# Contributing | Firebase Admin Node.js SDK

Thank you for contributing to the Firebase community!

- [Have a usage question?](#question)

- [Think you found a bug?](#issue)

- [Have a feature request?](#feature)

- [Want to submit a pull request?](#submit)

- [Need to get set up locally?](#local-setup)

## <a name="question"></a>Have a usage question?

We get lots of those and we love helping you, but GitHub is not the best place for them. Issues

which just ask about usage will be closed. Here are some resources to get help:

- Go through the [guides](https://firebase.google.com/docs/admin/setup/)

- Read the full [API reference](https://firebase.google.com/docs/reference/admin/node/)

If the official documentation doesn't help, try asking a question on the

[Firebase Google Group](https://groups.google.com/forum/#!forum/firebase-talk/) or one of our

other [official support channels](https://firebase.google.com/support/).

\*\*Please avoid double posting across multiple channels!\*\*

## <a name="issue"></a>Think you found a bug?

Yeah, we're definitely not perfect!

Search through [old issues](https://github.com/firebase/firebase-admin-node/issues) before

submitting a new issue as your question may have already been answered.

If your issue appears to be a bug, and hasn't been reported,

[open a new issue](https://github.com/firebase/firebase-admin-node/issues/new). Please use the

provided bug report template and include a minimal repro.

If you are up to the challenge, [submit a pull request](#submit) with a fix!

## <a name="feature"></a>Have a feature request?

Great, we love hearing how we can improve our products! Share you idea through our

[feature request support channel](https://firebase.google.com/support/contact/bugs-features/).

## <a name="submit"></a>Want to submit a pull request?

Sweet, we'd love to accept your contribution!

[Open a new pull request](https://github.com/firebase/firebase-admin-node/pull/new/master) and fill

out the provided template.

\*\*If you want to implement a new feature, please open an issue with a proposal first so that we can

figure out if the feature makes sense and how it will work.\*\*

Make sure your changes pass our linter and the tests all pass on your local machine. We've hooked

up this repo with continuous integration to double check those things for you.

Most non-trivial changes should include some extra test coverage. If you aren't sure how to add

tests, feel free to submit regardless and ask us for some advice.

Finally, you will need to sign our

[Contributor License Agreement](https://cla.developers.google.com/about/google-individual),

and go through our code review process before we can accept your pull request.

### Contributor License Agreement

Contributions to this project must be accompanied by a Contributor License

Agreement. You (or your employer) retain the copyright to your contribution.

This simply gives us permission to use and redistribute your contributions as

part of the project. Head over to <https://cla.developers.google.com/> to see

your current agreements on file or to sign a new one.

You generally only need to submit a CLA once, so if you've already submitted one

(even if it was for a different project), you probably don't need to do it

again.

### Code reviews

All submissions, including submissions by project members, require review. We

use GitHub pull requests for this purpose. Consult

[GitHub Help](https://help.github.com/articles/about-pull-requests/) for more

information on using pull requests.

## <a name="local-setup"></a>Need to get set up locally?

### Prerequisites

1. Node.js 10.10.0 or higher.

2. NPM 5 or higher (NPM 6 recommended).

3. Google Cloud SDK ([`gcloud`](https://cloud.google.com/sdk/downloads) utility)

### Initial Setup

Run the following commands from the command line to get your local environment set up:

```bash

$ git clone https://github.com/firebase/firebase-admin-node.git

$ cd firebase-admin-node # go to the firebase-admin-node directory

$ npm install # install local npm build / test dependencies

```

In order to run the tests, you also need to authorize the `gcloud` utility with

Google application default credentials:

```bash

$ gcloud beta auth application-default login

```

### Running the Linter

Source files are written in [TypeScript](https://www.typescriptlang.org/) and linted using

[TSLint](https://palantir.github.io/tslint/). Run the following command to kick off the linter:

```bash

$ npm run lint

```

### Running Tests

There are two test suites: unit and integration. The unit test suite is intended to be run during

development, and the integration test suite is intended to be run before packaging up release

candidates.

To run the unit test suite:

```bash

$ npm test # Lint and run unit test suite

```

If you wish to skip the linter, and only run the unit tests:

```bash

$ npm run test:unit

```

The integration tests run against an actual Firebase project. Create a new

project in the [Firebase Console](https://console.firebase.google.com), if you

do not already have one suitable for running the tests against. Then obtain the

following credentials from the project:

1. \*Service account certificate\*: This can be downloaded as a JSON file from

the "Settings > Service Accounts" tab of the Firebase console. Copy the

file into the repo so it's available at `test/resources/key.json`.

