Inedo BuildMaster Plugin

Plugin Information
View Inedo BuildMaster Plugin on the plugin site for more information.

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Plugin globally and unconditionally disables SSL/TLS certificate validation
- Plain text password shown in configuration form

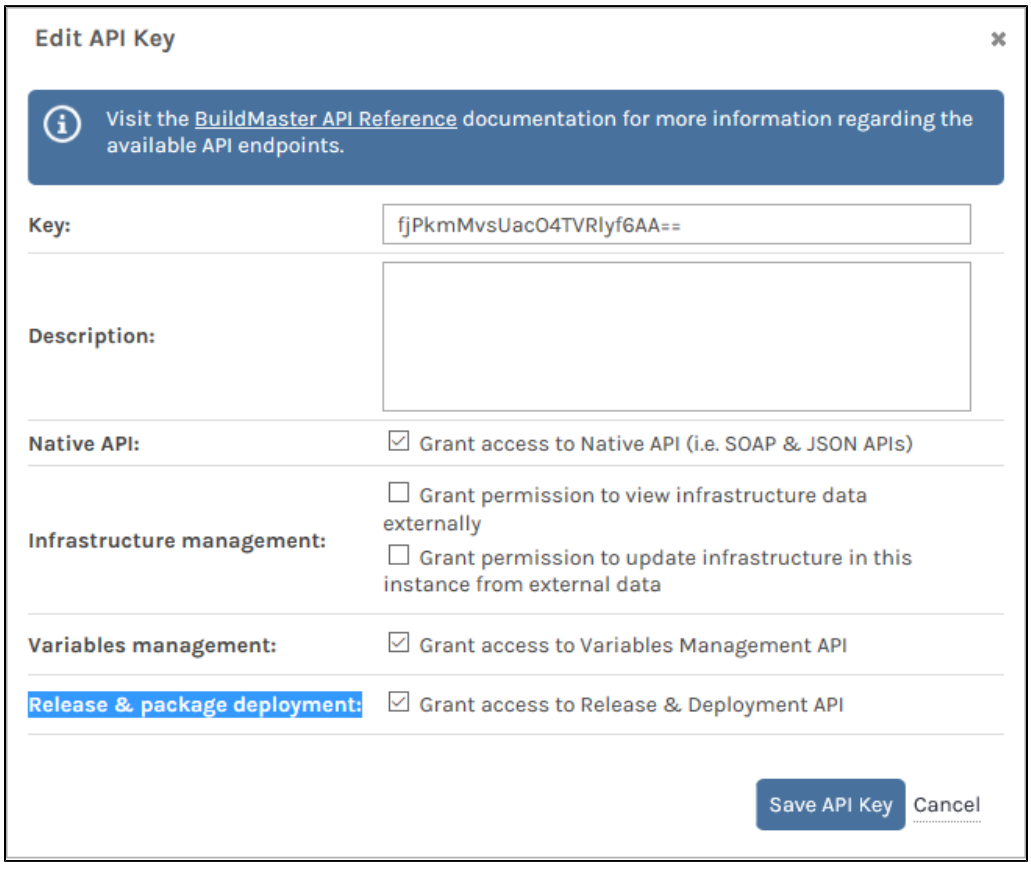
This plugin integrates Inedo BuildMaster to Jenkins allowing Jenkins jobs to gather version information from BuildMaster and trigger builds on an application.

