PlasticSCM Plugin

Plugin Information
View PlasticSCM on the plugin site for more information.

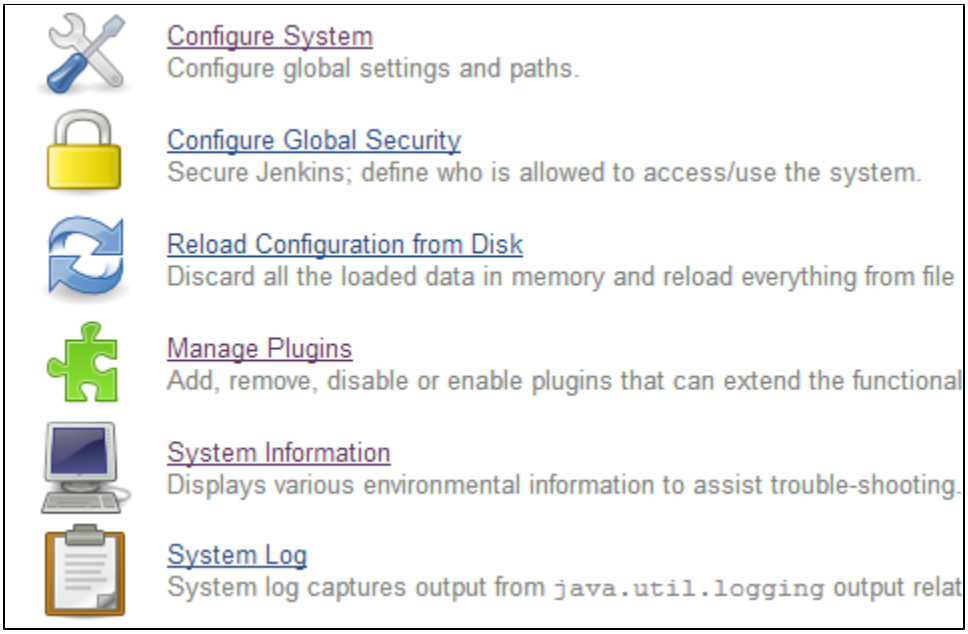
This plugin integrates Plastic SCM with Jenkins.

Configuration

System configuration

In order to configure the plugin, you need the `cm` executable installed in the Jenkins server machine. Then, please follow these steps:

1. Open the system configuration page "Manage Jenkins" and navigate to "Configure System"
2. Scroll down to the "Plastic SCM" section and enter the path where the `cm` executable (CLI client) is located
3. Click "Check" to verify that the path you set is correct
4. You can leave the field empty: "`cm`" will be used by default (i.e. the executable will be loaded from `$PATH`)



Plastic SCM

Command line client executable: cm

Location of Plastic SCM command line client.

Can be a binary name or a full path to the executable. If full path is not specified then the target binary must be present in one of system `PATH` directories. "`cm`" will be used if left blank.

