Package Repository Client and Server

Joe Mistachkin @ Tcl 2016
https://eyrie.solutions/

What are they?

- The **Package Repository Client** is a set of Tcl scripts that are capable of locating, downloading, and installing packages for both Tcl and Eagle. Packages can be installed in advance or on-demand.

- The **Package Repository Server** is a set of TH1 and Tcl scripts running on a carefully configured instance of Fossil with the package data stored in a SQLite database.

Why?

- The existing solutions for distributing packages for Tcl are:
  - Insecure, requiring the server machine itself to be fully trusted.
  - Overly complex, requiring knowledge of arcane tools, settings, and associated terminology.
  - Write things in various locations on the target machine.
  - Non-portable and/or proprietary.

Why? (continued)

- The existing solutions for distributing packages for Tcl also:
  - Write files to user-specific locations on target machines. Why?
  - Make extensive use of registry settings on Windows. Why?

Package Metadata

- Tcl relies upon “pkgIndex.tcl” files for metadata necessary to provide a given package.
- What if we could securely query a remote server instead?
- What if we could configure that server to securely serve our private packages in addition to those provided by the community?

Package Files

- Most packages for Tcl consist of one or more files.
- What if we could securely download these files on-demand?
- What if we always want the latest version of a rapidly changing package?
- What if we could use this to reduce the cost of deploying our Tcl-based applications?

What does the server do?

- It allows packages for Tcl (and Eagle) to be viewed, added, and managed via any web browser.

- There are two parts, both of which are run on Fossil:
  - The package metadata server.
  - The package file server.

What does the client do?

- It enables packages for Tcl (and Eagle) to be located, downloaded, and optionally persisted (i.e. installed) locally.
- All downloaded files are signed with PGP.
- All downloaded Eagle files are **also** signed with Harpy.

Demo

How does it work?

- The client sends a lookup request to the “package repository server” that includes the name and version of the package being sought.
- The request also includes a list of API keys that were configured for use with the package repository client.

How does it work? (continued)

- The server attempts to find all matching packages, based on the name, TIP #268 version requirement, and the supplied list of API keys.

- If a match is found, the server responds to the request with a script that has been signed with either OpenPGP or Harpy.

How does it work? (continued)

- The client verifies the signature on the script received from the server and then evaluates it using the appropriate target language (i.e. native Tcl or Eagle).

- The script is free to perform any actions necessary to obtain the package; typically, it downloads a list of files using the included “package downloader client”.

How does it work? (continued)

- All package files downloaded using the “package downloader client” must be signed using OpenPGP.
- All signatures will be verified prior to the package being made available to the interpreter.

FAQ

- Is it possible to setup private package repository servers and/or private package file servers?
  - Yes.
- Is it possible for package authors to maintain a set of packages and grant the general public access to a subset of them?
  - Yes.

FAQ (continued)

- How is the package repository (metadata) server managed?
  - Using a web interface, usable from any reasonably recent web browser.
- How is the package file server managed?
  - Using Fossil and/or a web browser.

FAQ (continued)

- How is access to the package repository server controlled?
  - Each account is given two API keys.
  - The “full” API key (which should be kept private) allows package metadata associated with the account to be read, listed, added, modified, or deleted.
  - The “read” API key (which may be shared) allows the package metadata associated with the account to be read via the “lookup” or “list” operations.

FAQ (continued)

- How is access to the package file server controlled?
  - Since it is a Fossil instance, managing users is accomplished via the Fossil command line and/or web interface.
  - Generally, the “user name” is actually one of the API keys issued to the account.

FAQ (continued)

- Since Fossil does not currently support per-directory access controls, only public package files should be published to it.

- Private package files can be supported using another (private) instance of Fossil.

Future directions…

- Based on customer feedback:
  - Additional packages will be supported.
  - New features may be added to the client and/or server.

What about open source?

- The **Package Repository Client** is open source, using the Tcl license terms.
- The **Package Repository Server** is closed source (for the time being).

Questions & Answers

Contact Information

- Eyrie Solutions
  sales@eyrie.solutions
- Package Repository Client
  https://eyrie.solutions/cgi-bin/pkgs
- Me (Joe Mistachkin)
  joe@mistachkin.com

EYRIE SOLUTIONS™