Publish Over

Goal
The goal of the Publish Over plugins is to provide a consistent set of features and behaviours when sending build artifacts ... somewhere.

Publish Over plugins

- Publish Over CIFS Plugin - send artifacts to a windows share
- Publish Over FTP Plugin - send artifacts to an FTP server
- Publish Over SSH Plugin - send artifacts to an SSH server (using SFTP) and/or execute commands over SSH

Overview

Host configurations are created in the main Jenkins configuration (Manage Jenkins > Configure System).
The host configurations define how to initiate connections to the remote server.

The #publisher is configured in a job by selecting the checkbox next to Publish Over ... (eg. Publish over FTP)
(The publisher is also available as an action during a promotion if you have installed the Promoted Builds Plugin

One or more servers are selected and configured as destinations for transfer sets.

One or more transfer sets are configured to select the files to send, and where to send them.
Configuration

Host configuration

Create one or more configurations that can be selected from the job configuration page. To change the order that the configurations appear in the drop down on the job configuration page, drag the square icon (next to FTP/SSH server) to reorder the list.

As the host configuration specifies how to connect to the remote server, the configurations can be quite specific to the plugin (ie. SSH keys, passive mode FTP, etc), but some common options are listed below.

When first configuring or changing a configuration, always click the "Test Configuration" to ensure that the configuration will work when it is used from a Job.

If all is well, then you should see "SUCCESS", otherwise you will see a message in red which should hopefully give some information to help fixing the configuration.
Name

Give the configuration a name, this is the name that appears in the drop down box on the job configuration page. Having a name for the configuration allows multiple configurations to the same host (to login with different users, or set different remote directories).

Hostname

The hostname or IP address of the server. The hostname must be resolvable by any of the machines that may initiate a connection to the remote server.

Credentials

Not a field in itself, but the connection will need to know how to authenticate to the remote server.
The Publish Over FTP Plugin uses Username/Password.
The Publish Over SSH Plugin can use Username/Password, or SSH keys to authenticate when logging in as Username.

- **Require credentials to access the server within a job**
  
  The credentials can now be overridden when configuring the publisher within a job. This means that you can now specify all of the properties of the Host Configuration, but leave the credentials blank - ensuring that the server can only be used if the person configuring the job can supply the required credentials. Test the Host Configuration with some known good credentials to ensure that it is configured correctly, then remove the username/password and test the connection again to ensure that it cannot be used without first supplying good credentials.

- **Encrypted passwords**
  
  From version 0.4 (Publish Over FTP Plugin 0.4, Publish Over SSH Plugin 0.5) the password and passphrase value is encrypted in the configuration files and also in the Jenkins UI. Make sure that you backup $JENKINS_HOME/secret.key

Remote directory

A directory on the remote server that will serve as the effective root directory for this configuration.
If specified, the remote directory must exist, it will not be created.
If not absolute, then the directory will be relative to the directory which the user is in when they login using the credentials supplied.
If the directory is not supplied, then the effective root directory will be the directory which the user is in when they login using the credentials supplied.
A job that uses this configuration will not be able to put files on the server outside this directory (the SSH Exec command is free to do anything that the configured user is free to do)

Advanced options

Clicking the "Advanced" button will reveal the following options.

Port

The port that the Server is running on can be changed if the server that you are connecting to is not running on the default port for the service that the plugin talks to.

Timeout (ms)

The connect timeout can be configured (in milliseconds)

Publisher

There are some options that are set at the publisher level - options that affect all of the server connections within.
All of the publisher options are only exposed when you click on the "Advanced" button (at the bottom of the configuration for this plugin).

Advanced options

Publish to other servers if an error occurs

By default, when an error occurs during publishing the publisher will fail and return immediately.
Set this option to cause the publisher to try to publish to other servers after publishing to a previous configuration has failed.

Fail the build if an error occurs

By default, when an error occurs, the publisher will set the build result to UNSTABLE.
Setting this option will cause a failure in the publisher to set the build result to FAILED.
This option is especially useful in the case of a promotion where the main action is to Publish Over ...

Always send from master
By default, the publisher will connect from the host that has the files that need to be transferred - if a build is performed on a slave, then the transfer would be initiated from the slave. By selecting this option, the connection will be initiated from the Jenkins master. May be useful for those with exciting network configurations / administrators.

**Give the master a NODE_NAME**

![Legacy option](image)

In Jenkins 1.414, the Jenkins master will be assigned a NODE_NAME (‘master’).
For the reason above, this option will not appear when this plugin is installed on Jenkins 1.414 and later. Additionally, the option now defaults to ‘master’ when this plugin is installed on a Jenkins older than 1.414.

