Package: oskeyring (via r-universe)

June 7, 2024

Title Raw System Credential Store Access from R

Version 0.1.6.9000

Description Aims to support all features of the system credential store, including non-portable ones. Supports 'Keychain' on 'macOS', and 'Credential Manager' on 'Windows'. See the 'keyring' package if you need a portable 'API'.

License MIT + file LICENSE

URL https://github.com/r-lib/oskeyring#readme, https://r-lib.github.io/oskeyring/

BugReports https://github.com/r-lib/oskeyring/issues

Depends R (>= 3.6)

Suggests askpass, covr, testthat (>= 3.0.0), withr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://r-lib.r-universe.dev

RemoteUrl https://github.com/r-lib/oskeyring

RemoteRef HEAD

RemoteSha 4fb5b3d70ff6cd45d9c7ba4471751cb02ad29857

Contents

macos_keychain *Query and manipulate the macOS Keychain*

Description

`macos_item_*` functions add, delete, update and search Keychain items.
`macos_keychain_*` functions create, delete, list, lock, unlock keychains.

`macos_item_classes()` lists the supported Keychain item classes. `macos_item_attr()` lists the supported attributes for these classes. `macos_item_match_options()` lists the options supported by the `match` argument of `macos_item_search()`.

Usage

```
macos_item_classes()

macos_item(value, attributes = list(), class = "generic_password")

macos_item_add(item, keychain = NULL)

macos_item_search(
  class = "generic_password",
  attributes = list(),
  match = list(),
  return_data = FALSE,
  keychain = NULL
)

macos_item_update(
  class = "generic_password",
  attributes = list(),
  match = list(),
  update = list(),
  keychain = NULL
)

macos_item_delete(
  class = "generic_password",
  attributes = list(),
  match = list(),
  keychain = NULL
)

macos_keychain_create(keychain, password = NULL)

macos_keychain_list(domain = c("all", "user", "system", "common", "dynamic"))

macos_keychain_delete(keychain)
```

```
macos_keychain_lock(keychain = NULL)

macos_keychain_unlock(keychain = NULL, password = NULL)

macos_keychain_is_locked(keychain = NULL)

macos_item_attr()

macos_item_match_options()
```

Arguments

value	Value of the item, a password, key or certificate. It must a raw vector or a string. If it is a string, then it is converted to UTF-8.
attributes	Narrow the search by indicating the attributes that the found item or items should have.
class	Type of items to search, see macos_item_classes() for possible values.
item	Keychain item, creted via macos_item() or returned by oskeyking itself.
keychain	Keychain to use. NULL means the default one.
match	Condition the search in a variety of ways. For example, you can limit the results to a specific number of items, control case sensitivity when matching string attributes, etc. See 'Search parameters' below.
return_data	Whether to include the secret data in the search result. If this is set to TRUE, then you'll have to set the limit parameter (in the match argument) to a finite value. If this is TRUE, then macOS will prompt you for passwords if necessary. You might get multiple password prompts, if you set limit to a larger than one value.
update	Named list specifying the new values of attributes.
password	Password to unlock the keychain, or new password to set when creating a new keychain. May be NULL in interactive sessions, to force a secure password dialog.
domain	The preference domain from which you wish to retrieve the keychain search list: • "all": include all keychains currently on the search list, • "user": user preference domain, • "system": system or daemon preference domain, • "common": keychains common to everyone, • "dynamic": dynamic search list (typically provided by removable keychains such as smart cards).

Value

macos_item_classes() returns a character vector, the names of the supported keychain item classes.

macos_item() returns a new oskeyring_macos_item object.

`macos_item_add()` returns `NULL`, invisibly.

`macos_item_search()` returns a list of keychain items.

`macos_item_update()` returns `NULL`, invisibly.

`macos_item_delete()` returns `NULL`, invisibly.

`macos_keychain_create()` returns `NULL`, invisibly.

`macos_keychain_list()` returns a data frame with columns:

- `path`: Path to the file of the keychain.
- `is_locked`: Whether the keychain is locked.
- `is_readable`: Whether the keychain is readable by the user.
- `is_writeable`: Whether the keychain is writeable by the user.

`macos_keychain_delete()` returns `NULL`, invisibly.

