TeamCity Artifactory Plug-in

Overview

Artifactory provides tight integration with TeamCity CI Server through the TeamCity Artifactory Plug-in. Beyond managing efficient deployment of your artifacts to Artifactory, the plug-in lets you capture information about artifacts deployed, dependencies resolved, environment data associated with the TeamCity build runs and more, that effectively provides full traceability for your builds.

From version 2.1.0 the TeamCity Artifactory Plug-in provides powerful features for release management and promotion. For details please refer to TeamCity Artifactory Plugin - Release Management.

Before you begin

Please refer to the general information about **Artifactory's Build Integration** before using the TeamCity Artifactory Plugin.

Source Code Available!

The TeamCity Artifactory Plugin is an open source project on GitHub which you can freely browse and fork.

Build Runner Support

The TeamCity Artifactory plugin supports most build runner types, including: **Maven2, Maven 3**, **Ivy/Ant** (with Ivy modules support), **Gradle** , **NAnt**, **MSBuild**, **FxCop** and **Ipr**.

Installing the Plugin

Plugins are deployed to TeamCity by placing the packaged plugin into the *$<TeamCity Data Directory>/plugins directory* and restarting TeamCity. You can also accomplish this via the TeamCity UI via **Administration | Plugins List | Upload Plugin Zip** and choosing the zip-file from your file-system. You will need to restart TeamCity (tomcat) for the plugin to take effect.

Download the latest version of the plugin:

Remove older versions

If you have an older version of the plug-in, be sure to remove it before upgrading to a newer one

Page Contents

Read More

- TeamCity Artifactory Plugin - Release Management

Configuration

To use the TeamCity Artifactory plugin you first need to configure your Artifactory servers in TeamCity's server configuration. You can then set up a project build runner to deploy artifacts and Build Info to a repository on one of the Artifactory servers configured.

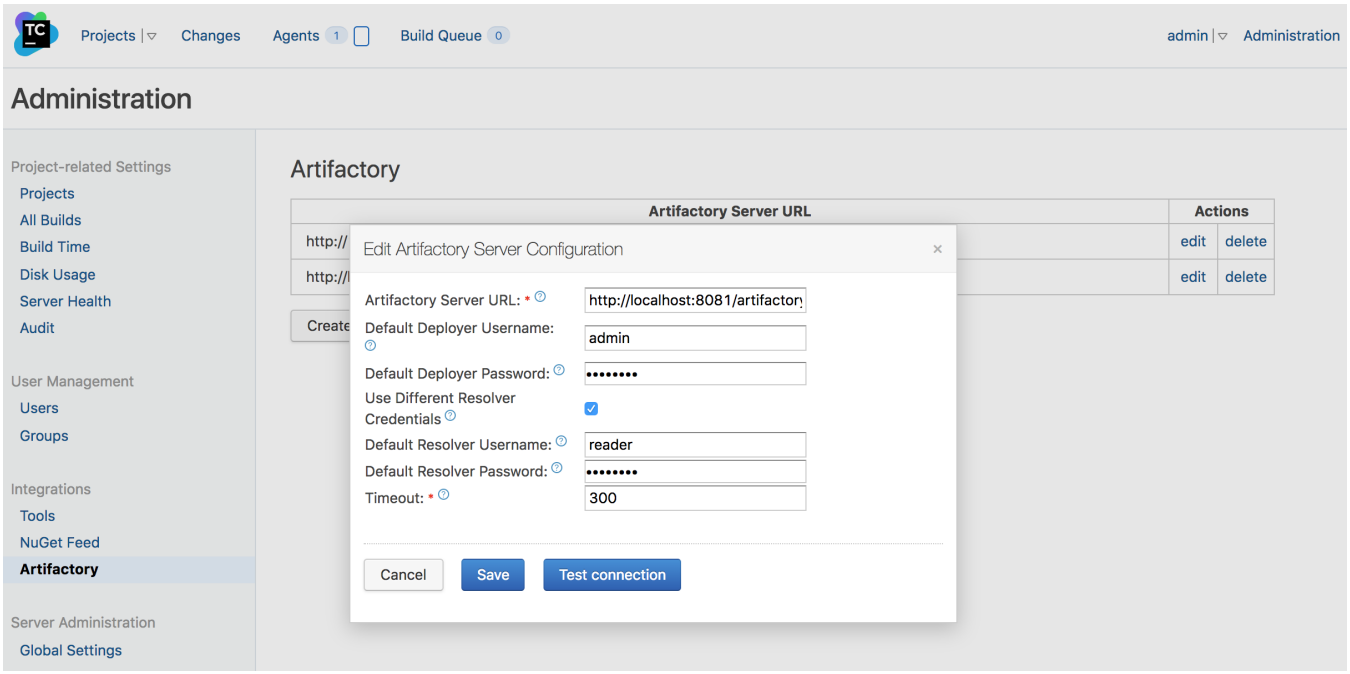
Configuring System-wide Artifactory Servers

To make Artifactory servers globally available to project runner configurations, they must be defined in **Administration | Integrations | Artifactory**.

Select **Create new Artifactory server configuration** and fill in the URL of the Artifactory server.

Deployer credentials can be set at the global level for all builds, but they can also be overridden and set at a project build level.

Specifying a username and password for the resolver repository is optional. It is only used when querying Artifactory's REST API for a list of configured repositories and then only if the target instance does not allow anonymous access.



Configuring Project-specific Runners

Editing Project-specific Configuration

To set up a project runner to deploy build info and artifacts to Artifactory go to **Administration | Projects** and select the project you want to configure.

Then, under the **Build Configurations** section, click the **Edit** link for the build you want to configure.

Under **Build Configuration Settings,** select the relevant **Build Step** and click the **Edit** link for the build step you want to configure.

