



How commit policies enable end-to-end traceability and faster code reviews

Code commit rules your developer team will follow



Who is Midori?



Experience

10+ years in the Atlassian Ecosystem

Top Vendor

(earlier was called “Atlassian Verified”)

Reliable maintenance, guaranteed support and top-notch docs

Powering 4000+ customers

Bank of America, BMW, Northrop Grumman, Lenovo, etc.

What is Better Commit Policy?



On the market since 2015

The must-have app for
every developer team

“Set it and forget it” commit rules

Code change, tag and branch verification for
Git, Bitbucket, GitHub, GitLab, Subversion & Mercurial



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team



Why do you need commit policies?

Key reasons for using commit policies



Code repositories need to follow processes/regulations

Commit policies ensure compliance in industries like:

- Avionics
- Automotive
- Financial
- Defence
- Medical
- Pharmaceutical
- (Other safety-critical)



Changes to source code need to be connected to requirements

The relation needs must be traceable between code changes and:

- Requirements
- Test executions
- Bug reports
- Version releases
- User stories



A well-controlled code base makes reviews and audits easy

Commit Policies help answer questions like:

- Why was this changed?
- Was it tested?
- Is this within the scope of the current release?
- Who approved this merge?
- Is there a programmatic test for this story?



Why do you need commit policies?

What are the risks of an uncontrolled repository?

The screenshot shows a CNN Money article from May 29, 2017. The article title is "Computer meltdown may cost **YOUR COMPANY** over \$100 million". The author is Ivana Kottasová. The article features a photograph of a crowded airport check-in area with a sign that says "BRITISH AIRWAYS We're here to help". A red banner at the bottom of the photo reads "British Airways' costly computer system crash". To the right of the photo is a "Personal Finance" table with columns for "Mortgage", "Personal Loans", and "Credit Cards". The table lists loan types and their corresponding rates and APRs.

Mortgage	Personal Loans	Credit Cards
Loan Type	Rate	APR
30-yr fixed	4.25%	4.342%
15-yr fixed	3.875%	3.927%
5/1 ARM	4%	4.708%

