Tin: A Tcl Package Manager

Version 0.4.3

Alex Baker

https://github.com/ambaker1/Tin

April 26, 2023

Abstract

Tin is a package installer for Tcl. With Tin, you can easily install packages directly from GitHub.

The Tin List

Tin installs packages that are in the “Tin List”, which can be modified in the current session with the commands *tin add* and *tin remove*.

```
tin add $name $version $repo $tag $file
```

`$name`	Package name.
`$version`	Package version.
`$repo`	Github repository URL.
`$tag`	Github release tag for version.
`$file`	Installer file path in repo.

```
tin remove $name <$version> <$repo>
```

`$name`	Package name.
`$version`	Package version to remove (optional, default all versions).
`$repo`	Repository to remove (optional, default all repositories).

Example 1: Adding a package to the Tin List

Code:

```
package require tin
tin add foo 1.0 https://github.com/username/foo v1.0 install_foo.tcl
```

Auto Packages

The commands *tin add* and *tin remove* also have alternative syntax for adding and removing Auto-Tin packages. An Auto-Tin package is one which has a GitHub repository that has release tags corresponding directly with the package versions, such as "v1.2.3". To be specific, version release tags must match the following regular expression:

```
^v(0|[1-9]\d*)(\.(0|[1-9]\d*))*([ab](0|[1-9]\d*)(\.(0|[1-9]\d*))*)?$
```

```
tin add -auto $name $repo $file <<-exact> $version> <$reqs ...>
```

$name	Package name.
$repo	Github repository URL.
$file	Installer file path in repo.
-exact	Option to specify exact version.
$version	Package version.
$reqs ...	Package version requirements, mutually exclusive with -exact option.

```
tin remove -auto $name <$repo> <$file>
```

$name	Package name.
$repo	Repository to remove (optional, default all repositories).
$file	Installer file path to remove (optional, default all installer files).

Then, if a package is configured as an Auto-Tin package, the Tin List can be populated with versions available for installation with the command *tin fetch*.

```
tin fetch <$name>
```

$name	Package name (optional, default "-all" fetches for all Auto-Tin packages).

The Official Tin List

The “tinlist.tcl” file in the Tin installation initializes the Tin List. As of Tin version 0.4.3, these are the packages in the official Tin List:

Auto-Tin Packages

Package	Repo	File	Version Requirements
tin	https://github.com/ambaker1/Tin	install.tcl	0
tintest	https://github.com/ambaker1/Tin-Test	install.tcl	0-
wob	https://github.com/ambaker1/wob	install.tcl	0.1.3-
flytrap	https://github.com/ambaker1/flytrap	install.tcl	0-

Saving, Clearing, and Resetting the Tin List

The state of the Tin List can be saved for future sessions with *tin save*, cleared with *tin clear*, and reset to default or factory settings with *tin reset*. Note that *tin save* does not modify the "tinlist.tcl" file in the Tin installation. Rather, it saves to a hidden user-config file located in the user's home directory.

```
tin save
```

```
tin clear
```

```
tin reset <-hard>
```

`-hard`	Option to reset to factory settings.

Example 2: Saving changes to the Tin List

Code:

```
tin reset -hard
tin add foo 1.0 https://github.com/username/foo v1.0 install_foo.tcl
tin save
```

Output:
"~/.tinlist.tcl" :

```
tin add foo 1.0 https://github.com/username/foo v1.0 install_foo.tcl
```

Accessing the Tin List

The command *tin get* queries basic information about Tin, and returns blank if the requested entry does exist. Similar to *tin add* and *tin remove*, it has two forms, one for querying Tin packages and one for querying Auto-Tin packages. Returns a dictionary associated with the supplied arguments.

```
tin get $name <$version> <$repo>
tin get -auto $name <$repo> <$file>
```

$name	Package name.
$version	Package version.
$repo	Github repository URL.
-auto	Option to query Auto-Tin packages.
$file	Installer file path in repo.

Additionally, the available packages in the Tin List can be queried with the command *tin packages*, and the available versions for each Tin package can be queried with the command *tin versions*.

```
tin packages <-auto> <$pattern>
```

-auto	Option to search for Auto-Tin packages. By default searches Tin packages only.
$pattern	Optional "glob" pattern, default "*", or all packages.

```
tin versions $name <<-exact> $version> <$reqs ...>
```

$name	Package name.
-exact	Option to specify exact version.
$version	Package version.
$reqs ...	Package version requirements, mutually exclusive with -exact option.

