wslwinreg Documentation

Release 1.0.6

Rebecca Ann Heineman

Dec 25, 2022

CONTENTS

The `wslwinreg` module is a drop in replacement for winreg for systems running under Cygwin, MSYS2, or Windows Subsystem for Linux.

- Documentation is found at https://wslwinreg.readthedocs.io
- Doxygen generated documentation is found at https://wslwinreg.readthedocs.io/en/latest/doxygen
- Python Packing Index (PyPI): https://pypi.python.org/pypi/wslwinreg
- Source code and issue tracker: https://github.com/burgerbecky/wslwinreg

CHAPTER

ONE

COMPATIBILITY

- Python 2.7.1 or higher
- Python 3.4 or higher

CHAPTER

TWO

INSTALLATION

Type in `pip install -U wslwinreg`. Some platforms may require the `sudo` prefix.

CHAPTER

THREE

BUGS

If you find a bug, issue or have a feature request, please submit a bug report by emailing becky@burgerbecky.com and mention python version, integer size (32 bit or 64 bit) and what platform was used (Windows / MSYS2 / Cygwin / Windows Subsystem for Linux).

CHAPTER

FOUR

TABLE OF CONTENTS

4.1 Why wslwinreg?

Python runs on several Windows hosted platforms such as Windows itself, Cygwin, MSYS2 and Windows Subsystem for Linux (WSL). Each of these platforms has a Windows Subsystem which contains the Windows registry. Under Windows native versions of Python, the module `winreg` allows access to the Windows Registry API. However, this API exists under the other three platforms, but winreg is not available.

That's where `wslwinreg` comes in!

This module will grant access to all `winreg` calls on Windows native platforms, either by passing through calls to `winreg` or implementing custom code to grant access.

4.1.1 Windows support

On Windows native platform, `winreg` is directly imported and all calls simply are passed through so the native implementation is used instead of the alternate versions.

4.1.2 Cygwin and MSYS2 support

CDLL on Cygwin and MSYS2 versions of Python allow access to the underlying native Windows APIs, so `wslwinreg` implements `winreg` calls through Python code that calls the native Windows API as needed.

4.1.3 Windows Subsystem for Linux support

Ubuntu and other distributions of Linux running under WSL run in a virtual machine that is isolated from the Window host machine to the point that native calls are not possible. To solve this issue, a Windows native backend is launched and through a loopback TCP/IP socket, communication is established so remote procedure calls can be issued and results returned to `wslwinreg` to allow access to the native Windows registry.

4.1.4 Testing

A unit test package, lovingly ripped off from Python 3.9.1 is run on all four platforms to ensure the same behavior is exhibited across all supported platforms. It is located in the folder `unittests` in the github source code tree.

4.1.5 Building the backend

To build the C++ backend, use Visual Studio 2022 with the 2017 XP compiler for x86 and x64 compilation. ARM and ARM64 are built with the 2022 C++ tool chain. The Release target will place the executables in the wslwinreg/bin folder for you. This backend is only used when running `wslwinreg` under WSL. It's not used on any other platform.

4.1.6 Credits

`wslwinreg` is the insane creation of Rebecca Ann Heineman. If bugs are found, please send all information on how to recreate the bug to becky@burgerbecky.com

4.2 Constants

4.2.1 wslwinreg constants

These values are set up on instantiation of the `wslwinreg` module.

PY2

`wslwinreg.common.`**`PY2 = 2`**

True if the interpreter is Python 2.x.

IS_LINUX

`wslwinreg.common.`**`IS_LINUX = sys.platform.startswith("linux")`**

Running on linux?

IS_CYGWIN

`wslwinreg.common.`**`IS_CYGWIN = sys.platform.startswith("cygwin")`**

Running on Cygwin.

IS_MSYS

`wslwinreg.common.IS_MSYS = sys.platform.startswith("msys")`

Running on MSYS.

IS_WSL

`wslwinreg.common.IS_WSL = IS_LINUX and "icrosoft" in platform.platform()`

Running on Windows Subsystem for Linux.

4.2.2 Windows constants

These integers are used to pass to Windows Registry calls.

ERROR_SUCCESS

`wslwinreg.common.ERROR_SUCCESS = 0x00000000`

The operation completed successfully.

ERROR_FILE_NOT_FOUND

`wslwinreg.common.ERROR_FILE_NOT_FOUND = 0x00000002`

The system cannot find the file specified.

ERROR_MORE_DATA

`wslwinreg.common.ERROR_MORE_DATA = 0x000000ea`

More data is available.

HKEY_CLASSES_ROOT

`wslwinreg.common.HKEY_CLASSES_ROOT = 0x80000000`

Registry entries subordinate to this key define types (or classes) of documents and the properties associated with those types.

HKEY_CURRENT_USER

`wslwinreg.common.HKEY_CURRENT_USER = 0x80000001`

Registry entries subordinate to this key define the preferences of the current user.

HKEY_LOCAL_MACHINE

`wslwinreg.common.HKEY_LOCAL_MACHINE = 0x80000002`

Registry entries subordinate to this key define the physical state of the computer, including data about the bus type, system memory, and installed hardware and software.

HKEY_USERS

`wslwinreg.common.HKEY_USERS = 0x80000003`

Registry entries subordinate to this key define the default user configuration for new users on the local computer and the user configuration for the current user.

HKEY_PERFORMANCE_DATA

`wslwinreg.common.HKEY_PERFORMANCE_DATA = 0x80000004`

Registry entries subordinate to this key reference the text strings that describe counters in US English.

HKEY_CURRENT_CONFIG

`wslwinreg.common.HKEY_CURRENT_CONFIG = 0x80000005`

Contains information about the current hardware profile of the local computer system.

HKEY_DYN_DATA

`wslwinreg.common.HKEY_DYN_DATA = 0x80000006`

Windows registry hive that contains information about hardware devices, including Plug and Play and network performance statistics.

KEY_QUERY_VALUE

`wslwinreg.common.KEY_QUERY_VALUE = 0x00000001`

Required to query the values of a registry key.

KEY_SET_VALUE

`wslwinreg.common.KEY_SET_VALUE = 0x00000002`

Required to create, delete, or set a registry value.

KEY_CREATE_SUB_KEY

`wslwinreg.common.KEY_CREATE_SUB_KEY = 0x00000004`

Required to create a subkey of a registry key.

KEY_ENUMERATE_SUB_KEYS

`wslwinreg.common.KEY_ENUMERATE_SUB_KEYS = 0x00000008`

Required to enumerate the subkeys of a registry key.

KEY_NOTIFY

`wslwinreg.common.KEY_NOTIFY = 0x00000010`

Required to request change notifications for a registry key or for subkeys of a registry key.

KEY_CREATE_LINK

`wslwinreg.common.KEY_CREATE_LINK = 0x00000020`

Reserved for system use.

KEY_WOW64_32KEY

`wslwinreg.common.KEY_WOW64_32KEY = 0x00000200`

Indicates that an application on 64-bit Windows should operate on the 32-bit registry view.