Source: <https://www.cnn.com/2017/05/29/airline-computer-crash/index.html>



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

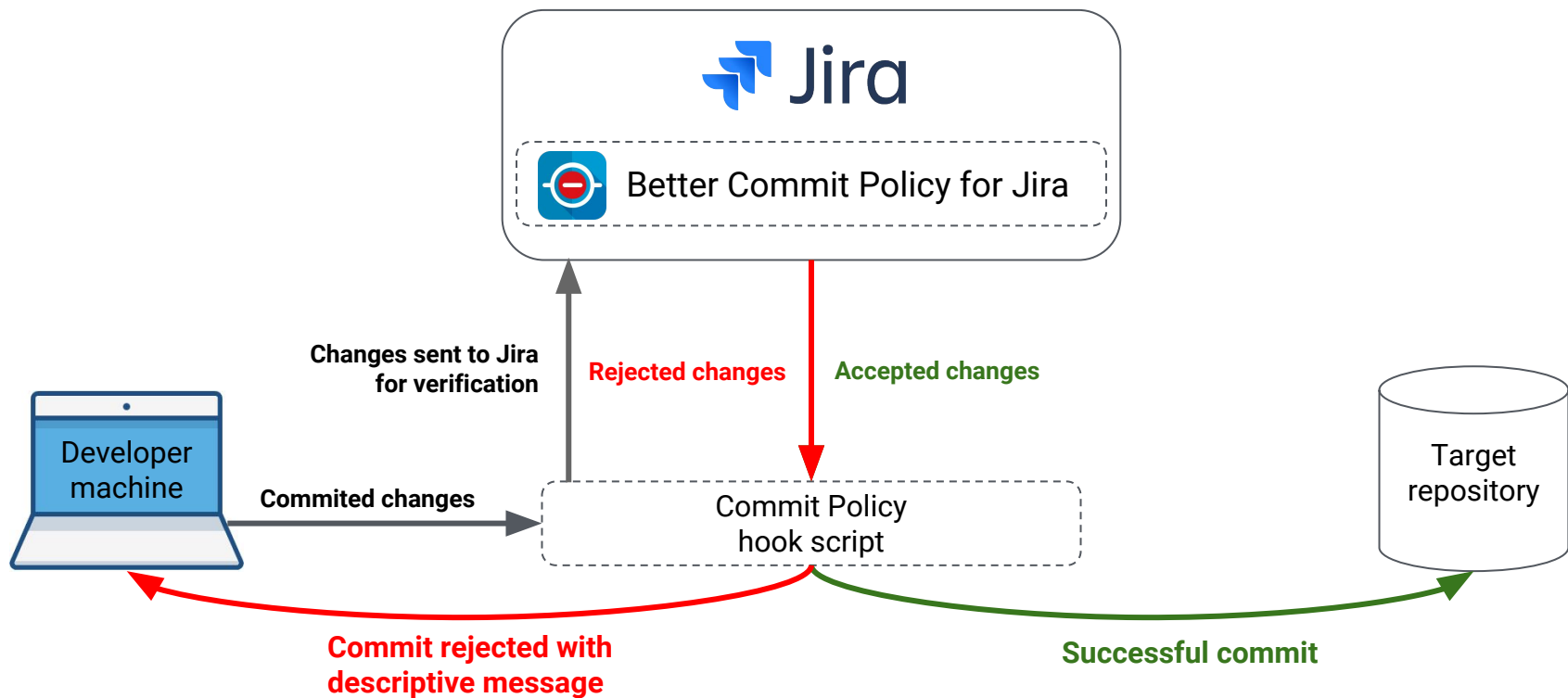
Commit policies in any environment

Introducing commit policies to your team



How do commit policies work?

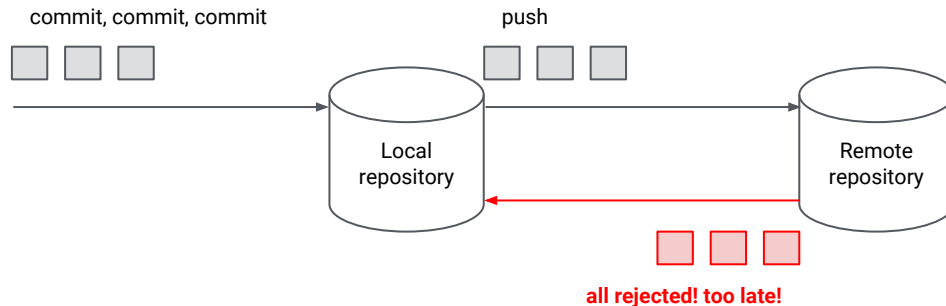
The commit verification process



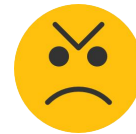
How do commit policies work?

The KILLER feature you won't find elsewhere: local commit verification with Git

Usual approach:
Verify remotely when pushing

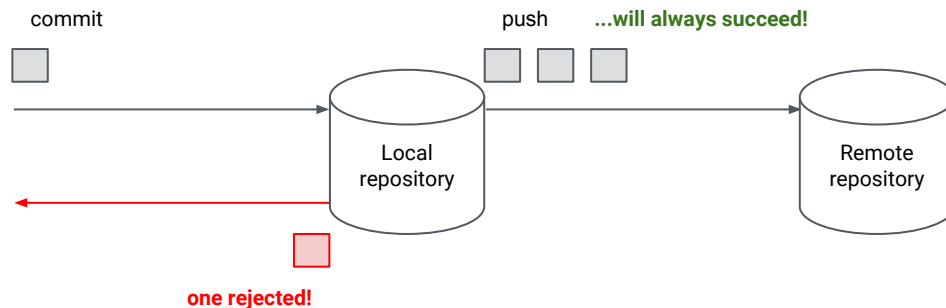


level of frustration
when rejected



Arghhh!
I must use interactive
rebasing to fix all those...

The Midori approach:
Verify locally when committing



Okay, let me just
quickly re-commit the
very last one.



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team



Defining commit policies

Commit policy = rules for who can change what under what conditions in repositories

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Calendar Commits ▾ Planning poker **Create** Search 🔍 🗨️ ? ⚙️ 👤

Commit Policies

[Global Configuration](#)

i Commit policies are named sets of conditions that will be verified against every commit to be sent to the central Version Control System repository. You can create, update or delete them here.
[More about policies](#) →