Success

Check

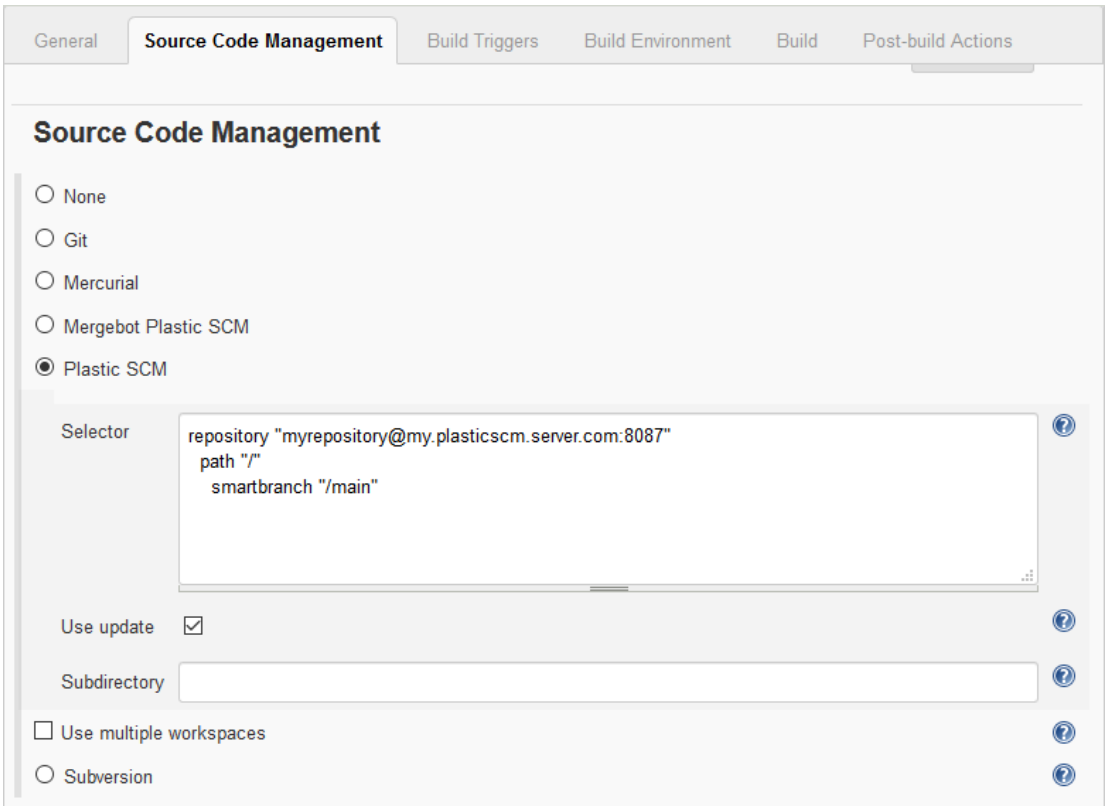
Job configuration

Freestyle projects

Scroll down to the Source Code Management section and select "Plastic SCM" (see screenshot below). You'll need to enter a valid **selector**, which will be used to check out the repository contents in every build. The "Poll SCM" trigger also uses that selector.

Every build will ensure that the Jenkins workspace contains a Plastic SCM before starting the `cm update` operation. If you'd like to create the PlasticSCM workspace in a particular subdirectory under the Jenkins job workspace, you can specify one in the **Subdirectory** field. This field is mandatory if you want to use multiple workspaces.

If you want to keep the workspace between builds, check the **Use update** box. Otherwise, the Plastic SCM workspace and its contents will be deleted before each build.



Selector format

The selector accepts branches, changesets or labels. You can setup not only branches (which is the normal case), but labels as well as changesets in the selector.

Branch selector

```
repository "myRepo@my.plasticscm.server.com:8087"
  path "/"
    smartbranch "/main/scm1773"
```

Changeset selector

```
repository "myRepo@my.plasticscm.server.com:8087"
  path "/"
    smartbranch "/main/scm1773" changeset "7030383"
```

Label selector

```
repository "myRepo@my.plasticscm.server.com:8087"
  path "/"
    label "1.0.32-preview_997"
```

You can also use build parameters in the selector.
Example: Let's say that you defined two build parameters in your project:

- `branchname` - default value (/main)
- `repositoryname` - default value (default)

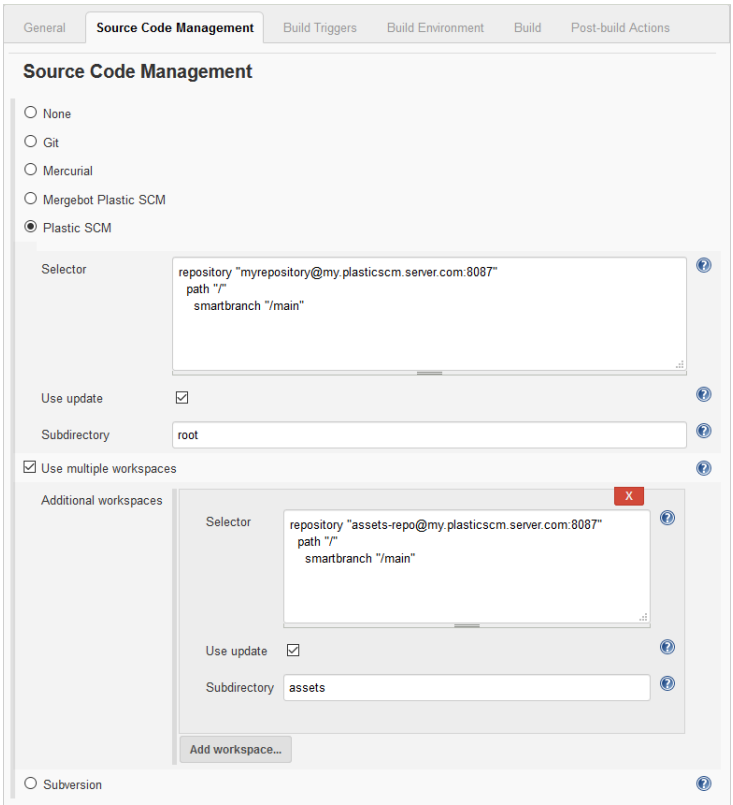
Then you can use those parameters in the Plastic SCM selector using the macro pattern `$parameter_name`:

```
repository '$repositoryname'
    path "/"
        smartbranch '$branchname'
```