Installing, Uninstalling, and Upgrading Packages

The command *tin install* installs packages directly from GitHub, and returns the version installed. The command *tin depend* installs packages only if they are not installed, and returns the version number installed (useful for installation scripts). The command *tin installed* returns the package version that is installed and meets the version requirements, or blank if it is not installed. The command *tin uninstall* uninstalls packages (as long as they are in the Tin List), and returns blank if successful. The command *tin upgrade* upgrades a package within the major version (for minor and patch upgrades) and returns the version number installed.

```
tin install $name <<-exact> $version> <$reqs ...>
```

```
tin depend $name <<-exact> $version> <$reqs ...>
```

```
tin installed $name <<-exact> $version> <$reqs ...>
```

```
tin uninstall $name <<-exact> $version> <$reqs ...>
```

```
tin upgrade $name <<-exact> $version> <$reqs ...>
```

`$name`	Package name.
`-exact`	Option to specify exact version.
`$version`	Package version.
`$reqs ...`	Package version requirements, mutually exclusive with -exact option.

Example 3: Upgrading Tin, and reloading within current interpreter

Code:

```
# Upgrade Tin
package require tin
tin upgrade tin
# Reload Tin
package forget tin
namespace delete tin
package require tin
```

Loading and Importing Packages

Tin also provides advanced tools for loading and importing packages. The command *tin require* is similar to the Tcl command *package require*, but with the added feature that if the package is missing, it will try to install it with *tin install*. The command *tin import* additionally handles most use-cases of *namespace import*. Both *tin require* and *tin import* return the version number of the package imported.

```
tin require $name <<-exact> $version> <$reqs ...>
```

`$name`	Package name.
`-exact`	Option to specify exact version.
`$version`	Package version.
`$reqs ...`	Package version requirements, mutually exclusive with -exact option.

```
tin import <-force> <$patterns from> $name <<-exact> $version> <$reqs ...> <as $ns>
```

`-force`	Option to overwrite existing commands.
`$patterns`	Commands to import, or "glob" patterns, default "`*`", or all commands.
`$name`	Package name.
`-exact`	Option to specify exact version.
`$version`	Package version.
`$reqs ...`	Package version requirements, mutually exclusive with -exact option.
`$ns`	Namespace to import into. Default global namespace, or "`::`".

Example 4: Importing all commands package "foo"

Code:

```
package require tin
tin import foo 1.0
```

Utilities for Package Development

In addition to commands for installing and loading packages, Tin provides a few commands intended to help in writing installation and build files for your packages.

Creating Package Directories

The command *tin mkdir* creates a library directory to install a package in, with a normalized naming convention that allows it to be uninstalled easily with *tin uninstall*.

```
tin mkdir <-force> <$basedir> $name $version
```

`-force`	Option to create fresh library directory (deletes existing folder).
`$basedir`	Base directory, default one folder up from the Tcl library folder.
`$name`	Package name.
`$version`	Package version.

See the example installation file for a package "foo" that requires the package "bar 1.2", and installs in library folder "foo-1.0".

Example 5: Example file "install_foo.tcl"

Code:

```
package require tin
tin depend bar 1.2
set dir [tin mkdir -force foo 1.0]
file copy README.md $dir
file copy LICENSE $dir
file copy lib/bar.tcl $dir
file copy lib/pkgIndex.pdf $dir
```

Building Library Files from Source with Configuration Variable Substitution

The command *tin bake* takes an input text file, and writes an output text file after substitution of configuration variables such as `@VERSION@`. This is especially helpful for ensuring that the package version is consistent across the entire project.

```
tin bake $inFile $outFile $config
```

`$inFile`	Source file to read from.
`$outFile`	File to write to after substitution.
`$config`	Dictionary of config variable names and values. Config variables must be uppercase alphanumeric.

See below for an example of how *tin bake* can be used to automatically update a "pkgIndex.tcl" file:

Example 6: Building a "pkgIndex.tcl" file

Code:

```
package require tin
tin bake pkgIndex.tin pkgIndex.tcl {VERSION 1.0}
```

Output:

"pkgIndex.tin" :

```
package ifneeded foo @VERSION@ [list source [file join $dir foo.tcl]]
```

"pkgIndex.tcl" :

```
package ifneeded foo 1.0 [list source [file join $dir foo.tcl]]
```