Usage

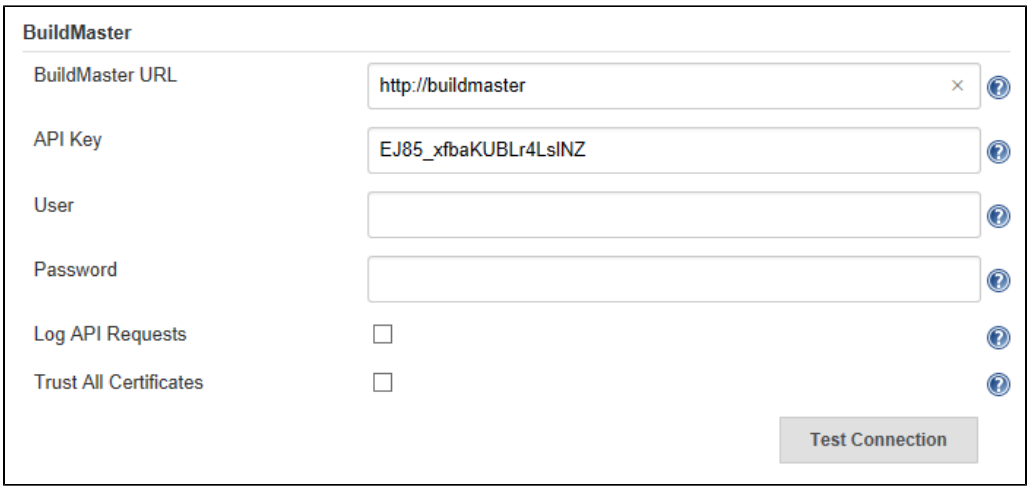
Note: A minimum BuildMaster version of 6.1.0 is required for this plugin

First, you need to ensure that an api key as been configured in BuildMaster at BuildMaster > Administration > Api Keys & Access Logs. Without this the plugin won't be able access BuildMaster. Ensure that the following items are checked:

- Native API
- Variables management
- Release & package deployment:



Next, you need to go to Jenkins' system config screen to tell Jenkins where's your BuildMaster resides. I have found that I need to configure it for NTLM authentication on our network otherwise my account gets locked out. Clicking the "Test Connection" button a few times will confirm whether this is required or not.

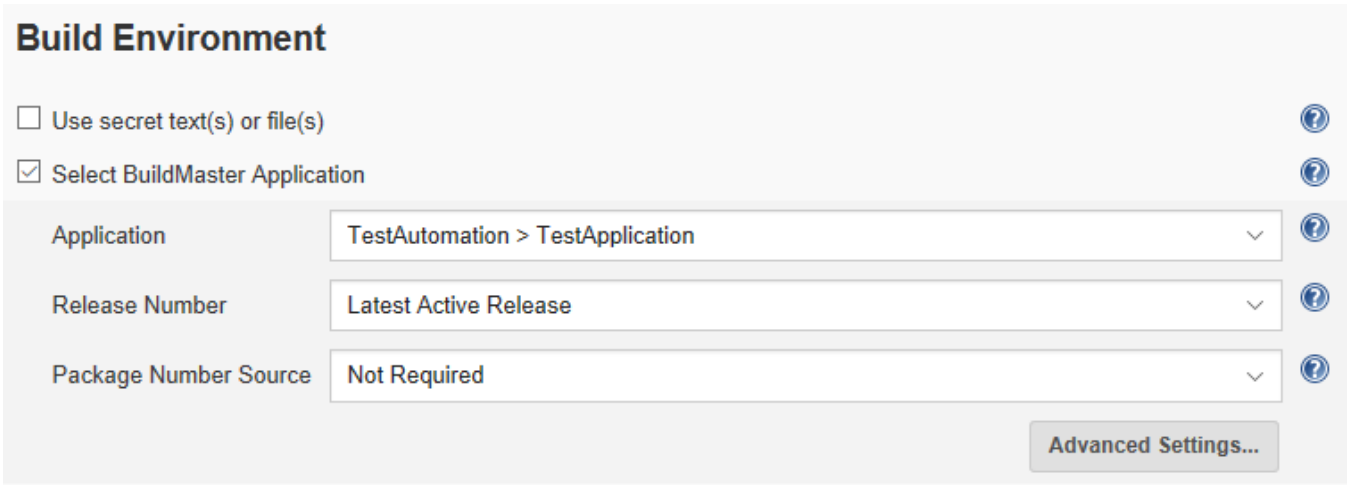


Lastly, you need to add a couple of steps to your Jenkins job. Because the Release and Build numbers could be required by several build and post-build actions selecting the application and has been separated from the trigger build action.