Multiple workspaces

The Plastic SCM plugin allows you to checkout multiple workspaces in a single Jenkins workspace. This can be useful if you need to fetch sources from two or more repositories in order to build your application.

To enable this feature, check the **Use multiple workspaces** box in the SCM configuration. You'll see a new button "Add workspace..." to append a new workspace to the list of additional workspaces. The configuration settings are identical to the root ones.



⚠️ **Be careful!** ⚠️ This setup requires you to specify a subdirectory value **for all workspaces**, and they must be different from one another. If you don't, you might risk having the plugin download the contents from two or more repositories into the same directory.

Pipelines

If you use scripted pipelines or you want to specify the pipeline script directly in the job configuration, you can take advantage of the `cm` command inside the groovy script:

cm command syntax

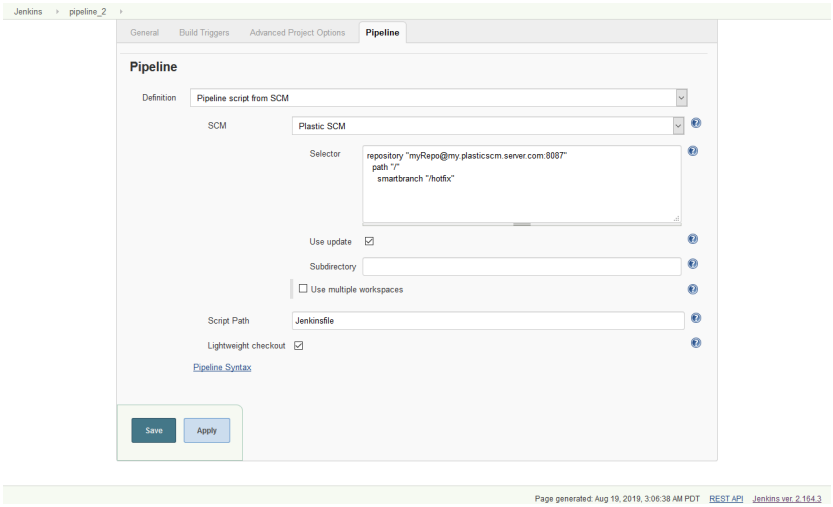
```
cm branch: '<full-branch-name>', repository: '<rep-name>', server: '<server-address>:<server-port>,
useUpdate: (true|false), directory: '<subdirectory-name>'
```

As you see, there's a one-to-one parameter mapping. To use multiple workspaces you simply need to add multiple `cm` statements, paying attention to the value of the `directory` parameter.

