Package 'uniset'

March 28, 2022

Type Package

Title Dynamic Settings File

Version 0.3.1

Maintainer Bernhard Pollner <bernhard.pollner@mac.com>

Description Any package (subsequently called 'target package') is enabled to provide its users an easily accessible, user-friendly and human readable text file where key=value pairs (used by functions defined in the target package) can be saved. This settings file lives in a location defined by the user of the target package, and its user-defined values remain unchanged even when the author of the target package is introducing or deleting keys, or when the target package is updated or re-installed.

URL https://bpollner.github.io/uniset/, https://github.com/bpollner/uniset

BugReports https://github.com/bpollner/uniset/issues

Language en-US

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 3.3.2)

Imports methods, easycsv

Suggests covr, knitr, rmarkdown, tidyverse, devtools, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Bernhard Pollner [cre, aut], Zoltan Kovacs [aut]

Repository CRAN

Date/Publication 2022-03-27 23:10:02 UTC

R topics documented:

`uniset` *Dynamic Settings File*

Description

Any package (subsequently called 'target package') is enabled to provide its users an easily accessible, user-friendly and human readable text file where key=value pairs (used by functions defined in the target package) can be saved. This settings file lives in a location defined by the user of the target package, and its user-defined values remain unchanged even when the author of the target package is introducing or deleting keys, or when the target package is updated or re-installed. In order to enable the target package to make use of the functionality offered by package 'uniset', three files have to be exported by 'uniset' and placed into the target package.

Details

There are two ways to generate the files required for the target package to make use of 'uniset':

-) Export and move manually Use `uniset_getFiles`, then move the 'xxx_settings.R' file ('xxx' for the name of the target package) into the 'inst' folder (create one if not already done) of the target package. Move the files 'uniset_globals.R' and 'uniset_functions.R' into the 'R' folder of the target package.

-) Write directly to target package (Recommended) Use `uniset_copyFilesToPackage` to copy the required files directly into the target package.

Every variable defined in the xxx_settings.R file is accessible in the code of the target package.
The target package has to list 'uniset' as an 'import', and then `uniset_updateSettings` or `uniset_autoUpS` can be used to manually or automatically update the settings,i.e. to read in the key=value pairs stored in the xxx_settings.R file. For an introduction and more detailed information please see `https://bpollner.github.io/uniset/`.

Advantage

The most imminent advantage of the 'uniset' settings-file system over using any static file for permanently storing settings for any package is the fact that the key=value pairs in the xxx_settings.R file get updated (added / deleted) dynamically.
So the developer of a package can delete keys or introduce new ones, and the new key=value pairs will be automatically added to or deleted from the local xxx_settings.R file. **Values changed by the user of the target package will be preserved**.
So the author of the target package can add or delete keys from the xxx_settings.R file without worrying that this will cause any effort or troubles for the user of the target package.

Links

Please see https://bpollner.github.io/uniset/ for a practical example; bug reports can be made at https://github.com/bpollner/uniset/issues.

Maintainer

Bernhard Pollner bernhard.pollner@mac.com

Functions for preparing the target package

uniset_copyFilesToPackage, uniset_getFiles

Functions to be called from within the target package

uniset_autoUpS, uniset_updateSettings, uniset_test, uniset_getstn, and only once: uniset_setup.

Examples

As the functions to update the settings file and to (automatically) source this settings are intended to be called from **within** the (installed) target package, please go to https://bpollner.github.io/uniset/ for a walk-through and for a real-life demonstration and examples how to use these functions in the code of the target package.

Author(s)

Bernhard Pollner, Zoltan Kovacs

---

uniset_autoUpS *Automatically update Settings*

Description

Use this function within your code to automatically update the settings from the users settings file

Usage

```
uniset_autoUpS(uniset_handover, setupFunc = NULL)
```

Arguments

`uniset_handover`
: List length two, containing two elements:
  1. `pkgname`: The name of the target package.
  2. `funcname`: The name of the function in the target package handing over the required values. See examples at `uniset`.

`setupFunc`
: Character length one. The name of the function in the target package performing the setup, i.e. the name of the function that is containing the `uniset` function `uniset_setup`. Defaults to 'NULL'; has to be changed.

Details

If 'autoUpdateSettings' in the local settings.R file is left at 'TRUE', the settings will be checked resp. updated automatically every time a function in the target package is calling `uniset_autoUpS`.