KEY_WOW64_64KEY

`wslwinreg.common.KEY_WOW64_64KEY = 0x00000100`

Indicates that an application on 64-bit Windows should operate on the 64-bit registry view.

KEY_WOW64_RES

`wslwinreg.common.KEY_WOW64_RES = 0x00000300`

Mask for *common.KEY_WOW64_32KEY* or'd with *common.KEY_WOW64_64KEY*.

KEY_WRITE

`wslwinreg.common.KEY_WRITE = 0x00020006`

Combines the STANDARD_RIGHTS_WRITE, *common.KEY_SET_VALUE*, and *common.KEY_CREATE_SUB_KEY* access rights.

KEY_EXECUTE

`wslwinreg.common.KEY_EXECUTE = 0x00020019`

Equivalent to KEY_READ.

KEY_READ

`wslwinreg.common.KEY_READ = 0x00020019`

Combines the STANDARD_RIGHTS_READ, *common.KEY_QUERY_VALUE*, *common.KEY_ENUMERATE_SUB_KEYS*, and *common.KEY_NOTIFY* values.

KEY_ALL_ACCESS

`wslwinreg.common.KEY_ALL_ACCESS = 0x000f003f`

Combines the STANDARD_RIGHTS_REQUIRED, *common.KEY_QUERY_VALUE*, *common.KEY_SET_VALUE*, *common.KEY_CREATE_SUB_KEY*, *common.KEY_ENUMERATE_SUB_KEYS*, *common.KEY_NOTIFY*, and *common.KEY_CREATE_LINK* access rights.

REG_OPTION_RESERVED

`wslwinreg.common.REG_OPTION_RESERVED = 0x00000000`

Default key option, same as *common.REG_OPTION_NON_VOLATILE*.

REG_OPTION_NON_VOLATILE

`wslwinreg.common.REG_OPTION_NON_VOLATILE = 0x00000000`

This key is not volatile; this is the default.

REG_OPTION_VOLATILE

`wslwinreg.common.REG_OPTION_VOLATILE = 0x00000001`

All keys created by the function are volatile.

REG_OPTION_CREATE_LINK

`wslwinreg.common.REG_OPTION_CREATE_LINK = 0x00000002`

This key is a symbolic link.

REG_OPTION_BACKUP_RESTORE

`wslwinreg.common.REG_OPTION_BACKUP_RESTORE = 0x00000004`

If this flag is set, the function ignores the samDesired parameter and attempts to open the key with the access required to backup or restore the key.

REG_OPTION_OPEN_LINK

`wslwinreg.common.REG_OPTION_OPEN_LINK = 0x00000008`

The key to be opened is a symbolic link.

REG_LEGAL_OPTION

`wslwinreg.common.REG_LEGAL_OPTION = REG_OPTION_RESERVED | \ REG_OPTION_NON_VOLATILE | \ REG_OPTION_VOLATILE | \ REG_OPTION_CREATE_LINK | \ REG_OPTION_BACKUP_RESTORE | \ REG_OPTION_OPEN_LINK`

Mask for all registry key option flags.

REG_CREATED_NEW_KEY

`wslwinreg.common.REG_CREATED_NEW_KEY = 0x00000001`

The key did not exist and was created.

REG_OPENED_EXISTING_KEY

`wslwinreg.common.REG_OPENED_EXISTING_KEY = 0x00000002`

The key existed and was simply opened without being changed.

REG_WHOLE_HIVE_VOLATILE

`wslwinreg.common.REG_WHOLE_HIVE_VOLATILE = 0x00000001`

If specified, a new, volatile (memory only) set of registry information, or hive, is created.

REG_REFRESH_HIVE

`wslwinreg.common.REG_REFRESH_HIVE = 0x00000002`

If set, the location of the subtree that the hKey parameter points to is restored to its state immediately following the last flush.

REG_NO_LAZY_FLUSH

`wslwinreg.common.REG_NO_LAZY_FLUSH = 0x00000004`

If set, disable lazy flushing.

REG_NOTIFY_CHANGE_NAME

`wslwinreg.common.REG_NOTIFY_CHANGE_NAME = 0x00000001`

Notify the caller if a subkey is added or deleted.

REG_NOTIFY_CHANGE_ATTRIBUTES

`wslwinreg.common.REG_NOTIFY_CHANGE_ATTRIBUTES = 0x00000002`

Notify the caller of changes to the attributes of the key, such as the security descriptor information.

REG_NOTIFY_CHANGE_LAST_SET

`wslwinreg.common.REG_NOTIFY_CHANGE_LAST_SET = 0x00000004`

Notify the caller of changes to a value of the key.

REG_NOTIFY_CHANGE_SECURITY

`wslwinreg.common.REG_NOTIFY_CHANGE_SECURITY = 0x00000008`

Notify the caller of changes to the security descriptor of the key.

REG_LEGAL_CHANGE_FILTER

`wslwinreg.common.REG_LEGAL_CHANGE_FILTER = REG_NOTIFY_CHANGE_NAME | \ REG_NOTIFY_CHANGE_ATTRIBUTES | \ REG_NOTIFY_CHANGE_LAST_SET | \ REG_NOTIFY_CHANGE_SECURITY`

Mask for all REG_NOTIFY flags.

REG_NONE

`wslwinreg.common.REG_NONE = 0x000000000`

No defined value type.

REG_SZ

`wslwinreg.common.REG_SZ = 0x000000001`

A null-terminated string.

REG_EXPAND_SZ

`wslwinreg.common.REG_EXPAND_SZ = 0x000000002`

A null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%")

REG_BINARY

`wslwinreg.common.REG_BINARY = 0x000000003`

Binary data in any form.

REG_DWORD

`wslwinreg.common.REG_DWORD = 0x000000004`

A 32-bit number.

REG_DWORD_LITTLE_ENDIAN

`wslwinreg.common.REG_DWORD_LITTLE_ENDIAN = 0x000000004`

A 32-bit number in little-endian format.

REG_DWORD_BIG_ENDIAN

`wslwinreg.common.REG_DWORD_BIG_ENDIAN = 0x000000005`

A 32-bit number in big-endian format.

REG_LINK

`wslwinreg.common.REG_LINK = 0x000000006`

A null-terminated Unicode string that contains the target path of a symbolic link that was created by calling the RegCreateKeyEx function with *common.REG_OPTION_CREATE_LINK*.

REG_MULTI_SZ

`wslwinreg.common.REG_MULTI_SZ = 0x000000007`

A sequence of null-terminated strings, terminated by an empty string (\0).

REG_RESOURCE_LIST

`wslwinreg.common.REG_RESOURCE_LIST = 0x000000008`

Device-driver resource list.

REG_FULL_RESOURCE_DESCRIPTOR

`wslwinreg.common.REG_FULL_RESOURCE_DESCRIPTOR = 0x000000009`

A list of hardware resources that a physical device is using, detected and written into the \HardwareDescription tree by the system.