ID	Name	Description	Actions
18	Application design guidelines	This policy is for the application (Android and iOS) design guidelines and documentation	Apply to a repository · 🔒 🗑️
7	Crypto Exchange Policy	Commits accepted only against unresolved issues in the CEP project.	Apply to a repository · 🔒 🗑️
12	Current CLD Sprint	Commits only against the "Cloud Hosting" Sprint will be accepted.	Apply to a repository · 🔒 🗑️
20	EVM - Solidity	This policy ensures that only .solidity files are committed and changed	Apply to a repository · 🔒 🗑️
9	FREEZE DISABLED	Commit policy to freeze the repository by rejecting all changes.	Apply to a repository · 🔒 🗑️
8	Image files only	Only image files in the ".jpg" ".jpeg" ".svg" ".png" ".gif" formats are accepted.	Apply to a repository · 🔒 🗑️
19	Sounds and music	This rule only allows audio files to be changed.	Apply to a repository · 🔒 🗑️
1	TOMCAT	Commit policy for Apache Tomcat, an open-source server and servlet container.	Apply to a repository · 🔒 🗑️

[+ Add policy](#)



Defining commit policies

Editing the details of commit policies

Navigation: Jira Software | Dashboards | Projects | Issues | Boards | Calendar | Commits | Planning poker | Create | Search | Help | Settings | Profile

Edit Commit Policy

Policy ID:

Name*:

Description:

Rejection message:

TIP If your Version Control System or Operating System console fails with international characters, consider using English-only characters.

Options

- Accept the commits that **already exist** in the repository (on another branch) without verification
The commits that have their SHA already existing in the repository will not be verified again. For example, when being merged to a new branch. (For Git only.)
- Accept the **merge** commits without verification
(For Git and Mercurial.)

Tag rules: The tag must satisfy **all** of these rules:



Defining commit policies

Adding tag, branch and commit rules to commit policies

Accept the **merge** commits without verification

(For Git and Mercurial.)

Tag rules The tag must satisfy **all** of these rules:

 No rules added yet.

+ Add rule

Branch rules The branch must satisfy **all** of these rules:

 No rules added yet.

+ Add rule

Commit rules The commit must satisfy **all** of these rules:

 No rules added yet.

+ Add rule

Save Cancel



Defining commit policies

Customizing the conditions within a tag rule

Tag rules The tag must satisfy **all** of these rules:

Rule 1 ✕

Apply this rule only if the tag name matches Glob ⓘ

Additional rejection message for this rule

The commit must satisfy all of these conditions:

Condition 1.1 **Tag name must match a pattern** ✕

RegEx ⓘ

▶ Examples

Condition 1.2 **Tag name must contain issue keys from a JQL query** ✕

exactly one issue in ✔

Allow and ignore the issue keys that don't match the JQL ⓘ

Leave the JQL blank to accept any existing issue key.
Non-existing issue keys are strictly rejected. Other references with similar syntax can be ignored by re-configuring the issue key pattern.

▶ Examples

+ Add condition ▾

+ Add rule



Defining commit policies

Customizing the conditions within a branch rule

Branch rules The branch must satisfy **all** of these rules:

Rule 2 ✕

Apply this rule only if the branch name matches Glob ⓘ

Additional rejection message for this rule

The commit must satisfy all of these conditions:

Condition 2.1 **Branch name must match a pattern** ✕

RegEx ⓘ

[▶ Examples](#)

Condition 2.2 **Branch name must contain issue keys from a JQL query** ✕

exactly one issue ▼ in ✔

Allow and ignore the issue keys that don't match the JQL ⓘ

Leave the JQL blank to accept any existing issue key.
Non-existing issue keys are strictly rejected. Other references with similar syntax can be ignored by re-configuring the issue key pattern.

[▶ Examples](#)

[+ Add condition](#) ▼

[+ Add rule](#)



Defining commit policies

Customizing the conditions for commit messages

Commit rules The commit must satisfy **all** of these rules:

Rule 1

Limit scope

Apply this rule only if the branch name matches Glob ⓘ

Apply this rule only for the files whose path matches Glob ⓘ

Additional rejection message for this rule

Enter the message shown for the committer when his commit is rejected by this specific rule (consider including an example that would be accepted)

TIP If your Version Control System or Operating System console fails with international characters, consider using English-only characters.

The commit must satisfy **all** of these conditions:

Condition 1.1

Commit message must match a pattern

Glob ⓘ

▸ Examples

+ Add condition

Defining commit policies

Customizing the conditions for commit messages

Condition 2.1

Commit message must match a pattern



Regex



Examples

Require at least 10 character long commit messages (excluding whitespace):

COPY REGEX `(\S|\S\s*){10,}`

Require a Jira issue key from the FOO or BAR projects in the start of the commit message:

COPY REGEX `(FOO|BAR)-\d+.*`

Enforce the 50/72 rule:

COPY REGEX `\S.{0,49}((\r\n|\r|\n)|(\r\n|\r|\n).{1,72}((\r\n|\r|\n){1,2}.{1,72}(\r\n|\r|\n)?))*`



Defining commit policies

Customizing the conditions for commit messages

Condition 2.2


Commit message must contain issue keys from a JQL query

exactly one issue



in project = "Beem for Business" and status= "In Progress" and assignee = current



Allow and ignore the issue keys that don't match the JQL 

Leave the JQL blank to accept any existing issue key.

