

TIDELIFT

REFERENCE ARCHITECTURE

JUNE 2022

CONTENTS

Executive summary	3
The Tidelift Subscription	3
Open source management tools	3
Tidelift catalog	4
Custom catalogs	4
Out-of-box functionality	5
Onboarding applications and using the open source management tools	6
Software bills of materials (SBOMs)	6
Organizing applications	7
New application development (default)	7
Production applications	7
Groups	8
Developers	8
Tidelift catalog	9
Getting data from the Tidelift catalog	9
Requesting new packages be reviewed and added to the Tidelift catalog	9
Custom catalogs	9
Catalog administration	10
Single catalog strategy	10
Multiple catalog strategy	10
Catalog administrators & catalog users	11
Tasks	11
Creating exceptions	12
Standards administration	12
Enabling standards	13
Licensing standard	13
Enforcing license compliance	14
Security vulnerability standard	14
Maintenance standard	15
CI flow	16
Starting your journey	16
Implementing the CLI	16
Enforcing your standards	16
Regression	17
Integrations	18
Artifactory integration	18

EXECUTIVE SUMMARY

The Tidelift Subscription provides a proven approach for managing the health and safety of the open source software supply chain and also streamlines the development process by removing obstacles that slow down developers. This document provides a technical explanation of all the functionality included in the Tidelift Subscription as well as a deep dive into how to integrate Tidelift into your organization’s software development lifecycle.

THE TIDELIFT SUBSCRIPTION

Before getting into deeper technical details, we’d like to introduce a few definitions that will make it easy to digest the content that follows. The Tidelift Subscription consists of three components as defined below:

Open source management tools provide visibility into the open source components being used within a subscriber’s applications, including the ability to continuously inventory application components and to create up-to-date **software bills of materials** (SBOMs) for all applications. An SBOM includes insights such as:

- Release and license information
- **Dependency chain** and provenance information
- Usage type (runtime or dev) and approval status
- **Security vulnerabilities** and **licensing issues**
- Dependency information with the ability to review and address issues

subscription-web-app bill of materials

Generated from these manifests: package-lock.json, package.json
subscription-web-app contains 1086 package releases.

Q Search for packages Filter package types... REQUEST ALL... CSV EXPORT


Name ↑	Version	License	Source	Type
abbrev	1.1.1 (package-lock.json)	ISC	transitive see dependency chain	development
accepts	1.3.5 (package-lock.json) 1.3.7 (package-lock.json)	MIT MIT	transitive see dependency chain	development development
acorn	6.4.1 (package-lock.json)	MIT	transitive see dependency chain	development
acorn-dynamic-import	4.0.0 (package-lock.json)	MIT	transitive see dependency chain	development
adm-zip	0.4.13 (package-lock.json)	MIT	transitive see dependency chain	development
after	0.8.2 (package-lock.json)	MIT	transitive see dependency chain	development
agent-base	4.2.1 (package-lock.json)	MIT	transitive see dependency chain	runtime
agentkeepalive	3.5.2 (package-lock.json)	MIT	transitive see dependency chain	runtime

The **Tidelift catalog** is a repository of open source components evaluated by Tidelift's team of security and licensing experts along with our network of partnered maintainers. **The Tidelift catalog** contains package release recommendations, security vulnerability recommendations, and verified license data for all evaluated packages and gives organizations the information they need to improve their application health.


ACTIVITY FEED **STANDARDS** **PACKAGES (5,017)** **ABOUT**

The Tidelift catalog is the source for security vulnerability and licensing information for all Tidelift-evaluated packages and releases. It provides a definitive approve or deny recommendation from Tidelift as to whether a particular package release should be used.


Friday, 28 Jan 2022

 **JOAN LIU DENIED 1 RELEASE** · 4:00 pm
 maven/org.apache.xmlgraphics:xmlgraphics-commons was affected by CVE-2020-11988
Denied 2.4
Notes: "Users should upgrade to 2.6 or later"



Thursday, 27 Jan 2022

 **LUIS VILLA UPDATED A PACKAGE'S METADATA** · 5:11 pm
 maven/org.jgrapht:jgrapht-core's license was set to LGPL-2.1 OR EPL-2.0 with the following note:

Custom catalogs provide a way for an organization to define and codify its own set of open source standards and then create, track, and manage repositories of pre-approved open source components. Using **custom catalogs**, organizations can track non-public, internally-developed (inner-source) packages included in applications as well.