REG_RESOURCE_REQUIREMENTS_LIST

`wslwinreg.common.REG_RESOURCE_REQUIREMENTS_LIST = 0x00000000a`

A device driver's list of possible hardware resources it or one of the physical devices it controls can use, from which the system writes a subset into the \ResourceMap tree.

REG_QWORD

`wslwinreg.common.REG_QWORD = 0x0000000b`

A 64-bit number.

REG_QWORD_LITTLE_ENDIAN

`wslwinreg.common.REG_QWORD_LITTLE_ENDIAN = 0x0000000b`

A 64-bit number in little-endian format.

FORMAT_MESSAGE_ALLOCATE_BUFFER

`wslwinreg.common.FORMAT_MESSAGE_ALLOCATE_BUFFER = 256`

The function allocates a buffer large enough to hold the formatted message, and places a pointer to the allocated buffer at the address specified by lpBuffer.

FORMAT_MESSAGE_IGNORE_INSERTS

`wslwinreg.common.FORMAT_MESSAGE_IGNORE_INSERTS = 512`

Insert sequences in the message definition such as %1 are to be ignored and passed through to the output buffer unchanged.

FORMAT_MESSAGE_FROM_STRING

`wslwinreg.common.FORMAT_MESSAGE_FROM_STRING = 1024`

The lpSource parameter is a pointer to a null-terminated string that contains a message definition.

FORMAT_MESSAGE_FROM_HMODULE

`wslwinreg.common.FORMAT_MESSAGE_FROM_HMODULE = 2048`

The lpSource parameter is a module handle containing the message-table resource(s) to search.

FORMAT_MESSAGE_FROM_SYSTEM

`wslwinreg.common.FORMAT_MESSAGE_FROM_SYSTEM = 4096`

The function should search the system message-table resource(s) for the requested message.

FORMAT_MESSAGE_ARGUMENT_ARRAY

`wslwinreg.common.FORMAT_MESSAGE_ARGUMENT_ARRAY = 8192`

The Arguments parameter is not a va_list structure, but is a pointer to an array of values that represent the arguments.

FORMAT_MESSAGE_MAX_WIDTH_MASK

`wslwinreg.common.FORMAT_MESSAGE_MAX_WIDTH_MASK = 255`

The function ignores regular line breaks in the message definition text.

LANG_NEUTRAL

`wslwinreg.common.LANG_NEUTRAL = 0x00`

String has no associated language.

SUBLANG_DEFAULT

`wslwinreg.common.SUBLANG_DEFAULT = 0x01`

User default sub language.

4.2.3 Windows C++ data types

These classes are used to describe the parameters to Windows functions.

LPCVOID

`wslwinreg.common.LPCVOID = c_void_p`

`const void *` C void pointer

LPVOID

`wslwinreg.common.LPVOID = c_void_p`

`void *` C void pointer

BOOL

`wslwinreg.common.BOOL = c_long`

BOOL 32 bit C integer type.

WORD

`wslwinreg.common.WORD = c_ushort`

`short int` 16 bit C integer type

DWORD

`wslwinreg.common.DWORD = c_ulong`

`unsigned int` 32 bit C integer type

PDWORD

`wslwinreg.common.PDWORD = POINTER(DWORD)`

`unsigned int *` 32 bit C integer pointer

LPDWORD

`wslwinreg.common.LPDWORD = PDWORD`

`FAR unsigned int *` 32 bit C pointer

QWORD

`wslwinreg.common.QWORD = c_ulonglong`

`unsigned long long` 64 bit C integer type

PQWORD

`wslwinreg.common.PQWORD = POINTER(QWORD)`

`unsigned long long *` 64 bit C integer pointer

LPQWORD

`wslwinreg.common.LPQWORD = PQWORD`

`FAR unsigned long long *` 64 bit C integer pointer

LONG

`wslwinreg.common.LONG = c_int`

`signed int` 32 bit signed C integer type

PLONG

`wslwinreg.common.PLONG = POINTER(LONG)`

`signed int *` 32 bit signed C integer pointer

PBYTE

`wslwinreg.common.PBYTE = c_char_p`

`signed char` 8 bit signed C integer type

LPBYTE

`wslwinreg.common.LPBYTE = PBYTE`

`unsigned char` 8 bit unsigned C integer type

LPSTR

`wslwinreg.common.LPSTR = c_char_p`

`char *` C string pointer

LPWSTR

`wslwinreg.common.LPWSTR = c_wchar_p`

`wchar_t *` C string pointer (16-bit)

LPCWSTR

`wslwinreg.common.LPCWSTR = LPWSTR`

`FAR wchar_t *` C string pointer (16-bit)

HANDLE

`wslwinreg.common.HANDLE = c_void_p`

HANDLE C data type from Windows (void *)

HKEY

`wslwinreg.common.HKEY = HANDLE`

HKEY C data type for Windows registry keys (void *)

PHKEY

`wslwinreg.common.PHKEY = POINTER(HKEY)`

HKEY * C pointer to Windows HKEY (void **)

HLOCAL

`wslwinreg.common.HLOCAL = c_void_p`

HLOCAL Windows HLOCAL data type (void *)

REGSAM

`wslwinreg.common.REGSAM = c_uint`

REGSAM Windows REGSAM C data type (unsigned int)

PFILETIME

`wslwinreg.common.PFILETIME = POINTER(FILETIME)`

FILETIME * pointer to Windows *FILETIME* structure

4.3 Classes

4.3.1 FILETIME

wslwinreg.common.FILETIME : public Structure

Structure to mimic the Windows *FILETIME* data type.

Parameters

None –

4.3.2 WindowsError

wslwinreg.cygwinapi.WindowsError : public OSError

This exception doesn't exist in Cygwin/MSYS, provide it.

Public Functions

__init__(*self*, *winerror*, *strerror=None*, *filename=None*)

Initialize a *WindowsError* exception.

Parameters

- **winerror** – The windows error code
- **strerror** – The string describing the error
- **filename** – Name of the file that caused this error, if applicable.

__str__(*self*)

Convert the class into a string.

Returns

String describing the error contained.

Public Members

winerror

Windows error code.

errno

Linux style error code.

strerror

Description string of the error.

filename

Name of the responsible file for this error.

4.3.3 PyHKEY

class PyHKEY

A Python object representing a win32 registry key.

This object wraps a Windows HKEY object, automatically closing it when the object is destroyed. To guarantee cleanup, you can call either the *Close()* method on the object, or the CloseKey() function.

All registry functions in this module return one of these objects.

All registry functions in this module which accept a handle object also accept an integer, however, use of the handle object is encouraged.

Handle objects provide semantics for **bool**() – thus

```
if handle:
    print("Yes")
```

will print Yes if the handle is currently valid (has not been closed or detached).

The object also support comparison semantics, so handle objects will compare true if they both reference the same underlying Windows handle value.

Handle objects can be converted to an integer (e.g., using the built-in int() function), in which case the underlying Windows handle value is returned. You can also use the *Detach()* method to return the integer handle, and also disconnect the Windows handle from the handle object.

Public Functions

__init__(*self*, *hkey*, *null_ok=False*)

Initialize the *PyHKEY* class.