Non-existing issue keys are strictly rejected. Other references with similar syntax can be ignored by re-configuring the [issue key pattern](#).

• Examples

Issues from project FOO:

```
COPY JQL project = FOO
```

Issues from project FOO and assigned to the commiter (for Git):

```
COPY JQL project = FOO and assignee = currentUser()
```

Issues from project FOO and assigned to the commiter (for Subversion):

```
COPY JQL project = FOO and assignee = "$committer.userName"
```

Issues from project FOO and assigned to the commiter or to anyone if the commiter is the project lead (for Git):

```
COPY JQL project = FOO and (assignee = currentUser() or project in projectsLeadByUser())
```

Issues from project FOO and assigned to the commiter or to anyone if the commiter is the project lead (for Subversion):

```
COPY JQL project = FOO and (assignee = "$committer.userName" or project in projectsLeadByUser("$committer.userName"))
```

User stories in the current sprint from project FOO:

```
COPY JQL project = FOO and issuetype = Story and sprint in openSprints()
```



Defining commit policies

Customizing the conditions for changed files

Condition 2.3

Changed paths (files) must match a pattern

Glob  

Examples

Avoid checking in *.obj, *.tmp, *.class files:

COPY GLOB !(*.obj, *.tmp, *.class)

Allow files with restricted types in a directory, e.g. **/images must contain image files:

COPY GLOB */images/*.{jpg,png,gif}

Lock a file, i.e. no changes are allowed on a file:

COPY GLOB !(*/*LICENSE.txt, LICENSE.txt)

Lock a directory, i.e. no changes are allowed on a directory and its descendants:

COPY GLOB !(production-config/)

Lock (freeze) the whole repository, i.e. to reject all changes in a repository:

COPY REGEX (?!.*)

Enforce naming conventions on Java files (tip: implement any naming convention similarly):

COPY REGEX (((.*\/|^)([A-Z][a-z0-9_-]+)\.java)|(((?!\.java\$)[\/\w.-]+))\$

Protect the Subversion repository structure (/trunk, /branches, and /tags):

COPY REGEX (.*\/)?(trunk|branches|tags)/*



Defining commit policies

Customizing the conditions for changed files

Condition 2.4

Changed paths (files) must contain issue keys from a JQL query

exactly one issue in

Allow and ignore the issue keys that don't match the JQL 

Leave the JQL blank to accept any existing issue key.

Non-existing issue keys are strictly rejected. Other references with similar syntax can be ignored by re-configuring the [issue key pattern](#).

Examples

Issues from project FOO:

`project = FOO`

Issues from project FOO and assigned to the commiter (for Git):

`project = FOO and assignee = currentUser()`

Issues from project FOO and assigned to the commiter (for Subversion):

`project = FOO and assignee = "${committer.userName}"`

Issues from project FOO and assigned to the commiter or to anyone if the commiter is the project lead (for Git):

`project = FOO and (assignee = currentUser() or project in projectsLeadByUser())`

Issues from project FOO and assigned to the commiter or to anyone if the commiter is the project lead (for Subversion):

`project = FOO and (assignee = "${committer.userName}" or project in projectsLeadByUser("${committer.userName}"))`

User stories in the current sprint from project FOO:

`project = FOO and issuetype = Story and sprint in openSprints()`



Defining commit policies

Customizing the conditions for committer users

Condition 2.5

Committer must have a valid JIRA account ✕

Committer's must identify a valid Jira user account

User must be member in any of these Jira groups:

e.g. the "jira-developers" group only

+ Add condition ▾



Defining commit policies

Customizing the conditions for committer users

Condition 2.6 **Committer attribute must match a pattern** ✕

Committer's must match the pattern ⓘ

▾ Examples

Accept from a specific email domain:

Accept from a list of usernames:

Accept nothing (freeze the repository):

+ Add condition ▾



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team



Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with a 'Commit Policies' sidebar and a 'Hook Script Wizard' dialog box. The wizard is currently on the first step of a six-step process. The background shows a table of existing commit policies, including 'TOMCAT' which is selected.