Organization: Pennington Education

[Pennington Education](#) > [Catalogs](#) > [Overview](#)

Pennington Education catalog IMPORT PACKAGES →

This is a new catalog for projects at Pennington Education to use. Get started by adding open source packages to the catalog and deciding which standards to enforce. You can reference this catalog in Tidelift CLI using `~catalog default`.

Catalog health

!

NEEDS ATTENTION

A security violation task for `npm/qs` has been open for 5 days ago

[VIEW TASK](#)

Project health

PROJECTS

7

There are 7 projects aligning to this catalog.

0 → 100% alignment
0 projects are 100% aligned to the catalog guidance

7 → need attention
7 projects are not in alignment

PROJECT ALIGNMENT (7 PROJECTS)

- `layer_sample_ruby_project` (0%, a month ago)
- `subscription-web-app` (23.94%, 18 hours ago)
- `monitor_backend_sample_python_project` (66.67%, a month ago)
- `maven_spring` (81.54%, 22 days ago)
- `billing_system` (84.13%, 20 days ago)
- `control_proxy_sample_java_project` (84.38%, 22

VERSION SPREAD ACROSS PROJECTS


`net.minidev:json-smart (maven)` 6 approved versions

`strip-ansi (npm)` 4 approved versions

`string-width (npm)` 4 approved versions

`serialize-javascript (npm)` 4 approved versions

`yallist (npm)` 3 approved versions



REFERENCE ARCHITECTURE

4

OUT-OF-BOX FUNCTIONALITY

Tidelift is able to deliver immediate value and information on open source packages even before your organization begins defining open source usage and management standards or onboarding development teams. Using Tidelift's package search functionality you can access package metadata for packages already in use or being considered for use. The metadata contains information such as:

- Number of package contributors
- Release history
- License and license format
- Known security issues and vulnerabilities

This information helps paint a holistic picture about package health, security history and practices, and it is crucial in evaluating whether a package should be approved for use or not.

TIDELIFT

Pillow (pypi)

Python Imaging Library (Fork).
342 contributors

Tidelift facilitates communication with the maintainers of this package.
[Send a secure message to the team](#)

CATALOGS
Overview
Tasks
Packages
Standards
Activity feed
Roles
Reports

PROJECTS

SETTINGS

RELEASES | **PROJECT USAGE** | **VULNERABILITIES** | **PACKAGE INFO**

OVERVIEW

License of latest release: HPND (SPDX)

License verified in a Tidelift-managed catalog
[Manually override license](#)

- Recent commit activity
- Semantic versioning versioning scheme
- Coordinated disclosure policy

OTHER DATA

Dependencies: 16 packages

[Homepage](#)
[Pillow on GitHub](#)
[Pillow on pypi](#)

CONTRIBUTORS (342)

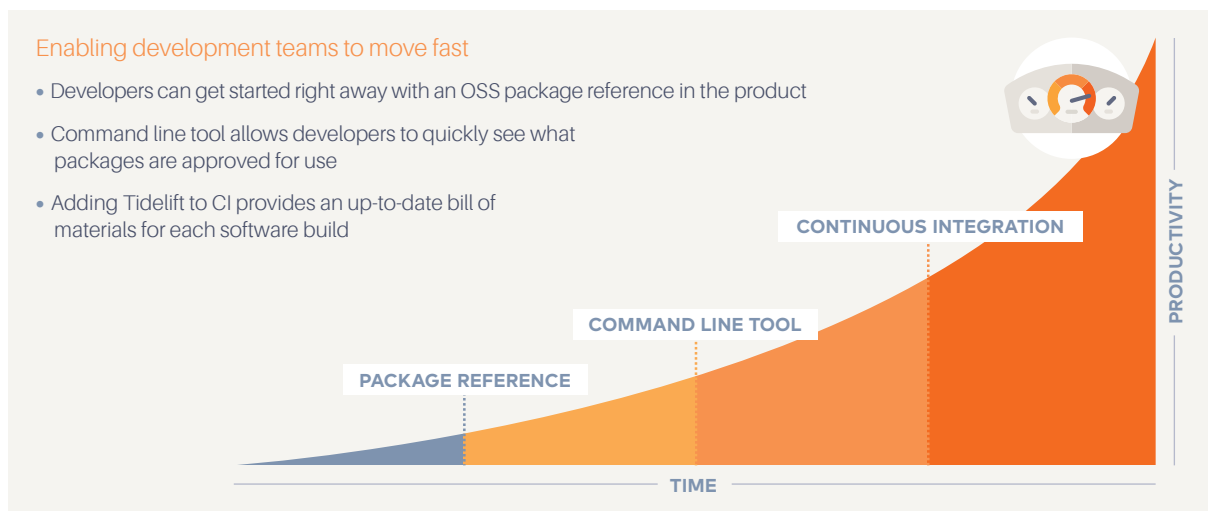
+330 contributors

VERSION GUIDANCE

Latest: 9.1.0 (1 Apr 2022)
Recommended: 9.1.0 (1 Apr 2022)

87 total releases since 2010.
The last release was 12 days ago.