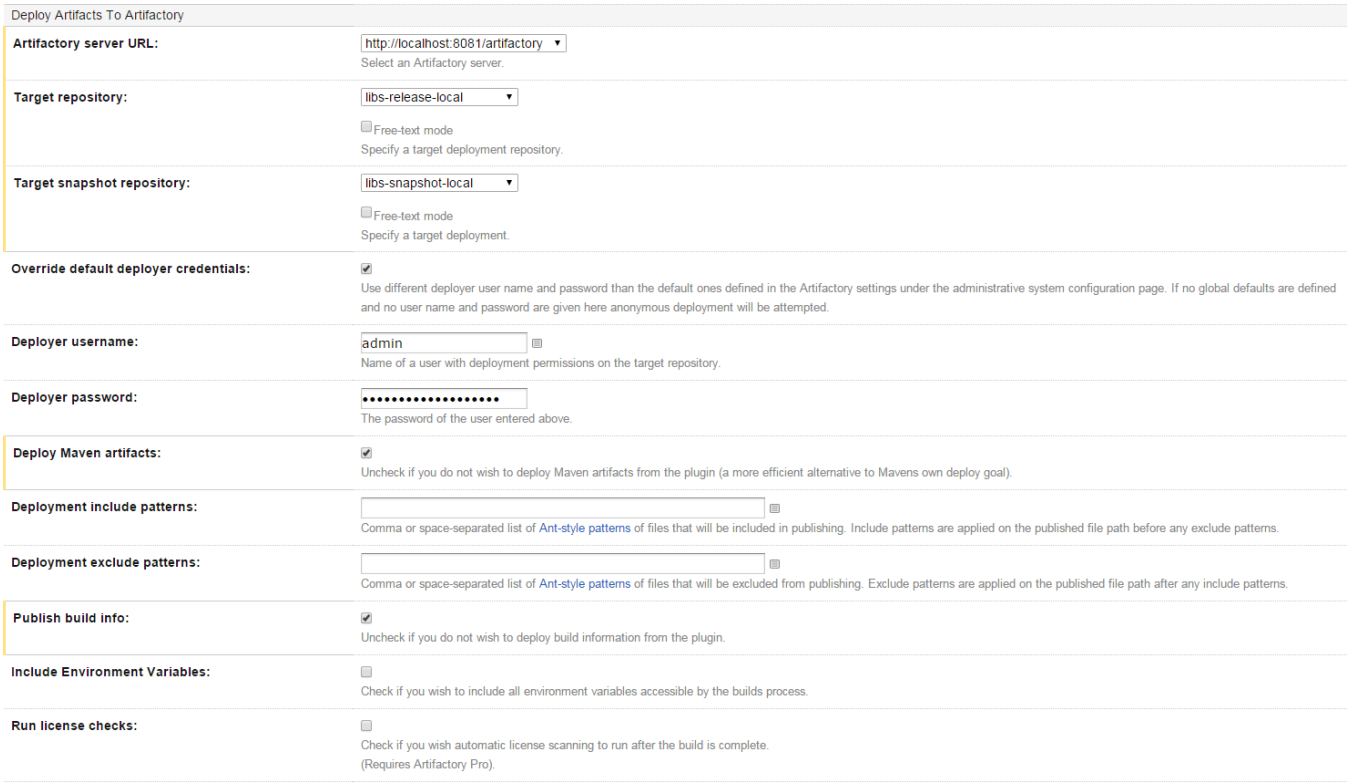
When you select a value in the **Artifactory server URL** field, the selected server is queried for a list of configured repositories (using the credentials configured in the corresponding **Artifactory Server Configuration**). This populates the **Target Repository** field with a list of repositories to which you can select to deploy.

Clicking on the **Free-text mode** checkbox enables you to type in repository name as free text. You may also include variables as part of the text. For example: libs-%variableName%

Configuration errors

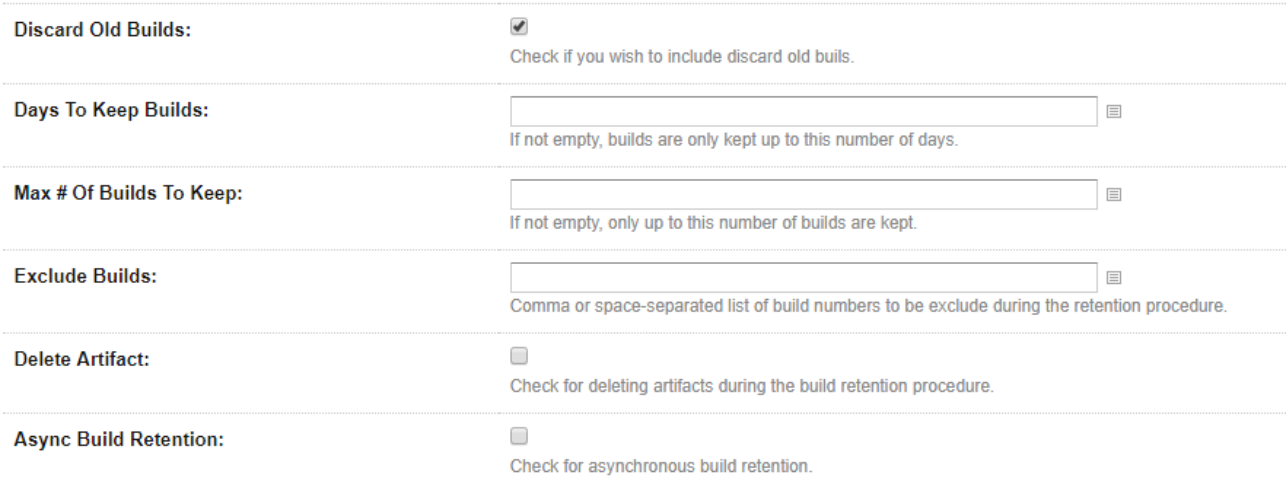
If the **Target Repository** list remains empty, check that the specified Artifactory server URL, credentials and proxy information (if provided) are valid.

Any information about communication errors that might occur can be found in the TeamCity server logs.