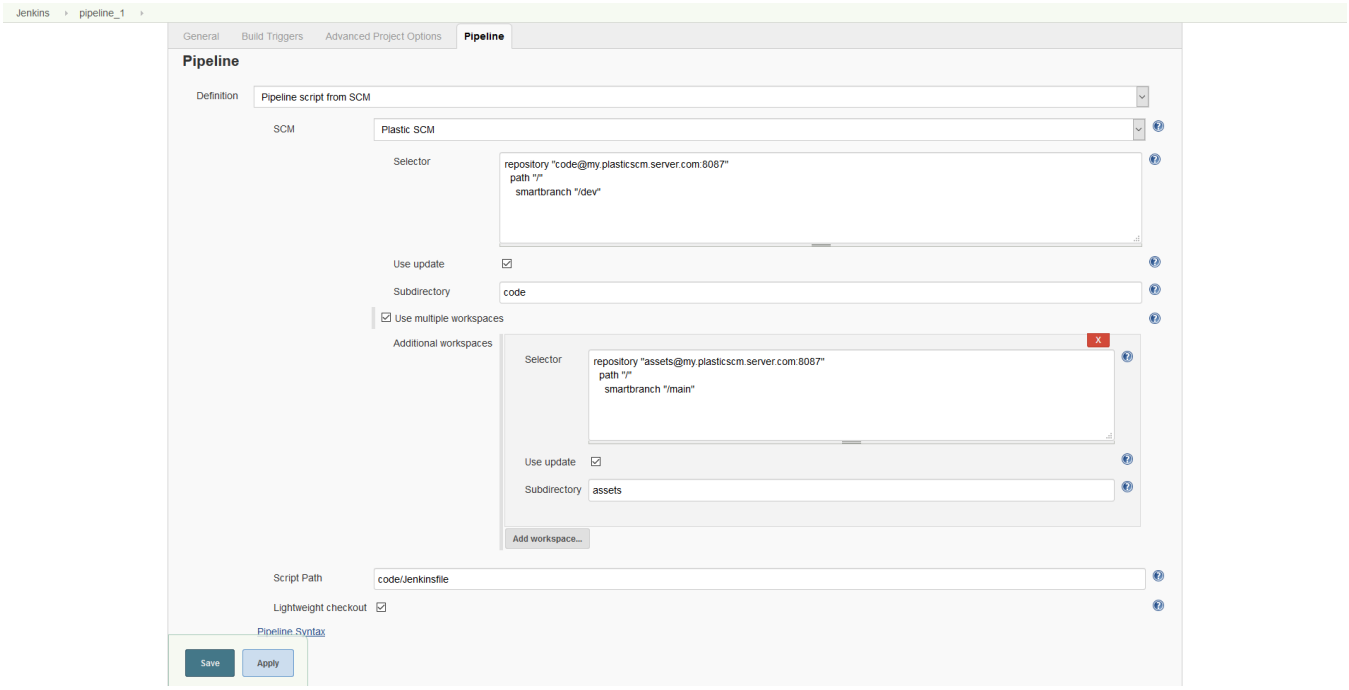
cm command example

```
cm branch: '/hotfix', repository: 'assets-repo', server: 'my.plasticscm.server.com:8087', useUpdate: true,
directory: 'assets'
```

If you'd rather use declarative pipelines (selecting the **Pipeline script from SCM** option), you'll have to specify the Plastic SCM configuration just as you'd do for a freestyle project. Then, two new parameters will appear: **Script path** and **Lightweight checkout**.



The script path tells Jenkins where to find the pipeline script file. **This path is relative to the Jenkins workspace!** So, if you defined subdirectories (regardless of how many additional repositories you described) you'll need to include the subdirectory in the Script path. See an example below:

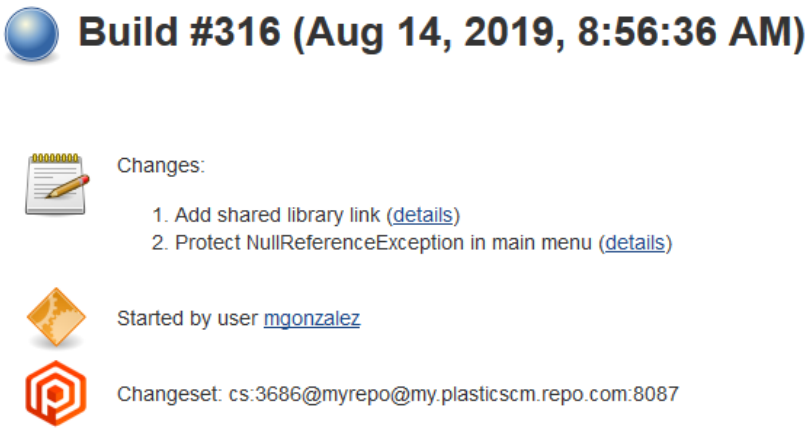


Enabling **lightweight checkout** lets Jenkins retrieve the pipeline script file directly, without a full Plastic SCM update.

