RegistryClass MFC Library Reference

Simplified Registry Access with MFC Data Types

CRegistry Class

This class is the root of all Windows Registry access provided by the RegistryClass library.

```
class CRegistry {
   CRegistryKey *KeyClassRoot();
   CRegistryKey *KeyCurrentUser();
   CRegistryKey *KeyLocalMachine();
   CRegistryKey *KeyUsers();
   CRegistryKey *KeyCurrentConfig();
   CRegistryKey *KeyDynData();
   CRegistryKey *KeyPerformanceData();
}
```

Each member function returns a pointer to a CRegistryKey class that can be used directly to open and create keys under a given Hive Key.

CRegistryKey Class

This class provides the majority of the RegistryClass functionality. This class is where you create keys and values and get information about them.

```
class CRegistryKey {
public:
   CRegistryKey();
   virtual ~CRegistryKey();
   CRegistryKey *OpenKey(LPCTSTR lpSubKey);
   CRegistryKey *OpenKey(LPCTSTR lpSubKey, REGSAM accessDesired);
   CRegistryKey *CreateKey (LPCTSTR lpSubKey);
   CRegistryKey *CreateKey(LPCTSTR lpSubKey, REGSAM accessDesired);
   LONG QueryMaxKeyNameLen();
   LONG QueryMaxValueLen();
   LONG QueryMaxValueNameLen();
   LONG GetValue(LPCTSTR lpValueName, CRegistryValue &rValue);
   LONG GetStringValue(LPCTSTR lpValueName, CString *rString);
   LONG GetIntegerValue(LPCTSTR lpValueName, LPDWORD lpdwValue);
   BOOL GetKeyNames(CArray<CString,CString> *pStrArray);
   BOOL GetValueNames(CArray<CString,CString> *pStrArray);
   LONG SetValue(LPCTSTR lpValueName, DWORD dwType, const BYTE*
lpData, DWORD cbData);
   LONG SetBinaryValue(LPCTSTR lpValueName, const BYTE* lpData,
DWORD cbData);
   LONG SetIntegerValue(LPCTSTR lpValueName, DWORD dwData);
   LONG SetStringValue(LPCTSTR lpValueName, LPCTSTR lpString);
   LONG SetExpandStringValue(LPCTSTR lpValueName, LPCTSTR lpString);
   BOOL CopyTo(CRegistryKey *pDestKey);
   LONG DeleteValue(LPCTSTR lpValueName);
   LONG DeleteKey (LPCTSTR lpSubKey);
   BOOL IsRegKey(HKEY hKey);           // For internal use only…
protected:
   HKEY m_myKey;
}
```

Member Variables

`m_myKey`

This is the HANDLE to the key that the class represents. This can be used with flat Win32 API functions to perform more advanced tasks on the key.

OpenKey

Syntax

```
CRegistryKey *OpenKey(LPCTSTR lpSubKey);
CRegistryKey *OpenKey(LPCTSTR lpSubKey, REGSAM accessDesired);
```

Parameters

`lpSubKey`

Path of the sub-key you want to open

`accessDesired`

The access limit you wish, it may be one or more of the following values:

`KEY_ALL_ACCESS`
Combination of KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY, KEY_CREATE_SUB_KEY, KEY_CREATE_LINK, and KEY_SET_VALUE access.

`KEY_CREATE_LINK`
Permission to create a symbolic link.

`KEY_CREATE_SUB_KEY`
Permission to create sub-keys.

`KEY_ENUMERATE_SUB_KEYS`
Permission to enumerate sub-keys.

`KEY_EXECUTE`
Permission for read access.

`KEY_NOTIFY`
Permission for change notification.

`KEY_QUERY_VALUE`
Permission to query subkey data.

`KEY_READ`
Combination of KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS, and KEY_NOTIFY access.

`KEY_SET_VALUE`
Permission to set sub-key data.

`KEY_WRITE`
Combination of KEY_SET_VALUE and KEY_CREATE_SUB_KEY access.

Returns

Returns a pointer to a CRegistryKey class that represents the opened key. The pointer must be deleted when it is no longer needed.

CreateKey

Syntax

```
CRegistryKey *CreateKey (LPCTSTR lpSubKey);
CRegistryKey *CreateKey(LPCTSTR lpSubKey, REGSAM accessDesired);
```

Parameters

`lpSubKey`

Path of the sub-key you want to create or open

`accessDesired`

The access limit you wish, it may be one or more of the following values:

KEY_ALL_ACCESS
Combination of KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY, KEY_CREATE_SUB_KEY, KEY_CREATE_LINK, and KEY_SET_VALUE access.

KEY_CREATE_LINK
Permission to create a symbolic link.

KEY_CREATE_SUB_KEY
Permission to create sub-keys.

KEY_ENUMERATE_SUB_KEYS
Permission to enumerate sub-keys.

KEY_EXECUTE
Permission for read access.

KEY_NOTIFY
Permission for change notification.

KEY_QUERY_VALUE
Permission to query subkey data.

KEY_READ
Combination of KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS, and KEY_NOTIFY access.