Triggering Build Retention in Artifactory

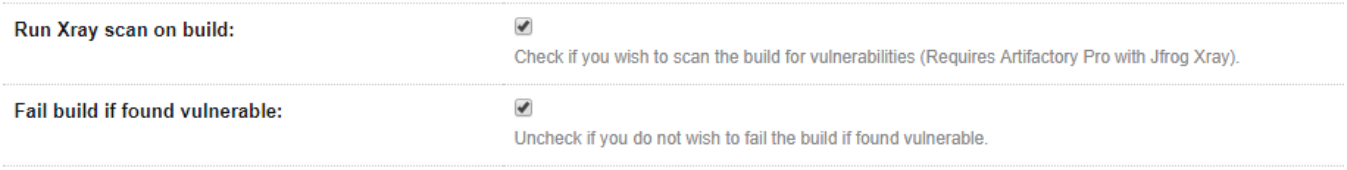
You can trigger build retention when publishing build-info to Artifactory.



Scanning Builds with JFrog Xray

The TeamCity Artifactory Plugin is integrated with JFrog Xray through JFrog Artifactory, allowing you to have build artifacts scanned for vulnerabilities and other issues. If issues or vulnerabilities are found, you may choose to fail a build job. The scan result details are always printed into the build log. This integration requires **JFrog Artifactory v4.16** and above and **JFrog Xray v1.6** and above.

For Xray to scan builds, you need to configure a **Watch** with the right filters that specify which artifacts and vulnerabilities should trigger an alert, and set a Fail Build Job Action for that Watch. You can read more about CI/CD integration with Xray here.



Running License Checks

If you are using Artifactory Pro, you can benefit from the License Control feature to discover and handle third party dependency licensing issues as part of the build.
If you check the **Run License Checks** checkbox, Artifactory will scan and check the licenses of all dependencies used by this build. You can also specify a list of recipients who should receive any license violation notifications by email.

Generic Build Integration

Generic build integration provides Build Info support for the following runner types:

- Command Line
- FxCop
- MSBuild
- Rake
- Powershell
- XCode Project
- NuGet Publish
- NAnt
- Visual Studio (sln)
- Visual Studio 2003
- SBT, Scala build tool

This allows the above builds to:

1. Upload any artifacts to Artifactory, together with custom properties metadata, and keep published artifacts associated with the TeamCity build.
2. Download artifacts from Artifactory that are required by your build.

You can define the artifacts to upload and download by either using "File Specs" or "Legacy Patterns".

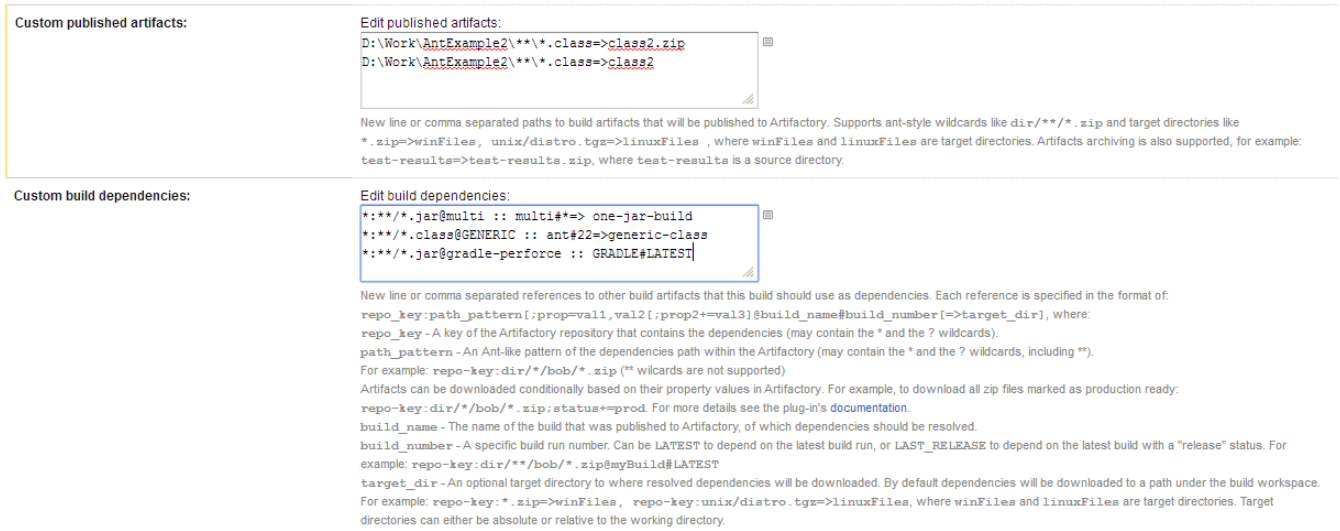
File Specs

File Spec are specified in JSON format. You can read the File Spec schema here.

Legacy Patterns (deprecated)

Legacy patterns are deprecated since version 1.8.0 and will be removed in future releases.

Custom published artifacts	Allows you to specify which artifact files produced by the build should be published to Artifactory. At the end of the build the plugin locates artifacts in the build's checkout directory according to the specified artifact patterns, and publishes them to Artifactory to one or more locations, optionally applyinga mapping for the target path of each deployed artifact. The pattern and mapping syntax for Published Artifacts is similar to the one used by TeamCity for Build Artifacts.
Custom build dependencies	Allows you specify dependency patterns for published artifacts that should be downloaded from Artifactory before the build is run. You can have detailed control over which artifacts are resolved and downloaded by using query-based resolution, adding to your artifact paths a query with the properties that the artifact should have before it can be downloaded. For further information read here about Resolution by Properties.