Hook Script Wizard

I want to apply the **TOMCAT** commit policy

to a **Git** repository

More help about Git hook scripts

so that the commits are verified **in the central repository (remote)**

hosted on **in my local repository (clone)**

Select the OS of the server which hosts the Git repositories and hooks.

Next

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

+ Add policy

Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with the 'Commit Policies' section active. A 'Hook Script Wizard' dialog box is open, guiding the user through the installation of a commit policy. The wizard is currently on the first step, where the user selects the commit policy to apply, the repository type, the verification location, and the operating system.

Hook Script Wizard

I want to apply the **TOMCAT** commit policy

to a **Git** repository

so that the commits are verified **in the central repository (remote)**

hosted on **Linux, Mac OS X (All U*x variants)**

Next

Commit policy for Apache Tomcat, an open-source server and servlet container.

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with a 'Commit Policies' page. A modal dialog titled 'Hook Script Wizard' is open, displaying a progress bar with four steps. The first step, 'Install Python for the hook scripts', is active. The dialog contains the following text:

Hook Script Wizard [Close]

Install Python for the hook scripts

If this is the **very first** hook script for this Git server, make sure that the software required for executing the hook scripts are installed to the server. You only need to **execute this step once**, by clicking the link below and following the instructions. If you already completed this step before, you can safely skip it now.

[Show instructions](#)

[Back](#) [I have installed Python](#)

The background interface shows a table of commit policies:

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

At the bottom of the wizard, there is a note: 'Commit policy for Apache Tomcat, an open-source server and servlet container.'

Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with a 'Commit Policies' page. A 'Hook Script Wizard' dialog box is open, displaying a progress bar and a 'Download' button. The background shows a table of commit policies and a list of actions for each policy.

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Calendar Commits ▾ Planning poker **Create** Search 🔍 🔔 ? ⚙️ 👤

Commit Policies

⚙️ Global Configuration

i Commit policies are named sets of...
More about policies →

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

+ Add policy

Hook Script Wizard

×

Download the hook script package

Click the button, then a ZIP archive with all necessary files will be downloaded to your browser.

⬅️ **Download**

Back

I have downloaded the ZIP

update or delete them here.

Actions

- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️

Commit policy for Apache Tomcat, an open-source server and servlet container.



Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with the 'Commit Policies' section active. A 'Hook Script Wizard' dialog box is open, displaying the first step: 'Install the hook script'. The wizard includes a progress bar, a list of steps, and instructions for unpacking a ZIP file to the correct location. The background shows a table of existing commit policies and a list of actions for each.

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Calendar Commits ▾ Planning poker **Create** Search 🔍 🔔 ? ⚙️ 👤

Commit Policies

⚙️ Global Configuration

i Commit policies are named sets of rules that control what can be committed to a repository. [More about policies →](#)

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

+ Add policy

Hook Script Wizard

×

Install the hook script

1

Unpack the ZIP file to the location where Git expects it

You will need write access to the filesystem where Git stores the repositories and hooks. Ask for the help of your system administrator when in doubt.

The destination directory is here by default:

```
<CENTRAL_GIT_REPO>/ .git/hooks
```

...or here for a Git bare repository (if you don't know what it is, then use the first location):

Back

I have installed the hook script

Commit policy for Apache Tomcat, an open-source server and servlet container.

update or delete them here.