Build information

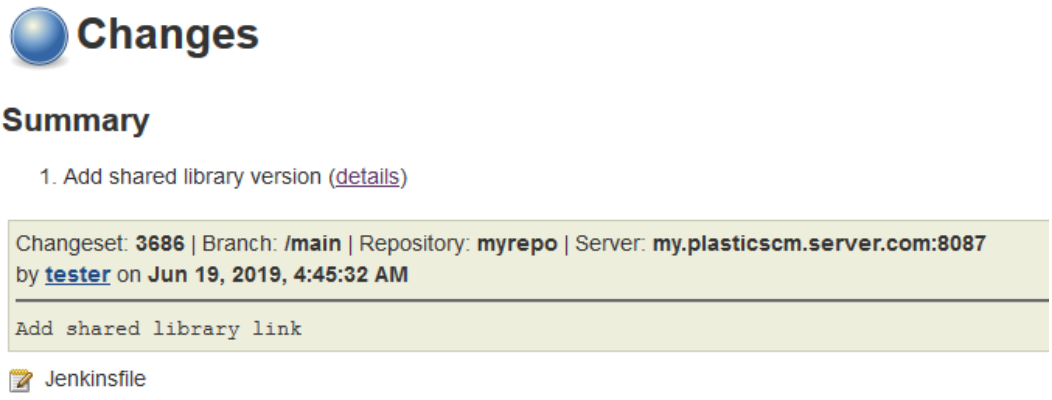
Summary

The build summary includes a reference to the Plastic SCM changeset spec that was built.



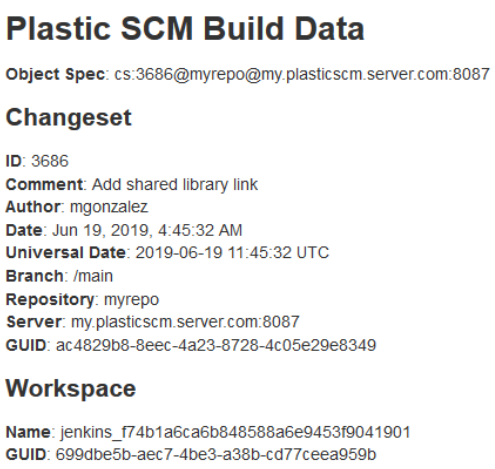
Changes list

The changes page in each build will have details about each of the changesets that were included in the build, linked from the summary list.



Plastic SCM Build Data

You'll find a link in the build page sidebar to display the Plastic SCM data for that build.



Environment variables

If the checkout operation succeeds, these environment variables will be populated with the appropriate values for the build:

- `PLASTICSCM_CHANGESET_ID`: Number of the currently built changeset
- `PLASTICSCM_CHANGESET_GUID`: GUID of the currently built changeset
- `PLASTICSCM_BRANCH`: Name of the branch in Plastic SCM
- `PLASTICSCM_AUTHOR`: Name of the user who created the currently build changeset
- `PLASTICSCM_REPSPEC`: The configured repository specification for the current build

Additional workspaces will include their position in the list, like this:

- `PLASTICSCM_1_CHANGESET_GUID`
- `PLASTICSCM_5_AUTHOR`
- `PLASTICSCM_9_CHANGESET_ID`
- etc.

Requirements

- Jenkins `2.60.3` or newer
- Plastic SCM command line client `8.0.16.3400` or newer

Upgrading from 2.x to 3.x

The upgrade process is mostly seamless. You'll only need to review the configuration parameters of your jobs **if you configured them to use multiple workspaces**. Since the subdirectory name was inferred from the workspace name before and that parameter is now gone, the **Subdirectory** parameter (used now to specify the subdirectory name explicitly) will be empty. Builds might download all workspaces in the same directory!

Plugin information

This plugin has been developed by Codice Software S.L., owner of the Plastic SCM product.

Visit us at http://www.plasticscm.com

You can meet the team here!

We really appreciate PR and contributions in the plugin GitHub repository.

Feel the power of merging branches easier than ever with SemanticMerge!

Change log

Version 3.2 29 Aug 2019

Fixes

- The REST API returned serialization errors. We adapted the Plastic SCM Build Data class and its dependencies to fix them.
- Changes in version 3.0 had broken compatibility with cloud repositories. It's fixed now.
- If you deleted the last built changeset of a job in the Plastic SCM repository, then all builds of new checked in changesets would fail because they wouln't be able to calculate the change log. Now the plugin will keep going back the build history until it detects a built changeset that still exists.

Version 3.1 26 Aug 2019

Fixes

- SCM Polling was broken in the previous release: deserialization was inconsistent for the command that calculates the new changesets between last build and current polling iteration.

Version 3.0 19 Aug 2019

Version 3 is out! This new major version enhances build information and improves SCM configuration parameters. Big shout-out to Housemarque, who contributed enormously to this new batch of changes. Enjoy!