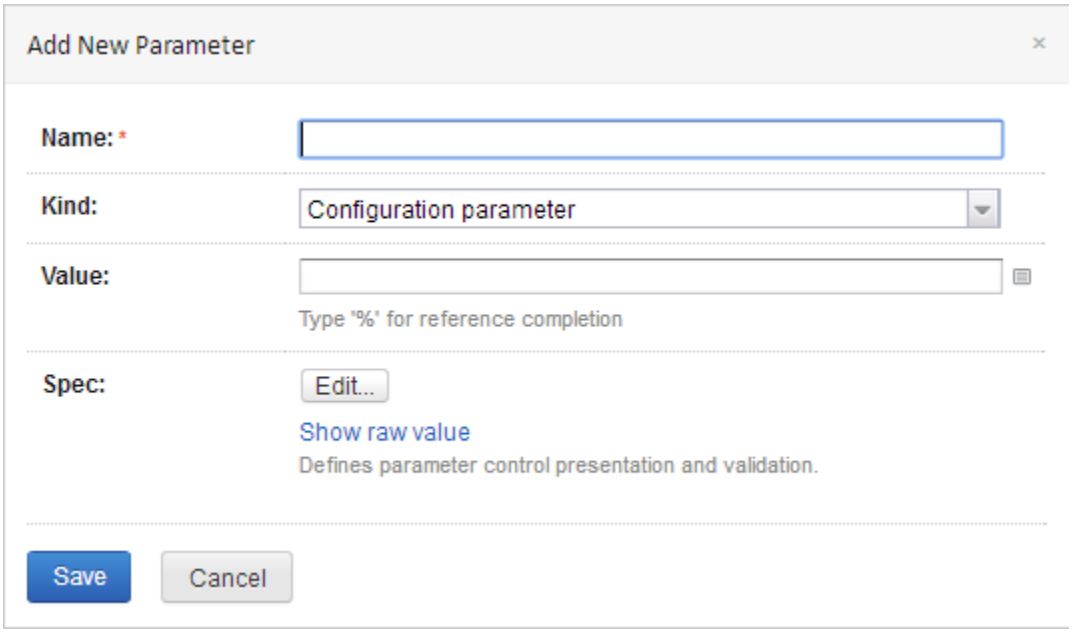
As of version 2.1.4, the above configuration is not backward compatible and you may need to re-save the builds configuration for them to run properly.

If no matching artifacts are found, remember that these parameters may be case sensitive depending on the operating system, the agent and the server they are running on.

Attaching Searchable Parameters to Build-Info and to Published Artifacts

In the **Build Configuration Settings** you can select **Parameters** to define system properties or environment variables that should be attached to artifacts and their corresponding build info.

To define a parameter click on the **Add new parameter** button.



FIll in the corresponding fields.

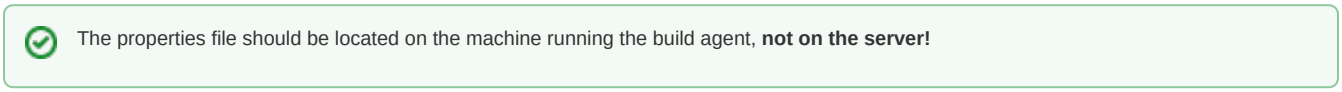
Parameters relevant for builds run through Artifactory are:

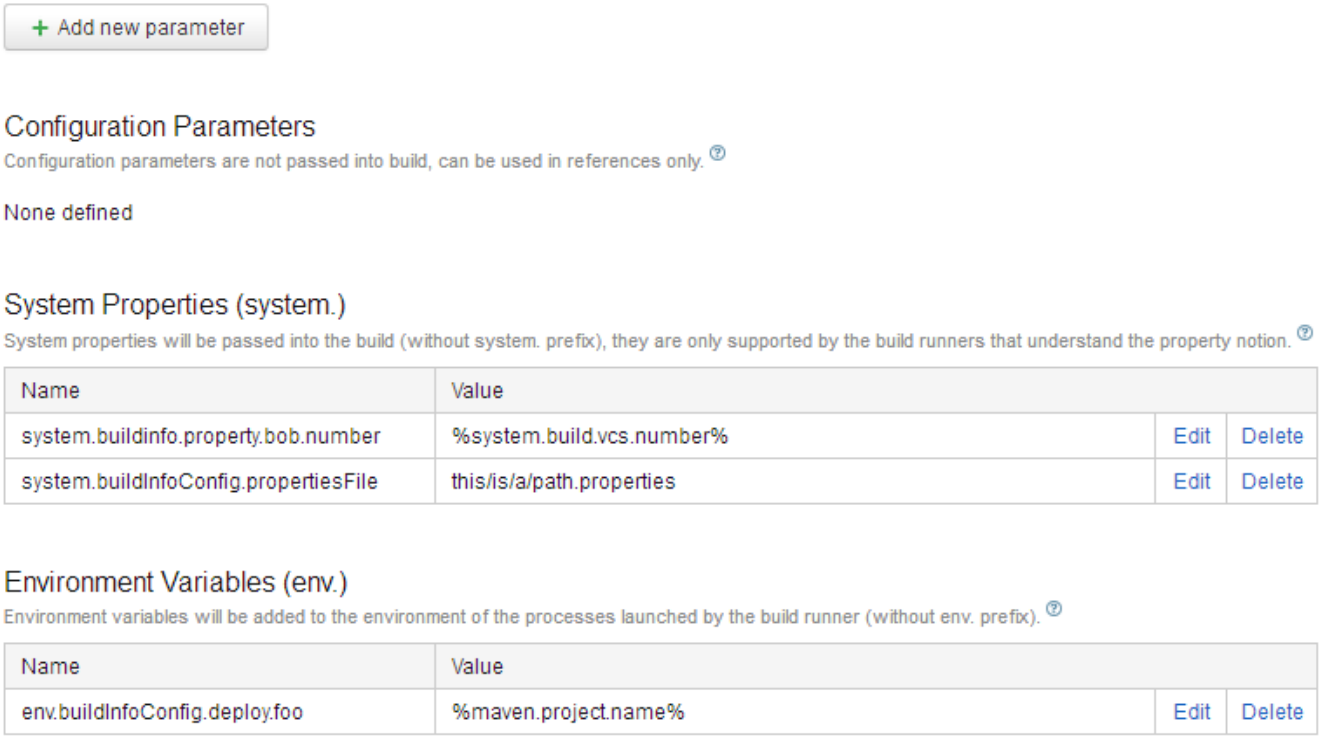
- `buildInfo.property.*` - All properties starting with this prefix are added to the root properties of the build-info
- `artifactory.deploy.*` - All properties starting with this prefix are attached to any deployed produced artifacts

You can specify all the properties in a single file, and then define another property pointing to it.

