



ENGINEERING SOFTWARE DEVELOPMENT MADE EASY

---

**Build Configurator plugin  
for Jenkins CI  
User Guide**

February, 2017

## Contents

1.	Welcome to Build Configurator .....	3
2.	Requirements and installation .....	4
2.1.	Build server installation .....	4
3.	Getting started .....	5
3.1.	Create new configuration command .....	5
3.2.	Action commands.....	8
3.3.	Edit command .....	9
3.4.	Delete command .....	9
3.5.	Copy command.....	9
4.	Administration.....	10
4.1.	Approve/Reject command .....	10
4.2.	Create job command.....	10
4.3.	Delete job command .....	11
4.4.	Update job command .....	11
4.5.	Plugin role privileges .....	11
5.	Examples & Specifications.....	13
5.1.	Build example.....	13
5.2.	Plugin “Builders.xml” file specification.....	14
5.3.	Plugin “Platforms.xml” file specification .....	14
5.4.	Plugin “ScriptType.xml” .....	15
5.5.	Buildserver “builders.xml” file specification.....	16
6.	Uninstallation .....	17
7.	About AMC Bridge .....	18

## 1. Welcome to Build Configurator

Build Configurator plugin extends the functionality of Jenkins CI and provides a simplified way of creating and managing various project's configurations and settings as well as generates necessary builds, artifacts etc.

This plugin features the following functionality:

- create and manage various build configurations;
- produce individual pre-configured job for every build;
- use several projects in one build as well as several build settings for each project;
- set pre/post build scripts (like comments for Jenkins administrator), e-mail notifications and produced artifacts;
- supports most of popular builder templates (such as .NET, Maven & Ant, XCode);
- view build's configuration's additional information (such as state, update date and other).

## 2. Requirements and installation

In order to use Build Configurator plugin, please install Jenkins CI at first. This plugin works efficiently starting from Jenkins version of 2.13. Build Configurator plugin can be installed either from [Jenkins CI plugin repository](#) or manually by uploading “build-configurator.hpi” file from Jenkins plugin manager advanced tab.

This plugin depends on following plugins:

- Mailer plugin;
- Description setter plugin;
- Conditional buildstep plugin;
- Copy to slave plugin
- Multiple SCMs plugin
- Workspace cleanup plugin (in case you need to use it)
- Subversion plugin
- Git plugin

### 2.1. Buildserver installation

Buildserver is a java application that actually produces artifacts by compiling source code, interpreting it or whatever it's needed to be done. Buildserver **is not** installed automatically with the Jenkins plugin and each Jenkins node (including leading machine, if needed) should have buildserver rolled up manually.

To setup, follow these steps:

1. Download buildserver.jar.zip archive from github.
2. Unpack it to the chosen folder. There should be 2 files: buildserver.jar and builders.xml (see section 5.5.).
3. Set up BUILDER\_PATH environment variable to folder with “*buildserver.jar*”.

Buildserver parses “*builders.xml*” file and launches corresponding builder with specified parameters. It produces build process information to STDOUT and STDERR streams which in turn are displayed at Jenkins console log.

### 3. Getting started

After successful plugin installation a new Jenkins menu item should appear (fig. 1). If the new menu item hasn't been added, try to restart Jenkins and log in; also you can describe the problem at plugin's comment section at Jenkins plugins repository and we will try to solve it together.

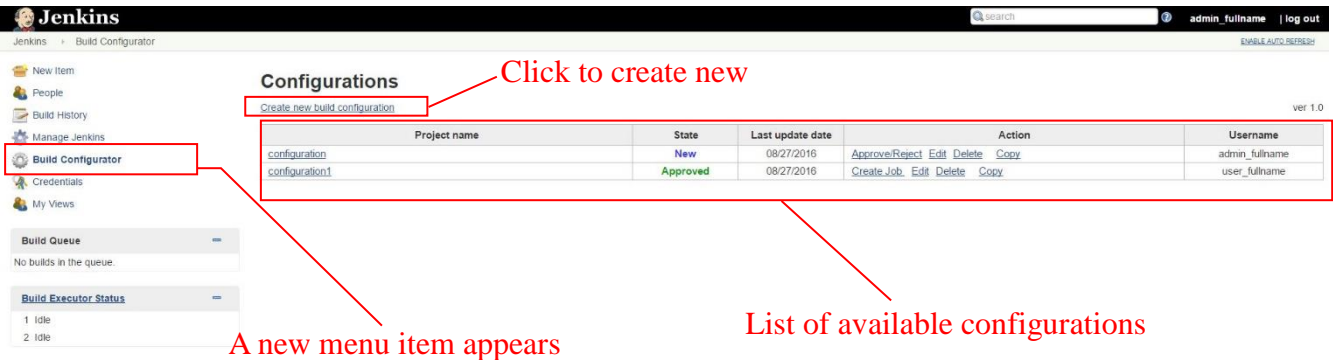


Fig. 1. "Build configurator" menu

Click "Build configurator" menu item with a list of currently available configurations and their status will be loaded.