⚠️ **Important** ⚠️ Jenkins baseline version is now 2.60.3 and Plastic SCM client minimum version is 8.0.16.3400! Please ensure you have your software up to date before upgrading the Plastic SCM plugin.

Improvements

- Configuration changed: the Workspace Name parameter was replaced by a new Subdirectory parameter. This allows you full control over where Plastic SCM will download the contens without having to worry about workspace name duplicity. Workspace names are now randomly generated every time the CLI client needs to create a workspace.
    - The Subdirectory parameter has smart validation: it can be empty if there's only a single workspace but required if additional workspaces are selected.
- All builds will have a "last changeset info" now, even if an old build is re-run or there aren't any new changes in the Plastic SCM server.
- Show Plastic SCM changeset information in the *Job status* view.
- Added the *Plastic SCM Build Data* view
- Configuration forms are protected against CSRF attacks. This ensures that only users with configuration permissions can check/validate repository access.
- Massive code refactoring to match common Java coding standars
- Included Checkstyle to verify code style and quality
- Added translation support to UI snippets
- Added a README file in the GitHub repository with software requirements and instructions about how to develop the plugin

Fixes

- All issues related to workspace name collision are gone now. As detailed in *improvements,* you'll no longer need to configure workspace names or worry about duplicates. You can specify the actual subdirectory to check out the sources and the workspace name is randomly generated.
- All parameters have a default value when an additional workspace is added.
- Changelog calculations:

- Branch is now taken into account to show incremental changes
- Fixed issues when the Plastic SCM server and the Jenkins server are in different time zones
- Fixed issues when the build was performed in separate Agents

Version 2.23 08 Jul 2019

Since version 2.22, the plugin is using paths instead of names to find out whether a workspace exists. However, that caused an issue with Windows agents that had their root paths specified with forward slashes (such as C:/jenkins/myAgent). This is fixed now.

Version 2.22 26 Jun 2019

Changes in previous version broke how workspace names were set, which was fixed in this release.

The 'use multiple workspaces' checkbox was broken as well in freestyle and declarative pipeline projects because it was always set as true. Fixed.

We improved how shared library projects are detected to avoid inconsistent workspace names like 'shl-[number]'.

We added support for build parameters in Jenkinsfile pipelines that have lightweight checkout enabled.

The cm path field in the plugin global configuration section now has a validation button.

Version 2.21 12 Jun 2019

There were issues with shared libraries when two or more projects were consuming a single shared library. They were related to the workspace names assigned to the shared library workspace for each project, which turned out to be always the same. We fixed that to make every shared library workspace have its own self-generated workspace name.

Version 2.20 10 Aug 2018

We improved how the plugin reports errors in a Pipeline with Lightweight checkout. Before this, if the Jenkinsfile download failed only a 'NULL' message was printed. Now the complete command execution is displayed.

Fixed an incompatibility with other plugins if they require the SCM plugin to support the ChangeLogSet.Entry.getAffectedfiles() method.

Version 2.19 07 Aug 2018

Added support to SCM environment variables for pipelines.

Now, you can check the available ones here: https://<your-jenkins>/env-vars.html

Version 2.18 19 Jul 2018

- The "${workspace-path} is not in a workspace" error was thrown the next time that Jenkins started a build if the workspace had been previously removed. Fixed.
- The find changeset operation used a wrong branch when the specified branch value was different from the default one in parameterized builds. Fixed

Version 2.17 03 Jul 2018

- The checkout process will undo all local changes in the workspace if there are any, to make sure the update operation won't fail.
- The environment variables weren't properly set if the current or previous build checkout failed. Fixed.
- The Plastic SCM plugin didn't work with pipeline projects. This is a regression of 2.16 version. Fixed.

Version 2.16 08 Jun 2018

The Plastic SCM plugin had a file path issue that prevented it from working as expected when the master and slave instances had different OS. Fixed.

Version 2.15 16 May 2018

The parameters of the plastic workspace name were not correctly resolved. It means, it used the exact workspace name string (e.g. 'Jenkins-${JOB_NAME}-${NODE_NAME}') without resolving the parameters JOB_NAME and NODE_NAME (e.g. 'Jenkins-project-MASTER'). Fixed.

Version 2.14 03 May 2018

The Plastic SCM plugin can work with multiple plastic workspaces or just a single plastic workspace. Now, the jenkins workspace and the plastic workspace paths will match in the single workspace mode.

Therefore, some jenkins features (such as pipeline shared libraries) that need both paths to match will correctly work.

