# Contributing

All contributors sending pull requests (PRs) must have a Contributor

License Agreement on file as either an

[individual](https://www.cloudfoundry.org/pdfs/CFF\_Individual\_CLA.pdf)

or via their

[employer](https://www.cloudfoundry.org/pdfs/CFF\_Corporate\_CLA.pdf).

All new Github issues and PRs will be reviewed by the Project Management

Committee (PMC) on the

[Weekly Call](https://github.com/openservicebrokerapi/servicebroker/wiki/Weekly-Call)

and will have a PMC member assigned to them who is responsible for following up

with the author and bringing the issue/PR to a resolution.

## Minor Changes

Minor change proposals to the specification, changes such as editorial bugs

or enhancements that do not modify the semantics of the specification or

syntax of the API, can be suggested via a Github issue or pull request (PR).

If there is a need for some discussion around how best to address the concern,

then opening an issue prior to doing the work to develop a PR would be best.

These minor issues do not need to follow the [Major Changes](#major-changes)

process described below, rather the issue should be used to come to a consensus

around a PR that will eventually be submitted. If the proposed change is

uncontroversial (e.g. a typo) then a PR can be submitted directly without

opening an issue. Either way, once a PR is submitted it will be reviewed per the

[process](#prissue-review-process) described below.

### Major Changes

Major changes will be tracked through a number of stages before becoming part of

the specification. The working group will use a set of pre-defined Github labels

to highlight the current stage of each proposal. Note that these labels will not be

used for [minor changes](#minor-changes).

New design proposals to the API spec should be submitted by opening a

Github issue with a link to a Google Doc containing the proposal. Proposals

should focus primarily on motivation, problem statement, and use cases before

suggesting a solution or change. Collaboration on the design, fleshing out of

use cases, etc can occur as comment discussions in the Google Doc, as well

as on our weekly calls. Once the working group have agreed to focus on a

proposal, it will move into the first stage, as described below.

The stages that a proposal will go through are as follows:

- `1 - reviewing proposal`:

The API working group is actively reviewing a proposal that has been submitted

as a Github issue with the aim of validating both the problem statements and

any proposed solutions. Critical information such as new API endpoints and

response codes should be reviewed at this stage. For changes largely impacting

the Open Service Broker API actors (Platforms, Service Authors, etc), it is

recommended to solicit feedback from these actors and leave enough time (say

2 weeks) for feedback to be provided, and for the potentially received

objections/suggestions to be handled.

Once the design has been finalized in the Google Doc, the proposed set of

changes to the specification should be made available for review. This could

be done by pointing to a branch in a Github repo with the proposed edits or by

creating a PR with the `do not merge` label assigned to it. Reviewers, or

potential implementers of the feature, can then easily see the exact changes

being proposed so that they can comment on them. If the proposal adds or

changes an object model or resource endpoint definition, it is expected that

the PR will also include the necessary updates to the

[Open API document](openapi.yaml) and [Swagger document](swagger.yaml).

Before moving to the next stage, consensus must be agreed on the weekly call

that the proposal is ready to be validated.

- `2 - validating through implementation`:

One or more platforms are actively working on the proposal with the aim of

providing feedback on the end-to-end implementation of the proposed solution.

The related issue and/or PR should be kept updated with any activity or

feedback that is collected whilst the change is being implemented. It is

expected that during this implementation phase there will be changes made to

the design to accurately represent the current status of the proposal.

Before moving to the next stage, this feedback must have been presented on the

weekly call and any issues found during this stage must have been discussed

with the group and the issue and/or PR should have been updated accordingly.

- `3 - reviewing PR`:

Feedback has been received on the implementation of the proposed solution and

a pull request has been created containing the validated specification

changes. By this point, the API interactions should be well understood and

there should be no technical surprises; we expect the only discussion

necessary on PRs to be for wordsmithing and formatting. The PR will then be

reviewed as described in the [process](#prissue-review-process) below.

Note that not all issues will need to go through the

`validation through implementation` stage, and proposals can move back to a

previous label at any time.

## PR/Issue Review Process

All proposals (either pull requests or issues) will follow the process

described below:

- All proposals must be available for review for no less than one week before

they are approved. This will provide each dedicated committer enough time

to review the proposal without unnecessarily delaying forward progress.

Any non-trivial edit to the proposal (e.g. edits larger than typos) will

reset the clock.

- Any dedicated committer can veto (via a "NOT LGTM" comment in the proposal).

The comment must include the reasoning behind the veto.