KEY_SET_VALUE
Permission to set sub-key data.

KEY_WRITE
Combination of KEY_SET_VALUE and KEY_CREATE_SUB_KEY access.

Returns

Returns a pointer to a CRegistryKey class that represents the opened or created key. The pointer must be deleted when it is no longer needed.

Query Function

Syntax

```
   LONG QueryMaxKeyNameLen();
   LONG QueryMaxValueLen();
   LONG QueryMaxValueNameLen();
```

These functions queries information about the key.

Parameters

None of these functions take any parameters

Return

QueryMaxKeyNameLen returns the length, in characters, of the key with the longest name. QueryMaxValueNameLen returns the length, in characters, of the value with the longest name. QueryMaxValueLen returns the length, in bytes, of the largest value.

GetValue Functions

Syntax

```
LONG GetValue(LPCTSTR lpValueName, CRegistryValue &rValue);
LONG GetStringValue(LPCTSTR lpValueName, CString *rString);
LONG GetIntegerValue(LPCTSTR lpValueName, LPDWORD lpdwValue);
```

These functions retrieve values from an opened key.

Parameters

`lpValueName`

The name of the value you want to retrieve.

`rValue`

A reference to a CRegistryValue class that will be filled with the value information and data.

`rString`

A pointer to a CString class that will be filled with the value.

`lpdwValue`

A pointer to a DWORD that will be set to the value.

Return

Returns ERROR_SUCCESS if successful, the failure error value otherwise.

GetNames Functions

Syntax

```
BOOL GetKeyNames(CArray<CString,CString> *pStrArray);
BOOL GetValueNames(CArray<CString,CString> *pStrArray);
```

These function collect the names of sub-keys or values for a opened key.

Parameters

`pStrArray`

A pointer to a CArray class that will be filled with the names of the sub-keys or values.

Return

Returns TRUE on success, FALSE on error.

SetValue Functions

Syntax

```
LONG SetValue(LPCTSTR lpValueName, DWORD dwType, const BYTE* lpData,
DWORD cbData);
LONG SetBinaryValue(LPCTSTR lpValueName, const BYTE* lpData, DWORD
cbData);
LONG SetIntegerValue(LPCTSTR lpValueName, DWORD dwData);
LONG SetStringValue(LPCTSTR lpValueName, LPCTSTR lpString);
LONG SetExpandStringValue(LPCTSTR lpValueName, LPCTSTR lpString);
```

These functions sets values in an opened key.

Parameters

`lpValueName`

Name of the value you want to set.

`DwType`

The data of the value. This may be one of the following constants:

`REG_BINARY`
Binary data in any form.

`REG_DWORD`
A 32-bit number.

`REG_DWORD_LITTLE_ENDIAN`
A 32-bit number in little-endian format.

Microsoft® Windows® is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_DWORD in the Windows header files.

`REG_DWORD_BIG_ENDIAN`
A 32-bit number in big-endian format.

Some UNIX systems support big-endian architectures.

`REG_EXPAND_SZ`
Null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%"). It will be a Unicode or ANSI string depending on whether you use the Unicode or ANSI functions. To expand the environment variable references, use the ExpandEnvironmentStrings function.

`REG_LINK`
Reserved for system use.

`REG_MULTI_SZ`
Array of null-terminated strings, terminated by two null characters.

`REG_NONE`
No defined value type.

`REG_SZ`
Null-terminated string. It will be a Unicode or ANSI string, depending on whether you use the Unicode or ANSI functions.

`lpData`

A pointer to the data to set the value to.

`cbData`

The size, in bytes, of the data.

`dwData`

A DWORD to set the value to.

`lpString`

A pointer to a string to set the value to.

Return

Returns ERROR_SUCCESS if successful, the failure error value otherwise.

Delete Function

Syntax

```
LONG DeleteValue(LPCTSTR lpValueName);
LONG DeleteKey (LPCTSTR lpSubKey);
```

These function delete a value or a sub-key, its sub-keys and values.

Parameters

`lpValueName/lpSubKey`

The name of the value or sub-key to delete

Note

When deleting a sub-key, the key does not have to be empty. The DeleteKey function will recursively delete the sub-key, its sub-keys, and their values.

Return

Returns ERROR_SUCCESS if successful, the failure error value otherwise.

CopyTo

Syntax

```
BOOL CopyTo(CRegistryKey *pDestKey);
```

Copies a key’s sub-keys, and their values, recursively, to another opened key.

Parameters

`pDestKey`

A pointer to a CRegistryKey class that the key’s contents will be copied to.

Return

Returns TRUE on success, FALSE on error.

CRegistryValue Class

This class raw access to a retrieved value. This class is here for future expansion.

```
class AFX_CLASS_EXPORT CregistryValue
{
   ULONG Type();
   DWORD GetLength();
   LPBYTE GetData();
   CRegistryValue();
   virtual ~CRegistryValue();
};
```

The Type function returns a ULONG that for the type of value this class represents.

The GetLength function returns the size, in bytes, of the values.

The GetData function returns a pointer to the value data.