ReleaseManagement

Apache ZooKeeper Release Management Process

This document is currently a work in process and should not be considered authoritative.

This document describes the release management policies used by the ZooKeeper project. As described herein, this policy is not set in stone and may be adjusted by the Release Manager.

Who can make a release?

Technically, anyone can make a release of the source code due to the Apache Software License. However, only members of the Apache ZooKeeper Project (committers) can make a release designated with Apache. Other people must call their release something other than "Apache" unless they obtain written permission from the Apache Software Foundation.

Following our official release policies, we will only accept release binaries from members of the Apache ZooKeeper Project for inclusion on our website. This ensures that our binaries can be supported by members of the project. Other people are free to make binaries, but we will not post them on our website.

Who is in charge of a release?

The release is coordinated by the Release Manager (hereafter, abbreviated as RM). Since this job requires coordination of the development community (and access to subversion), only committers to the project can be RM. However, there is no set RM. Any committer may perform a release at any time. A committer proposes to manage a new release. Once the proposal is approved according to the ZooKeeper Bylaws, the committer becomes the Release Manager for that release. If the Release Manager is starting a new **minor** release, they will create a new **major.minor** branch in subversion and begin collecting changes into that branch to form a release. Although the release manager determines what is included in the release, the release must be approved according to the Bylaws.

Although the Release Manager may change between every fix release in a **major.minor** series of releases, it is desirable that the same Release Manager manage all of the fix releases in a **major.minor** series.

What power does the RM yield?

Regarding what makes it into a release, the RM is the unquestioned authority. No one can contest what makes it into the release. The community will judge the release's quality after it has been issued, but the community can not force the RM to include a feature that they feel uncomfortable adding. Remember that this document is only a guideline to the community and future RMs - each RM may run a release in a different way.

How does an impending release affect development?

It can not. Let's repeat that: an impending release can not affect development of the project. It is the RM's responsibility to identify what changes should make it into the release. The RM may have an intermediate tag, so the RM can merge in or reject changes as they are committed to the repository's HEAD.

Committers may voluntarily refrain from committing patches if they wish to ease the burden on the RM, but they are under no obligation to do so. This is one reason why we recommend that the RMs have plenty of time on their hands - they may have to deal with a rapidly changing target. It's not an easy job.

Release Numbering

ZooKeeper uses a **major.minor** release numbering. A change in the **major** number means that the release includes some substantial changes from the previous version such as the introduction of new protocols, new formats, major new functionality, etc. A **minor** release includes new features and fixes from the main line development branch, referred to as **trunk**. The first release of the **major.minor** version of the code, will be called **major.minor.0**, where **0** designates a **fix** number. Bugs found after the release of **major.minor.0** may need to be incorporated into the **major.minor** series, which will result in a new fix release. Fix releases are meant for bug fixes and should not include new functionality unless it is necessary to fix a bug.

Backward Compatibility

All fix releases to a **major.minor** release should be backwards compatible. Otherwise we do one **major.minor** release backward compatibility: a **major.minor** release of ZooKeeper must be backwards compatible with the previous minor release, **major.(minor-1)**; a new major release, **major.0**, must be backwards compatible with current **minor** release for the previous **major** number.

Here are some examples of backward compatibility:

Old release	New release	Must be compatible
1.1.0	1.1.3	yes
1.1.1	1.1.x	yes
1.1.2	1.2.0	yes
1.1.x	1.2.y	yes

1.1.1	1.3.0	no
1.3.2	2.0.0	yes if 1.3.2 was the current release when 2.0.0 was released

Code Changes

Changes will be done against **trunk**. Patch contributors make their patches with respect to **trunk**. The only exceptions to this would be in the case of a bug fix that was only in a fix release and not in **trunk** or for a backport of a bug fix to a fix release.