# How To Contribute

## Requirements

Oozie is a Java project and it requires:

- Unix environment.
- JDK 1.6.
- Maven 3.0.
- (Optional) An IDE (ie. Eclipse, IntelliJ, Netbeans can be easily used with the IDEs Maven support).
- Be online for the first build to download required plugins and dependencies.

## Getting the Source

Get Oozie's source code from one of the Apache Version Control Systems.

## Development, in Which Branch?

Most of the development is done on "master" and then backported (if required) to release branches.

Development can be done directly in a release branch only if the change is not meaningful on "master".

Oozie committers can create release and development branches. A development branch is commonly used to work on a complex issue(s) that requires several JIRAs and eventually it will be merged back into "master".

Note: Prior to migrating from svn to git, the branch was named "trunk". As a consequence, many developers use "trunk" and "master" interchangeably, and many places, including JIRA, still refer to it as "trunk".

## Making Changes to Oozie

First create an issue in Oozie Issue Tracking System (JIRA) describing the bug, improvement, new feature, task; and discuss if appropriate. It is important that the subject of the issue to be descriptive of the issue, and concise, as it will be used for the RELEASE LOG. All discussions should be done in the Issue Tracking System to make easier for developers to follow or come back in the future.

Oozie development follows a Review Then Commit model.

Work on the issue in your local development environment and when you think it is ready for review or integration upload your patch generated by doing `git diff --no-prefix` to the JIRA issue. The naming convention of the patch should be `OOZIE-<JIRA NUMBER>-001.patch`. For eg: OOZIE-2555-001.patch.

After attaching the patch, click on 'Submit Patch' to change the status of the JIRA to 'Patch Available'. Marking the JIRA as 'Patch Available' it will trigger Oozie **test-patch** Jenkins job that will do an automated analysis of the patch and reported to the JIRA.

Typically, after **test-patch** +1 the patch a developer/committer will do a review.

Refine the patch based on the review discussions. For new patches with changes after review, you can increment the number at the end. For eg: OOZIE-2555-2.patch

Once your patch has been +1ed by Oozie committers the patch can be committed to the Version Control system. If you are a committer, you can commit it yourself. If you are not a committer, one of the committers that +1ed the patch will commit it on your behalf.

If required, backports for releases branches should follow. If the backports are trivial there is no need for a new review.

The corresponding issue in the Issue Tracking and in the Code Review systems must be resolved as 'fixed' by the committer once the patch has been committed to all relevant branches.

**IMPORTANT:** Do not 'Close' the issue, resolve it as 'Fixed'. The issue will be closed by the release manager of the of the target release branch at release time.

If the reviewers consider that it will easier to review the patch to Apache Code Review System, he/she may upload the patch to Apache Code Review System, or ask the contributor of the patch to do so. Once the review is complete in Apache Code Review System, the contributor must upload the final patch to the Apache JIRA where it will be +1'd.

## Patches General Guidelines

### Coding Conventions

Oozie uses Java standard coding conventions with the following changes:

- Indentation is 4 spaces instead of 2 spaces.
- Maximum line lenght is 132 characters instead of 80 characters.
- No **@author** tags.

In addition:

- All files must include the Apache License header unless the file type does not support comments.
- All public classes/methods should have Javadocs. The code should have inline comments when appropriate to clarify things.
- No stale/commented-out code.
- Remove all trailing spaces.
- Don't reformat code that is not part of the patch.
- else/catch/finally start on a new line.

### Test Cases

All patches should include test cases whenever possible. Test cases should have as much coverage as possible (within reasonable boundaries).

### Test Patch

Attach the patch to the corresponding JIRA and mark the JIRA as **patch available**, this will trigger a **test-patch** build that will do an automatic diagnostic of the patch and report it to the JIRA.

You can run **test-patch** locally in trunk, it is available at **bin/test-patch**, run it with not arguments to get the help.

### Documentation

If the patch is adding/modifying functionality that is user facing, it must include documentation.

### Release Log File

The Release Log file must be edited by the committer committing the patch (the patch must not include changes to the Release Log file). This is to avoid unnecessary conflicts.

## Easy Tasks (for newcomers)

Some easy-to-do tasks can be found via this JIRA search result. These tasks are marked with the label 'newbie' and are generally easy tasks to do, that shouldn't require in-depth knowledge of Oozie to attack.