Sets NODE_NAME to the value specified if the environment variables contain a variable NODE_NAME that does not have a value.
If you are running on multiple nodes and you have executors on the master Jenkins and the build may occur on the master, then if you set this option, the NODE_NAME for the master will be set to the configured value.
This may be useful if you want to use $NODE_NAME in a Transfer Sets Remote directory eg. builds/$BUILD_NUMBER/$NODE_NAME.

**Parameterized publishing**

Publish to servers by matching labels against a regular expression provided by a parameter or an environment variable.
For each server the label will be matched against the expression, and if matched, the publish will continue, otherwise that server will be skipped.
If a label is not set for a server it will default to the empty string.
Configured labels have whitespace removed from the start and end of them, which means that an all whitespace label will be evaluated as an empty string.
The same label can be used multiple times, e.g. UAT for a database server and again for the web server.
The regular expression syntax is the [Java syntax](https://docs.oracle.com/javase/8/docs/api/). The labels can use the standard Jenkins environment variables e.g. $NODE_NAME, or build variables such as a matrix axis.

**Parameter name**
The name of the parameter or environment variable that will contain the expression for matching the labels.

**Server**

One or more servers need to be configured to tell the publisher where to send the files.
To add another server, click the “Add Server” button.
The order in which the servers are used during a build can be changed by left clicking on the small square icon above Name (next to SSH/ FTP Server) and dragging to a new location.

**Name**
The name of the Server (Host configuration) to use when connecting.

**Advanced options**

Clicking the “Advanced” button directly beneath the Name drop down will reveal the following options

**Verbose output in console**

Select this option to print lots of detail in the Jenkins console when the publisher is run.
This may be useful for diagnosing problems with publishing, such as authorization issues on the remote server, but in general should probably be left off as it will fill the console with a lot of unnecessary detail.

**Credentials**

Check to set the credentials to use to connect to the server.
This option enables the credentials set in the Host Configuration to be overridden in the job configuration page.
This option gives the server administrator the ability to create incomplete Host Configurations defining everything bar the credentials (i.e. leaving the Username blank) which means that configuring the server for use in a job will require the credentials to be set.
When enabling this option, you will expose a Test Connection button that can be used to ensure that you have entered the credentials correctly.

**Retry**

Check to enable the plugin to make more than one attempt to publish the artifacts.
Any files that were successfully transferred will not be transferred again.

**Retries**
The number of times to try to publish after the initial publish fails.

Delay

The number of milliseconds to wait before attempting to publish again.

Label

Set a label to be used with Parameterized publishing

Label

The label for this server. This label can also contain the Jenkins environment variables such as $NODE_NAME and build variables such as a matrix axis.

Transfer set

Each server will have one or more transfer sets to specify which files to send where.

Click the "Add Transfer Set" button to add more sets.

The order in which the transfers are performed during a build, left click the small square icon next to Transfer Set and drag to a new location.

Source files

This is an ant include pattern see Patterns in the Apache Ant Manual. Multiple includes can be specified by separating each pattern with a comma.

See Examples below.

Remove prefix

When transferring files, unless "Flatten files" has been selected, then the entire directory structure will be transferred from the base directory used for the Source files (usually the Workspace).

This option allows the removal of the higher parts of the directory structure (nearest the base directory).

This option is will be matched and removed from the front of the file path - whilst Jenkins environment variables will be substituted, it will not be expanded with shell like glob syntax or ant style patterns.

Remote directory

If specified, the files will be transferred below this directory (which is relative to the Remote Directory specified in the Host Configuration for this server).

If the directory does not exist it will be created.

See Examples below.

Advanced options

Clicking the "Advanced" button will reveal the following options.

Exclude files

This is an ant exclude pattern see Patterns in the Apache Ant Manual. Multiple excludes can be specified by separating each pattern with a comma.

E.g. .git/,doc/**/*.log

Pattern separator

The regular expression that is used to separate the Source files and Exclude files patterns.

The Source files and Exclude files both accept multiple patterns that by default are split using [; ]+ (any number of consecutive commas or spaces) which is how Ant, by default, handles multiple patterns in a single string.

The above expression makes it difficult to reference files or directories that contain spaces. This option allows the expression to be set to something that will preserve the spaces in a pattern e.g. a single comma.
No default excludes

There is a default set of patterns for Exclude files (e.g. `**/.git/**`) - check this option to disable them. Expand the help for this option to see the complete list of default exclude patterns.