Parameters

- **hkey** – Integer value representing the pointer to the HKEY
- **null_ok** – True if None is acceptable.

__del__(*self*)

Called when this object is garbage collected.

Close(*self*)

Closes the underlying Windows handle.

Note: If the handle is already closed, no error is raised.

Detach(*self*)

Detaches the Windows handle from the handle object.

After calling this function, the handle is effectively invalidated, but the handle is not closed. You would call this function when you need the underlying win32 handle to exist beyond the lifetime of the handle object.

On 64 bit windows, the result of this function is a long integer.

Note: The result is the value of the handle before it is detached. If the handle is already detached, this will return zero.

Returns

Previous HKEY integer.

__enter__(*self*)

Called when object is entered.

Note: Needed for the Python `with` statement.

__exit__(*self*, *exc_type*, *exc_value*, *exc_traceback*)

Release handle on class destruction.

Note: Needed for the Python `with` statement.

__hash__(*self*)

Convert the object into a hash.

Note: The implementation returns the id, which should be random enough.

__int__(*self*)

Converting a handle to an integer returns the Win32 handle.

__nonzero__(*self*)

Handles with an open object return true, otherwise false.

__bool__(*self*)

Handles with an open object return true, otherwise false.

__repr__(*self*)

Return descriptive string for the class object.

__str__(*self*)

Return short string for the handle object.

Public Members

hkey

Integer that represents the HANDLE pointer.

Public Static Functions

make(*hkey*, *null_ok=None*)

Convert a pointer into a *PyHKEY* object.

4.3.4 WinRegKey

class WinRegKey

Registry key helper class.

Manage entries in a registy key by using indexing and scanning.

Public Functions

__init__(*self*, *root_key*, *subkey*, *access*)

Initialize the class.

Parameters

- **root_key** – Root key enumeration
- **subkey** – String or None for name of sub key
- **access** – Access flags to pass to OpenKeyEx()

__enter__(*self*)

Enable enter/exit functionality.

__exit__(*self*, *exception_type*, *exception_value*, *traceback*)

Release resources on class release.

Parameters

- **exception_type** – Ignored
- **exception_value** – Ignored
- **traceback** – Ignored

close(*self*)

Release existing key, if any.

open_subkey(*self*, *subkey*, *access=None*)

Open a sub key.

Parameters

- **subkey** – Name of the subkey to open
- **access** – Desired access to the subkey.

Returns

An instance of the *WinRegKey* of the new key.

get_subkeys(*self*)

Return the list of all of the sub key names.

```
Iterate over all of the sub keys and place their names in a list.
Return the list. Unicode is properly handled.
```

Returns

list of names of all the subkeys.

get_value(*self*, *value_name=None*)

Read a value from a registry key.

Note: If the type returned is REG_SZ, scan for a null and trucate. If the type returned is REG_EXPAND_SZ, call

```
ExpandEnvironmentStrings()
```

Parameters
value_name – String of the name of the value, None for default.

Returns
Value returned from QueryValueEx()

get_all_values(*self*)

Return a dict of all key name and associated values.

Returns
dict with each item is the returned value from QueryValueEx()

__getitem__(*self*, *subkey*)

Call *open_subkey()* with subscript.

```
Uses the access flags from this key to open the sub key.
```

Parameters
subkey – Requested sub key.

Returns
An instance of the *WinRegKey* of the new key.

__iter__(*self*)

Iterator of the sub keys.

```
Call get_subkeys() and encapsulate the value in an iter()
```

Returns
iter of *get_subkeys()*

Public Members

key

Key handle of opened key.

access

Access used to open this key, used as default to open sub keys.

4.4 Functions

4.4.1 Helper functions

wslwinreg.common.winerror_to_errno

`wslwinreg.common.winerror_to_errno(winerror)`

Convert a Windows error code into the matching errno code.

Parameters
winerror – Integer error code number returned by Windows

Returns
Corresponding errno error code or EINVAL if no match.

wslwinreg.common.convert_to_utf16

`wslwinreg.common.convert_to_utf16(input_string)`

Convert the input string into utf-16-le.

Parameters
input_string – string encoded in locale.getpreferredencoding()

Returns
Byte array in little endian UTF-16

wslwinreg.common.to_registry_bytes

`wslwinreg.common.to_registry_bytes(value, typ)`

Convert input data into appropriate Windows registry type.

Exception
`ValueError` for invalid input or `TypeError` for bad type.

Parameters

- **value** – Value to convert
- **typ** – Windows registry type to convert to (Example REG_DWORD)

Returns
A ctypes.c_char array.

wslwinreg.common.from_registry_bytes

`wslwinreg.common.from_registry_bytes(input_data, input_size, typ)`

Convert raw Windows registry data into an appropriate Python object.

Parameters

- **input_data** – Raw binary data
- **input_size** – Size in bytes of the input data
- **typ** – Windows registry type to convert from (Example REG_DWORD)

Returns

Data converted to appropriate Python object, or None.

wslwinreg.get_HKCU

`wslwinreg.get_HKCU()`

Open HKEY_CURRENT_USER.

Returns

Read only *WinRegKey* for HKEY_CURRENT_USER

wslwinreg.get_HKLM_32

`wslwinreg.get_HKLM_32()`

Open HKEY_LOCAL_MACHINE, 32 bit view.

Returns

Read only *WinRegKey* for HKEY_LOCAL_MACHINE with KEY_WOW64_32KEY

wslwinreg.get_HKLM_64

`wslwinreg.get_HKLM_64()`

Open HKEY_LOCAL_MACHINE, 64 bit view.

Returns

Read only *WinRegKey* for HKEY_LOCAL_MACHINE with KEY_WOW64_64KEY

4.4.2 Null implementation

On operating systems such as macOS and Linux that doesn't have a Windows operating system underpinning, all functions will raise a `NotImplementedError` exception.

wslwinreg.nullapi.CloseKey

`wslwinreg.nullapi.CloseKey(hkey)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.ConnectRegistry

`wslwinreg.nullapi.ConnectRegistry(computer_name, key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.CreateKey

`wslwinreg.nullapi.CreateKey(key, sub_key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.CreateKeyEx

`wslwinreg.nullapi.CreateKeyEx(key, sub_key, reserved=0, access=KEY_WRITE)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.DeleteKey

`wslwinreg.nullapi.DeleteKey(key, sub_key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.DeleteKeyEx

`wslwinreg.nullapi.DeleteKeyEx(key, sub_key, access=KEY_WOW64_64KEY, reserved=0)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.DeleteValue

`wslwinreg.nullapi.DeleteValue(key, value)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.EnumKey

`wslwinreg.nullapi.EnumKey(key, index)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.EnumValue

`wslwinreg.nullapi.EnumValue(key, index)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.ExpandEnvironmentStrings

`wslwinreg.nullapi.ExpandEnvironmentStrings(str)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.FlushKey

`wslwinreg.nullapi.FlushKey(key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.LoadKey

`wslwinreg.nullapi.LoadKey(key, sub_key, file_name)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.OpenKey

