Using the Dependency submission API

In this article

You can use the Dependency submission API to submit dependencies for projects, such as the dependencies resolved when a project is built or compiled.

Note: The ability to use the REST API for dependency submission is currently in public beta and subject to change.

About the Dependency submission API

You can use the REST API to submit dependencies for a project. This enables you to add dependencies, such as those resolved when software is compiled or built, to GitHub's dependency graph feature, providing a more complete picture of all of your project's dependencies.

The dependency graph shows any dependencies you submit using the API in addition to any dependencies that are identified from manifest or lock files in the repository (for example, a `package-lock.json` file in a JavaScript project). For more information about viewing the dependency graph, see "Exploring the dependencies of a repository."

Submitted dependencies will receive Dependabot alerts and Dependabot security updates for any known vulnerabilities. You will only get Dependabot alerts for dependencies that are from one of the supported ecosystems for the GitHub Advisory Database. For more information about these ecosystems, see "About the GitHub Advisory database." Submitted dependencies will be surfaced in dependency review or your organization's dependency insights.

Note: The dependency review API and the dependency submission API work together. This means that the dependency review API will include dependencies submitted via the dependency submission API. This feature is currently in public beta and subject to change.

Dependencies are submitted to the dependency submission API in the form of a snapshot. A snapshot is a set of dependencies associated with a commit SHA and other metadata, that reflects the current state of your repository for a commit. Snapshots can be generated from your dependencies detected at build time or from a software bill of materials (SBOM). There are GitHub Actions that support either of these use cases. For more information about the Dependency submission API, see the Dependency submission REST API documentation.

Submitting dependencies at build-time

You can use the Dependency submission API in a GitHub Actions workflow to submit dependencies for your project when your project is built.

Using pre-made actions

The simplest way to use the Dependency submission API is by adding a pre-made action to your repository that will gather and convert the list of dependencies to the required snapshot format and submit the list to the API. Actions that complete these steps for various ecosystems are available on GitHub Marketplace. Some of these actions are provided by third parties. You can find links to the currently available actions in the table below.

Ecosystem	Action	Maintained by GitHub
Go	Go Dependency Submission	✓
Gradle	Gradle Dependency Submission	×
Maven	Maven Dependency Tree Dependency Submission	✓
Mill	Mill Dependency Submission	×
Scala	Sbt Dependency Submission	×

For example, the following Go Dependency Submission workflow calculates the dependencies for a Go build-target (a Go file with a `main` function) and submits the list to the Dependency submission API.

```yaml
name: Go Dependency Submission
on:
  push:
    branches:
      - main

# The API requires write permission on the repository to submit dependencies
permissions:
  contents: write

# Environment variables to configure Go and Go modules. Customize as necessary
env:
  GOPROXY: '' # A Go Proxy server to be used
  GOPRIVATE: '' # A list of modules are considered private and not requested from
GOPROXY
jobs:
  go-action-detection:
    runs-on: ubuntu-latest
    steps:
      - name: 'Checkout Repository'
        uses: actions/checkout@v4

      - uses: actions/setup-go@v4
        with:
          go-version: ">=1.18.0"

      - name: Run snapshot action
        uses: actions/go-dependency-submission@v1
        with:
            # Required: Define the repo path to the go.mod file used by the
            # build target
            go-mod-path: go-example/go.mod
            #
            # Optional. Define the repo path of a build target,
            # a file with a `main()` function.
            # If undefined, this action will collect all dependencies
```

```
            # used by all build targets for the module. This may
            # include Go dependencies used by tests and tooling.
            go-build-target: go-example/cmd/octocat.go
```

Creating your own action

Alternatively, you can write your own action to submit dependencies for your project at build-time. Your workflow should:

1. Generate a list of dependencies for your project.
2. Translate the list of dependencies into the snapshot format accepted by the Dependency submission API. For more information about the format, see the body parameters for the "Create a repository snapshot" API operation in the Dependency submission REST API documentation.
3. Submit the formatted list of dependencies to the Dependency submission API.

GitHub Enterprise Cloud maintains the Dependency Submission Toolkit, a TypeScript library to help you build your own GitHub Action for submitting dependencies to the Dependency submission API. For more information about writing an action, see "Creating actions".

Generating and submitting a software bill of materials (SBOM)

An SBOM is a formal, machine-readable inventory of a project's dependencies and associated information (such as versions, package identifiers, and licenses). SBOMs help reduced supply chain risks by:

- providing transparency about the dependencies used by your repository
- allowing vulnerabilities to be identified early in the process
- providing insights in the license compliance, security, or quality issues that may exist in your codebase
- enabling you to better comply with various data protection standards

To generate an SBOM, you can use:

- the GitHub user interface. For more information about how to export an SBOM for a repository using information from the dependency graph, see "Exporting a software bill of materials for your repository."
- the REST API. For more information, see "Software bill of materials (SBOM)."
- GitHub Actions. The following actions will generate an SBOM for your repository and attach it as a workflow artifact which you can download and use in other applications. For more information about downloading workflow artifacts, see "Downloading workflow artifacts."

Action	Details	Maintained by GitHub
SBOM-generator-action	Uses the information in your dependency graph to generate an SPDX SBOM	✓
Anchore SBOM Action	Uses Syft to create SPDX 2.2 compatible SBOMs with the supported ecosystems	×
sbom-tool by Microsoft	Scans your dependencies and creates an SPDX compatible	×

creates an SPDX compatible SBOM

You can then upload and submit the SBOM to the dependency submission API using one of the following actions so that you can receive Dependabot alerts on any dependencies that have known vulnerabilities. Actions that appear in both tables can be configured to both generate and submit an SBOM.

Action	Details	Maintained by GitHub
SPDX Dependency Submission Action	Uses Microsoft's SBOM Tool to create SPDX 2.2 compatible SBOMs with the supported ecosystems	✓
Anchore SBOM Action	Uses Syft to create SPDX 2.2 compatible SBOMs with the supported ecosystems	×
SBOM Dependency Submission Action	Uploads a CycloneDX SBOM to the dependency submission API	×

For example, the following SPDX Dependency Submission Action workflow calculates the dependencies for a repository, generates an exportable SBOM in SPDX 2.2 format, and submits it to the dependency submission API.

```yaml
name: SBOM upload

on:
  workflow_dispatch:
  push:
    branches: ["main"]

jobs:
  SBOM-upload:

    runs-on: ubuntu-latest
    permissions:
      id-token: write
      contents: write

    steps:
    - uses: actions/checkout@v4
    - name: Generate SBOM
      # generation command documentation: https://github.com/microsoft/sbom-tool#sbom-generation
      run: |
        curl -Lo $RUNNER_TEMP/sbom-tool https://github.com/microsoft/sbom-tool/releases/latest/download/sbom-tool-linux-x64
        chmod +x $RUNNER_TEMP/sbom-tool
        $RUNNER_TEMP/sbom-tool generate -b . -bc . -pn $ -pv 1.0.0 -ps OwnerName -nsb https://sbom.mycompany.com -V Verbose
    - uses: actions/upload-artifact@v3
      with:
        name: sbom
        path: _manifest/spdx_2.2
    - name: SBOM upload
      uses: advanced-security/spdx-dependency-submission-action@v0.0.1
      with:
        filePath: "_manifest/spdx_2.2/"
```