To point the plugin to a properties file, define a property called `buildInfoConfig.propertiesFile` and set its value to the absolute path of the properties file.

It is also possible to point the plugin to a properties file containing the aforementioned properties.

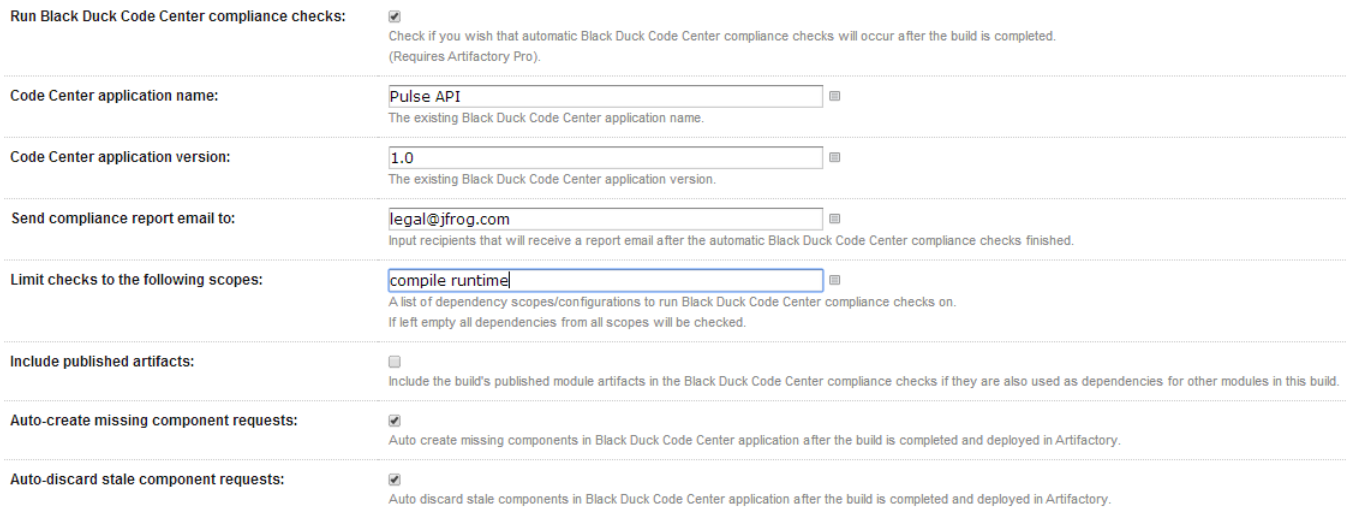




Black Duck Code Center Integration (deprecated)

This feature is no loner supported since version 5 of Artifactory.

If you are using Artifactory Pro and have an account with Black Duck Code Center, you can run the build through an automated, non-invasive, open source component approval process, and monitor for security vulnerabilities.



Viewing Project-specific Configuration

Existing project configuration can be viewed in **Settings** under **Projects | $PROJECT_NAME | $BUILD_NAME**:

Build Step: Maven edit »

Step 1:

Runner type: **Maven** (Runs Maven builds)

Execute: **If all previous steps finished successfully (zero exit code)**

POM file path: **maven-example/pom.xml**

Goals: **clean install**

Maven used: default

Additional Maven command line parameters: **none specified**

User settings provided by default

Maven metadata disabled: **false**

Use own local artifact repository: **false**

Build only modules affected by changes (incremental building): **false**

JDK home path: **not specified**

Build working directory: **not specified**

JVM command line parameters: **not specified**

Deploy artifacts to Artifactory: **enabled**

- Artifactory server: **http://10.0.0.235:8080/artifactory**
- Target repository: **libs-release-local**
- Target snapshot repository : **libs-snapshot-local**
- Deployer username: **admin**
- Deploy artifacts: **true**
- Deployment include patterns: **not specified**
- Deployment exclude patterns: **not specified**
- Publish Build Info: **true**
- Include Environment Variables: **true**
- Environment Variables Include Patterns: **not specified**
- Environment Variables Exclude Patterns: ***password*,*secret***
- Run license checks (Requires pro): **true**
- License violation notifications recipients: **none specified**
- Limit checks to the following scopes: **none specified**
- Include published artifacts: **false**
- Disable automatic license discovery: **false**
- Enable Artifactory release management: **true**
- VCS tags base URL/name: **none**
- Git release branch name prefix: **REL-BRANCH-**
- Alternative Maven goals: **none**
- Alternative Maven command line parameters: **none**
- Default module version configuration: **Global**

