/<step
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

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/<i
http://teamcity:8111/httpAuth/app/rest/buildTypes/<buildTypeLocator>/triggers/<i
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

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, projectId (use projectId to associate a VCS root with a specific project)

Project Settings

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

Since TeamCity 7.0
Create a new empty project: POST plain text (name) to
http://teamcity:8111/httpAuth/app/rest/projects/
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/<par
(accepts/produces text/plain)

Superuser access

If you add "rest.use.authToken=true" internal property, any user can perform superuser operation if authToken is passed in 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.

Data Backup

Start backup: POST http://teamcity:8111/httpAuth/app/rest/server/backup?includeConfigs=true&includeDatabase=true&includeBuildLogs=true&fileName=<fileNam
where <fileName> in the prefix of the file to save backup to
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

Since TeamCity 7.0
Agent's single field: GET/PUT
http://teamcity:8111/httpAuth/app/rest/agents/<agentLocator>/<field name>

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.

REST API Versions

As REST API evolves from one TeamCity version to another there can be incompatible changes into the protocol.
Under http://teamcity:8111/app/rest/ URL the latest version is available.
Under http://teamcity:8111/app/rest/<version> URL, other versions CAN be available. Our general policy is to supply TeamCity with ONE previous version.
In TeamCity 7.0 you can use "6.0" instead of <version> to get previous version of the protocol.

Please note that additions to the objects returned (such as new XML attributes or elements) are not considered major changes and does not cause the protocol version to increment.

Development links

Sources

If you need to extend the plugin with your functionality, you can base your plugin on the current REST API plugin code, but ensure that your plugin does not interfere with the bundled REST plugin. To achieve this, change `teamcity-plugin.xml` file to have different plugin name and different value for "api.path" parameter. Once this is done, your patched plugin and original REST api plugin can work in the single TeamCity installation.

TeamCity Versions Compatibility

TeamCity 5.0 and above.

Labels: None

1 Comment

Yegor Yarko Jan 23, 2012

Also a bit of details and curl command line description for build configuration editing in TW-8394