PyWin32ctypes Documentation

Release 0.2.3.dev1

David Cournapeau, Ioannis Tziakos

Sep 03, 2023

Contents

A reimplementation of pywin32 that is pure python. The default behaviour will try to use cffi (>= 1.3.0), if available, and fall back to using ctypes. Please note that there is no need to have a compiler available on installation or at runtime.

CHAPTER 1

Usage

Example:

```
# Equivalent to 'import win32api' from pywin32.
from win32ctypes.pywin32 import win32api

win32api.LoadLibraryEx(sys.executable, 0, win32api.LOAD_LIBRARY_AS_DATAFILE)
```

Note: Currently pywin32ctypes implements only a very small subset of pywin32, for internal needs at Enthought. We do welcome additional features and PRs, though.

CHAPTER 2

Development setup

The following should be good enough:

```
pip install -r test_requirements.txt
python install -e .
```

Note:

- While pywin32-ctypes should regularly be tested on windows, you can also develop/test on unix by using wine

CHAPTER 3

Reference

3.1 PyWin32 Compatibility Layer

The win32ctypes library provides a compatibility layer with PyWin32. We attempt to keep the signature of the Functions and their behaviour as close as possible to PyWin32. For details and behaviour please refer to the Pywin32 documentation and the related MSDN reference pages.

win32ctypes.pywin32.win32api	A module, encapsulating the Windows Win32 API.
win32ctypes.pywin32.win32cred	Interface to credentials management functions.
win32ctypes.pywin32.pywintypes	A module which supports common Windows types.

3.1.1 win32api

A module, encapsulating the Windows Win32 API.

Functions

BeginUpdateResource(filename, delete)	Get a handle that can be used by the *UpdateResource()*.
EndUpdateResource(handle, discard)	End the update resource of the handle.
EnumResourceLanguages(hModule, lpType, lpName)	List languages of a resource module.
EnumResourceNames(hModule, resType)	Enumerates all the resources of the specified type within a module.
EnumResourceTypes(hModule)	Enumerates resource types within a module.
FreeLibrary(hModule)	Free the loaded dynamic-link library (DLL) module.
GetSystemDirectory()	Get the `System` directory.

Continued on next page

Table 2 – continued from previous page

GetTickCount()	The number of milliseconds that have elapsed since startup
GetWindowsDirectory()	Get the `Windows` directory.
LoadLibraryEx(fileName, handle, flags)	Loads the specified DLL, and returns the handle.
LoadResource(hModule, type, name[, language])	Find and Load a resource component.
UpdateResource(handle, type, name, data[, ...])	Update a resource.

`win32ctypes.pywin32.win32api.`**`BeginUpdateResource`**(*filename*, *delete*)

Get a handle that can be used by the *UpdateResource()*.

Parameters

- **fileName** (*unicode*) – The filename of the module to load.
- **delete** (*bool*) – When true all existing resources are deleted

Returns **result** (*hModule*) – Handle of the resource.

`win32ctypes.pywin32.win32api.`**`EndUpdateResource`**(*handle*, *discard*)

End the update resource of the handle.

Parameters

- **handle** (*hModule*) – The handle of the resource as it is returned by *BeginUpdateResource()*
- **discard** (*bool*) – When True all writes are discarded.

`win32ctypes.pywin32.win32api.`**`EnumResourceLanguages`**(*hModule*, *lpType*, *lpName*)

List languages of a resource module.

Parameters

- **hModule** (*handle*) – Handle to the resource module.
- **lpType** (*str or int*) – The type or id of resource to enumerate.
- **lpName** (*str or int*) – The type or id of resource to enumerate.

Returns **resource_languages** (*list*) – List of the resource language ids.

`win32ctypes.pywin32.win32api.`**`EnumResourceNames`**(*hModule*, *resType*)

Enumerates all the resources of the specified type within a module.