The "*Select BuildMaster Application*" build envrionment setting allows you to select the the BuildMaster application you are dealing with and the settings will be used to inject these environment variables into the job at build time:

- BUILDMASTER_APPLICATION_ID
- BUILDMASTER_RELEASE_NUMBER
- BUILDMASTER_BUILD_NUMBER
- BUILDMASTER_DEPLOYABLE_ID

Ignore the Deployable option for the moment, that will be covered in the advanced topics section later in this document.



The "*Create BuildMaster Package*" action can be added as either a build step or post build action. The choice of which to use will be largely dependent on how you import the build artifacts into BuildMaster and your personal preference:

1. You are using the BuildMaster Jenkins Build Importer Build Step which imports build artifacts from Jenkins: the post build action is required
2. You are using a standard BuildMaster build step and importing files from a folder that you've placed the artifacts into from the Jenkins build (eg using ArtifactDeployer Plugin): either the post build or build step actions will be fine
3. You use an external artifact repository such as Nexus or Artifactory: either the post build or build step actions will be fine

If you haven't used the "Select BuildMaster Application" action then you will need to go into the advanced section and set the application id, release and build number details.

To deploy the build to the first environment ensure that you have the post-build step action "Auto-Promote Build to the Next Environment" set in BuildMaster.

The *"Deploy BuildMaster Package To Stage"* action can be used to deploy (or re-deploy) a package to a specified stage by specifying a Stage Name, or deploy to the next stage in the pipeline by leaving Stage Name empty.

Pipeline Script Support

Script can be generated using the pipeline syntax snippet generator, although with the complexities of these plugins the results will have to be tweaked.

Scripted Pipeline Example

```
node {
        buildMasterWithApplicationRelease(applicationId: '1', packageNumberSource: 'BUILDMASTER') {
                echo """
                  Application id = $BUILDMASTER_APPLICATION_ID
                Release Number = $BUILDMASTER_RELEASE_NUMBER
                Package Number = $BUILDMASTER_PACKAGE_NUMBER
                  """

                BUILDMASTER_PACKAGE_NUMBER = buildMasterCreatePackage(applicationId:
BUILDMASTER_APPLICATION_ID, releaseNumber: BUILDMASTER_RELEASE_NUMBER, packageNumber:
BUILDMASTER_PACKAGE_NUMBER, packageVariables: [variables: "hello=world\nsource=$BUILD_URL"])

                echo "BUILDMASTER_PACKAGE_NUMBER = $BUILDMASTER_PACKAGE_NUMBER"

                buildMasterDeployPackageToStage(applicationId: BUILDMASTER_APPLICATION_ID, releaseNumber:
BUILDMASTER_RELEASE_NUMBER, packageNumber: BUILDMASTER_PACKAGE_NUMBER, waitTillBuildCompleted:
[printLogOnFailure: true])
        }
}
```

