npm / registry Public

Code Pull requests 5 Actions Security Insights

master

registry / docs / responses / package-metadata.md

dr-js Add later introduced fields to dist object History

3 contributors

258 lines (210 sloc) | 10.4 KB

Package Metadata

Package *metadata* describes a package for its consumers: who wrote it, where its repository is, and what versions of it have been published. It also contains a description of each *version* of a package present in the registry, listing its dependencies, giving the url of its tarball, and so on. Package metadata is useful for finding packages and for installing them.

You can request *package metadata* from this endpoint:

```
GET https://registry.npmjs.org/:package
```

The registry responds with a JSON-formatted string containing metadata for the package named, either in full or abbreviated form depending on what you request in the `Accept` header. If you provide no Accept header, the full document is returned. To request an *abbreviated* document with only the fields required to support installation, set the `Accept` header in your request to the following string:

```
application/vnd.npm.install-v1+json
```

A more typical accept header might request json as a fallback, like this:

```
application/vnd.npm.install-v1+json; q=1.0, application/json; q=0.8, */*
```

The formats are described in detail below, but you can compare them by making requests using any tool you like. To request package metadata documents with httpie:

```
http GET https://registry.npmjs.org/npm
http GET https://registry.npmjs.org/npm Accept:application/vnd.npm.install-v1+json
```

With the less user-friendly but ubiquitous curl:

```
curl -H "Accept: application/vnd.npm.install-v1+json" https://registry.npmjs.org/npm
```

`tiny-tarball` is a small package with only one version and no dependencies. Its abbreviated metadata looks like this:

```
{
    "dist-tags": {
        "latest": "1.0.0"
    },
    "modified": "2015-05-16T22:27:54.741Z",
    "name": "tiny-tarball",
    "versions": {
        "1.0.0": {
            "_hasShrinkwrap": false,
            "directories": {},
            "dist": {
                "shasum": "bbf102d5ae73afe2c553295e0fb02230216f65b1",
                "tarball": "https://registry.npmjs.org/tiny-tarball/-/tiny-tarball-1
            },
            "name": "tiny-tarball",
            "version": "1.0.0"
        }
    }
}
```

The full metadata for `tiny-tarball` looks like this:

```
{
    "_attachments": {},
    "_id": "tiny-tarball",
    "_rev": "3-085759e977d42299e64a35aedc17d250",
    "author": {
        "email": "ben@npmjs.com",
        "name": "Ben Coe"
    },
    "description": "tiny tarball used for health checks",
    "dist-tags": {
        "latest": "1.0.0"
    },
    "license": "ISC",
    "maintainers": [
        {
            "email": "ben@npmjs.com",
            "name": "bcoe"
        }
    ],
    "name": "tiny-tarball",
    "readme": "# TinyTarball\n\ntiny-tarball used for health checks\n\n**don't unpub
    "readmeFilename": "README.md",
    "time": {
        "1.0.0": "2015-03-24T00:12:24.039Z",
        "created": "2015-03-24T00:12:24.039Z",
        "modified": "2015-05-16T22:27:54.741Z"
    },
    "versions": {
        "1.0.0": {
            "_from": ".",
            "_id": "tiny-tarball@1.0.0",
            "_nodeVersion": "1.5.0",
            "_npmUser": {
                "email": "bencoe@gmail.com",
                "name": "bcoe"
            },
            "_npmVersion": "2.7.0",
            "_shasum": "bbf102d5ae73afe2c553295e0fb02230216f65b1",
            "author": {
                "email": "ben@npmjs.com",
                "name": "Ben Coe"
            },
            "description": "tiny tarball used for health checks",
            "directories": {},
            "dist": {
                "shasum": "bbf102d5ae73afe2c553295e0fb02230216f65b1",
                "tarball": "https://registry.npmjs.org/tiny-tarball/-/tiny-tarball-1
            },
            "license": "ISC",
            "main": "index.js",
            "maintainers": [
                {
                    "email": "bencoe@gmail.com",
                    "name": "bcoe"
                }
            ],
            "name": "tiny-tarball",
            "scripts": {
                "test": "echo \"Error: no test specified\" && exit 1"
            },
            "version": "1.0.0"
        }
    }
}
```

The size difference is more exaggerated for packages with many versions or many stars, such as `npm` or `lodash`. For some packages in the registry, the full metadata is over 10MB uncompressed. If the information you wish to use for a package is present in the abbreviated version, you should prefer it over the full version.

Components of the metadata

human

Human objects have at least one of the following fields defined:

- `name` : a freeform string name
- `email` : an email address
- `url` : a url for a web page with more information about the author

Historically no validation has been performed on those fields; they are generated by parsing user-provided data in package.json at publication time.

