libnpmpublish

`libnpmpublish` is a Node.js library for programmatically publishing and unpublishing npm packages. Give it a manifest as an object and a tarball as a Buffer, and it'll put them on the registry for you.

Table of Contents

Example

```
const { publish, unpublish } = require('libnpmpublish')
```

Install

```
$ npm install libnpmpublish
```

API

`opts` for `libnpmpublish` commands `libnpmpublish` uses `npm-registry-fetch`. Most options are passed through directly to that library, so please refer to its own `opts` documentation for options that can be passed in.

A couple of options of note:

- `opts.defaultTag` - registers the published package with the given tag, defaults to `latest`.
- `opts.access` - tells the registry whether this package should be published as public or restricted. Only applies to scoped packages, which default to restricted.
- `opts.token` - can be passed in and will be used as the authentication token for the registry. For other ways to pass in auth details, see the n-r-f docs.

`> libpub.publish(manifest, tarData, [opts]) -> Promise` Sends the package represented by the `manifest` and `tarData` to the configured registry.

`manifest` should be the parsed `package.json` for the package being published (which can also be the manifest pulled from a packument, a git repo, tarball, etc.)

`tarData` is a `Buffer` of the tarball being published.

If `opts.npmVersion` is passed in, it will be used as the `_npmVersion` field in the outgoing packument. You may put your own user-agent string in there to identify your publishes.

If `opts.algorithms` is passed in, it should be an array of hashing algorithms to generate `integrity` hashes for. The default is `['sha512']`, which means you end up with `dist.integrity = 'sha512-deadbeefbadc0ffee'`. Any algorithm supported by your current node version is allowed – npm clients that do not support those algorithms will simply ignore the unsupported hashes.

Example

```
// note that pacote.manifest() and pacote.tarball() can also take
// any spec that npm can install.  a folder shown here, since that's
// far and away the most common use case.
const path = '/a/path/to/your/source/code'
const pacote = require('pacote') // see: http://npm.im/pacote
const manifest = await pacote.manifest(path)
const tarData = await pacote.tarball(path)
await libpub.publish(manifest, tarData, {
  npmVersion: 'my-pub-script@1.0.2',
  token: 'my-auth-token-here'
}, opts)
// Package has been published to the npm registry.
```

`> libpub.unpublish(spec, [opts]) -> Promise` Unpublishes `spec` from the appropriate registry. The registry in question may have its own limitations on unpublishing.

`spec` should be either a string, or a valid `npm-package-arg` parsed spec object. For legacy compatibility reasons, only `tag` and `version` specs will work as expected. `range` specs will fail silently in most cases.

Example

```
await libpub.unpublish('lodash', { token: 'i-am-the-worst'})
//
// `lodash` has now been unpublished, along with all its versions
```