`macos_keychain_lock()` returns `NULL`, invisibly.

`macos_keychain_unlock()` returns `NULL`, invisibly.

`macos_keychain_is_locked()` returns `TRUE` or `FALSE`.

`macos_item_attr()` returns a list of lists of character scalars, the description of keychain item attributes, for each keychain item class.

`macos_item_match_options()` returns a list of character scalars, the description of the supported match options.

Keychain items

`macos_item_classes()` returns the currently supported Keychain item classes.

```
macos_item_classes()
#> [1] "generic_password"  "internet_password"
```

`macos_item()` creates a new Keychain item. See the next section about the attributes that are supported for the various item types.

```
it <- macos_item("secret", list(service = "My service", account = "Gabor"))
it
#> <oskeyring_macos_item: generic_password>
#>  account: Gabor
#>  service: My service
#>  value: <-- hidden -->
```

`macos_item_add()` adds an item to the keychain. If there is already an item with the same primary keys, then it will error.

```
macos_item_add(it)
```

`macos_item_search()` searches for Keychain items. If `return_data` is `TRUE` then it also returns the secret data. Returning the secret data might create a password entry dialog. If `return_data` is `TRUE` then you need to set the `limit` match condition to a (small) finite number.

```
macos_item_search(attributes = list(service = "My service"))
#> [[1]]
#> <oskeyring_macos_item: generic_password>
#>  account: Gabor
#>  creation_date: 2023-11-03 12:30:13
#>  label: My service
#>  modification_date: 2023-11-03 12:30:13
#>  service: My service
```

`macos_item_update()` updates existing Keychain items.

```
macos_item_update(
  attributes = list(service = "My service", account = "Gabor"),
  update = list(account = "Gabor Csardi")
)
macos_item_search(attributes = list(service = "My service"))
#> [[1]]
#> <oskeyring_macos_item: generic_password>
#>  account: Gabor Csardi
#>  creation_date: 2023-11-03 12:30:13
#>  label: My service
#>  modification_date: 2023-11-03 12:30:13
#>  service: My service
```

`macos_item_delete()` deletes one or more Keychain items. Note that all matching items will be deleted.

```
macos_item_delete(attributes = list(service = "My service"))
macos_item_search(attributes = list(service = "My service"))
#> list()
```

Keychain Item Attributes:

- The set of supported attributes depends on the class of the item.
- oskeyring supports the following item classes currently: generic_password, internet_password.
- A subset of the attributes form a *primary key*. It is not possible to add more than one item with the same primary key. See the primary keys for the various classes below.
- oskeyring does not currently support all attributes that the Keychain Services AIP supports.
- Some attributes are read-only. If you try to set them when adding or updating items, they will be ignored.
- If an attribute is not included in the return value of `macos_item_search()` then it is not set, and its default value is in effect.

Attributes for generic passwords:

- `creation_date`: [.POSIXct(1)][read-only] The date the item was created.
- `modification_date`: [.POSIXct(1)][read-only] The last time the item was updated.
- `description`: [character(1)] User-visible string describing this kind ofitem (for example, 'Disk image password').

- comment: [character(1)] User-editable comment for this item.
- label: [character(1)] User-visible label for this item.
- is_invisible: [logical(1)] TRUE if the item is invisible (that is, should not be displayed).
- is_negative: [logical(1)] Indicates whether there is a valid password associated with this keychain item. This is useful if your application doesn't want a password for some particular service to be stored in the keychain, but prefers that it always be entered by the user.
- account: [character(1)][key] Account name.
- service: [character(1)][key] The service associated with this item.
- generic: [character(1)] User-defined attribute.
- synchronizable: [logical(1)] Indicates whether the item in question is synchronized to other devices through iCloud.