Example:

```
{
    "email": "ben@example.com",
    "name": "Ben The Example"
}
```

dist

The `dist` object is generated by npm and may be relied upon. Each dist object has at least two fields:

- `tarball` : the url of the tarball containing the payload for this package

- `shasum` : the SHA-1 sum of the tarball
- `integrity` : since Apr 2017, string in the format `<hashAlgorithm>-<base64-hash>` , refer the Subresource Integrity and cacache package for more
- `fileCount` : since Feb 2018, the number of files in the tarball, folder excluded
- `unpackedSize` : since Feb 2018, the total byte of the unpacked files in the tarball
- `npm-signature` : since Apr 2018, a PGP signature of `<package>@<version>:<integrity>` , refer the npm blog and doc for more
- (in the future) a SHA-2 512 sum of the tarball

Example:

```
{
    "shasum": "bbf102d5ae73afe2c553295e0fb02230216f65b1",
    "tarball": "https://registry.npmjs.org/tiny-tarball/-/tiny-tarball-1.0.0.tgz"
}
```

repository

An object specifying the repository where the source for this package might be found. It has two fields:

```
"repository": {
    "type": "git",
    "url": "git://github.com/npm/npm.git"
}
```

Abbreviated metadata format

This form of the package metadata exists to provide a smaller payload designed to support installation. It contains a whitelisted subset of fields from the full metadata set. The top-level fields are:

- `name` : the package name
- `modified` : ISO string of the last time this package was modified
- `dist-tags` : a mapping of dist tags to the versions they point to
- `versions` : a mapping of version numbers to objects containing the information needed to install that version

Example:

```
{
    "name": "<package-name>",
    "modified": "2017-03-21T21:40:18.939Z",
    "dist-tags": {
        "latest": "<semver-compliant version string>",
        "<dist-tag-name>": "<semver-compliant version string>"
    },
    "versions": {
        "<version>": <version object>,
        "<version>": <version object>
    }
}
```

Abbreviated version object

Each abbreviated version object contains the following fields:

- `name` : the package name
- `version` : the version string for this version
- `deprecated` : the deprecation warnings message of this version
- `dependencies` : a mapping of other packages this version depends on to the required semver ranges
- `optionalDependencies` : an object mapping package names to the required semver ranges of *optional* dependencies
- `devDependencies` : a mapping of package names to the required semver ranges of *development* dependencies
- `bundleDependencies` : an array of dependencies bundled with this version
- `peerDependencies` : a mapping of package names to the required semver ranges of *peer* dependencies
- `bin` : a mapping of bin commands to set up for this version
- `directories` : an array of directories included by this version
- `dist` : a dist object
- `engines` : the node engines required for this version to run, if specified
- `_hasShrinkwrap` : `true` if this version is known to have a shrinkwrap that must be used to install it; `false` if this version is known not to have a shrinkwrap. If this field is undefined, the client must determine through other means if a shrinkwrap exists.

The `name` , `version` , and `dist` fields will always be present. The others will be absent if they are irrelevant for this package version.

Full metadata format

Top-level fields, in lexical order:

- `_id` : the package name, used as an ID in CouchDB
- `_rev` : the revision number of this version of the document in CouchDB
- `dist-tags` : a mapping of dist tags to versions. Every package will have a `latest` tag defined.
- `name` : the package name
- `time` : an object mapping versions to the time published, along with `created` and `modified` timestamps
- `users` : an object whose keys are the npm user names of people who have starred this package
- `versions` : a mapping of semver-compliant version numbers to version data

The following fields are hoisted to the top-level of the package json from the latest version published:

- `author` : human object
- `bugs` : url
- `contributors` : array of human objects
- `description` : a short description of the package
- `homepage` : url
- `keywords` : array of string keywords
- `license` : the SPDX identifier of the package's license
- `maintainers` : array of human objects for people with permission to publish this package; not authoritative but informational
- `readme` : the first 64K of the README data for the most-recently published version of the package
- `readmeFilename` : The name of the file from which the readme data was taken.
- `repository` : as given in package.json, for the latest version

Each package version data object contains all of the fields in the abbreviated document, plus the fields listed above as hosted, plus at least the following:

- `_id` : `package@version` , such as `npm@1.0.0`
- `_nodeVersion` : the version of node used to publish this
- `_npmUser` : the author object for the npm user who published this version
- `_npmVersion` : the version of the npm client used to publish this

- `main` : the package's entry point (e.g., index.js or main.js)

The full version object will also contain any other fields the package publisher chose to include in their package.json file for that version.