`wslwinreg.nullapi.OpenKey(key, sub_key, reserved=0, access=KEY_READ)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.OpenKeyEx

`wslwinreg.nullapi.OpenKeyEx(key, sub_key, reserved=0, access=KEY_READ)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.QueryInfoKey

`wslwinreg.nullapi.QueryInfoKey(key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.QueryValue

`wslwinreg.nullapi.QueryValue(key, sub_key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.QueryValueEx

`wslwinreg.nullapi.QueryValueEx(key, value_name)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.SaveKey

`wslwinreg.nullapi.SaveKey(key, file_name)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.SetValue

`wslwinreg.nullapi.SetValue(key, sub_key, type, value)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.SetValueEx

`wslwinreg.nullapi.SetValueEx(key, value_name, reserved, type, value)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.DisableReflectionKey

`wslwinreg.nullapi.DisableReflectionKey(key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.EnableReflectionKey

`wslwinreg.nullapi.EnableReflectionKey(key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.QueryReflectionKey

`wslwinreg.nullapi.QueryReflectionKey(key)`

Not implemented.

Exception

`NotImplementedError` is always thrown.

wslwinreg.nullapi.convert_to_windows_path

`wslwinreg.nullapi.convert_to_windows_path(path_name)`

Convert pathname to Windows.

This is the null function, it returns the path_name unchanged.

Return

path_name as is.

Parameters

path_name – Absolute Windows pathname

wslwinreg.nullapi.convert_from_windows_path

`wslwinreg.nullapi.convert_from_windows_path(path_name)`

Convert pathname from Windows.

This is the null function, it returns the path_name unchanged.

Return

path_name as is.

Parameters
path_name – Absolute Windows pathname

wslwinreg.nullapi.get_file_info

`wslwinreg.nullapi.get_file_info(path_name, string_name)`

Not implemented.

Return
None

Parameters

- **path_name** – Name of the windows file.
- **string_name** – Name of the data chunk to retrieve

4.4.3 Cygwin / MSYS2 implementation

On Cygwin and MSYS2 platforms, the `CDLL` exposes the Windows API directly so these python functions mimic the C code from Python for Windows and calls the Windows API to perform the low level work.

wslwinreg.cygwinapi.CloseKey

`wslwinreg.cygwinapi.CloseKey(hkey)`

Closes a previously opened registry key.

The hkey argument specifies a previously opened key.

Note: If hkey is not closed using this method (or via hkey.Close()), it is closed when the hkey object is destroyed by Python.

Parameters
hkey – *PyHKEY* object or None.

wslwinreg.cygwinapi.ConnectRegistry

`wslwinreg.cygwinapi.ConnectRegistry(computer_name, key)`

Establishes a connection to a predefined registry handle.

Establishes a connection to a predefined registry handle on another computer, and returns a handle object.

Exception
WindowsError or `FileNotFileError`

Parameters

- **computer_name** – Is the name of the remote computer, of the form `r"\\computername"`. If `None`, the local computer is used.

- **key** – Is the predefined handle to connect to.

Returns
PyHKEY object

wslwinreg.cygwinapi.CreateKey

`wslwinreg.cygwinapi.CreateKey(key, sub_key)`

Creates or opens the specified key.

If key is one of the predefined keys, sub_key may be None. In that case, the handle returned is the same key handle passed in to the function.

If the key already exists, this function opens the existing key.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or one of the predefined HKEY_* constants.
- **sub_key** – Is a string that names the key this method opens or creates.

Returns
Handle of the opened key.

wslwinreg.cygwinapi.CreateKeyEx

`wslwinreg.cygwinapi.CreateKeyEx(key, sub_key, reserved=0, access=KEY_WRITE)`

Creates or opens the specified key.

If key is one of the predefined keys, sub_key may be None. In that case, the handle returned is the same key handle passed in to the function.

If the key already exists, this function opens the existing key.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or one of the predefined HKEY_* constants
- **sub_key** – Is a string that names the key this method opens or creates.
- **reserved** – Is a reserved integer, and must be zero. The default is zero.
- **access** – Os an integer that specifies an access mask that describes the desired security access for the key. Default is *common.KEY_WRITE*.

Returns
Handle of the opened key.

wslwinreg.cygwinapi.DeleteKey

`wslwinreg.cygwinapi.DeleteKey(key, sub_key)`

Deletes the specified key.

If the method succeeds, the entire key, including all of its values, is removed.

Exception
WindowsError

Note: This method can not delete keys with subkeys.

Parameters

- **key** – Is an already open key, or one of the predefined HKEY_* constants.
- **sub_key** – Is a string that must be a subkey of the key identified by the key parameter. This value must not be `None`, and the key may not have subkeys.

wslwinreg.cygwinapi.DeleteKeyEx

`wslwinreg.cygwinapi.DeleteKeyEx(key, sub_key, access=KEY_WOW64_64KEY, reserved=0)`

Deletes the specified key.

If the method succeeds, the entire key, including all of its values, is removed.

Exception
WindowsError

Note: This method can not delete keys with subkeys.

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Os a string that must be a subkey of the key identified by the key parameter. This value must not be `None`, and the key may not have subkeys.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is *common.KEY_WOW64_64KEY*.
- **reserved** – Is a reserved integer, and must be zero. The default is zero.

wslwinreg.cygwinapi.DeleteValue

`wslwinreg.cygwinapi.DeleteValue(key, value)`

Removes a named value from a registry key.

Exception

WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **value** – Is a string that identifies the value to remove.

wslwinreg.cygwinapi.EnumKey

`wslwinreg.cygwinapi.EnumKey(key, index)`

Enumerates subkeys of an open registry key, returning a string.

The function retrieves the name of one subkey each time it is called. It is typically called repeatedly until an `OSError` exception is raised, indicating no more values are available.

Exception

WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **index** – Is an integer that identifies the index of the key to retrieve.

wslwinreg.cygwinapi.EnumValue

`wslwinreg.cygwinapi.EnumValue(key, index)`

Enumerates values of an open registry key, returning a tuple.

The function retrieves the name of one subkey each time it is called. It is typically called repeatedly, until an `OSError` exception is raised, indicating no more values.

Index	Meaning
0	A string that identifies the value.
1	An object that holds the value data, and whose type depends on the underlying registry type
2	An integer that identifies the type of the value data

Exception

WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.

- **index** – Is an integer that identifies the index of the value to retrieve.

Returns
A tuple of 3 items.

wslwinreg.cygwinapi.ExpandEnvironmentStrings

`wslwinreg.cygwinapi.ExpandEnvironmentStrings(str)`

Expands environment variables.

Expands environment variable placeholders NAME% in strings like REG_EXPAND_SZ.

Exception
WindowsError

Parameters
str – String to expand.

Returns
Expanded string

wslwinreg.cygwinapi.FlushKey

`wslwinreg.cygwinapi.FlushKey(key)`

Writes all the attributes of a key to the registry.