Attributes for internet passwords:

- creation_date: [.POSIXct(1)][read-only] The date the item was created.
- modification_date: [.POSIXct(1)][read-only] The last time the item was updated.
- description: [character(1)] User-visible string describing this kind ofitem (for example, 'Disk image password').
- comment: [character(1)] User-editable comment for this item.
- label: [character(1)] User-visible label for this item.
- is_invisible: [logical(1)] TRUE if the item is invisible (that is, should not be displayed).
- is_negative: [logical(1)] Indicates whether there is a valid password associated with this keychain item. This is useful if your application doesn't want a password for some particular service to be stored in the keychain, but prefers that it always be entered by the user.
- account: [character(1)][key] Account name.
- synchronizable: [logical(1)] Indicates whether the item in question is synchronized to other devices through iCloud.
- security_domain: [character(1)][key] The item's security domain.
- server: [character(1)][key] Contains the server's domain name or IP address.
- protocol: [character(1)][key] The protocol for this item.
- authentication_type: character[1][key] Authentication type.
- port: [integer(1)][key] Internet port number.
- path: [character(1)][key] A path, typically the path component of the URL

Search Parameters

osxkeychain only supports a limited set of search parameters. You can provide these for macos_item_search() as the match argument:

- limit: [numeric(1)] This value specifies the maximum number of results to return or otherwise act upon. Use Inf to specify all matching items.

Keychains

macOs supports multiple keychains. There is always a default keychain, which is the user's login keychain, unless configured differently. There is also a keychain search list. Keychains may belong into four non-exclusive categories, see the domain argument of macos_keychain_list(). A keychain is stored in an encrypted file on the disk, see the first column of the output of macos_keychain_list().

`macos_item_*()` functions have a `keychain` argument to direct or restrict the operation to a single keychain only. These are the defaults:

- `macos_item_add()` adds the item to the default keychain.
- `macos_item_search()` searches all keychains in the search list.
- `macos_item_update()` updates matching items on all keychains in the search list.
- `macos_item_delete()` deletes matching items from all keychains in the search list.

`macos_keychain_create()` creates a new keychain.

`macos_keychain_list()` lists all keychains on the search list.

```
new <- "~/Library/Keychains/test.keychain-db"
macos_keychain_create(new, password = "secret")
macos_keychain_list()
```

```
##                                                      path is_unlocked
## 1 /Users/gaborcsardi/Library/Keychains/login.keychain-db        TRUE
## 2 /Users/gaborcsardi/Library/Keychains/shiny.keychain-db       FALSE
## 3  /Users/gaborcsardi/Library/Keychains/test.keychain-db        TRUE
## 4                     /Library/Keychains/System.keychain       FALSE
##   is_readable is_writeable
## 1        TRUE         TRUE
## 2        TRUE        FALSE
## 3        TRUE         TRUE
## 4        TRUE        FALSE
```

`macos_keychain_lock()` locks a keychain. `macos_keychain_unlock()` unlocks a keychain. `macos_keychain_is_locked` checks if a keychain is locked.

```
macos_keychain_lock(new)
macos_keychain_is_locked(new)
```

```
## [1] TRUE
```

```
macos_keychain_unlock(new, password = "secret")
macos_keychain_is_locked(new)
```

```
## [1] FALSE
```

`macos_keychain_delete()` deletes a keychain: it removes it from the search list and deletes the data from the disk. It currently refuses to delete the user's login keychain and the system keychain. Use Keychain Access instead if you want to delete these. (Only do this if you are aware of the bad consequences.)

```
macos_keychain_delete(new)
macos_keychain_list()
```

```
##                                                          path is_unlocked
## 1 /Users/gaborcsardi/Library/Keychains/login.keychain-db        TRUE
## 2 /Users/gaborcsardi/Library/Keychains/shiny.keychain-db       FALSE
## 3                     /Library/Keychains/System.keychain       FALSE
##   is_readable is_writeable
## 1        TRUE         TRUE
## 2        TRUE        FALSE
## 3        TRUE        FALSE
```

See Also

The Keychain Services API documentation at https://developer.apple.com/documentation/security/keychain_services.

Examples

```
# See above
```

`windows_credentials` *Query and manipulate the Windows Credential Store*

Description

`windows_item_*` functions read, write, delete and list credentials.