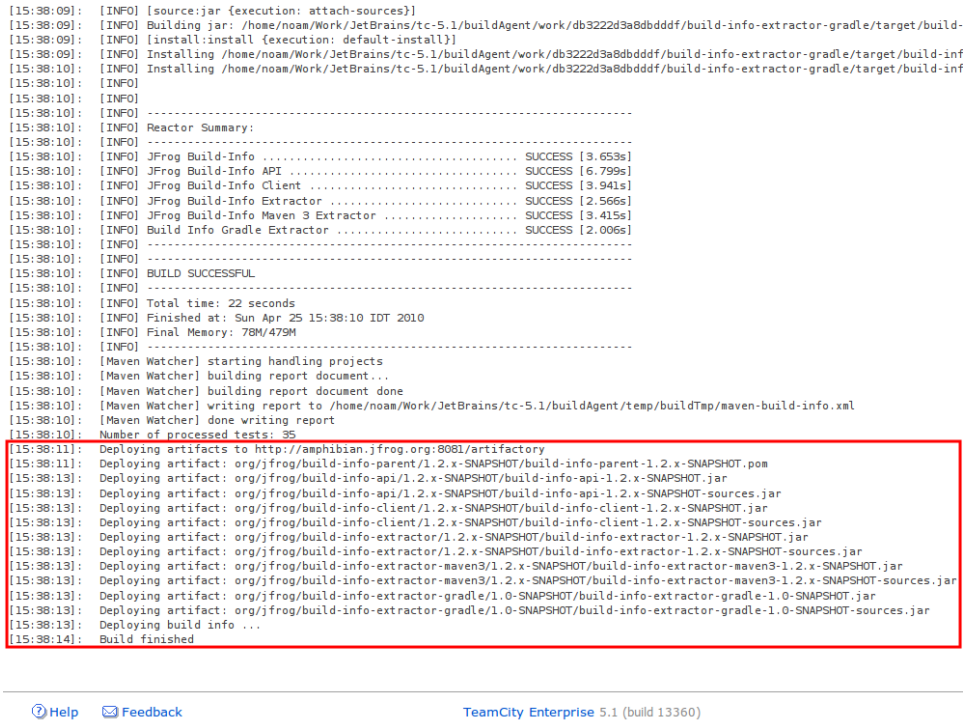
Java code coverage: **disabled**

Running a Build with the Artifactory Plugin

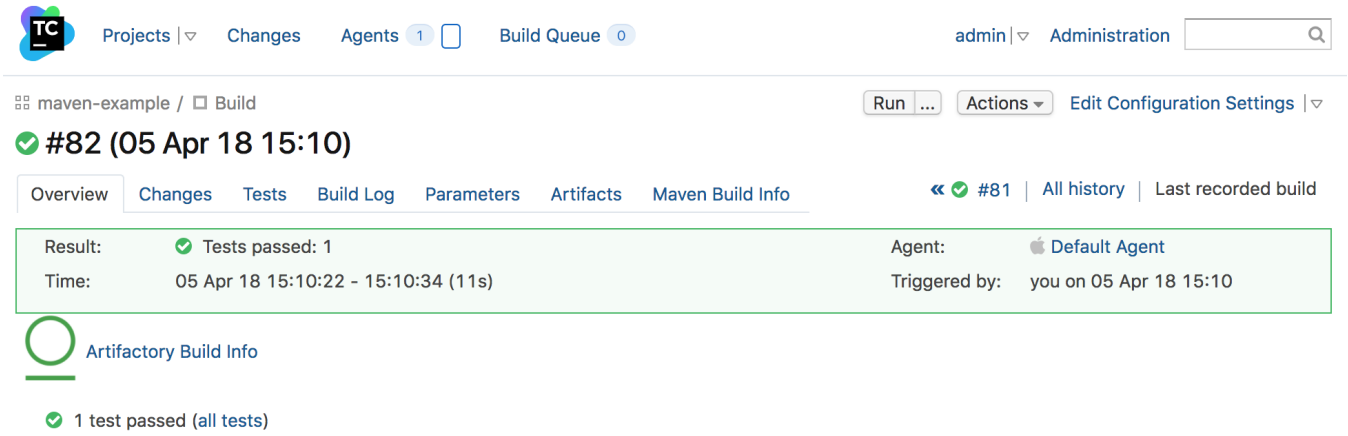
Once you have completed setting up a project runner you can run a project build. The Artifactory plugin takes effect at the end of the build and does the following:

1. For all build runner types - Publishes the specified Published Artifacts to the selected target repository and applies corresponding path mappings.
2. For Maven or Ivy/Ant build runner - Deploys all artifacts to the selected target repository together at the end of the build (as opposed to deploying separately at the end of each module build as done by Maven and Ivy).

3. Deploys the Artifactory BuildInfo to Artifactory, providing full traceability of the build in Artifactory, with links back to the build in TeamCity.



You can also link directly to the build information in Artifactory from a build run view:



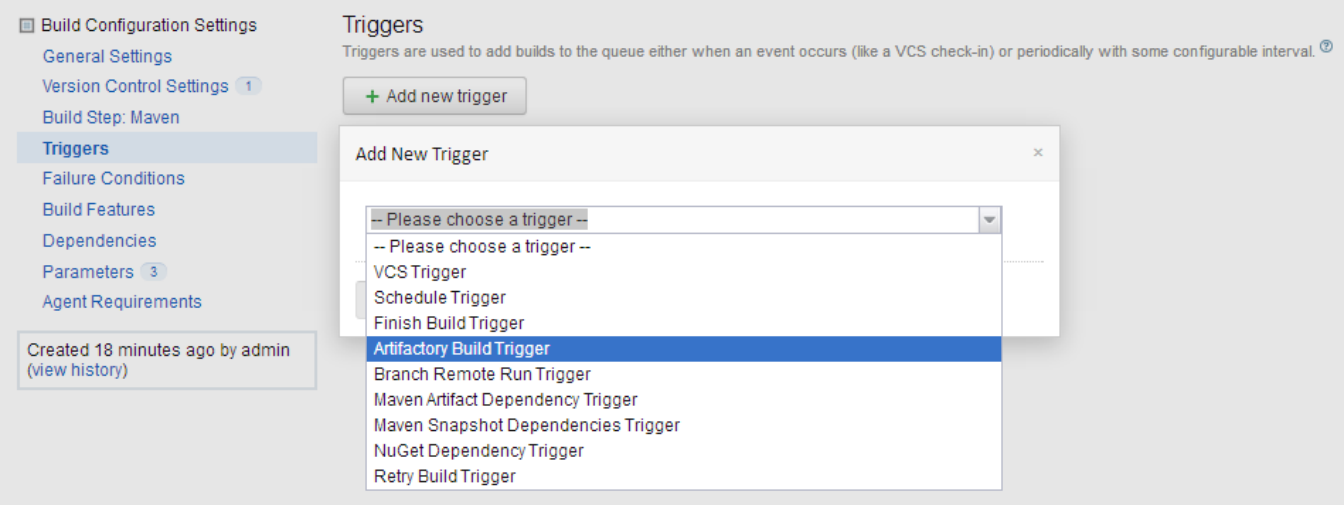
Triggering Builds in Reaction to Changes in Artifactory

The plugin allows you to set a new type of trigger that periodically polls a path in Artifactory, a folder or an individual file. Whenever a change is detected in the polled element, the TeamCity build is triggered. For example, the build could be triggered when new artifacts have been deployed to the specified path in Artifactory.

