NAME

Keywords - files to extend the number of keywords avaiable for plist

DESCRIPTION

pkg-create(8) and pkg-register(8) can parse a plist file, which describes the content of a package to be created/registered. The plist format is documented in the pkg-create(8) manpage .

Keywords are files that extends the number of keywords available to a plist. Those files are in UCL format.

The format is the following:

actions Array of actions which defines what do to the argument(s) of the keywords. Valid actions are:

dir Consider the argument of the keyword as a directory and register it as such in the package

file Consider the argument of the keyword as a regular file and register it as such in the package

arguments

Boolean which tells pkg(8) if it should parse the argument of the keyword or not. If *true* then pkg will make split it using spaces as token and make the argument available to:

actions Any action will now accept a number that will represent the argument passed to the action via parenthesis.

shell and lua scripts

New variables will be available to scripts: %<number>.

attributes

attributes that can be set to a file or a directory depending if actions has been set. It will take the precedence over the attributes that may have been set when calling the keyword. Attributes can be:

owner *string*

Name of the owner of the file or directory.

group *string*

Name of the group of the file or directory.

mode *string*
mode of the file or directory, this mode can be in numeric or string form.

deprecated
Boolean to mark a keyword as deprecated

deprecation_message
Message to be show if the keyword is used and mark as deprecated

preformat_arguments
Boolean to activate the preformating the arguemnts of the keywords repescting the escape sequences descibred below.

prepackaging
lua script which is executed at the packaging time. Useful to add some input validation.

pre-install
shell script to be run during the pre-install phase. It will be merged with any existing pre-install scripts. The script will be formatted respecting the escape sequences define later.

post-install
shell script to be run during the post-install phase. It will be merged with any existing post-install scripts. The script will be formatted respecting the escape sequences define later.

pre-deinstall
shell script to be run during the pre-deinstall phase. It will be merged with any existing pre-deinstall scripts. The script will be formatted respecting the escape sequences define later.

post-deinstall
shell script to be run during the post-deinstall phase. It will be merged with any existing post-deinstall scripts. The script will be formatted respecting the escape sequences define later.

pre-install-lua
Lua script to be run during the pre-install phase. It will be appended with any existing array of lua pre-install scripts The script will be formatted respecting the escape sequences define later.

post-install-lua
Lua script to be run during the post-install phase. It will be appended with any existing array of lua post-install scripts The script will be formatted respecting the escape sequences define later.

pre-deinstall-lua

Lua script to be run during the pre-deinstall phase. It will be appended with any existing array of lua pre-deinstall scripts The script will be formatted respecting the escape sequences define later.

post-deinstall-lua

Lua script to be run during the post-deinstall phase. It will be appended with any existing array of lua post-deinstall scripts The script will be formatted respecting the escape sequences define later.

messages

Array of test messages that can be passed to the users. Valid information by entry in the array are:

message *string*

actual message to be shown to the users.

type [*upgrade* | *remove* | *install*]

defines in which contect the message should be shown to the users. If not set, the message will always be printed

ESCAPE SEQUENCE

If *line* contains any of the following sequences somewhere in it, they will be expanded inline. For the following examples, assume that **@cwd** is set to */usr/local* and the last extracted file was *bin/emacs*.

%F Expands to the last filename extracted (as specified), in the example case *bin/emacs*.

%D Expand to the current directory prefix, as set with **@cwd**, in the example case */usr/local*.

%B Expand to the "basename" of the fully qualified filename, that is the current directory prefix, plus the last filespec, minus the trailing filename. In the example case, that would be */usr/local/bin*.

%f Expand to the filename part of the fully qualified name, or the converse of **%B**, being in the example case, *emacs*.

SEE ALSO

pkg_create(3), pkg_printf(3), pkg_repos(3), pkg-lua-script(5), pkg-repository(5), pkg-script(5), pkg-triggers(5), pkg.conf(5), pkg(8), pkg-add(8), pkg-alias(8), pkg-annotate(8), pkg-audit(8), pkg-autoremove(8), pkg-check(8), pkg-clean(8), pkg-config(8), pkg-create(8), pkg-delete(8), pkg-fetch(8), pkg-info(8), pkg-install(8), pkg-lock(8), pkg-query(8), pkg-register(8), pkg-repo(8), pkg-rquery(8), pkg-search(8), pkg-set(8), pkg-shell(8), pkg-shlib(8), pkg-ssh(8), pkg-stats(8), pkg-triggers(8), pkg-update(8), pkg-updating(8), pkg-upgrade(8), pkg-version(8), pkg-which(8)