Declarative Pipeline Example

```
pipeline {
  agent any
  environment {
        // Prevents echo statement from breaking if packageNumberSource not supplied in
        // buildMasterWithApplicationRelease step below leaving BUILDMASTER_PACKAGE_NUMBER
        // variable undefined
    BUILDMASTER_PACKAGE_NUMBER = null
  }

  stages {
    stage('Main') {
      steps {
        buildMasterWithApplicationRelease(applicationId: '1', packageNumberSource: 'BUILDMASTER') {
            echo """
                      Application id = $BUILDMASTER_APPLICATION_ID
                      Release Number = $BUILDMASTER_RELEASE_NUMBER
                      Package Number = $BUILDMASTER_PACKAGE_NUMBER
            """
            // Jenkins declarative pipeline script has a somewhat restricted syntax.  Unfortunately to return
package
            // number you need to wrap this in a script block
            // See: https://jenkins.io/doc/book/pipeline/syntax/#script
            script {
                BUILDMASTER_PACKAGE_NUMBER = buildMasterCreatePackage(applicationId:
BUILDMASTER_APPLICATION_ID, releaseNumber: BUILDMASTER_RELEASE_NUMBER, packageNumber:
BUILDMASTER_PACKAGE_NUMBER, deployToFirstStage: [waitUntilDeploymentCompleted: true, printLogOnFailure: true],
packageVariables: [variables: "hello=package\nsource=$JENKINS_URL"])
            }

            echo "BUILDMASTER_PACKAGE_NUMBER = $BUILDMASTER_PACKAGE_NUMBER"
            buildMasterDeployPackageToStage(stage: 'Integration', applicationId: BUILDMASTER_APPLICATION_ID,
releaseNumber: BUILDMASTER_RELEASE_NUMBER, packageNumber: BUILDMASTER_PACKAGE_NUMBER,
waitUntilDeploymentCompleted: true, printLogOnFailure: true, deployVariables: [variables: "hello=stage"]))

            echo "Redeploy to Integration"
            buildMasterDeployPackageToStage(stage: 'Integration', applicationId: BUILDMASTER_APPLICATION_ID,
releaseNumber: BUILDMASTER_RELEASE_NUMBER, packageNumber: BUILDMASTER_PACKAGE_NUMBER,
waitUntilDeploymentCompleted: true, printLogOnFailure: true)
        }
      }
    }
  }
}
```

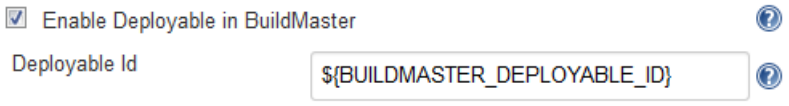
Advanced Topics

Multiple Jenkins Jobs Pointing At A Single BuildMaster Application

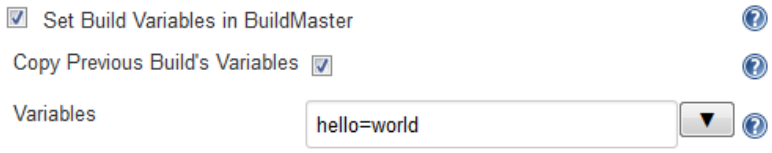
If you have multiple Jenkins jobs all triggering a build for the same BuildMaster application it is vital that you use the "*Select BuildMaster Application*" build step as this this build step will queue any jobs attempting to run for the same BuildMaster application while another is already running so that you cannot get two jobs triggering a build in BuildMaster at the same time.

The following two options have been supplied as a means to ensure that the triggered BuildMaster build picks up artifacts from only the Jenkins jobs that have build for its release:

1. "Enable Deployable in BuildMaster": selectively enables deployables for a release
    a. In BuildMaster you must ensure that Deployables are disabled by default for a release
    b. The "Select BuildMaster Application" Deployable option must be set to the correct deployable



2. "Copy Previous Build's Variables": if checked will gather the variables from the previous build and add them to the list of variables being passed in for this build, overriding any that are being set for this build.
    a. This could be useful if you're only passing in pointers to a version, eg you keep your artfacts in Artifactory or Nexus



History

Version 2.5.0 (14 Sep, 2019)

- Fix SECURITY-1513 secret viewable as plaintext in configuration form

Version 2.4.0 (6 Feb, 2019)

UI compatibility with BuildMaster 6.10

- Update UI to use Build instead of Package to reflect latest BuildMaster
- Update API to use new build rather than obsolete package API

NOTE:

To retain backwards compatibility these have not been altered, but will likely be updated in a future release

- plugin populating BUILDMASTER_PACKAGE_NUMBER variable
- pipeline script function names using "package" in name

Version 2.3.0 (25 Nov, 2018)

Warning: this release contains breaking changes.

While this release has a number of minor updates that improve the overall usability of the plugin, there is no pressing need to update to this release.