9.1 is the current recommended release stream (latest version released on 1 Apr 2022)
9.1.0 is the current recommended version on 9.1.* and is also the latest



ONBOARDING APPLICATIONS AND USING THE OPEN SOURCE MANAGEMENT TOOLS

Software bills of materials (SBOMs)

The first step for organizations is to make sure they have a clear understanding of the open source components already in use, and then to make informed decisions on which components to use moving forward. The best approach for gathering the data required for this analysis is through creating an SBOM. Tidelift stores individual application SBOM data through its project functionality, where projects and applications have a one-to-one relationship.

project bill of materials

dependencyci contains 247 package releases.

Filter package types... ADD TO CATALOG EXPORT AS CSV

Package	Version	License	Source	Type
crack current release	0.4.3	MIT	transitive	runtime
crass current release	1.0.6	MIT	transitive	runtime
dalli current release	2.7.10 >= 0	MIT MIT	direct	runtime runtime
database_cleaner current release	1.7.0 >= 0	MIT MIT	direct	runtime runtime
deb_control current release	0.0.1	MIT	transitive	runtime
debug_inspector current release	0.0.3	MIT	transitive	runtime
declarative current release	0.0.20	MIT	transitive	runtime
declarative-option current release	0.1.0	MIT	transitive	runtime
deep_merge current release	1.2.1	MIT	transitive	runtime
diff-lcs current release	1.4.4	MIT AND Artistic-2.0 AND GPL-2.0+	transitive	runtime

With Tidelift's CLI, you can create itemized lists for your applications direct and transitive package dependencies which can be exported in .CSV, SPDX, and CycloneDX formats. These exported formats make it easier to integrate into any internal processes that require an SBOM. Once you have created SBOMs for all of your applications, you can import the complete list of packages to your own custom catalog. This gives you a continuously updated view of the open source in use in your applications.

Building a custom catalog from these SBOMs can make onboarding more systematic and targeted across your application development teams. While bringing all your open source into Tidelift at once can sound appealing, this approach is, more often than not, overwhelming. We instead recommend bringing in sets of applications to make the process easier to manage which in turn allows you to become familiar with Tidelift workflows.

```
Alignment score: 98%

1161 package releases identified
1146 package releases approved for use ██████████
15 package releases not approved █

Your alignment score shows what percentage (%) of package releases in your project have been approved.

Denied Releases
-----
npm yargs-parser 11.1.1
  notes: Unfortunately this has a known security vulnerabilitiy.
npm glob-parent 3.1.0
npm ws 6.2.1
npm jszip 3.5.0

Requested releases
-----
npm ansi-regex 4.1.0
npm ansi-regex 3.0.0
npm ansi-regex 5.0.0
npm glob-parent 5.1.1
npm ssri 6.0.1
npm ws 3.3.3
npm hosted-git-info 2.8.8
npm minimist 0.0.10
npm postcss 7.0.35
```

Organizing applications

Quick wins and early successes are a crucial part of adopting new technologies. Generally, we find that the older an application is the more technical debt it will contain in the form of deprecated packages, security disclosure, and out-of-date packages. Categorizing the applications that you plan to bring into Tidelift and putting them into lifecycle buckets is a useful way to decide which order to onboard applications. New development, ongoing development, legacy development, maintenance, and sunset are examples of buckets you can start assigning to your applications.

New application development (default)

Starting with your newest development projects helps leverage Tidelift recommendations as early as possible and can help you stay current with the appropriate software versions available. Newer development paired with the introduction of DevOps practices results in faster interactions and quicker deployments. At the end of the day, it is easier to keep a newer package up-to-date than it is to bring a legacy application’s dependencies to the most current package versions.

With new applications we expect to see fewer, if any, maintenance standard violations. When reviewing standard violations for a new application, maintenance tasks can alert you to issues related to open source components being pulled into current development. This process helps your teams course correct before the application gets further into its development cycle.

After adding your most current and active development projects to Tidelift, move onto the next bucket, working your way towards the oldest applications. Even if you decide to bring in applications by the development team, use the same process of newest to oldest as you create your Tidelift projects.

