Log4Delphi Project Bylaws

Table of contents

1. What is the Log4Delphi Project?

The Log4Delphi Project is a community governed by the principle of meritocracy that strives to meet the Log4Delphi Mission Statement.

2. Meritocracy

The Log4Delphi Project operates in a fashion whereby anybody interested can easily become involved by sending patches or suggestions, participating in the mailing lists and making other contributions.

As soon as a person has made sufficient contributions they are granted the privilege of becoming a part of the development community by being granted access to the code repository, in effect becoming a commiter. Once a committer has made a sufficient number of contributions and proven his commitment to the project, he may be nominated to become a project administrator. This principle is known as meritocracy, the govern of merrit.

This implies that the project is governed by the people who contribute the most.

3. Roles and Responsibilties

The project defines a set of roles with associated rights and responsibilities. These roles govern what tasks an individual may perform within the project.

The roles are: User | Developer | Committer | Administrator

3.1. Users

The most important participants in the project are the people who use our software. They contribute to the project by providing feedback to developers in the form of bug reports and feature requests. Users participate in the community by helping other users on mailing lists and user support forums. The passive users are also known as *lurkers*.

3.2. Developers

All of the volunteers who are contributing time, code, documentation, or resources to the project are known as developers. A developer that makes sustained, welcome contributions to the project may be invited to become a Committer, though the exact timing of such invitations depends on many factors.

3.3. Committers

The project's Committers are responsible for the project's technical management. All committers have write access to the project's source repositories. Committers may cast binding votes on any technical discussion regarding the project.

Committer access is by invitation only and must be approved by lazy consensus of the active Project Administrators. A Committer is considered emeritus by their own declaration or by not contributing in any form to the project for over six months. An emeritus committer may request reinstatement of commit access from the Administrators. Such reinstatement is subject to lazy consensus of active Administrators.

Commit access can be revoked by a unanimous vote of all the active Administrators (except the committer in question if they are also an Administrator).

A committer who makes a sustained contribution to the project may be invited to become a Project Administrator. The form of contribution is not limited to code. It can also include code review, helping out users on the mailing lists, documentation, etc.

3.4. Administrators

Administrators are responsible for the management of the project. Duties include:

- Deciding what is distributed as products of the Log4Delphi Project. In particular all releases and release dates must be approved by the Administrators.
- Maintaining the project's shared resources, including the codebase repository, mailing lists, websites.
- Speaking on behalf of the project.
- Nominating new administrators and committers.
- Maintaining these bylaws and other guidelines of the project.

Membership to the Administrators is by invitation only and must be approved by a lazy consensus of active Administrators. An Administrator is considered "emeritus" by their own declaration or by not contributing in any form to the project for over six months. An emeritus member may request reinstatement to the Administrators. Such reinstatement is subject to lazy consensus of the active Administrators. Membership of the Administrators can be revoked by an unanimous vote of all the active Administrators other than the member in question.

4. Project Management and Collaboration

The project is managed using a collaborative, consensus-based process.

4.1. Communication

Communication is done via mailing lists. These identify "virtual meeting rooms" where conversations happen asynchronously, which is a general requirement for groups that are so geographically distributed to cover all time zones.

In general, asynchronous communication is much more useful because it allows archives to be created and it's more tolerant on the volunteer nature of the community.

4.2. Documentation

Documentation is imperitive as it gives users and new comers to the project vital information. As such the project strives to create complete and high quality documentation for the products.

All documentation is marked up using XML which is then converted to HTML and PDF formats. The documentation is also used for the project's website.

4.3. Decision Making

Within the project, different types of decisions require different forms of approval. For example, the previous section describes several decisions which require "lazy consensus" approval. This section defines how voting is performed, the types of approvals, and which types of decision require which type of approval.

4.3.1. Voting

Decisions regarding the project are made by votes on the developer mailing list. Votes are clearly indicated by subject line starting with [VOTE] or [ADMIN-VOTE]. Voting is carried out by replying to the vote mail. Voting may take one of three flavours:

+1	A positive vote indicating "Yes," "Agree," or "the action should be performed."
0	Abstain, have no opinion. This constitutes a veto. No explanation for veto is required.
-1	A negative vote indicating "No," "Do Not Agree" or "the action should not be performed." It may also be appropriate for a -1 vote to include an alternative course of action.

Table 1: Votes

All participants in the project are encouraged to show their agreement with or against a particular action by voting. For technical decisions, only the votes of active committers are binding. Non binding votes are still useful for those with binding votes to understand the

perception of an action in the wider community. For Administrator decisions, only the votes of Administrators are binding.

4.3.2. Approvals

These are the types of approvals that can be sought. Different actions require different types of approvals

Consensus	For this to pass, all voters with binding votes must vote and there can be no binding vetoes (0). Consensus votes are rarely required due to the impracticality of getting all eligible voters to cast a vote.
Lazy Consensus	Lazy consensus requires at least 3 binding (+1) votes and no binding vetoes.
Lazy Approval	An action with lazy approval is implicitly allowed unless a (-1) vote is received, at which time, depending on the type of action, a lazy consensus approval must be obtained.
Majority	Some actions require a 2/3 majority of active committers or administrators to pass. Such actions typically affect the foundation of the project (e.g. adopting a new codebase to replace an existing product). The higher threshold is designed to ensure such changes are strongly supported. To pass this vote requires at least 2/3 of binding vote holders to vote (+1)

Table 1: Approvals

4.3.3. Vetoes

A valid, binding veto cannot be overruled. If you disagree with a valid veto, you must lobby the person casting the veto to withdraw their veto. If a veto is withdrawn, depending on the situation, a re-vote may ensue.

4.3.4. Actions

This section describes the various actions which are undertaken within the project, the corresponding approval required for that action and those who have binding votes over the action.

Action	Description	Approval	Binding Votes

Code Change	A change made to a codebase of the project and committed by a committer. This includes source code, documentation, website content, etc.	Lazy Approval	Active committers
Release Plan	Defines the timetable and actions for a release.	Lazy Approval	Active committers
Adoption of New Codebase	When the codebase for an existing, released product is to be replaced with an alternative codebase. If such a vote fails to gain approval, the existing code base will continue. This also covers the creation of new sub-projects within the project.	Majority	Active committers
Product Release	When a release of one of the project's products is ready, a vote is required to accept the release as an official release of the project.	Lazy Approval	Active Administrators
New Committer	When a new committer is nominated for the project.	Lazy Consensus	Active Administrators
New Administrator	When a committer is proposed for Administrator	Lazy Consensus	Active Administrators
Committer Removal	When removal of commit privileges is sought.	Consensus	Active Administrators
Administrator Removal	When removal of an Administrator is sought.	Consensus	Active Administrators

Table 1: Actions

4.3.5. Voting Timeframes

Votes are open for a period of 1 week to allow all active voters time to consider the vote. Votes relating to code changes are not subject to a strict timetable but should be made as timely as possible.