Contributing integrations back

What integrators can give back to Exponential and why it pays - fixes found while integrating, generic connectors as public extensions, documentation of an integration pattern - how to separate generic from customer…

Source file: `doc/eng-us/help/integrators/contributing-integrations-back.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/integrators/contributing-integrations-back.md

This page is for integrators and integration companies. Integration projects find bugs at the edges of a system, build connectors that other customers need too, and learn patterns worth writing down. Giving those back makes the next project cheaper for you and for everyone, and it is how Exponential grows. The general guide is Contribute extensions back.

Contents

1. What to give back
2. Separate generic from customer code
3. Licensing points
4. From private repository to public
5. What it brings you

What to give back

Contribution	Example	Where it goes
A kernel or extension fix found while integrating	A datatype's `fromString()` stored an invalid value; a REST route answered the wrong status	A pull request to the repository on https://github.com/se7enxweb (Contributing back with pull requests)
A bug report with a reproduction	A setting that does not do what its comment says	An issue in the repository (Writing good issues)
A generic connector	A mail transport for a provider's API, a login handler for an identity provider, a search engine plugin	A public extension of its own
An extension point	A filter or event the kernel should announce so integrations need no override	A pull request with the event and its documentation
Documentation	A pattern, a pitfall, a recipe	A pull request to the documentation (Contributing docs and translations)
A change to behavior that is deliberate but awkward for your integration	A default that suits most sites but not yours	An issue first; it may be a negative feature others depend on, so justify it clearly, expect several review rounds and propose an option

Separate generic from customer code

Design the integration as two extensions from the start (Building an integration as an extension):

- **The connector**: the client of the other system's API, its error handling, the mail transport or login handler. Configured by settings, with no customer names, node ids, URLs or credentials in the code.
- **The customer extension**: the mapping, the classes, the schedule, the customer's settings.

Check the connector before publishing:

☐ No customer names, domains, node ids, internal URLs or test data from the customer.

☐ No credentials in code, settings files, tests or the Git history (search the history, not only the last commit).

☐ Settings documented, with safe defaults.

☐ Tests that run without the other system.

☐ A README with the purpose, requirements, installation and an example configuration.

Licensing points

The Exponential kernel is licensed under the GNU General Public License v2 (or any later version). An extension that runs inside Exponential and is distributed to others must respect that license; choosing GPL v2 or later for a public extension keeps it compatible. Using an extension only yourself, or for a customer who receives it as part of a project, has different obligations than publishing it. Read Contribute extensions back and Licensing for partners, and agree in the contract who owns what (Statements of work). This is not legal advice.

From private repository to public

1. Agree with the customer, in writing, that the generic connector may be published.
2. Move the connector into its own repository with a clean history (or start a new one from the cleaned code).
3. Add `LICENSE`, `README.md`, `CHANGELOG.md`, `composer.json`, `ezinfo.php` and `extension.xml` with the same version (Releasing).
4. Publish a first release; tags are permanent, so check before tagging (Tags are permanent).
5. Register the package on Packagist and announce it (Composer and Packagist).
6. Let the customer extension require the published connector.

What it brings you

- Fixes upstream mean no private patches to carry through every upgrade.
- A public connector is reviewed and improved by others, and shows your company's competence to prospects.
- For Exponential Partners, community work counts toward the points of the partner program (Earning points through community work).

Related pages

- Contribute extensions back
- Contributing extensions back as a partner
- Contributing code back
- Part 5 of the extension book: extend, fork, contribute
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.