Production applications

Not all organizations will have the same amount of new applications to use with Tidelift. The reality is open source packages have been in use for decades now and many of our customer applications are just as old. To address these older production applications, Tidelift can apply the same standard across an organization’s open source centrally—an advantage from a policy management point-of-view. With older applications, the maintenance standards can be

used to provide insights into areas of improvement that need to be considered, in addition to the provided security and licensing standards.

Groups

Using Tidelift groups provides a way to categorize projects and users for enhanced sorting, filtering, and reporting purposes. A group can represent:

- Development teams
- Application tiers
- Business units
- Cloud deployment regions
- Or a combination of the categories above

A group can have multiple projects and users assigned to it. Projects and users can also be assigned to multiple groups allowing for flexible group configurations. Groups have no hierarchy or inheritance at this time in Tidelift. It is still recommended to think about what groups need to be created in a logical way that fits your organization. Arbitrarily creating groups can become difficult to administer and maintain over time.

Developers

After an application has been added to a Tidelift project, it is important that developers begin to use the approved packages in a catalog and discontinue using packages that are denied. A catalog administrator will approve or deny packages based on the catalog's policy which is composed of catalog standards.

The primary way that developers work with Tidelift is through the Tidelift command line interface (CLI). The following commands should be regularly used during a development:

- ***tidelift alignment***: Get a bill of materials and check if package releases are approved in your organization's custom catalog
- ***tidelift release list***: Display releases for one of your catalogs
- ***tidelift releases lookup***: Display approved releases for a package in your catalog
- ***tidelift request***: Request a new package be approved and added to your organization's catalog

TIDELIFT CATALOG

There are two key ways for users to interact with the Tidelift catalog:

- Getting data from the Tidelift catalog
- Requesting review of new packages

Getting data from the Tidelift catalog

Tidelift catalog releases and data can be browsed directly from tidelift.com/catalogs.

All custom catalogs receive security vulnerability recommendations and license data from the Tidelift catalog. You'll automatically see these recommendations when resolving security or licensing issues.

Requesting new packages be reviewed and added to the Tidelift catalog

We are constantly growing the list of packages that are reviewed and managed as part of the Tidelift catalog. We aim to proactively add new packages that customers are using to the catalog so we can deliver better recommendations and data to our customers. If you would like to check with us on the progress of adding packages that you use to the catalog, send an email to support@tidelift.com.

Only publicly-available packages can be included in the Tidelift catalog. We have a screening mechanism in place to prevent any organization's internal packages unintentionally ending up in this public catalog.

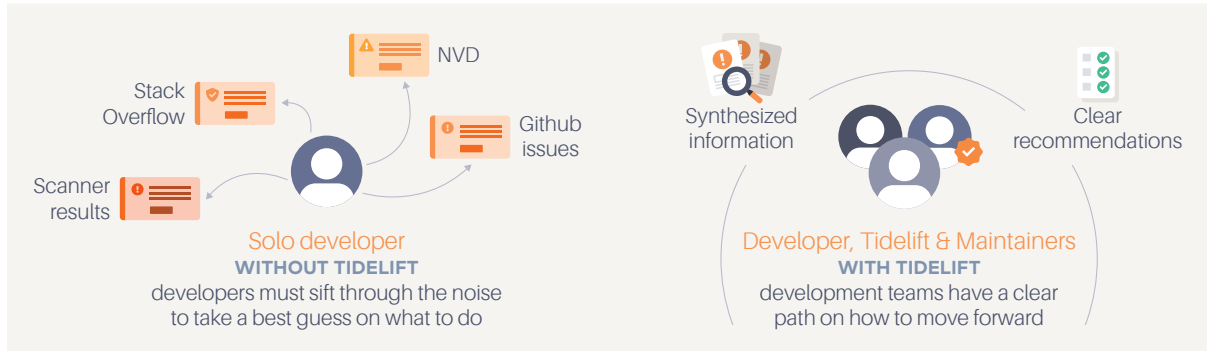
CUSTOM CATALOGS

Custom catalogs give organizations the ability to track only the packages being used by its developers. They include all the packages that are approved-for-use at your organization, improving organizational alignment and developer experience when using open source. Custom catalogs also provide:

- **Version guidance** so that you can ensure that only pre-approved packages are being used.
- **Centralized issue resolution** workflows to streamline and automate updating the catalog (e.g. when there are new security vulnerabilities, licensing issues, or requests from your team to start using new packages).
- **Standardized open source release management**, to reduce the complexity of managing your open source software supply chain.

Organizations also have the added flexibility to create custom catalogs based on an individual development team's requirements, setting up unique standards and policies applicable to that team and its development initiatives.