#### 3.1. Create new configuration command

To create new configuration, click the link at the top of the page (see fig. 1). After specifying all needed parameters and settings for the new build, click "Save" (fig. 2).

Jenkins > Build Configurator

**Create new build configuration**

Project Name:

SCM:

☐ Job E-mails notifications

Configuration E-mails notifications:

Regular expression to parse build version:

Use workspace cleanup plugin: ☒

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle  
2 Idle

Project url:

Credentials:

Branch name:

Local module directory (optional):

Project to build:

Path to artifacts:

☐ Comments

Users with access to configuration:

Add user:

user1 | X user2 | X user3 | X

Fig. 2. Create new build configuration page view

Below is the detailed description of each field.

- **Project name** – required field, represents build name;
- **SCM** – drop-down list with supported source code management systems;
- **Build machine configuration** – list of available nodes (slaves);
- **Job e-mails notification** – a whitespace-separated list of recipient addresses; when not empty it will be used to send notifications when job makes build;
- **Configuration E-mails notifications** – a whitespace-separated list of recipient addresses;

when not empty it will be used to send notifications when configuration state changes;

- **Regular expression to parse build version** – this is description setter plugin regular expression field. If configured, the regular expression will be applied to each line in the build log. A build description will be set based on the first match;
- **Use workspace cleanup plugin** – this plugin deletes the workspace before the build. In case you want to use it, you should check this checkbox. It will enable plugin and add configuration file to ignore list;
- **Project URL** – a valid URL-address of project's repository path;
- **Credentials** - drop-down list with available user credentials;
- **Set as default button** (for admins only) – set selected credentials as default for all users;
- **Branch name** – repository branch to build;
- **Local module directory** – specifies relative to the Jenkins workspace root path folder structure where repository content will be downloaded; if left empty, the last repository's path entry will be taken as a folder name; if marked as dot ("."), the content will be downloaded right into workspace;
- **Project to build** – name of the project's main file; some project types (such as Maven or Ant) do not require main project;
- **Path to artifacts** – a list of [valid ANT-fileset pattern format](#) of subdirectory path structures to create artifacts (relatively to project root directory);
- **Version files** – enable and add version file path to list if you want Description setter plugin to set build description based on version from your file.
- **Add builder** button allows to add multiple build settings for every project. It will add a new group of fields which are described below.
- **Builder** – drop-down list of builders;
- **Platform** – target build platform; plugin doesn't check each builder's available platform support;
- **Configuration** – describes target configuration; *"Release"* and *"Debug"* configurations are used by .NET projects; when checking *"Other"* configuration checkmark, specify at the field below some specific build parameters.
- **Command line arguments** (for administrator only) – this field value will be added to the end of the builder command line. Set additional parameters for the selected builder if necessary.
- **Script type** - drop-down list of supported scripts.
- **Pre/post build scripts** – scripts that will be executed before/after job build. Actually they're just text blocks for administrator.

- **Comments** – enable and fill in to provide some additional information for administrator.
- **Users with access to configuration** – defines users which have full access to this configuration. User can be added by username
- **Add project to build** - add another projects and specify custom build settings for them. It will add a new group of fields which are described below.

If some group of fields has been added by mistake, press “X” sign at the top-right corner which will close it and erase all data that has been filled there.

To save the configuration, press ‘Save’ button at the bottom of the page or “Cancel’ to discard changes.

### 3.2. Action commands

“Action” column contains all available commands that can be applied to particular configuration. The only available for regular user actions are: Edit, Delete, Copy. “New” configuration can be approved or rejected, edited or deleted by admin with corresponding action command (fig. 3).

#### Configurations

[Create new build configuration](#)

Project name	State	Last update date	Action	Username
My Test Project	New	04/28/2015	Approve/Reject Edit Delete	admin_admin

Fig. 3. Create new build configuration page view

Below is the table of possible commands for each configuration state (table 1).