Actions

Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️

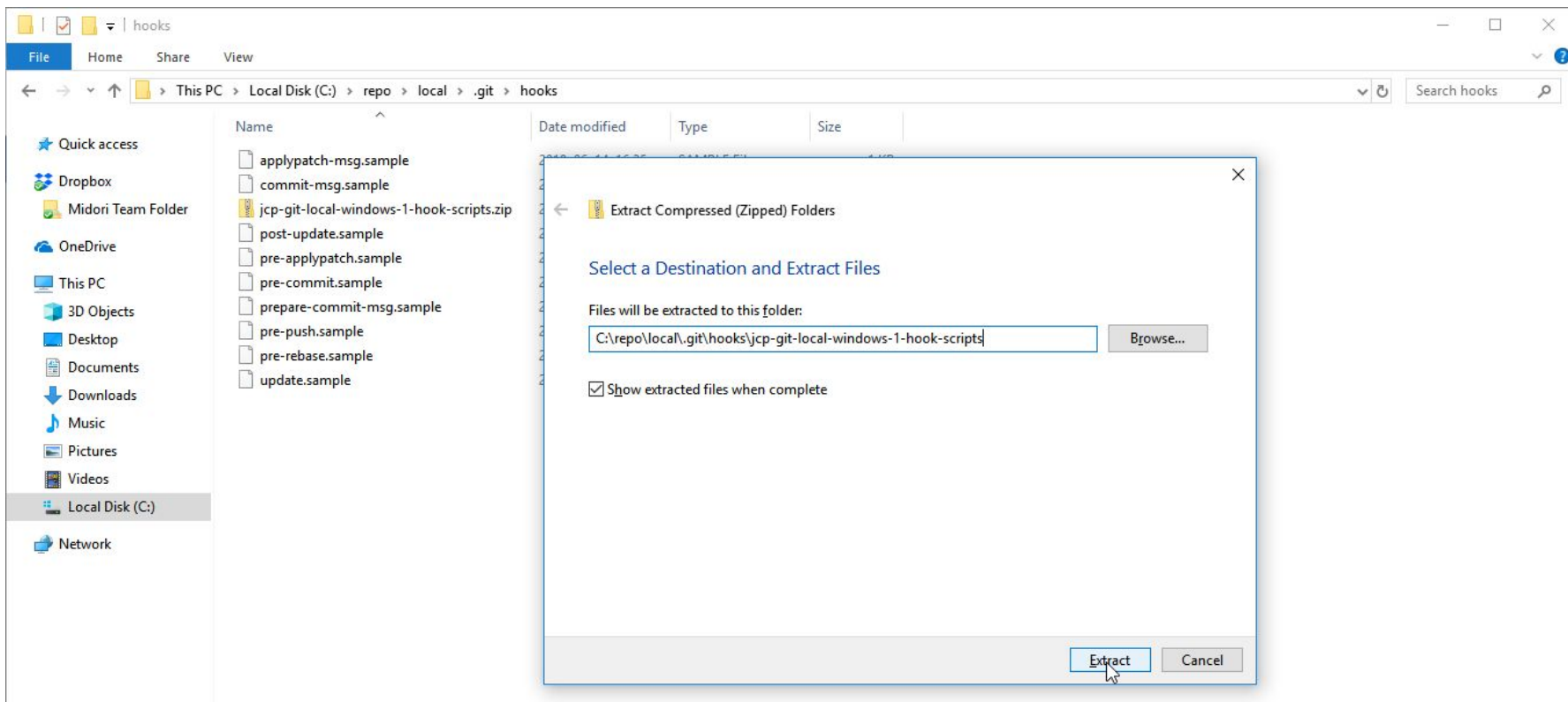
Apply to a repository · 🔒 🗑️

Apply to a repository · 🔒 🗑️



Installing commit policies

Installing and applying commit policies to repositories



Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with the 'Commit Policies' page. A 'Hook Script Wizard' dialog box is open, displaying instructions for testing configurations. The background shows a table of existing policies and a list of actions for each.

Jira Software Dashboards ▾ Projects ▾ Issues ▾ Boards ▾ Calendar Commits ▾ Planning poker **Create** Search 🔍 🔔 ? ⚙️ 👤

Commit Policies

⚙️ Global Configuration

i Commit policies are named sets of rules that control what can be committed to a repository. [More about policies →](#)

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

+ Add policy

update or delete them here.

Actions

- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️
- Apply to a repository · 🔒 🗑️

Hook Script Wizard

Test your configuration

- 1 Try to commit something that should be rejected**
Ex: if your policy requires at least one issue key, then make a commit with the message "No issue key here".
Check if it is rejected.
- 2 Try to commit something that should be accepted**
Ex: if your policy requires at least one issue key, include one in the commit message: "Fix for FOO-17".
Check if it is accepted.

Back **I have tested it**

Commit policy for Apache Tomcat, an open-source server and servlet container.