CATALOG ADMINISTRATION



The core benefits of using the Tidelift catalog and custom catalogs:

1. Version guidance so you can ensure only pre-vetted and approved packages are being used in production environments.
2. Centralized issue resolution workflows to streamline and automate updating the catalog.
3. Standardized open source release management to reduce the complexity of managing your open source software supply chain.

This helps your organization adopt a proactive approach for managing open source by establishing policies. Both the Tidelift catalog and custom catalogs act as the hub for your open source policy creation and enforcement.

A catalog represents an **ideal recommended state** of open source package usage amongst the packages that are approved for use within an organization. Catalogs are a top level construct which contain their own set of standards and each catalog sits at the same hierarchy level within Tidelift. Catalogs include information about packages and their versions that are approved for use within an organization. By enabling or disabling catalog standards, you can establish your policy for that catalog. The decision to use one catalog or multiple catalogs depends on the needs of your organization.

Single catalog strategy

Using a single catalog is the default recommendation for most organizations and comes with the advantage of having all applications compared to a central set of policies. The administrative overhead is lower in a single catalog, with all projects, reports, and task outputs reflecting decisions made on a singular policy. The tradeoff of using this strategy is that in some cases, an organization may require a different set of policies for their applications and a single catalog may not fulfill every application's specific needs.

Multiple catalog strategy

An advanced catalog administrator can get creative with multiple catalogs. A catalog can slice an organization in different ways since it is a flat construct. Individual catalogs can represent:

- Application tiers
- Business units
- Risk profiles
- License distribution needs
- Or a combination of the categories above

Typically a multiple catalog strategy is implemented when different applications within an organization require diverging policies. However, an individual development team’s needs may not justify the increase in administrative overhead that is incurred when using multiple catalogs. Each catalog will require its standards to be maintained independently of other catalogs, but the tradeoff of this increased overhead is more flexibility.

Catalog administrators & catalog users

Catalog roles can be generally viewed as either a catalog administrator or a catalog user. A catalog administrator creates policies and makes decisions based on those policies. A catalog user, here meaning a member (a user that benefits from the output of policy decisions) or a developer, takes the outputs of a catalog administrator’s decisions and applies them to their applications. At this stage, it is important to consider who plays the role of a catalog administrator and a catalog user. A catalog administrator could be one individual or a team based on the size of the organization. It is important that catalog administrators are armed with a set of standards and practices that help them decide which open source components to approve and which to reject. Furthermore, it is critical that these standards and policies are defined in a centralized manner, taking into account the needs of all the stakeholders who are impacted in the case of a vulnerability. Defining catalog standards and correctly assigning the decision maker(s) is an important early step in adopting the Tidelift Subscription.

Catalog administrators are also responsible for managing which standards are enabled and ensuring remediation tasks are addressed by appropriate individuals.

Tasks

At a high level, the primary goal of catalog administration is to answer the question, **“Do I want my developers to be using this package release?”**. To achieve this, a catalog requires curation by a catalog administrator in your organization. The primary method in which a catalog administrator curates one or more catalogs is through processing catalog tasks.

When an open source package violates a catalog standard, a resulting task is created to prompt for a decision to be made on that package’s usage. We notify you and create tasks when there are standards violations, taking the guesswork (and legwork) out of catalog management. In its ideal state, the catalog should become the curated source of truth for whether or not a package release should continue to be used or start being used by developers in your organization.

Catalog tasks 52

Task type	Name	Affected	Date created	Tags
MAINTENANCE	npm/request	site_system_sample_javascript_project, subscription web app 2.88.0, 2.88.2	11 Apr 2022	...
LICENSING	rubygems/internal-package 1.0.0	layer_sample_ruby_project 1.0.0	11 Apr 2022	...
LICENSING	npm/internal-package 1.0.0	site_system_sample_javascript_project 1.0.0	11 Apr 2022	...
LICENSING	rubygems/mongrel 1.1.5	layer_sample_ruby_project 1.1.5	11 Apr 2022	...
SECURITY	npm/glob-parent	site_system_sample_javascript_project, subscription-web-app and 1 other... 2.0.0, 3.1.0, 5.1.1	11 Apr 2022	7.5 CVE Severity ...
SECURITY	npm/ini	site_system_sample_javascript_project, subscription-web-app 1.3.5	11 Apr 2022	7.3 CVE Severity ...

Creating exceptions

There will be instances where it is allowable to use a package release that is not in line with your organization's standards. By creating an exception, you are indicating that the standard violation in question does not apply to a particular catalog or applications. As such, **exceptions** are granted at the catalog level and apply to a specific package version and name.