**Table 1. Possible configuration states and action commands**

State	Action
New	Approve/Reject, Edit, Delete, Copy
Updated	Approve/Reject, [Delete Job], Edit, Delete, Copy
Approved	Create job, [Update Job, Delete Job], Edit, Delete, Copy
Rejected	Edit, Delete, [Delete Job], Copy
For Deletion	Delete permanently, Restore, [Delete Job], Copy

Commands in square brackets only appear when there is job generated from configuration by clicking “Create job” reference.



### 3.3. Edit command

*“Edit”* command will open a new page with current settings that can be changed.

Press *“Save”* to save changes or press *“Cancel”* to discard them. If changes have been changed, configuration state will be changed to *“Updated”*.

### 3.4. Delete command

Click on *“Delete”* action command on plugin main page to delete some configuration.

Configuration’s state will be changed to *“For Deletion”*. Then only admin can either revert to previous state (*“Restore”* action command) or delete it permanently by pressing *“Delete permanently”*.

### 3.5. Copy command

Any available configuration can be copied. Click *“Copy”* link and fill new configuration name field. Copied configuration will appear in configuration list with *“New”* state.

## 4. Administration

### 4.1. Approve/Reject command

After creating or modifying some configuration approve or reject it by clicking corresponding action command.

Approved configuration state is changed to *“Approved”*. After configuration becomes *“Approved”*, corresponding job can be created or updated with particular action commands.

After pressing *“Reject”* button a dialog window appears where the reason of rejection should be specified (fig. 4).

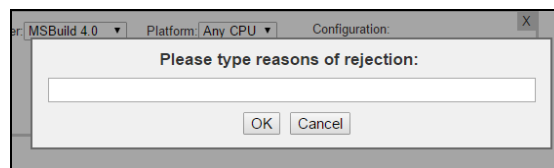


Fig. 4. Rejection dialog window

After that configuration's state changes to *“Rejected”*, rejection message will be written at the configuration's page header (fig. 5).



Fig. 5. Rejection message when making changes into configuration

### 4.2. Create job command

*“Create Job”* action command is available only for approved configurations.

By default, job will take following settings:

- *“Log Rotation”* strategy with *“Days to keep builds”* value of 300
- *“Max # of builds to keep”* as 30.

Also it takes SCM tool and fills it with corresponding values from configuration settings.

“*Build triggers*” value is set up as “Poll SCM” with schedule cron value as “H \* \* \* \*” (hourly).

Apart of that plugin uses “*conditional-buildstep*” plugin features: it creates 2 triggers which check OS name and execute corresponding build command.

For Windows systems it calls following command:

```
java -jar "%BUILDER_PATH%\buildserver.jar" -nodeName "%NODE_NAME%" -jobName  
"%JOB_NAME%" -workspace "%WORKSPACE%" -jenkinsHome "%JENKINS_HOME%"
```

For Unix-like systems it calls next shell command:

```
java -jar "$BUILDER_PATH/buildserver.jar" -nodeName "$NODE_NAME" -jobName "$JOB_NAME" -  
workspace "$WORKSPACE" -jenkinsHome $JENKINS_HOME
```

It starts *buildserver.jar* Java application at specified by BUILDER\_PATH environment variable location.

### 4.3. Delete job command

To delete job, click “*Delete job*” action command reference. Job configuration persists after deleting.

### 4.4. Update job command

When there are some changes in configuration and corresponding job already exists, “*Update job*” action command appears and changes can be reflected into the job.

### 4.5. Plugin role privileges

Build Configurator plugin uses role privileges specified by Jenkins global security settings. Administrative rights allow user to manage configurations with full access. If there are some restrictions in account settings (i.e. set by “*Matrix-based security*” strategy) access to concrete

commands is provided only.

For example, user with no administrative rights is able only to create configuration and mark it for editing or deletion. Then users with administrative privileges will make final decision about whether or not to adopt proposed changes.

Also non-privileged user can't create jobs from configurations as well as see configurations of other users.

Users with administrative rights can view all configurations, create jobs and decide whether or not delete or approve/reject changes, though they can't make proposals for editing or deletion of some configuration (unlike its configuration owner).

## 5. Examples & Specifications

### 5.1. Build example

User should set up BUILDER\_PATH environment variable to folder with “*buildserver.jar*” and “*builders.xml*” files on his nodes (slaves).

“*buildserver.jar*” file parses “*builders.xml*” file and launches corresponding builder with specified parameters. It produces build process information to STDOUT and STDERR streams which in turn are displayed at Jenkins console log.

If for some reason an error occurs at some build step of any project within same configuration the whole build is marked as “*Failure*”; otherwise it is defined as “*Success*”.