Value

Is primarily called for its side effects, i.e to automatically update / (re-)source the settings file. Returns (invisible) 'FALSE' if the the update was unsuccessful, otherwise an (invisible) list with the settings sourced from the settings.R file.

Important

This function is meant to be called from within the target package.

Note

Please refer to `uniset` for a link to examples and a real-world demo.

Examples

```
{
## Not run:
# to be called from within the target package
uniset_autoUpS(uniset_handover, "dogPack_demo_setup")

## End(Not run)
}
```

`uniset_copyFilesToPackage`

Copy Uniset Files into Target Package

Description

Generate the four files required in the target package (i.e. the package that should be enabled to use the package 'uniset'). The generated files will be copied directly into their required destination folders in the target package. The name of the target package will be extracted from the description file.

Usage

```
uniset_copyFilesToPackage(
  pathToPackage,
  setupFunc = NULL,
  taPaSH = "def",
  taPaObj = "settings",
  tmpl = "_TEMPLATE"
)
```

Arguments

pathToPackage	Character length one. The path to the root of the target package.
setupFunc	Character length one. The name of the function **in the target package** that is containing the setup-function uniset_setup.
taPaSH	Character length one. The name of the variable to be defined in the '.Renviron' file, leading to the place where the settings.R file for the target package will be stored. If left at the default 'def', 'taPaName_SH' will be used, with 'taPaName' being the value provided at the argument 'taPaName'.
taPaObj	Character length one. The name of the object holding the list with the key-value pairs that can be defined to be used in the target package. Can be left at the default 'settings'.
tmpl	Character length one. the Character string that will be appended to the fresh settings file that is copied (by the target package) to the users settings home directory if updating the key=value pairs was not successful. Can be left at the default '_TEMPLATE'.

Value

Writes the four required files directly into a valid R-package folder structure. Returns (invisible) NULL.

Note

Please refer to uniset for a link to examples and a real-world demo.

See Also

uniset_getFiles

Examples

```
{
library(uniset)
# first copy the target package example into tempdir
to <- tempdir()
from <- paste0(path.package("uniset"), "/examples/dogPack")
file.copy(from, to, recursive = TRUE)
# now copy the four required files directly into the package 'dogPack'
```

```
path <- paste0(to, "/dogPack")
uniset_copyFilesToPackage(path, "nameOfSetupFunc")
}
```

`uniset_getFiles` *Get Uniset Files*

Description

Function to generate the four files required in the target package (i.e. the package that should be enabled to use the package 'uniset').

Usage

```
uniset_getFiles(
  taPaName = NULL,
  setupFunc = NULL,
  where = NULL,
  taPaSH = "def",
  taPaObj = "settings",
  tmpl = "_TEMPLATE"
)
```

Arguments

`taPaName`	Character length one. The name of the target package.
`setupFunc`	Character length one. The name of the function **in the target package** that is containing the setup-function `uniset_setup`.
`where`	Character length one. The location where the folder with the resulting four files should be copied to. Defaults to 'NULL'. If left at the default 'NULL', the location should be selectable interactively. Provide a character length one holding a valid path to an existing folder to copy the folder containing the four required files there.
`taPaSH`	Character length one. The name of the variable to be defined in the '.Renviron' file, leading to the place where the settings.R file for the target package will be stored. If left at the default 'def', `'taPaName_SH'` will be used, with 'taPaName' being the value provided at the argument 'taPaName'.
`taPaObj`	Character length one. The name of the object holding the list with the key-value pairs that can be defined to be used in the target package. Can be left at the default 'settings'.
`tmpl`	Character length one. the Character string that will be appended to the fresh settings file that is copied (by the target package) to the users settings home directory if updating the key=value pairs was not successful. Can be left at the default '_TEMPLATE'.

Details

Look at the content of the four generated files for information on where they should be moved.

Value

Creates a folder at the location specified at argument 'where' with the four files to be moved into the target package in it. Returns an (invisible) character holding the path of the folder where the three files were written into.

Note

Please refer to uniset for a link to examples and a real-world demo.

See Also

uniset_copyFilesToPackage

Examples

```
{
library(uniset)
# first copy the target package example into tempdir
to <- tempdir()
from <- paste0(path.package("uniset"), "/examples/dogPack")
file.copy(from, to, recursive = TRUE)
# now copy the four required files
uniset_getFiles("dogPack", setupFunc="nameOfSetupFunc", where=to)
# Now manually move the four files according to the instructions contained
# in them.
}
```

---

uniset_getstn *Get Settings Object description Source the list holding the key=value pairs from the settings.R file.*

Description

Get Settings Object description Source the list holding the key=value pairs from the settings.R file.