Make empty dirs

The default behaviour of this plugin is to match files, and then create any directories required to preserve the paths to the files. Selecting this option will create any directories that match the Source files pattern, even if empty.

Flatten files

Only create files on the server, don't create directories (except for the Remote directory, if present).

- **Flatten files**
  
  All files that have been selected to transfer must have unique filenames. The publisher will stop and fail as soon as a duplicate filename is found when using the flatten option.

See Examples below.

Remote directory is a date format

This will format the build date with the string configured for Remote directory. For details on the date format see the javadoc for SimpleDateFormat. The Remote directory will first have any Jenkins environment variables expanded, and will then be used to format the build date.

- **Remote directory is a date format**
  
  All characters in the range `[a-zA-Z]` are reserved by SimpleDateFormat, so any that should not be replaced need to be quoted.

See Examples below.

Promotions

When the publisher is executed during a promotion, then:

- The Source files are rooted in the artifacts directory. (see below to change)
  Any artifact that you want to transfer during a promotion should be "archived" in the post-build actions of the build. This will make it available during a promotion. This means that if you have builds 1, 2 and 3 and you choose to "promote" build #2, then the artifacts for build 2 will be available to the publisher.
- The original builds environment variables are restored. This means that $BUILD_NUMBER will be replaced by the build number of the build under promotion. The environment variables for the promotion itself are available with the prefix `promotion_`, so to get the build number for the promotion, use the variable `promotion_BUILD_NUMBER`.
- If the "Remote directory is a date format" option is selected, then the date used is the date of the original build, not the date of the promotion. (see below to change)

When configuring a publisher for a promotion, the following extra configuration options are available at the "Server" level (options will appear below the "Verbose in console" option).

Use the workspace

Use the workspace as the base directory for the Source files. This option can be used if files are generated in the workspace during a promotion which then need to be transferred.

Use promotion timestamp

This option will only have any effect if the "Remote directory is a date format" option is selected. When selected, this option will use the time of the promotion when formatting the Remote directory, instead of using the time of the original build that is currently undergoing promotion.

Examples

To help illustrate the examples, a contrived workspace layout is presented below:
build.xml
src/my/code/HelloWorld.java
src/my/code/HelloWorldImpl.java
src/my/code/Main.java
target/classes/my/code/HelloWorld.class
target/classes/my/code/HelloWorldImpl.class
target/classes/my/code/Main.class
target/jar/hello-world.jar
target/test-classes/my/code/HelloWorldImplTest.class
test/my/code/HelloWorldImplTest.java

For the sake of the examples, this publisher is being run at the end of build number 99, at 3:45 pm and 55 seconds on the 7th November 2010.

Eg 1 Transfer directory

Source files target/classes/**/*
Remove prefix
Remote directory

Result:

target/classes/my/code/HelloWorld.class
target/classes/my/code/HelloWorldImpl.class
target/classes/my/code/Main.class

Eg 2 Remove prefix

Source files target/classes/
Remove prefix target
Remote directory

Result:

classes/my/code/HelloWorld.class
classes/my/code/HelloWorldImpl.class
classes/my/code/Main.class

i target/classes/ == target/classes/** == target/classes/**/*

Eg 3 Environment variables

Transfer all files and folders beneath a directory and place in a directory named with the job name and then build number (given the job is called "Hello World")
Source files target/classes/
Remove prefix target/classes
Remote directory $JOB_NAME/$BUILD_NUMBER

Result:

Hello World/99/my/code/HelloWorld.class
Hello World/99/my/code/HelloWorldImpl.class
Hello World/99/my/code/Main.class

Eg 4 Transfer .class files

Source files target/**/*.class
Remove prefix target
Remote directory

Result:
classes/my/code/HelloWorld.class
classes/my/code/HelloWorldImpl.class
classes/my/code/Main.class
test-classes/my/code/HelloWorldImplTest.class

Eg 5 Transfer files with flatten

Source files **/*.java
Remove prefix
Remote directory /java
Flatten files checked

Result:
java/HelloWorld.java
java/HelloWorldImpl.java
java/Main.java
java/HelloWorldImplTest.java

Eg 6 Remote directory is a date format

Source files target/**/*.jar
Remove prefix
Remote directory 'builds/yyy/MM/dd/build-$BUILD_NUMBER'
Flatten files checked
Remote directory is a date format checked

Result:
builds/2010/11/07/build-99/hello-world.jar

Note The whole of Remote directory is quoted apart from the date tokens and the separators (which are not letters)