This release includes:

- Restructured arguments for createPackage and deployToStage tasks to rename the waitTillBuildCompleted property to waitUntilDeploymentCompleted, and move it to the deployToFirstStage section (*this is the breaking change, see below for migration instructions)*
- Default createPackage task's deployToFirstStage section to checked
- More informative logging
- Updated help text for Jenkins UI
- Last of 'Build' to 'Package' refactoring
- Bug Fix: Do not attempt to get deployment logs on deployment failure if login credentials not configured

Migration Instructions

There is no automatic migration of any changed settings, you will need to either update these via the UI, or direction in the job configuration XML files.

For pipeline script

- buildMasterCreatePackage(..., **deployToFirstStage: true, waitTillBuildCompleted: [printLogOnFailure: true]**) becomes ..., deployToFirstStage: [waitUntilDeploymentCompleted: true, printLogOnFailure: true])
- buildMasterDeployPackageToStage(..., stage: 'Integration', **waitTillBuildCompleted: [printLogOnFailure: true]**) becomes ..., stage: 'Integration', waitUntilDeploymentCompleted: true, printLogOnFailure: true)
  The waitUntilDeploymentCompleted and printLogOnFailure default to true, so only need to be supplied if you want to set these to false.

For freestyle job

I'd recommend using the Jenkins UI to edit the job configuration and recheck the "Wait until deployment completed" and "Show deployment log on failure" in the "Create BuildMaster Package" and "Deploy BuildMaster Package To Stage" build steps and post build actions.

If you have a large number of jobs and wish to edit the job config.xml files manually, these are the changes:

package creation tasks xml

```
<deployToFirstStage>true</deployToFirstStage>
<waitTillBuildCompleted>
  <printLogOnFailure>true</printLogOnFailure>
</waitTillBuildCompleted>
```

becomes

```
<deployToFirstStage>
  <waitTillDeploymentCompleted>true</waitTillDeploymentCompleted>
```

```
    <printLogOnFailure>true</printLogOnFailure>
</deployToFirstStage>
```

deploy to stage tasks xml

```
<waitTillBuildCompleted>
    <printLogOnFailure>true</printLogOnFailure>
</waitTillBuildCompleted>
```

becomes

```
<waitTillDeploymentCompleted>true</waitTillDeploymentCompleted>
<printLogOnFailure>true</printLogOnFailure>
```

Version 2.2.0 (23 Nov, 2018)

- When waiting for a build to complete, also wait for any automatically triggered deployments to subsequent stages to also complete
- Create Package Post Build label rename create package rather than trigger build to align with other tasks
- Bug Fix: Create Package Post Build sets BUILDMASTER_PACKAGE_NUMBER environment variable when finished

Version 2.1.1 (Oct 10, 2018)

- Renamed SetBuildVariables to PackageVariables createPackage task and add variable support to deployToStage task

Version 2.0.2 (Sep 2, 2018)

- Fixed JENKINS-53336 mark build and deployments as successful when status of warned returned from buildmaster

Version 2.0.1 (Aug 28, 2018)

- Fixed JENKINS-53263 passing build variables to BuildMaster createPackage method
- Fixed Serializable error in EnableReleaseDeployable and SetBuildVariables objects

Version 2.0 (Jul 29, 2018)

- Fix security issue
- This version of the plugin is not backwards compatible as it has gone under a complete rewrite to:
    - Move away from the BuildMaster Native APIs to the Release & Package Deployment APIs where possible
    - Add Pipeline script support

This has only been tested with BuildMaster 6, but should work against BuildMaster 5.5+

Version 1.6 (Nov 14, 2016)

- Prevent caching of requests

Version 1.5 (Nov 14, 2016)

- Deploy to stage build step

Version 1.4 (Oct 19, 2016)

- Support BuildMaster 5.5.0

Version 1.3 (May 15, 2015)

- Expand variable values defined in “Set Build Variables in BuildMaster” property
- Queue Jenkins builds targeting same BuildMaster application

Version 1.2 (May 14, 2015)

- Create Build waits for previous execution on BuildMaster to finish before new one is requested
- Improved wait for build to complete handling
- Create Build request triggers build regardless of whether BuildMaster application has build step or not

Version 1.0 (May 1, 2015)

- Initial Release