Example of “*Success*” Jenkins console log:

```
D:\jenkins_folder\workspace\1>java -jar "D:\buildserver_folder\buildserver.jar" -nodeName "master"
-jobName "1" -workspace "D:\jenkins_folder\workspace\1" -jenkinsHome "D:\jenkins_folder"
Mar 15, 2016 10:50:55 AM com.amcbridge.buildserver.server.Main main
```

INFO:

STEP 1: START BUILD SERVER

```
Mar 15, 2016 10:50:55 AM com.amcbridge.buildserver.server.Main main
```

INFO:

STEP 2: INITIALIZE BUILD SERVER

```
Mar 15, 2016 10:50:55 AM com.amcbridge.buildserver.server.Main main
```

INFO:

STEP 3: EXECUTE BUILD SERVER

COMMAND:

cmd /c mvn clean package

...maven log goes here...

RETURN CODE: 0

## 5.2. Plugin “Builders.xml” file specification

File “*Builders.xml*” located in “JENKINS\_HOME\plugins\build-configurator\config” on Jenkins master, contains list of available *builders* that will be shown in BAMT view when creating new configuration (fig. 6). The root tag `<builders>` has list of its child tags `<builder>`, that contains attribute “value” with builder’s name. To add/change builder, check builder with the same name in “builders.xml” file in buildserver directory to be added/changed.

Example of “Builders.xml” file structure:

```
<builders>
  <builder key="VS_2013" value="VS 2013"/>
  ...
</builders>
```



Builder: VS 2013 Platform: x86

Configuration: ☒ Release ☐ Debug ☐ Other

Command line args:

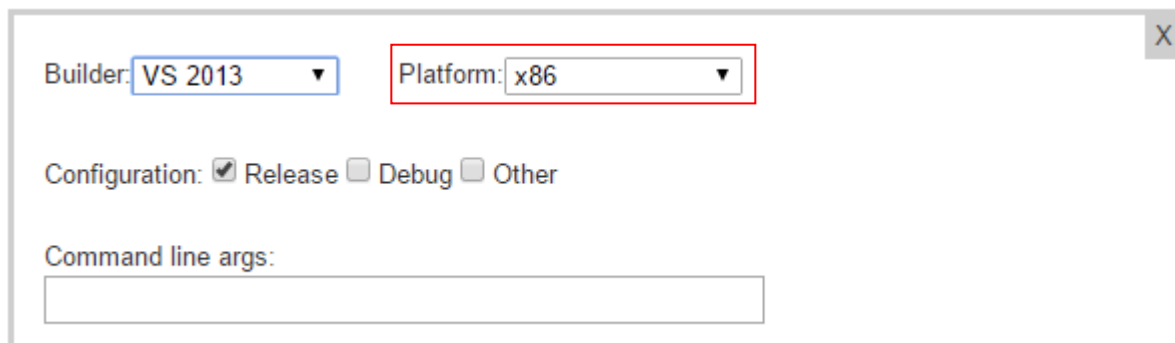
Fig. 6. Builder view

## 5.3. Plugin “Platforms.xml” file specification

File “*Platforms.xml*” is located in “JENKINS\_HOME\plugins\build-configurator\config” on Jenkins master. There is a list of platforms for builders (fig. 7). While changing this list, check if buildserver’s file “builders.xml” is changed, where builder’s platform can be added/changed.

Example of “Platforms.xml” file structure:

```
<platforms>
  <platform>
    <platformName>x86</platformName>
  </platform>
  ...
</platforms>
```

A screenshot of a 'Builder view' window. It contains two dropdown menus at the top: 'Builder:' with 'VS 2013' selected and 'Platform:' with 'x86' selected. Below these are three radio buttons for 'Configuration': 'Release' (checked), 'Debug', and 'Other'. At the bottom is a text input field labeled 'Command line args:'.

Builder: VS 2013 Platform: x86

Configuration: ☒ Release ☐ Debug ☐ Other

Command line args:

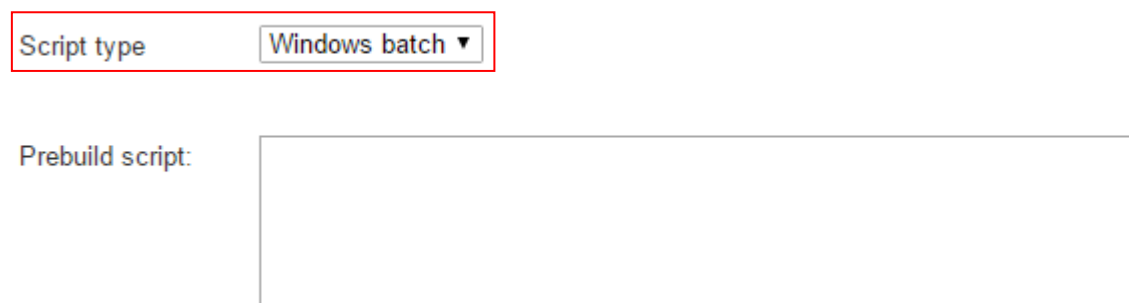
Fig. 7. Builder view

## 5.4. Plugin “ScriptType.xml”

File “*ScriptType.xml*” is located in “JENKINS\_HOME\plugins\build-configurator\config”. It contains a list of supported scripts (fig. 8), there are already “windows batch”, “unix shell” and “python” script types. Add own script types according to the existing list.

Example of “ScriptType.xml” file structure:

```
<scriptTypes>
  <scriptType>
    <scriptTypeName>Windows batch</scriptTypeName>
  </scriptType>
  ...
</scriptTypes>
```

A screenshot of a form for selecting a script type. It has a label 'Script type' and a dropdown menu with 'Windows batch' selected. Below this is a label 'Prebuild script:' followed by a large empty text area.

Script type Windows batch

Prebuild script:

Fig. 8. Pre/post script type

## 5.5. Buildserver “builders.xml” file specification

File “*builders.xml*” consists of configurations of available *builders* that will execute particular project builds (in most cases they’re abstraction on top of compilers). The root tag `<builders>` has list of its child tags `<builder>`, each of which describes information about particular builder's configuration. Below attributes of each `<builder>` tag are described:

1. *name* - represents name of the builder; it should be exactly the same name which is available as an option on plugin's configuration creation page;
2. *architecture* - it's an obsolete value (because 64-bit builders can still build 32-bit applications when particular command option is specified);
3. *executeBuild* - represents absolute path to builder's location;
4. *commandLine* - describes command pattern and parameters to launch builder; consists of next parameters:

-*executeBuild* - substitutes by aforementioned path to concrete builder application;

-*solution* - substitutes by retrieved from config.xml value; determines solution file name (e.g. \*.sln, \*.csproj etc.);

-*config* - substitutes by retrieved from config.xml value; determines target configuration (e.g. 'Debug', 'Release');

-*architecture* - substitutes by retrieved from config.xml value; determines target project architecture (e.g. 'Win32', 'x86\_64', 'Any CPU' etc.)

Remarks:

-square brackets determine 'Other' configuration where custom attributes for particular builder command can be specified; as a result, inner content is replaced by value retrieved from config.xml; only one pair of brackets is allowed;

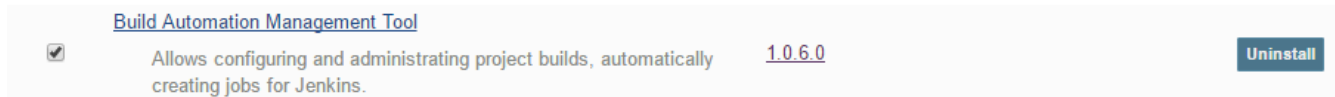
-&quot; - denotes XML-friendly quotes (simple "quote" sign will cause buildserver's failure at parsing XML file).



## 6. Uninstallation

To uninstall Build Configurator plugin run next steps on its nodes (slaves):

1. Go to plugin manager in Jenkins account and find "Build Automation Management Tool" in the list of plugins
2. Click Uninstall button.



3. Remove BUILDER\_PATH environment variable and folder with "buildserver.jar" and "builders.xml" files on each Jenkins node and leading Jenkins machine.

## About AMC Bridge

AMC Bridge is a vendor of choice for software development services in the areas of computer aided design, engineering, manufacturing and construction. Since 1999 we have been delivering solutions for CAD, CAE, CAM, PDM, BIM and PLM applications. For over 15 years we have participated in the development of commercial software products and custom solutions for the engineering markets based on the variety of platforms from desktop and web to mobile and clouds.

AMC Bridge helps to improve engineering process overhead by the development of 3D and 2D modeling software products, data, document and community management technologies, CAD data interoperability, and many other aspects of software development for the engineering markets.

Feel free to use wide experience of AMC Bridge team to find out all features and intricacies of software development process. Contact us any time and we will do our best to turn your ideas into reality.

303 Wyman Street, Suite 300 Waltham, MA 02451, USA

+1 866-575-4791

[www.amcbridge.com](http://www.amcbridge.com)

**For all online inquiries, please contact:** [contact@amcbridge.com](mailto:contact@amcbridge.com)

**Technical Support:** [support@amcbridge.com](mailto:support@amcbridge.com)