Usage

```
windows_item_types()

windows_item(
  credential_blob,
  target_name,
  type = "generic",
  comment = NULL,
  persist = c("local_machine", "session", "enterprise"),
  attributes = list(),
  target_alias = NULL,
  username = NULL
)

windows_item_read(target_name, type = "generic")

windows_item_write(item, preserve = FALSE)

windows_item_delete(target_name, type = "generic")

windows_item_enumerate(filter = NULL, all = FALSE)
```

Arguments

credential_blob
: The secret credential, a password, certificate or key. See also https://learn.microsoft.com/en-us/windows/win32/api/wincred/ This can be a raw vector, or a string. If it is a string, then it will be converted to Unicode, without the terminating zero. It can also be NULL, to be used with the preserve = TRUE argument of windows_item_write().

target_name
: The name of the credential. The target_name and type members uniquely identify the credential. This member cannot be changed after the credential is created. Instead, the credential with the old name should be deleted and the credential with the new name created. This member cannot be longer than CRED_MAX_GENERIC_TARGET_NAME_LENGTH (32767) characters. This member is case-insensitive.

type
: The type of the credential. This member cannot be changed after the credential is created. See windows_item_types() for possible values.

comment
: If not NULL, then a string comment from the user that describes this credential. This member cannot be longer than CRED_MAX_STRING_LENGTH (256) characters. It is stored as a Unicode string.

persist
: Defines the persistence of this credential.
  - "local_machine": The credential persists for all subsequent logon sessions on this same computer. It is visible to other logon sessions of this same user on this same computer and not visible to logon sessions for this user on other computers.
  - "session": The credential persists for the life of the logon session. It will not be visible to other logon sessions of this same user. It will not exist after this user logs off and back on.
  - "enterprise": The credential persists for all subsequent logon sessions on this same computer. It is visible to other logon sessions of this same user on this same computer and to logon sessions for this user on other computers.

attributes
: Application-defined attributes that are associated with the credential. This is NULL or a named list of raw or string vectors. String vectors are converted to Unicode, without the terminating zero. A credential can have at most 64 attributes, the names of the attributes cannot be longer than CRED_MAX_STRING_LENGTH (256) characters each, and the attributes themselves cannot be longer than CRED_MAX_VALUE_SIZE (256) bytes.

target_alias
: Alias for the target_name member. This member can be read and written. It cannot be longer than CRED_MAX_STRING_LENGTH (256) characters. It is stored in Unicode.

username
: NULL or the user name of the account used to connect to target_name.

item
: oskeyring_windows_item object to write.

preserve
: The credential BLOB from an existing credential is preserved with the same credential name and credential type. The credential_blob of the passed oskeyring_windows_item object must be NULL.

filter
: If not NULL, then a string to filter the credentials. Only credentials with a target_name matching the filter will be returned. The filter specifies a name

prefix followed by an asterisk. For instance, the filter "FRED*" will return all credentials with a target_name beginning with the string "FRED".

all — Whether to use the CRED_ENUMERATE_ALL_CREDENTIALS flag to enumerate all credentials. If this is TRUE, then filter must be NULL. If this is TRUE, then the target name of each credential is returned in the "namespace:attribute=target" format.

Details

windows_item_types():

windows_item_types() lists the currently supported credential types.

```
windows_item_types()
#> [1] "generic"                 "domain_password"
#> [3] "domain_certificate"      "domain_visible_password"
```

windows_item():

windows_item() creates a Windows credential, that can be then added to the credential store.

```
it <- windows_item("secret", "my-host-password")
it
#> <oskeyring_windows_item: generic>
#>  target_name: my-host-password
#>  persist: local_machine
#>  credential_blob: <-- hidden -->
```

windows_item_write():

Writes an item to the credential store.

```
windows_item_write(it)
```

windows_item_read():

Reads a credential with the specified type and target_name.

```
windows_item_read("my-host-password")
```

windows_item_enumerate():

List all credentials that match a prefix.

```
windows_item_enumerate(filter = "my-*")
```

windows_item_delete():

Delete a credential:

```
windows_item_delete("my-host-password")
windows_item_enumerate(filter = "my-*")
```

Value

`windows_item_types()` returns a character vector, the currently supported credential types.

`windows_item()` returns an `oskeyring_windows_item` object.

`windows_item_read()` returns an `oskeyring_windows_item` object.

`windows_item_write()` returns `NULL`, invisibly.

`windows_item_delete()` returns `NULL`, invisibly.

`windows_item_enumerate()` returns a list of `oskeyring_windows_item` items.

See Also

The API documentation at https://learn.microsoft.com/en-us/windows/win32/api/wincred/

Examples

```
# See above
```

Index