It is not necessary to call `FlushKey()` to change a key. Registry changes are flushed to disk by the registry using its lazy flusher. Registry changes are also flushed to disk at system shutdown. Unlike `CloseKey()`, the `FlushKey()` method returns only when all the data has been written to the registry. An application should only call `FlushKey()` if it requires absolute certainty that registry changes are on disk.

Exception
WindowsError

Note: If you don't know whether a `FlushKey()` call is required, it probably isn't.

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

wslwinreg.cygwinapi.LoadKey

`wslwinreg.cygwinapi.LoadKey(key, sub_key, file_name)`

Creates a subkey under the specified key.

Creates a subkey under the specified key and stores registration information from a specified file into that subkey.

This file must have been created with the SaveKey() function. Under the file allocation table (FAT) file system, the filename may not have an extension.

A call to `LoadKey()` fails if the calling process does not have the SE_RESTORE_PRIVILEGE privilege.

If key is a handle returned by `ConnectRegistry()`, then the path specified in fileName is relative to the remote computer.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is a handle returned by `ConnectRegistry()` or one of the constants *common.HKEY_USERS* or *common.HKEY_LOCAL_MACHINE*.
- **sub_key** – Is a string that identifies the sub_key to load
- **file_name** – Is the name of the file to load registry data from.

wslwinreg.cygwinapi.OpenKey

`wslwinreg.cygwinapi.OpenKey`(*key*, *sub_key*, *reserved=0*, *access=KEY_READ*)

Opens the specified key, returning a handle object.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that identifies the sub_key to open
- **reserved** – Is a reserved integer, and must be zero. Default is zero.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is KEY_READ.

Returns
A new handle to the specified key.

wslwinreg.cygwinapi.OpenKeyEx

`wslwinreg.cygwinapi.OpenKeyEx`(*key*, *sub_key*, *reserved=0*, *access=KEY_READ*)

Opens the specified key, returning a handle object.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that identifies the sub_key to open
- **reserved** – Is a reserved integer, and must be zero. Default is zero.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is KEY_READ.

Returns
A new handle to the specified key.

wslwinreg.cygwinapi.QueryInfoKey

`wslwinreg.cygwinapi.QueryInfoKey(key)`

Returns information about a key, as a tuple.

Index	Meaning
0	An integer giving the number of sub keys this key has.
1	An integer giving the number of values this key has.
2	An integer giving when the key was last modified (if available) as 100's of nanoseconds since Jan 1, 1601.

Exception
WindowsError or `FileNotFileError`

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

Returns
A tuple of 3 items.

wslwinreg.cygwinapi.QueryValue

`wslwinreg.cygwinapi.QueryValue(key, sub_key)`

Retrieves the unnamed value for a key, as a string.

Values in the registry have name, type, and data components. This method retrieves the data for a key's first value that has a NULL name. But the underlying API call doesn't return the type, so always use `QueryValueEx()` if possible.

Exception
WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that holds the name of the subkey with which the value is associated. If this parameter is None or empty, the function retrieves the value set by the `SetValue()` method for the key identified by `key`.

wslwinreg.cygwinapi.QueryValueEx

`wslwinreg.cygwinapi.QueryValueEx(key, value_name)`

Retrieves the type and data for a specified value name.

Retrieves the type and data for a specified value name associated with an open registry key.

Index	Meaning
0	The value of the registry item.
1	An integer giving the registry type for this value.

Exception

WindowsError or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **value_name** – Is a string indicating the value to query.

Returns

A tuple of 2 items.

wslwinreg.cygwinapi.SaveKey

`wslwinreg.cygwinapi.SaveKey(key, file_name)`

Saves the specified key, and all its subkeys to the specified file.

If key represents a key on a remote computer, the path described by file_name is relative to the remote computer. The caller of this method must possess the SeBackupPrivilege security privilege.

This function passes NULL for security_attributes to the API.

Exception

WindowsError or `FileNotFileError`

Note: Privileges are different than permissions.

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **file_name** – Is the name of the file to save registry data to. This file cannot already exist. If this filename includes an extension, it cannot be used on file allocation table (FAT) file systems by the `LoadKey()` method.

wslwinreg.cygwinapi.SetValue

`wslwinreg.cygwinapi.SetValue(key, sub_key, type, value)`

Associates a value with a specified key.

If the key specified by the sub_key parameter does not exist, the SetValue function creates it.

Value lengths are limited by available memory. Long values (more than 2048 bytes) should be stored as files with the filenames stored in the configuration registry. This helps the registry perform efficiently.

The key identified by the `key` parameter must have been opened with *common.KEY_SET_VALUE* access.

Exception

`TypeError`, *`WindowsError`* or `FileNotFileError`

Parameters

- **`key`** – Is an already open key, or any one of the predefined HKEY_* constants.
- **`sub_key`** – Is a string that names the subkey with which the value is associated.
- **`type`** – is an integer that specifies the type of the data. Currently this must be REG_SZ, meaning only strings are supported. Use the `SetValueEx()` function for support for other data types.
- **`value`** – Is a string that specifies the new value.

wslwinreg.cygwinapi.SetValueEx

`wslwinreg.cygwinapi.SetValueEx(key, value_name, reserved, type, value)`

Stores data in the value field of an open registry key.

This method can also set additional value and type information for the specified key. The key identified by the key parameter must have been opened with KEY_SET_VALUE access.

To open the key, use the CreateKeyEx() or OpenKeyEx() methods.

Value lengths are limited by available memory. Long values (more than 2048 bytes) should be stored as files with the filenames stored in the configuration registry. This helps the registry perform efficiently.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **`key`** – Is an already open key, or any one of the predefined HKEY_* constants.
- **`value_name`** – Is a string that names the subkey with which the value is associated.
- **`reserved`** – can be anything – zero is always passed to the API.
- **`type`** – Is an integer that specifies the type of the data.
- **`value`** – Is a string that specifies the new value.

wslwinreg.cygwinapi.DisableReflectionKey

`wslwinreg.cygwinapi.DisableReflectionKey(key)`

Disables registry reflection.

Disables registry reflection for 32-bit processes running on a 64-bit operating system.

If the key is not on the reflection list, the function succeeds but has no effect. Disabling reflection for a key does not affect reflection of any subkeys.

Exception
WindowsError

Parameters
key – Is an already open key, or one of the predefined HKEY_* constants.

wslwinreg.cygwinapi.EnableReflectionKey

`wslwinreg.cygwinapi.EnableReflectionKey(key)`

Restores registry reflection for the specified disabled key.

Restoring reflection for a key does not affect reflection of any subkeys.

Exception
WindowsError

Parameters
key – Is an already open key, or one of the predefined HKEY_* constants.

wslwinreg.cygwinapi.QueryReflectionKey

`wslwinreg.cygwinapi.QueryReflectionKey(key)`

Determines the reflection state for the specified key.

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

Returns
`True` if reflection is disabled.

wslwinreg.cygwinapi.convert_to_windows_path

`wslwinreg.cygwinapi.convert_to_windows_path(path_name)`

Convert a MSYS/Cygwin path to windows if needed.

If the path is already Windows format, it will be returned unchanged.