> **Artifactory Pro required**
>
> Triggering builds is only available with Artifactory Pro

To configure a new build trigger, under **Administration,** select **$PROJECT_NAME | $BUILD_NAME.** Then, select **Triggers.**

Click the **Add new trigger** button to select an **Artifactory Build Trigger**

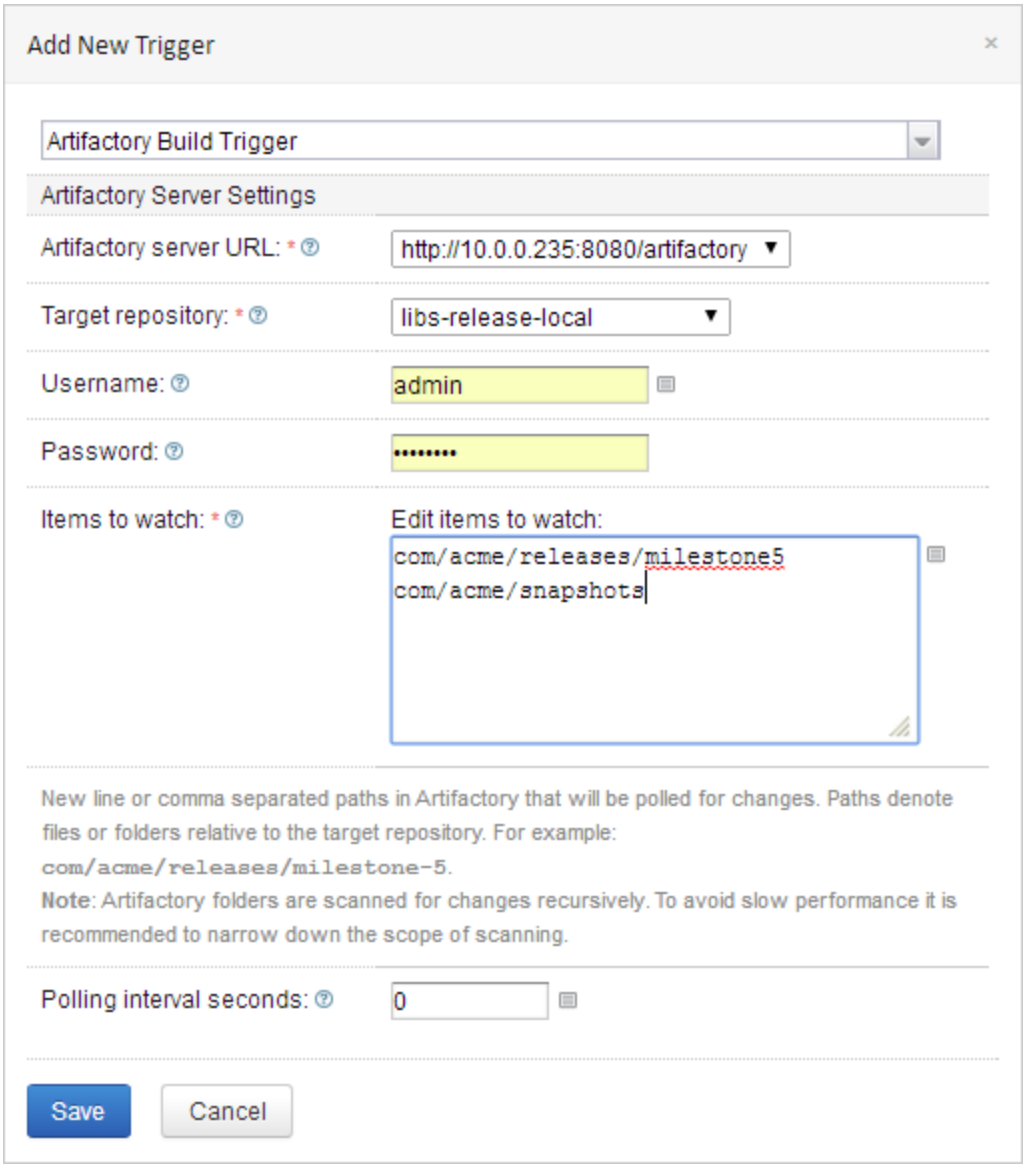


Select the **Artifactory Server URL** and the **Target repository.**

Complete the username and a password fields of a valid deployer for the selected repository.

> **Deploy permssion**
>
> The specified user must have deploy permissions on the repository

Then, in **Items to watch,** specify the paths in the selected repository in which a change should automatically trigger a build.



Be as specific as possible in Items to watch

In order to establish if there has been a change, Artifactory must traverse all the folders and their sub-folders specified in **Items to watch**. If the specified folders have a lot of content and sub-folders, this is a resource intensive operation that can take a long time.

Therefore, we recommend being as specific as possible when specifying folders in **Items to watch**.