Parameters

- **hModule** (*handle*) – The handle to the module.
- **resType** (*str or int*) – The type or id of resource to enumerate.

Returns **resource_names** (*list*) – The list of resource names (unicode strings) of the specific resource type in the module.

`win32ctypes.pywin32.win32api.`**`EnumResourceTypes`**(*hModule*)

Enumerates resource types within a module.

Parameters **hModule** (*handle*) – The handle to the module.

Returns **resource_types** (*list*) – The list of resource types in the module.

`win32ctypes.pywin32.win32api.`**`FreeLibrary`**(*hModule*)
Free the loaded dynamic-link library (DLL) module.

If necessary, decrements its reference count.

Parameters **handle** (*hModule*) – The handle to the library as returned by the LoadLibrary function.

`win32ctypes.pywin32.win32api.`**`GetSystemDirectory`**()
Get the `System` directory.

Returns **result** (*str*) – The path to the `System` directory.

`win32ctypes.pywin32.win32api.`**`GetTickCount`**()
The number of milliseconds that have elapsed since startup

Returns **counts** (*int*) – The millisecond counts since system startup.

`win32ctypes.pywin32.win32api.`**`GetWindowsDirectory`**()
Get the `Windows` directory.

Returns **result** (*str*) – The path to the `Windows` directory.

`win32ctypes.pywin32.win32api.`**`LoadLibraryEx`**(*fileName*, *handle*, *flags*)
Loads the specified DLL, and returns the handle.

Parameters

- **fileName** (*unicode*) – The filename of the module to load.
- **handle** (*int*) – Reserved, always zero.
- **flags** (*int*) – The action to be taken when loading the module.

Returns **handle** (*hModule*) – The handle of the loaded module

`win32ctypes.pywin32.win32api.`**`LoadResource`**(*hModule*, *type*, *name*, *language=0*)
Find and Load a resource component.

Parameters

- **handle** (*hModule*) – The handle of the module containing the resource. Use None for current process executable.
- **type** (*str or int*) – The type of resource to load.
- **name** (*str or int*) – The name or Id of the resource to load.
- **language** (*int*) – Language to use, default is LANG_NEUTRAL.

Returns **resource** (*bytes*) – The byte string blob of the resource

`win32ctypes.pywin32.win32api.`**`UpdateResource`**(*handle*, *type*, *name*, *data*, *language=0*)
Update a resource.

Parameters

- **handle** (*hModule*) – The handle of the resource file as returned by *BeginUpdateResource()*.
- **type** (*str or int*) – The type of resource to update.
- **name** (*str or int*) – The name or Id of the resource to update.

- **data** (*bytes*) – A bytes like object is expected.

  Note: PyWin32 version 219, on Python 2.7, can handle unicode inputs. However, the data are stored as bytes and it is not really possible to convert the information back into the original unicode string. To be consistent with the Python 3 behaviour of PyWin32, we raise an error if the input cannot be converted to *bytes*.

- **language** (*int*) – Language to use, default is LANG_NEUTRAL.

3.1.2 win32cred

Interface to credentials management functions.

Functions

CredDelete(TargetName, Type[, Flags])	Remove the given target name from the stored credentials.
CredEnumerate([Filter, Flags])	Remove the given target name from the stored credentials.
CredRead(TargetName, Type[, Flags])	Retrieves a stored credential.
CredWrite(Credential[, Flags])	Creates or updates a stored credential.

win32ctypes.pywin32.win32cred.**CredDelete**(*TargetName*, *Type*, *Flags=0*)

Remove the given target name from the stored credentials.

Parameters

- **TargetName** (*unicode*) – The target name to fetch from the keyring.
- **Type** (*int*) – One of the CRED_TYPE_* constants.
- **Flags** (*int*) – Reserved, always use 0.

win32ctypes.pywin32.win32cred.**CredEnumerate**(*Filter=None*, *Flags=0*)

Remove the given target name from the stored credentials.