See also:

convert_from_windows_path

Return
Pathname converted to Windows.

Parameters
path_name – Windows or Linux pathname

wslwinreg.cygwinapi.convert_from_windows_path

`wslwinreg.cygwinapi.convert_from_windows_path(path_name)`

Convert an absolute Windows path to Cygwin/MSYS2.

If the path is already Cygwin/MSYS2 format, it will be returned unchanged.

See also:

convert_to_windows_path

Return
Pathname converted to Linux.

Parameters
path_name – Absolute Windows pathname

wslwinreg.cygwinapi.get_file_info

`wslwinreg.cygwinapi.get_file_info(path_name, string_name)`

Extract information from a windows exe file version resource.

Given a windows exe file, extract the "StringFileInfo" resource and parse out the data chunk named by string_name.

Full list of resource names: https://docs.microsoft.com/en-us/windows/desktop/menurc/stringfileinfo-block

Examples

```
file_version = get_file_info("devenv.exe", "FileVersion")
product_version =  get_file_info("devenv.exe", "ProductVersion")
```

Return
None if no record found or an error, or a valid string

Parameters

- **path_name** – Name of the windows file.
- **string_name** – Name of the data chunk to retrieve

4.4.4 Windows Subsystem for Linux implementation

On Windows Subsystem for Windows, the calls are sent to a server that will issue the calls directly in the Windows host which performs the actual the low level work.

wslwinreg.wslapi.CloseKey

`wslwinreg.wslapi.CloseKey(hkey)`

Closes a previously opened registry key.

The hkey argument specifies a previously opened key.

Note: If hkey is not closed using this method (or via hkey.Close()), it is closed when the hkey object is destroyed by Python.

Parameters

hkey – PyHKEY object or None.

wslwinreg.wslapi.ConnectRegistry

`wslwinreg.wslapi.ConnectRegistry(computer_name, key)`

Establishes a connection to a predefined registry handle.

Establishes a connection to a predefined registry handle on another computer, and returns a handle object.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **computer_name** – Is the name of the remote computer, of the form `r"\\computername"`. If `None`, the local computer is used.
- **key** – Is the predefined handle to connect to.

Returns

PyHKEY object

wslwinreg.wslapi.CreateKey

`wslwinreg.wslapi.CreateKey(key, sub_key)`

Creates or opens the specified key.

If key is one of the predefined keys, sub_key may be None. In that case, the handle returned is the same key handle passed in to the function.

If the key already exists, this function opens the existing key.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or one of the predefined HKEY_* constants.
- **sub_key** – Is a string that names the key this method opens or creates.

Returns
Handle of the opened key.

wslwinreg.wslapi.CreateKeyEx

`wslwinreg.wslapi.CreateKeyEx(key, sub_key, reserved=0, access=KEY_WRITE)`

Creates or opens the specified key.

If key is one of the predefined keys, sub_key may be None. In that case, the handle returned is the same key handle passed in to the function.

If the key already exists, this function opens the existing key.

Exception
`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or one of the predefined HKEY_* constants
- **sub_key** – Is a string that names the key this method opens or creates.
- **reserved** – Is a reserved integer, and must be zero. The default is zero.
- **access** – Os an integer that specifies an access mask that describes the desired security access for the key. Default is *common.KEY_WRITE*.

Returns
Handle of the opened key.

wslwinreg.wslapi.DeleteKey

`wslwinreg.wslapi.DeleteKey(key, sub_key)`

Not implemented.

Exception
`NotImplementedError` is always thrown.

wslwinreg.wslapi.DeleteKeyEx

`wslwinreg.wslapi.DeleteKeyEx(key, sub_key, access=KEY_WOW64_64KEY, reserved=0)`

Deletes the specified key.

If the method succeeds, the entire key, including all of its values, is removed.

Exception
`WindowsError`

Note: This method can not delete keys with subkeys.

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Os a string that must be a subkey of the key identified by the key parameter. This value must not be `None`, and the key may not have subkeys.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is *common.KEY_WOW64_64KEY*.
- **reserved** – Is a reserved integer, and must be zero. The default is zero.

wslwinreg.wslapi.DeleteValue

`wslwinreg.wslapi.DeleteValue(key, value)`

Removes a named value from a registry key.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **value** – Is a string that identifies the value to remove.

wslwinreg.wslapi.EnumKey

`wslwinreg.wslapi.EnumKey(key, index)`

Enumerates subkeys of an open registry key, returning a string.

The function retrieves the name of one subkey each time it is called. It is typically called repeatedly until an `OSError` exception is raised, indicating no more values are available.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **index** – Is an integer that identifies the index of the key to retrieve.

wslwinreg.wslapi.EnumValue

`wslwinreg.wslapi.EnumValue(key, index)`

Enumerates values of an open registry key, returning a tuple.

The function retrieves the name of one subkey each time it is called. It is typically called repeatedly, until an `OSError` exception is raised, indicating no more values.

Index	Meaning
0	A string that identifies the value.
1	An object that holds the value data, and whose type depends on the underlying registry type
2	An integer that identifies the type of the value data

Exception
`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **index** – Is an integer that identifies the index of the value to retrieve.

Returns
A tuple of 3 items.

wslwinreg.wslapi.ExpandEnvironmentStrings

`wslwinreg.wslapi.ExpandEnvironmentStrings(str)`

Expands environment variables.

Expands environment variable placeholders NAME% in strings like REG_EXPAND_SZ.

Exception
`WindowsError`

Parameters
str – String to expand.

Returns
Expanded string

wslwinreg.wslapi.FlushKey

`wslwinreg.wslapi.FlushKey(key)`

Writes all the attributes of a key to the registry.

It is not necessary to call `FlushKey()` to change a key. Registry changes are flushed to disk by the registry using its lazy flusher. Registry changes are also flushed to disk at system shutdown. Unlike `CloseKey()`, the `FlushKey()` method returns only when all the data has been written to the registry. An application should only call `FlushKey()` if it requires absolute certainty that registry changes are on disk.

Exception
`WindowsError`

Note: If you don't know whether a `FlushKey()` call is required, it probably isn't.

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

wslwinreg.wslapi.LoadKey

`wslwinreg.wslapi.LoadKey(key, sub_key, file_name)`

Creates a subkey under the specified key.

Creates a subkey under the specified key and stores registration information from a specified file into that subkey.

This file must have been created with the SaveKey() function. Under the file allocation table (FAT) file system, the filename may not have an extension.

A call to `LoadKey()` fails if the calling process does not have the SE_RESTORE_PRIVILEGE privilege.

If key is a handle returned by `ConnectRegistry()`, then the path specified in fileName is relative to the remote computer.

Exception
`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is a handle returned by `ConnectRegistry()` or one of the constants *common.HKEY_USERS* or *common.HKEY_LOCAL_MACHINE*.
- **sub_key** – Is a string that identifies the sub_key to load
- **file_name** – Is the name of the file to load registry data from.

wslwinreg.wslapi.OpenKey

`wslwinreg.wslapi.OpenKey(key, sub_key, reserved=0, access=KEY_READ)`

Opens the specified key, returning a handle object.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that identifies the sub_key to open
- **reserved** – Is a reserved integer, and must be zero. Default is zero.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is KEY_READ.