It is then expected that the dedicated committers will discuss the concerns

and determine the next step for proposal - either close/reject the proposal

or address the concerns raised such that the "NOT LGTM" can be rescinded.

- A proposal requires at least 3 "LGTM" comments from at least 3 different

organizations to be approved.

- Once a "design change" issue is approved, it will be tagged with an

"proposal finalized" label. This indicates that it is ready to be

implemented by a platform developer, see the [process](#contributing) above.

- Once a pull request is approved, it will be merged into the 'master' branch

and labels should be removed any related issues and PRs as necessary.

- If the pull request is significant enough to warrant noting a difference in

compatibility between platforms, the feature should be noted in the table for

[Platform Compatibility for OSBAPI](compatibility.md).

## Release Process

Any member of the PMC can request a specific SHA on master (the

\*\*Release RC SHA\*\*) is ready to be released into a new version of the spec. They

will do this by creating a new PR with the title of the proposed release. For

example, \*\*"Release Proposal: v$major.$minor"\*\*.

### Prepare a PR

1. In a fork, create a new branch called "v$major.$minor-rc" from the

\*\*Release RC SHA\*\*.

2. Create a new commit titled `prepare release` with the following changes:

\* Update [release-notes.md](release-notes.md) detailing the changes that are

to be released in this version. Include a versioned link to the new branch's

version of the spec.

\* Update [README.md](README.md) with an updated \_Latest Release\_ subheading

and links to the latest version of the documents (`spec.md`, `profile.md`,

`openapi.yaml`, etc).

\* Update [spec.md](spec.md) with an updated \_Changes Since v...\_ section (and

link from table of contents) containing a copy of the relevant release notes,

and with any references to the previous version of the specification (i.e. the

`X-Broker-API-Version` headers) updated. Do not update the header

`Open Service Broker API (master - might contain changes that are not yet released)`

(this will be done if and when the release proposal is approved).

\* Update [diagram.md](diagram.md) to show the new version number in the

header and link to the correct version of the Google Drawing from the

[OSBAPI Google Drive Folder](https://drive.google.com/drive/u/0/folders/0B427Up4C9IE0VmM0ZlhHTG1Rc0E).

3. Open a new pull request titled \*\*Release Proposal: v$major.$minor\*\* from the

branch of the fork to the master branch of the repository, with the following

description:

```

Announcing a new release candidate as described in the

[Release Process](https://github.com/openservicebrokerapi/servicebroker/blob/master/CONTRIBUTING.md#release-process).

\*\*Release version\*\*: <v$major.$minor>

\*\*Release RC SHA\*\*: <Release RC SHA>

\*\*Target release date\*\*: <YYYY-MM-DD> (one week after the next weekly call)

The one-week

[Review Process](https://github.com/openservicebrokerapi/servicebroker/blob/master/CONTRIBUTING.md#review-process)

will be triggered on the next weekly call.

```

4. Announce the release proposal on the next weekly call and notify the mailing

list of the proposal, triggering the start of the

[Review Process](#review-process) as outlined below.

### Review Process

- All release proposals must be available for review for no less than one

week before they are approved. This will provide each dedicated committer

enough time to review the release proposal without unnecessarily delaying

forward progress.

- Any dedicated committer can veto (via a "NOT LGTM" comment in the proposal).

The comment must include the reasoning behind the veto. It is then expected

that the dedicated committers will discuss the concerns and determine the next

steps for release proposal. The submitter should either close/reject the

proposal or address the concerns raised such that the "NOT LGTM" can be

rescinded.

- A release proposal requires at least 3 "LGTM" comments from at least

3 different organizations to be approved.

### Once Approved

Once the release is approved, the following actions should be taken by

any PMC member:

1. Merge the release proposal PR into the master branch of the repository. There

should not be any conflicts as the text in the files that have changed should

only be changed during this release process.

1. Checkout the \*\*Release RC SHA\*\* (either to a local branch or in 'detached HEAD'

state). This is done in order to ensure that changes that were merged into

master after the release candidate was created are not included in the

release.

1. Cherry pick the `prepare release` commit.

1. Update [spec.md](spec.md), [profile](profile.md),

[openapi.yaml](openapi.yaml) and [swagger.yaml](swagger.yaml) to include the

version of the release `v$major.$minor`.

1. Create a new commit with these changes called `finalise release`.

1. Tag the commit with the name `v$major.$minor`.

1. Push the tag to the repository with `git push origin v$major.$minor`.

1. Notify the mailing list of the new release.

1. Update the [Roadmap & Release Planning](https://github.com/openservicebrokerapi/servicebroker/projects/1)

project.

1. The PMC will create a blog post for the new release.