Installing commit policies

Testing your commit rule

```
Terminal File Edit View Search Terminal Help
/projects/guava$ git status
On branch feature/nullability-review
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   guava/src/com/google/common/collect/Table.java
        modified:   guava/src/com/google/common/collect/Tables.java

no changes added to commit (use "git add" and/or "git commit -a")
/projects/guava$ git commit -am "Add Nullable annotations to all contains() methods"
=====
REJECTED!
=====
Start the commit message with the key of an unresolved in-progress issue!
-----
1.1 COMMIT [NEW COMMIT] on [feature/nullability-review] Exactly one issue key must be mentioned in the commit message "Add Nullable anno..."
1.2 COMMIT [NEW COMMIT] on [feature/nullability-review] Commit message "Add Nullable anno..." must match the pattern <GVA-\d+.*> (regex)
=====
REJECTED!
=====
/projects/guava$ git commit -am "GVA-3 Add Nullable annotations to all contains() methods"
=====
REJECTED!
=====
Start the commit message with the key of an unresolved in-progress issue!
-----
1.1 COMMIT [NEW COMMIT] on [feature/nullability-review] Issue key [GVA-3] not found in the JQL query result: <project = GVA AND status = "In Progress" AND resolution IS EMPTY> (by user <admin>)
-----
REJECTED!
=====
/projects/guava$ git commit -am "GVA-4 Add Nullable annotations to all contains() methods"
[feature/nullability-review d73085d] GVA-4 Add Nullable annotations to all contains() methods
 2 files changed, 2 insertions(+), 2 deletions(-)
/projects/guava$
```

Make a change, and commit it without linking to a Jira issue.

Rejected (immediately, not when pushing to the server)!

Commit again including a Jira issue in the commit message.

Rejected, as the issue key is not an unresolved in-progress issue!

Commit again including a Jira issue that meets the requirements.

Accepted.



Installing commit policies

Testing your tag rule

```
Terminal File Edit View Search Terminal Help
/projects/acne-dynamite$ git tag v1.2
/projects/acne-dynamite$ git push --tags
Counting objects: 15, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (15/15), 1.03 KiB | 0 bytes/s, done.
Total 15 (delta 0), reused 0 (delta 0)
remote: =====
remote: REJECTED!
remote: =====
remote: Tags must be semantic version names (e.g. "1.2.3" or "4.5.6-beta").
remote: =====
remote: 1.1 TAG [v1.2] Tag name must match the pattern <(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-
remote: \.([0-9a-zA-Z-]+)*)?> (regex)
remote: =====
remote: REJECTED!
remote: =====
To /tmp/git-shots/srv/website/
! [remote rejected] v1.2 -> v1.2 (pre-receive hook declined)
error: failed to push some refs to '/tmp/git-shots/srv/website/'
/projects/acne-dynamite$ git tag 1.2.0 v1.2
/projects/acne-dynamite$ git tag -d v1.2
Deleted tag 'v1.2' (was a blob)
/projects/acne-dynamite$ git push --tags
Counting objects: 15, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (15/15), 1.03 KiB | 0 bytes/s, done.
Total 15 (delta 0), reused 0 (delta 0)
remote:
To /tmp/git-shots/srv/website/
* [new tag] 1.2.0 -> 1.2.0
/projects/acne-dynamite$
```

Create the tag "v1.2" and push it.

Rejected as its format is not allowed!

Rename it to "1.2.0" (a valid semantic version name), and push again.

Accepted.



Installing commit policies

Testing your branch rule

```
Terminal File Edit View Search Terminal Help
/projects/acme-documentation$ svn cp ^/trunk ^/branches/authenticating-with-google -m "Start writing the Google auth doc"
Committing transaction...
svn: E165001: Commit blocked by pre-commit hook (exit code 1) with output:
=====
REJECTED!
=====
Branch names must contain a Task type issue key (e.g. "DOC-5-my-feature").
-----
1.1 BRANCH [authenticating-with-google] Exactly one issue key must be mentioned in the branch name
1.1 COMMIT [8] on [authenticating-with-google] Exactly one issue key must be mentioned in the branch name
-----
All changes committed to this repository must follow the ACME Documentation Change Control Procedure.
-----
/projects/acme-documentation$ svn cp ^/trunk ^/branches/DOC-6-authenticating-with-google -m "Start writing the Google auth doc"
Committing transaction...
Committed revision 8.
/projects/acme-documentation$
```

Start a new Subversion branch without including the related issue key in its name.

Rejected!

Start again with the link.

Accepted



Installing commit policies

Installing and applying commit policies to repositories

The screenshot shows the Jira Software interface with the 'Commit Policies' page. A 'Hook Script Wizard' dialog box is open, displaying a 'Congratulations!' message. The dialog includes a progress bar, a green checkmark icon, and the text: 'The TOMCAT commit policy is now active in the Git repository. Tip: read the commit policies with Git page. Happy coding!'. At the bottom of the dialog are 'Back' and 'Finish' buttons. The background shows a table of commit policies and an 'Apply to a repository' button for each.

ID	Name
18	Application design guidelines
7	Crypto Exchange Policy
12	Current CLD Sprint
20	EVM - Solidity
9	FREEZE DISABLED
8	Image files only
19	Sounds and music
1	TOMCAT

Commit policy for Apache Tomcat, an open-source server and servlet container.