Returns

A new handle to the specified key.

wslwinreg.wslapi.OpenKeyEx

`wslwinreg.wslapi.OpenKeyEx(key, sub_key, reserved=0, access=KEY_READ)`

Opens the specified key, returning a handle object.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that identifies the sub_key to open
- **reserved** – Is a reserved integer, and must be zero. Default is zero.
- **access** – Is an integer that specifies an access mask that describes the desired security access for the key. Default is KEY_READ.

Returns

A new handle to the specified key.

wslwinreg.wslapi.QueryInfoKey

`wslwinreg.wslapi.QueryInfoKey(key)`

Returns information about a key, as a tuple.

Index	Meaning
0	An integer giving the number of sub keys this key has.
1	An integer giving the number of values this key has.
2	An integer giving when the key was last modified (if available) as 100's of nanoseconds since Jan 1, 1601.

Exception
`WindowsError` or `FileNotFileError`

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

Returns
A tuple of 3 items.

wslwinreg.wslapi.QueryValue

`wslwinreg.wslapi.QueryValue(key, sub_key)`

Retrieves the unnamed value for a key, as a string.

Values in the registry have name, type, and data components. This method retrieves the data for a key's first value that has a NULL name. But the underlying API call doesn't return the type, so always use `QueryValueEx()` if possible.

Exception
`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that holds the name of the subkey with which the value is associated. If this parameter is None or empty, the function retrieves the value set by the `SetValue()` method for the key identified by `key`.

wslwinreg.wslapi.QueryValueEx

`wslwinreg.wslapi.QueryValueEx(key, value_name)`

Retrieves the type and data for a specified value name.

Retrieves the type and data for a specified value name associated with an open registry key.

Index	Meaning
0	The value of the registry item.
1	An integer giving the registry type for this value.

Exception
`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **value_name** – Is a string indicating the value to query.

Returns
A tuple of 2 items.

wslwinreg.wslapi.SaveKey

`wslwinreg.wslapi.SaveKey(key, file_name)`

Saves the specified key, and all its subkeys to the specified file.

If key represents a key on a remote computer, the path described by file_name is relative to the remote computer. The caller of this method must possess the SeBackupPrivilege security privilege.

This function passes NULL for security_attributes to the API.

Exception
`WindowsError` or `FileNotFileError`

Note: Privileges are different than permissions.

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **file_name** – Is the name of the file to save registry data to. This file cannot already exist. If this filename includes an extension, it cannot be used on file allocation table (FAT) file systems by the `LoadKey()` method.

wslwinreg.wslapi.SetValue

`wslwinreg.wslapi.SetValue(key, sub_key, type, value)`

Associates a value with a specified key.

If the key specified by the sub_key parameter does not exist, the SetValue function creates it.

Value lengths are limited by available memory. Long values (more than 2048 bytes) should be stored as files with the filenames stored in the configuration registry. This helps the registry perform efficiently.

The key identified by the `key` parameter must have been opened with *common.KEY_SET_VALUE* access.

Exception
`TypeError`, `WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **sub_key** – Is a string that names the subkey with which the value is associated.

- **type** – is an integer that specifies the type of the data. Currently this must be REG_SZ, meaning only strings are supported. Use the `SetValueEx()` function for support for other data types.
- **value** – Is a string that specifies the new value.

wslwinreg.wslapi.SetValueEx

`wslwinreg.wslapi.SetValueEx(key, value_name, reserved, type, value)`

Stores data in the value field of an open registry key.

This method can also set additional value and type information for the specified key. The key identified by the key parameter must have been opened with KEY_SET_VALUE access.

To open the key, use the CreateKeyEx() or OpenKeyEx() methods.

Value lengths are limited by available memory. Long values (more than 2048 bytes) should be stored as files with the filenames stored in the configuration registry. This helps the registry perform efficiently.

Exception

`WindowsError` or `FileNotFileError`

Parameters

- **key** – Is an already open key, or any one of the predefined HKEY_* constants.
- **value_name** – Is a string that names the subkey with which the value is associated.
- **reserved** – can be anything – zero is always passed to the API.
- **type** – Is an integer that specifies the type of the data.
- **value** – Is a string that specifies the new value.

wslwinreg.wslapi.DisableReflectionKey

`wslwinreg.wslapi.DisableReflectionKey(key)`

Disables registry reflection.

Disables registry reflection for 32-bit processes running on a 64-bit operating system.

If the key is not on the reflection list, the function succeeds but has no effect. Disabling reflection for a key does not affect reflection of any subkeys.

Exception

`WindowsError`

Parameters

key – Is an already open key, or one of the predefined HKEY_* constants.

wslwinreg.wslapi.EnableReflectionKey

`wslwinreg.wslapi.EnableReflectionKey(key)`

Restores registry reflection for the specified disabled key.

Restoring reflection for a key does not affect reflection of any subkeys.

Exception
`WindowsError`

Parameters
key – Is an already open key, or one of the predefined HKEY_* constants.

wslwinreg.wslapi.QueryReflectionKey

`wslwinreg.wslapi.QueryReflectionKey(key)`

Determines the reflection state for the specified key.

Parameters
key – Is an already open key, or any one of the predefined HKEY_* constants.

Returns
`True` if reflection is disabled.

wslwinreg.wslapi.convert_to_windows_path

`wslwinreg.wslapi.convert_to_windows_path(path_name)`

Convert a WSL path to windows if needed.

If the path is already Windows format, it will be returned unchanged.

See also:

convert_from_windows_path

Return
Pathname converted to Windows.

Parameters
path_name – Windows or Linux pathname

wslwinreg.wslapi.convert_from_windows_path

`wslwinreg.wslapi.convert_from_windows_path(path_name)`

Convert an absolute Windows path to WSL.

If the path is already Linux format, it will be returned unchanged.

See also:

convert_to_windows_path

Return
Pathname converted to Linux.

Parameters
path_name – Absolute Windows pathname

wslwinreg.wslapi.get_file_info

`wslwinreg.wslapi.get_file_info(path_name, string_name)`

Extract information from a windows exe file version resource.

Given a windows exe file, extract the "StringFileInfo" resource and parse out the data chunk named by string_name.

Full list of resource names: https://docs.microsoft.com/en-us/windows/desktop/menurc/stringfileinfo-block

Examples

```
file_version = get_file_info("devenv.exe", "FileVersion")
product_version =  get_file_info("devenv.exe", "ProductVersion")
```

Return
None if no record found or an error, or a valid string

Parameters

- **path_name** – Name of the windows file.
- **string_name** – Name of the data chunk to retrieve

4.5 License

4.5.1 MIT License

The gist of the license… Have fun using this code, I won't sue you and you can't sue me. However, please be nice about it and give me a credit in your software that you used my code in.

Please?

INDEX

Symbols

A

B

D

E

F

H

I

K

L

P

Q

R

S

W