2. \*Web API key\*: This is displayed in the "Settings > General" tab of the

console. Copy it and save to a new text file at `test/resources/apikey.txt`.

Then set up your Firebase/GCP project as follows:

1. Enable Firestore: Go to the Firebase Console, and select "Database" from

the "Develop" menu. Click on the "Create database" button. You may choose

to set up Firestore either in the locked mode or in the test mode.

2. Enable password auth: Select "Authentication" from the "Develop" menu in

Firebase Console. Select the "Sign-in method" tab, and enable the

"Email/Password" sign-in method, including the Email link (passwordless

sign-in) option.

3. Enable the Firebase ML API: Go to the

[Google Developers Console](

https://console.developers.google.com/apis/api/firebaseml.googleapis.com/overview)

and make sure your project is selected. If the API is not already enabled, click Enable.

4. Enable the IAM API: Go to the

[Google Cloud Platform Console](https://console.cloud.google.com) and make

sure your Firebase/GCP project is selected. Select "APIs & Services >

Dashboard" from the main menu, and click the "ENABLE APIS AND SERVICES"

button. Search for and enable the "Identity and Access Management (IAM)

API".

5. Grant your service account the 'Firebase Authentication Admin' role. This is

required to ensure that exported user records contain the password hashes of

the user accounts:

1. Go to [Google Cloud Platform Console / IAM & admin](https://console.cloud.google.com/iam-admin).

2. Find your service account in the list, and click the 'pencil' icon to edit it's permissions.

3. Click 'ADD ANOTHER ROLE' and choose 'Firebase Authentication Admin'.

4. Click 'SAVE'.

Finally, to run the integration test suite:

```bash

$ npm run integration # Build and run integration test suite

```

By default the integration test suite does not modify the Firebase security rules for the

Realtime Database. If you want to force update the rules, so that the relevant Database

integration tests can pass, launch the tests as follows:

```bash

$ npm run test:integration -- --updateRules

```

The integration test suite skips the multi-tenancy Auth tests by default.

If you want to run these tests, an

[Identity Platform](https://cloud.google.com/identity-platform/) project with multi-tenancy

[enabled](https://cloud.google.com/identity-platform/docs/multi-tenancy-quickstart#enabling\_multi-tenancy)

will be required.

An existing Firebase project can be upgraded to an Identity Platform project without

losing any functionality via the

[Identity Platform Marketplace Page](https://console.cloud.google.com/customer-identity).

Note that charges may be incurred for active users beyond the Identity Platform free tier.

The integration tests can be launched with these tests enabled as follows:

```bash

$ npm run test:integration -- --testMultiTenancy

```

### Repo Organization

Here are some highlights of the directory structure and notable source files

\* `src/` - Source directory, written in TypeScript.

\* `auth/` - Auth source files, including the user management, credential, and token generator

APIs.

\* `database/` - Database source files, imported as a minified JavaScript file from a separate

repo.

\* `messaging/` - Messaging source files, including the FCM send APIs.

\* `utils/` - Utilities for doing things such as sending requests, handling errors, and validating

inputs.

\* `index.ts` - Main SDK entry point which registers the Firebase services.

\* `index.d.ts` - Hand-crafted TypeScript declaration file.

\* `firebase-app.ts` - `FirebaseApp` implementation.

\* `firebase-namespace.ts` - `FirebaseNamespace` implementation.

\* `default-namespace.ts` - Exports a `FirebaseNamespace` instance which acts as the top-level SDK

export.

\* `lib/` - Output directory for all compiled files.

\* `test/` - Unit and integration test suites.

\* `unit/` - Unit test suite written in Mocha which extensively tests the source code using mocks

for all network requests.

\* `index.spec.js` - Main unit test entry point which imports the tests to be run.

\* `utils.js` - Testing utilities.

\* `integration/` - Integration test suite which actually hits live Firebase services.

\* `resources/` - Provides mocks for several variables as well as mock service account keys.

\* `.github/` - Contribution instructions as well as issue and pull request templates.

\* `createReleaseTarball.sh` - Generates a new release tarball and ensures it passes all tests.

\* `verifyReleaseTarball.sh` - Installs and validates the release tarballs created by

`createReleaseTarball.sh`.

\* `gulpfile.js` - Defines the `gulp` tasks necessary for building release artifacts.

\* `package-lock.json` - A snapshot of the dependency tree for development and CI purposes.

\* `tslint.json` - TypeScript linting rules.

\* `tsconfig.json` - TypeScript configuration options.

\* `tsconfig-lint.json` - TypeScript configuration options for the linter. This simply widens

the scope of `tsconfig.json` by including some test source files in the project.