Usage

```
uniset_getstn(uniset_handover)
```

Arguments

uniset_handover

List length two, containing two elements:

1. pkgname: The name of the target package.
2. funcname: The name of the function in the target package handing over the required values. See examples at uniset.

Value

A list holding the key=value pairs from the settings.R file on on success, NULL if sourcing the file was not successful.

Important

This function is meant to be called from within the target package.

Examples

```
{
## Not run:
# to be called from within the target package
uniset_getstn(uniset_handover)

## End(Not run)
}
```

`uniset_setup` *Perform Setup*

Description

Perform the required setup to enable the target package to make use of the functionality of package 'uniset'. Only has to be called once by the user of the target package.

Usage

```
uniset_setup(where = NULL, uniset_handover)
```

Arguments

`where`	Character length one, holding the path to the location where the folder containing the settings.R file should be located. Defaults to 'NULL'. If left at the default 'NULL', the location should be selectable interactively.
`uniset_handover`	List length two, containing two elements: 1. `pkgname`: The name of the target package. 2. `funcname`: The name of the function in the target package handing over the required values. See examples at `uniset`.

Details

This function is intended to be called from **within** the target package by the user of the target package. Only has to be called once to initiate the system, i.e. to

- Define the folder where the settings.R file will be located,
- Copy the settings.R file into this folder, and
- Create a corresponding entry in the .Renviron file (or create the .Renviron file if does not exist).

This setup has to be done manually (but only once!) by the user of the target package. However, if called repeatedly, it enables the user of the target package to conveniently change the settings-home directory and its corresponding variable in the .Renviron file. In that case, a factory-fresh version of the settings.R file will be copied into the new settings-home directory. For the user-defined values in the 'old' settings.R file not to be lost, the user then has to manually move / copy the settings from the old location to the new one.

Value

Called for its side effects, i.e. to initiate the dynamic settings file system (see Details.) Returns an (invisible) character length one holding the path to the settings-home directory.

Important

This function is meant to be called from within the target package.

Examples

```
{
## Not run:
# to be called from within the target package
uniset_setup(where, uniset_handover)

## End(Not run)
}
```

---

uniset_test *Simple Test*

Description

Test if the input package name etc. was correct / successful. This function is meant to be called from inside the target package.

Usage

```
uniset_test(uniset_handover)
```

Arguments

`uniset_handover`
: List length two, containing two elements:

1. `pkgname`: The name of the target package.
2. `funcname`: The name of the function handing over the required values.

Value

Is printing the parameters defined by the target package, and is returning those parameters in an (invisible) list.

Important

This function is intended to be called from within the target package.

Note

Please refer to `uniset` for a link to examples and a real-world demo.

Examples

```
{
## Not run:
# to be called from within the target package
uniset_test(uniset_handover_pkgname, uniset_handover_funcname)

## End(Not run)
}
```

`uniset_updateSettings` *Update Settings of Target Package*

Description

Manually read in the settings-file in the target package settings home directory as specified in the .Renviron file.

Usage

```
uniset_updateSettings(uniset_handover, setupFunc = NULL, silent = FALSE)
```

Arguments

`uniset_handover`
: List length two, containing two elements:

1. `pkgname`: The name of the target package.
2. `funcname`: The name of the function in the target package handing over the required values. See examples at `uniset`.

`setupFunc`
: Character length one. The name of the function in the target package performing the setup, i.e. the name of the function that is containing the `uniset` function `uniset_setup`. Defaults to 'NULL'; has to be changed.

`silent`
: Logical. If a confirmation should be printed. Defaults to 'FALSE'

Value

This function is called for its side effects, i.e to manually update / (re-)source the settings file. Returns (invisible) 'FALSE' if the update was unsuccessful, otherwise an (invisible) list with the settings sourced from the settings.R file.

Important

This function is meant to be called from within the target package.

Note

Please refer to `uniset` for a link to examples and a real-world demo.

Examples

```
{
## Not run:
# to be called from within the target package
uniset_updateSettings(uniset_handover, "dogPack_demo_setup")

## End(Not run)
}
```

Index