Version 2.13 20 Apr 2018

- Added support for the lightweight checkout feature in the pipeline jobs. It requires that the latest version of 'cm' is installed.
- The environment variables were not published when there were no new changes in the build. Fixed.

Version 2.12 10 Apr 2018

In Blue Ocean, if a build included multiple changesets, only the first one was rendered in the details. Also, the info for the commit and timestamp columns were not filled. Fixed.

Version 2.11 23 Mar 2018

Reduced the number of duplicated builds that can happen using the Plastic SCM plugin. Now, the scm polling takes into account the current build, avoiding to start a new build for the same changeset.

Version 2.10 11 Jan 2018

We fixed an issue configuring existing pipeline projects: the PlasticSCM entry didn't appear in the SCM dropdown list if the pipeline was set to get the script from SCM.

Also, now the Plastic SCM configuration will automatically propose a default workspace name for the first (mandatory) workspace.

Version 2.9 17 Mar 2017

From now on, each build will publish environment variables containing the data of the built changeset for each configured workspace. These are the variables exposed by the main workspace of the project:

- PLASTICSCM_CHANGESET_ID: Number of the currently built changeset
- PLASTICSCM_CHANGESET_GUID: GUID of the currently built changeset
- PLASTICSCM_BRANCH: Name of the branch in Plastic SCM
- PLASTICSCM_AUTHOR: Name of the user who created the currently build changeset
- PLASTICSCM_REPSPEC: The configured repository specification for the current build

Additional workspaces will include their position in the list, like this:

- PLASTICSCM_1_CHANGESET_GUID
- PLASTICSCM_5_AUTHOR
- PLASTICSCM_9_CHANGESET_ID
- etc.

Version 2.8 16 Feb 2017

- The required core version is now 1.580.1
- Added support for pipelines
- Fixed builds being triggered if connection with the Plastic SCM server was lost.
- Plastic SCM commands will be retried 3 times from now on, waiting 0.5 seconds between retries.

The pipeline script syntax for Plastic SCM is:

```
cm branch: '<full-branch-name>', changelog: (true|false), poll: (true|false), repository: '<rep-name>', server:
'<server-address>:<server-port>, useUpdate: (true|false), workspaceName: '<wk-name-using-jenkins-variables>'
```

For example:

```
cm branch: '/main', changelog: true, poll: true, repository: 'default', server: 'localhost:8087', useUpdate:
true, workspaceName: 'Jenkins-${JOB_NAME}-${NODE_NAME}'
```

Version 2.7 10 Oct 2016

- Fixed a problem causing parameterized builds to have their workspaces deleted before each build run.

Version 2.6 26 Jul 2016

- Replaced all relative paths (implicit or explicit) with full, explicit paths. This fixed several issues on Mac OS X since apparently the current working directory for VCS commands is being set to '/' by Jenkins.

Version 2.5 25 Apr 2016

- Cross-platform setups (linux server + windows agents) were deleting workspaces before each build, regardless of the actual "Use update" checkbox value. Fixed.

Version 2.4 29 Feb 2016

- Workspaces were being deleted before each build on Windows, regardless of the actual value of the "Use update" checkbox. Fixed.

Version 2.3 21 Oct 2015

- Added build parameters support in the Plastic SCM selector. Jenkins allows to define build parameters (https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Build), and Plastic SCM now can use those parameters in its selector.
  IMPORTANT NOTE: When using parametrized builds, maybe the Poll SCM feature may not work as expected because Jenkins performs the poll with the LAST USED workspace. So maybe the selector is not placed in the branch you expect the poll is performed.
  When using parametrized builds we recommend setting up two Jenkins projects:
  - One for the parametrized build
  - Other, with an static selector to perform the poll.
- Fixed: When a changeset was a result of a merge, Jenkins was not able to properly present modified elements in 'Changes' chapter.

Version 2.2 09 Dec 2014

- The Jenkins workspaces now support multiple Plastic SCM workspaces.

Version 2.1 02 Sep 2014

- Fixed an issue related to the non-ASCII characters included on the date on some cultures such as Korean culture.
- Support labels and changesets in the Plastic SCM selector.
- The plugin did not reuse Plastic SCM workspaces correctly when the 'Use update' preference was set. Fixed.

Version 2.0 20 Jan 2012

- Plugin adapted to Plastic SCM version 4.

Version 1.0 29 Mar 2011

- Initial version.