Proxy Configuration

If the Artifactory server is accessed via a proxy, you need to configure the proxy by setting the following properties in the `$TEAMCITY_USER_HOME/.BuildServer/config/internal.properties` file. If the file does not exist, you'll need to create it.

```
org.jfrog.artifactory.proxy.host
org.jfrog.artifactory.proxy.port
org.jfrog.artifactory.proxy.username
org.jfrog.artifactory.proxy.password
```

Since version 2.5.0, you can also define a proxy for specific build agents. You do that by adding the TeamCity agent name to the end of the above property names.

For example, if you wish to configure a proxy for the "my-agent" agent, the proxy properties configuration should look as follows:

```
org.jfrog.artifactory.proxy.host.my-agent
org.jfrog.artifactory.proxy.port.my-agent
org.jfrog.artifactory.proxy.username.my-agent
org.jfrog.artifactory.proxy.password.my-agent
```

In case your build agent name contains a white-space, you should replace the white-space in the property name with \u0020.

For example, here's how you define the proxy host for the "Default Agent":

org.jfrog.artifactory.proxy.host.Default\u0020Agent

Licence

The TeamCity Artifactory plugin is available under the Apache v2 License.

Release Notes

2.8.0 (3 Jan 2019)

1. Support Teamcity 2018 (TCAP-322)
2. Allow downloading using file spec which contains "build" only ("pattern" or "aql" are not mandatory) (TCAP-326)
3. Bug fix (TCAP-323)

2.7.1 (20 August 2018)

1. Fix build-info link in TeamCity 2018 (TCAP-318)
2. Allow modifying the build name in Artifactory (TCAP-210)
3. Parallel uploads to Artifactory when using File Specs (TCAP-315)
4. Bug fix (TCAP-317)

2.6.0 (9 April 2018)

1. Files larger than 5mb are downloaded concurrently using range-requests (TCAP-312)
2. Bug fixes (TCAP-305, TCAP-306, TCAP-307, TCAP-309, TCAP-311, TCAP-314)

2.5.0 (28 Sep 2017)

1. Allow proxy configuration per agent (TCAP-237)
2. Support pattern exclusion in File Specs (TCAP-300)
3. File specs AQL optimizations (TCAP-302)
4. Bug fixes (TCAP-297, TCAP-299, TCAP-301, TCAP-303)

2.4.0 (29 Jun 2017)

1. Support for retention policy within TeamCity Plugin (TCAP-283)
2. Support Xray build scan (TCAP-292)
3. Add upload and download from file specs support to generic jobs (TCAP-294)
4. Allow to extract supported formats using file specs (TCAP-295)
5. Bug fixes (TCAP-167, TCAP-293, TCAP-296)

2.3.1 (23 Jan 2017)

1. Support full path in specs (TCAP-287)
2. Add to file spec the ability to download artifact by build name and number (TCAP-288)
3. Change file Specs pattern (TCAP-285)

2.3.0 (13 Nov 2016)

1. Upload and download File Specs support to generic jobs (TCAP-284)

2.2.1 (19 May 2016)

1. Bug fix (TCAP-214)

2.2.0 (21 March 2016)

1. Bug fixes (TCAP-238, TCAP-239, TCAP-241, TCAP-244, TCAP-245, TCAP-247, TCAP-250, TCAP-258, TCAP-236)

2.1.13 (4 May 2015)

1. Support multi Artifactory Build Triggers (TCAP-222)
2. Support SBT build tool (TCAP-223)
3. Bug fix (TCAP-214)

2.1.12 (12 Mar 2015)

1. Adding support for free text repository configuration (TCAP-217)

2.1.11 (7 Dec 2014)

1. Compatibility with Gradle 2.x (TCAP-211)
2. Bug Fixed (TCAP-205)

2.1.10 (8 May 2014)

1. Bug Fixed (TCAP-206, TCAP-72)

2.1.9 (17 Apr 2014)

1. Adding Version Control Url property to the Artifactory Build Info JSON. (TCAP-203)
2. Support for TeamCity 8.1 Release management feature issues
3. Support working with maven 3.1.1
4. Bug Fixed (TCAP-197, TCAP-161)

2.1.8 (15 Jan 2014)

1. Allow remote repository caches to be used for build triggering - TCAP-196
2. Bug Fixes

2.1.7 (18 Dec 2013)

1. Add support for blackduck integration - TCAP-185

2.1.6 (03 Sep 2013)

1. TeamCity 8.0.x full compatability issue - TCAP-172
2. Global and build credentials issue - TCAP-153
3. Repositories refreshed by credential issue - TCAP-166
4. Generic deploymentresolution on Xcode builds - TCAP-180
5. Working directory in Gradle build issue - TCAP-125

2.1.5 (07 Jul 2013)

1. Fix security issue - TCAP-172
2. Improve generic resolution - BI-152

2.1.4 (21 Aug 2012)

1. Compatible with TeamCity7.1.
2. Bug Fixes

2.1.3 (30 May 2012)

1. Compatible with TeamCity7.
2. Support 'Perforce' in release management.
3. Support multiple deploy targets for the same source pattern in generic deploy.
4. Support for custom build dependencies resolution per build.

2.1.2 (12 Dec 2011)

1. Compatible with Gradle 1.0-milestone-6.

2.1.1 (09 Aug 2011)

1. Support for Gradle milestone-4
2. Better support for releasing nested Maven projects
3. Fixed minor Maven deployments discrepancies

2.1.0 (14 Jul 2011)

1. Release management capabilities
2. Bug fixes

2.0.1 (9 Jan 2011)

1. Auto Snapshot/Release target repository selection
2. Add ivy/artifact deploy patterns
3. Improved Gradle support
4. Bug fixes

2.0.0 (5 Dec 2010)

1. Support for Gradle builds
2. Support for maven3 builds
3. Default deployer add resolver credentials
4. Support for muti steps builds

1.1.3 (21 Nov 2010)

1. Include/exclude pattern for artifacts deployment

1.1.2 (7 Nov 2010)

1. Control for including published artifacts when running license checks
2. Limiting license checks to scopes
3. Control for turning off license discovery when running license checks