Parameters

- **Filter** (*unicode*) – Matches credentials' target names by prefix, can be None.
- **Flags** (*int*) – When set to CRED_ENUMERATE_ALL_CREDENTIALS enumerates all of the credentials in the user's credential set but in that case the Filter parameter should be NULL, an error is raised otherwise

Returns **credentials** (*list*) – Returns a sequence of CREDENTIAL dictionaries.

win32ctypes.pywin32.win32cred.**CredRead**(*TargetName*, *Type*, *Flags=0*)

Retrieves a stored credential.

Parameters

- **TargetName** (*unicode*) – The target name to fetch from the keyring.
- **Type** (*int*) – One of the CRED_TYPE_* constants.
- **Flags** (*int*) – Reserved, always use 0.

Returns **credentials** (*dict*) – `None` if the target name was not found or A dictionary corresponding to the PyWin32 `PyCREDENTIAL` structure.

`win32ctypes.pywin32.win32cred.`**`CredWrite`**(*Credential*, *Flags=0*)

Creates or updates a stored credential.

Parameters

- **Credential** (*dict*) – A dictionary corresponding to the PyWin32 `PyCREDENTIAL` structure.
- **Flags** (*int*) – Always pass `CRED_PRESERVE_CREDENTIAL_BLOB` (i.e. 0).

3.1.3 pywintypes

A module which supports common Windows types.

Functions

pywin32error()

`win32ctypes.pywin32.pywintypes.`**`pywin32error`**()

Exceptions

error(*args, **kw)

class `win32ctypes.pywin32.pywintypes.`**`error`**(**args*, ***kw*)

CHAPTER 4

Change Log

4.1 Version 0.2.2

- Use ctypes.set_last_error to avoid race conditions (#122)

4.2 Version 0.2.1

- Use faulthandler when testing and fix discovered errors (#115, #117).
- Fix support for None username in credentials to be consistent in all backends (#99).
- Test also on cp38, cp39, cp310, cp311 and use cp38 for linking (#114, #107, #100).
- Add support for CredEnumerate extending code from @markb-EE (#110, #109, #111)
- Remove support for older python versions < cp36 (#104, #120).

4.3 Version 0.2.0

- Fix syntax error when installing in python 3.7 (#81).
- Support testing on python 3.7 (#82).
- Support testing on python 3.3 and python 3.4 (#77).
- Do not use 2to3 (#75).
- Support lazy wrapping of win32 functions (#67).
- Add CRED_PERSIST constants (#79 contributed by @tivnet).

4.4 Version 0.1.2

(bugfix release)

- Fix implementation for the deprecated api (#64).

4.5 Version 0.1.1

(bugfix release)

- Update Manifest.in entries (#63)
- fix VERSION path in Manifest.in (#62 contributed by @MinchinWeb)

4.6 Version 0.1.0

- Update tests and provide better compatibility with pywin32 for Resource functions
- Fix: Python 3.5 and 3.6 support (#52).
- API additions to allow pywin32-ctypes to work with pyinstaller (#46 and #57 contributed by @virtuald).
- Fix: do not update the global copy of the windows dlls (#42)
- Add documentation and setup automatic builds in ReadTheDocs (#3, #36).
- Add cffi backend to be used when available (#31).
- Fix: EnumResourceTypes and EnumResourceNames would only return ints (#21, #30).
- Restructure package layout to split core wrapping modules from pywin32 emulation (#15, #17).

4.7 Version 0.0.1

7/04/2014

- Python 2.6 support (#13)
- Python 3 support (#12)
- Basic maintenance work (#11, #7)
- Fix error raising to be pywin32 compatible (#8)
- Package rename mini_pywin32 -> pywin32-ctypes
- Add travis-ci integration using wine! (#2)
- Support basic library and resource loading (#1)
- mini_pywin32 is born

CHAPTER 5

LICENSE

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index