Standards administration

The screenshot shows the Tidelift web interface for managing standards. The organization is 'Barrette Dynamics Corp' and the catalog is 'Externally Distributed'. The page is titled 'Catalog standards' and includes a description: 'Select standards to enforce for this catalog. When a package release violates a standard, Tidelift will generate a task to help you fix it.'

The interface is divided into two sections: 'Active standards' and 'Inactive standards'.

Active standards:

- SECURITY:** Releases have no vulnerabilities. Description: 'Configure this security standard to keep release streams with vulnerabilities out of your catalog. If necessary, [create an exception](#) for this standard.' Buttons: DEACTIVATE, CONFIGURE STANDARD.
- LICENSING:** Releases use approved licenses. Description: 'Configure this licensing standard to keep packages with denied or unknown licenses out of your catalog. If necessary, [create an exception](#) for this standard.' Buttons: DEACTIVATE, CONFIGURE STANDARD.
- MAINTENANCE:** Releases are actively maintained. Description: 'Use this maintenance standard to keep deprecated packages out of your catalog. If necessary, [create an exception](#) for this standard.' Button: DEACTIVATE.
- OTHER:** Releases must be manually reviewed. Description: 'Use this standard to make sure all requests are manually reviewed, even when there are no other standards violations. If necessary, [create an exception](#) for this standard.' Button: DEACTIVATE.

Inactive standards:

- MAINTENANCE:** Releases are up to date. Description: 'Configure this maintenance standard to keep old packages out of your catalog.' Buttons: ACTIVATE, CONFIGURE STANDARD.
- OTHER:** Releases are approved by Tidelift. Description: 'Configure this standard to keep any releases that are not in one of Tidelift's approved lists of packages (eg. Python data science) out of your catalog. If necessary, [create an exception](#) for this standard.' Buttons: ACTIVATE, CONFIGURE STANDARD.

Standards are how a catalog's policies are created and enforced. A catalog represents open source packages and package versions approved or denied for use in your organization's applications. The Tidelift standards page is designed to be easy to understand and configure. Standards are simply enabled or disabled by catalog administrators. This binary mechanism makes setting up a catalog's policy approachable from the start. For example, if you want to be alerted to security vulnerabilities, enable the 'Releases have no vulnerabilities' security standard. If you are not interested in being alerted about security issues in a catalog, you can easily disable the standard.

ENABLING STANDARDS

There are a variety of security, maintenance, and licensing-related standards that are available out-of-the-box with the Tidelift Subscription. Additionally, organizations can also build their own standards as needed. When a standard is enabled on a catalog, any OSS package that violates that standard will result in the creation of a **task**.

The screenshot shows the 'Active standards' configuration page in the Tidelift interface. On the left is a navigation sidebar with 'CATALOGS' selected, containing links for Overview, Tasks, Packages, Standards, Activity feed, Roles, and Reports. Below this are 'PROJECTS' and 'SETTINGS'. The main content area lists five active standards:

- SECURITY**: Releases have no vulnerabilities. Description: Configure this security standard to keep release streams with vulnerabilities out of your catalog. Buttons: DEACTIVATE, CONFIGURE STANDARD.
- SECURITY**: Unknown packages must be manually reviewed. Description: Configure this security standard to keep unknown, and potentially malicious, packages out of your catalog. If necessary, create an exception for this standard. Button: DEACTIVATE.
- LICENSING**: Releases use approved licenses. Description: Configure this licensing standard to keep packages with denied or unknown licenses out of your catalog. If necessary, create an exception for this standard. Buttons: DEACTIVATE, CONFIGURE STANDARD.
- LICENSING**: Releases have an identified license. Description: All package releases have a machine-readable SPDX expression. Buttons: DEACTIVATE, CONFIGURE STANDARD.
- MAINTENANCE**: Releases are actively maintained. Description: Use this maintenance standard to keep deprecated packages out of your catalog. If necessary, create an exception for this standard. Button: DEACTIVATE.

LICENSING STANDARD

The **licensing standard** provides more configuration options for setting up the licensing policies for a catalog. Tidelift has worked with its in-house licensing experts to create **licensing templates** to help users get started with a license policy ranging from least to most restrictive.

The default policy template applied is the 'Mobile app/Hardware' template, the most restrictive licensing policy. In the licensing configuration menu, a less-restrictive policy can be selected, or no template can be applied to create a licensing policy from scratch. Open source licensing is a specialized discipline in which Tidelift has significant knowledge and experience. It is recommended to start with a policy template and adjust as needed.