How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team

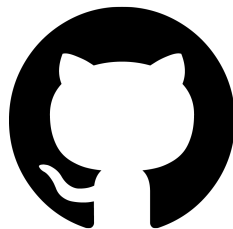


Installing commit policies

Supported Version Control Systems



git



Bitbucket



GitLab



Any custom VCS or environment
via REST API



Installing commit policies

Supported clients & IDE integrations

The screenshot shows the TortoiseGit commit dialog box. The commit message is "XLS-123 Fixed". The output window displays the following text:

```
git.exe push --progress "origin" master:master
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 991 bytes | 0 bytes/s, done.
Total 13 (delta 5), reused 0 (delta 0)
remote: Commit policy is violated.
remote: -----
remote: REJECTED!
remote: -----
remote: Include the keys of your open issues in the beginning part of the commit message: "XLS-123 Fixed"
remote: -----
remote: 1.1 COMMIT [ca785b03cfff] on [master] Commit message "Add a null check ..." must match the pattern <{?s}{?:XLS}-\d+.*
remote: 1.2 COMMIT [ca785b03cfff] on [master] At least one issue key must be mentioned in the commit message "Add a null check ..."
remote: -----
remote: THIS REPOSITORY IS FOR THE XLS PRODUCT CODE.
remote: -----
To https://midori.dyn dns.org:8445/scm/~levente.szabo/main.git
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'https://midori.dyn dns.org:8445/scm/~levente.szabo/main.git'

git did not exit cleanly (exit code 1) (2156 ms @ 2018. 06. 15. 13:43:31)
```

The TortoiseGit logo is visible at the bottom center of the window.

The screenshot shows the TortoiseSVN commit dialog box. The commit message is "Add support for P...". The output window displays the following text:

```
Commit Failed!

Action      Path
Command     Commit to file:///C:/Users/Levente_sz/Google%20Cloud%20Platform%20Projects/google-cloud-clients/google-cloud-biggery/src/main/java/com/google/cloud/bigquery/ExtractJobC
Modified    C:/Users/Levente_sz/Google Cloud Platform Projects/google-cloud-clients/google-cloud-biggery/src/main/java/com/google/cloud/bigquery/ExtractJobC
Sending content
Committing transaction...
Error      Commit failed (details follow):
Error      Commit blocked by pre-commit hook (exit code 1) with output:
Error      -----
Error      REJECTED!
Error      -----
Error      The commit must reference a user story!
Error      -----
Error      1.1 COMMIT [3] on [trunk] Commit message "Add support for P..." must match the
Error      pattern <hello> (glob)
Error      -----
Error      REJECTED!
Error      -----
Error      If you want to break the lock, use the 'Check For Modifications' dialog or the repository browser.
Completed!

The operation failed.
```

The TortoiseSVN logo is visible at the bottom center of the window.



REST API

Integrate with custom VCSs and custom environments

New REST end-points are added to the standard Jira REST API!

For commit verification:

`/commit-policy/{policy-id}/verification`

For listing existing commit policies:

`/commit-policy`

For generating hook scripts:

`/hook-script/{vcs}/{os}/{policy-id}`

– Learn more:

<http://www.midori-global.com/products/better-commit-policy-for-jira/documentation/rest-api>



How commit policies enable end-to-end traceability and faster code reviews

Why do you need commit policies?

How do commit policies work?

Defining commit policies

Installing commit policies

Commit policies in any environment

Introducing commit policies to your team



Introducing commit policies to your team

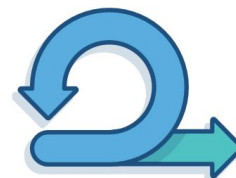
Making your job easier by getting everyone on board



**Don't surprise the team
with commit policies**



**Bring real-life examples to
support your case**



**Encourage using local
commit verification**



Introducing commit policies to your team

Use the Team Playbook by Atlassian

Use the 5 “Whys” Analysis

<https://www.atlassian.com/team-playbook>





Thank you!



Levente Szabo • levente.szabo@midori-global.com



Try our other apps, too (free)!



Better PDF Exporter for Jira

Easy emailing, sharing, archiving, printing for Jira data



Better Excel Exporter for Jira

Full-blown native Excel exports, spreadsheet reports and Business Intelligence for Jira



Better Content Archiving for Confluence

Usage tracking, expiration, review workflow, retention and clean-up for your Confluence pages