The screenshot shows a dialog box titled 'Choose a different template' with a close button in the top right. Below the title is a subtitle: 'Reset your lists by selecting the option that best meets your project's needs. You'll be able to customize this later.' The dialog contains five selectable options, each with a 'SELECT' button:

- Continue without a template**: Set up your catalog's license standards without a template.
- SaaS frontend**: *Least restrictive*. Licenses typically appropriate for code distributed to users as part of a web interface, typically JavaScript or similar.
- SaaS server**: *Less restrictive*. Licenses typically appropriate for code used in cloud-hosted services, limited to permissive licenses rated Lead or higher by Blue Oak.
- B2B Distributed application**: *Most restrictive*. Licenses typically appropriate for code that is distributed to corporate customers.
- Mobile App/Hardware**: *Most restrictive*. Licenses typically appropriate for code that is distributed to consumers, particularly over mobile or as part of an embedded device.

If your organization already has an open source software license policy in place, that policy can be mirrored in Tidelift by setting licenses to be approved or denied for use based on your organization's policy.

ENFORCING LICENSE COMPLIANCE

By choosing to use the **license compliance standard**, we ensure that all package releases in your catalog only use a license from your approved list of licenses. When enabling the standard, you can choose from a list of templates that fit your deployment scenario. You can also proceed without a template and begin classifying the licenses into the following three categories:

- **Approved:** Licenses that are always approved for use in the catalog
- **Uncategorized:** Licenses that will need additional review in the future
- **Denied:** Licenses that are never approved for use in the catalog

Configure license compliance standard VIEW EXCEPTIONS

0 exceptions have been created for denied licenses

Customize your license lists

Decide which licenses you want to automatically approve or deny. You can also [change your licensing template](#) from this page.

Q Search for a license EXPORT CSV SAVE CHANGES

Uncategorized in catalog (0)	✓ Approved in catalog (6)	⊘ Denied in catalog (0)
Uncategorized not in use (204)	BSD-3-Clause (SPDX) 135 releases	⊘ Denied not in use (138)
AAL (SPDX) Not in use	MIT (SPDX) 104 releases	AGPL-1.0 (SPDX) Not in use
Abstyles (SPDX) Not in use	Apache-2.0 (SPDX) 73 releases	AGPL-1.0-only (SPDX) Not in use
Adobe-2006 (SPDX) Not in use	BSD-2-Clause (SPDX) 10 releases	AGPL-1.0-or-later (SPDX) Not in use
Afmparse (SPDX) Not in use	ISC (SPDX) 2 releases	AGPL 3.0 (SPDX) Not in use

SECURITY VULNERABILITY STANDARD

By enabling the **'Releases have no vulnerabilities' standard**, you can ensure that all requested and approved packages are reviewed for any known vulnerabilities.

Configure releases have no vulnerabilities standard VIEW EXCEPTIONS

Customize how we manage the 'releases have no vulnerabilities' security standard for packages in your catalog. We'll use these configurations to create catalog tasks for security vulnerabilities you care about.

Select the minimum CVE score to generate tasks:

Low 1 2 3 Medium 5 6 High 8 Critical 10

Tidelift maintains a security database of vulnerabilities and proactively reviews these vulnerabilities. For each vulnerability, we share the upgrade path recommended by Tidelift or our partnered maintainers. This recommended path lets your designated catalog administrator make an informed decision about whether to remove the vulnerable release, to create an exception, or to remediate the vulnerability.

MAINTENANCE STANDARD

There are two ways to enforce maintenance compliance within your catalog:

1. Releases are actively maintained
2. Releases are up-to-date

Releases are actively maintained

Releases are actively maintained standard exceptions

Here are the exceptions that you've created to this standard, over time.

[CREATE NEW EXCEPTION](#) [EXPORT CSV](#)

Package manager	Package name	Releases	Project label	Date added
No exceptions!				

'Releases are actively maintained' is a standard that will check to see if a particular package has been marked as deprecated. Packages that are explicitly marked as deprecated are less likely to receive fixes if a security vulnerability is discovered. Stale, deprecated packages can become a nuisance for upgrading an application as they may rely on older versions of other packages.

Tidelift is regularly monitoring for package deprecation from the following sources:

- From the package manager, when a maintainer indicates that a package has been deprecated
- Directly from our partnered maintainers for instances when deprecation information has not been shared publicly

Enabling this standard will notify you when your team is using or wants to use deprecated packages and help you uphold this standard. We will also display any additional information that a maintainer has provided about the deprecation, which may include recommendations for alternate packages.

Releases are up-to-date

Configure releases are up to date maintenance standard

[VIEW EXCEPTIONS](#)

Customize how we manage the 'releases are up to date' maintenance standards for packages in your catalog. We'll use these configurations to make sure you're always using the safest release stream.

Set default that all releases should be no older than the latest release by the number of years specified below.
[See example](#) →

year(s) older than the latest release

[SAVE](#) [BACK TO STANDARDS](#)

The 'Releases are up-to-date' standard measures the risk of using outdated packages within your organization. Similar to deprecated packages, older releases are less likely to be patched. While the package itself may still be actively maintained, these older releases usually have security vulnerabilities and issues that are addressed in newer releases.

The longer your organization waits to update to a newer release, the harder it may be to use as more changes are made to the package and underlying APIs. With this standard, you can keep outdated package releases out of your catalog by enforcing a number of years that releases should be no older than.

Let's use an example. Suppose you set a default that all releases should be no more than 1 year older than the latest release. 2.0.0 is the latest release, but your projects are still using releases 1.5.0 and 1.0.0. In the example, Tidelift will alert you to update where you're using version 1.0.0, but not 1.5.0.

VERSION	RELEASE DATE	DECISION
2.0.0	1 Jan 2020	Allowed, latest release
1.5.0	1 Apr 2019	Allowed, < 1 year older than the latest release
1.0.0	1 Apr 2018	Not Allowed, > 1 year older than the latest release

CI FLOW

Starting your journey

The first step to effectively manage your organization's open source software supply chain is to clearly understand what is in use today. Integrating Tidelift into your continuous integration (CI) workflow is a great way to start getting value from your subscription from day one.

Once integrated, Tidelift helps prevent developers from pulling denied packages into their builds. Having the most up-to-date SBOMs sent to Tidelift with each build gives you the visibility into all the open source that your organization is using at any given moment.

Tidelift works with most CI tools including: Jenkins, CircleCI, Github Actions, Teamcity, Gitlab Pipelines, Azure Devops Pipelines, and Atlassian Bitbucket.

Implementing the CLI

Customers can keep their SBOMs current and up-to-date by integrating the Tidelift CLI into their CI pipeline. Most commonly, customers include our CLI as part of their build process for their application. This ensures that Tidelift can generate an accurate SBOM of the actual artifact that is going to be deployed to the development and production environments.

The first step to using the Tidelift CLI is to install it into the CI pipeline where your application is being built. The exact steps vary depending on the operating system running the build server, you can find the additional instructions in our [Installing the Tidelift CLI](#) document. Once the permissions have been properly configured for the Tidelift CLI, you can execute `tidelift alignment save --wait` to upload the SBOM to the Tidelift servers.

Enforcing your standards

Once an organization becomes mature enough in their open source management practices, they can start using the Tidelift CLI to gatekeep their CI pipelines to enforce their standards. This allows an organization to move towards being proactive about preventing problematic open source from entering their production environment, rather than fixing these issues after the fact.

Using the CLI to enforce standards is considered an advanced action taken after standards selection, team onboarding, and catalog administration are fully adopted and the downstream effects of failing a build are fully understood. SBOMs will be submitted at build time and the data can be used to populate a catalog as well as to continuously compare packages to an approved catalog.

Understanding when a build will fail and under what circumstances based on catalog curation is an important step to adopting Tidelift into your software development lifecycle. Failing builds should be considered only after your organization becomes comfortable with Tidelift catalog administration, package approvals, package denials, and establishing your organization's standards.

Regression

Regressions fail a build if new open source issues are introduced into an SBOM. A regression is when a new dependency is introduced and the dependency cannot be auto-approved in your catalog. The regression is compared relative to a baseline alignment scan. Anything introduced after that baseline has the capability to fail scans, while everything before that baseline will not cause a build to fail, allowing for your team to block builds when new violations are added.

Information related to regressions includes:

- The total number of regressions
- The type of violation (per package) that introduced a regression
- A severity score (for security vulnerabilities) that will allow your team to create a script to configure a build stage to fail



INTEGRATIONS

Artifactory integration

Several organizations often use artifact managers to manage their use of open source. Tidelift provides integrations with JFrog Artifactory with the ability to import Artifactory repository metadata directly into a catalog, helping inventory the packages available to your developers. While Artifactory does a good job of providing an inventory of packages and package versions, Tidelift provides additional capabilities such as supplying package metadata as well as giving customers the ability to apply security, license, and maintenance standards. Additionally, Tidelift enables you to aggregate all or multiple Artifactory repository data into a single Tidelift catalog for a centralized view. Applying Tidelift standards in this workflow allows you to use Tidelift as a barometer to identify and prioritize the areas of improvement you need to plan as you start onboarding development teams.

TIDELIFT