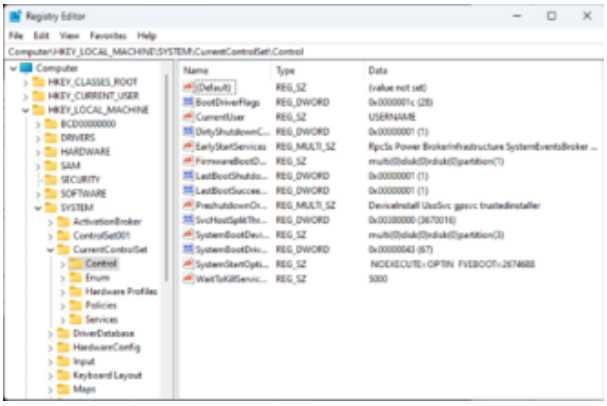
Windows Registry

By Contributors to Wikimedia projects

Published: 2004-11-13 · Archived: 2026-04-06 00:23:01 UTC

Windows Registry Editor



Registry Editor, the user interface for the registry, in Windows 11

Other names	regedit
Developer	Microsoft
Initial release	April 6, 1992; 33 years ago with Windows 3.1
Operating system	Microsoft Windows
Platform	IA-32, x86-64 and ARM (and historically DEC Alpha, Itanium, MIPS, and PowerPC)
Included with	Microsoft Windows
Type	Hierarchical database
License	Proprietary
Website	learn.microsoft.com/en-us/windows/win32/sysinfo/registry

The **Windows Registry** is a hierarchical database that stores low-level settings for the Microsoft Windows operating system and for applications that opt to use the registry. The kernel, device drivers, services, Security Accounts Manager, and user interfaces can all use the registry. The registry also allows access to counters for profiling system performance.

In other words, the registry or Windows Registry contains information, settings, options, and other values for programs and hardware installed on all versions of Microsoft Windows operating systems. For example, when a program is installed, a new subkey containing settings such as a program's location, its version, and how to start the program, are all added to the Windows Registry.

When introduced with Windows 3.1, the Windows Registry primarily stored configuration information for COM-based components. Windows 95 and Windows NT extended its use to rationalize and centralize the information in the profusion of INI files, which held the configurations for individual programs, and were stored at various locations.[1][2] It is not a requirement for Windows applications to use the Windows Registry. For example, .NET Framework applications use XML files for configuration, while portable applications usually keep their configuration files with their executables.

Prior to the Windows Registry, .INI files stored each program's settings as a text file or binary file, often located in a shared location that did not provide user-specific settings in a multi-user scenario. By contrast, the Windows Registry stores all application settings in one logical repository (but also in a number of discrete files) and in a standardized form. According to Microsoft, this offers several advantages over .INI files.[2][3] Since file parsing is done much more efficiently with a binary format, it may be read from or written to more quickly than a text INI file. Furthermore, strongly typed data can be stored in the registry, as opposed to the text information stored in .INI files. This is a benefit when editing keys manually using `regedit.exe`, the built-in Windows Registry Editor. Because user-based registry settings are loaded from a user-specific path rather than from a read-only system location, the registry allows multiple users to share the same machine, and also allows programs to work for less privileged users. Backup and restoration is also simplified as the registry can be accessed over a network connection for remote management/support, including from scripts, using the standard set of APIs, as long as the Remote Registry service is running and firewall rules permit this.

Because the registry is a database, it offers improved system integrity with features such as atomic updates. If two processes attempt to update the same registry value at the same time, one process's change will precede the other's and the overall consistency of the data will be maintained. Where changes are made to .INI files, such race conditions can result in inconsistent data that does not match either attempted update. Windows Vista and later operating systems provide transactional updates to the registry by means of the Kernel Transaction Manager, extending the atomicity guarantees across multiple key or value changes with traditional commit–abort semantics. (Note however that NTFS provides such support for the file system as well, so the same guarantees could, in theory, be obtained with traditional configuration files.)

Structure of Registry Key

[edit]

The registry contains two basic elements: **keys** and **values**. Registry *keys* are container objects similar to folders. Registry *values* are non-container objects similar to files. Keys may contain values and subkeys. Keys are referenced with a syntax similar to Windows' path names, using backslashes to indicate levels of hierarchy. Keys must have a case insensitive name without backslashes.

The hierarchy of registry keys can only be accessed from a known root key handle (which is anonymous but whose effective value is a constant numeric handle) that is mapped to the content of a registry key preloaded by the kernel from a stored "hive", or to the content of a subkey within another root key, or mapped to a registered service or DLL that provides access to its contained subkeys and values.

E.g. `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows` refers to the subkey "Windows" of the subkey "Microsoft" of the subkey "Software" of the HKEY_LOCAL_MACHINE root key.

There are seven predefined root keys, traditionally named according to their constant handles defined in the Win32 API, or by synonymous abbreviations (depending on applications):[4]

- HKEY_LOCAL_MACHINE or HKLM
- HKEY_CURRENT_CONFIG or HKCC
- HKEY_CLASSES_ROOT or HKCR
- HKEY_CURRENT_USER or HKCU
- HKEY_USERS or HKU
- HKEY_PERFORMANCE_DATA (only in Windows NT, but invisible in the Windows Registry Editor)[5]
- HKEY_DYN_DATA (only in Windows 9x, and visible in the Windows Registry Editor)

Like other files and services in Windows, all registry keys may be restricted by access control lists (ACLs), depending on user privileges, or on security tokens acquired by applications, or on system security policies enforced by the system (these restrictions may be predefined by the system itself, and configured by local system administrators or by domain administrators). Different users, programs, services or remote systems may only see some parts of the hierarchy or distinct hierarchies from the same root keys.

Registry *values* are name/data pairs stored within keys. Registry values are referenced separately from registry keys. Each registry value stored in a registry key has a unique name whose letter case is not significant. The Windows API functions that query and manipulate registry values take value names separately from the key path or handle that identifies the parent key. Registry values may contain backslashes in their names, but doing so makes them difficult to distinguish from their key paths when using some legacy Windows Registry API functions (whose usage is deprecated in Win32).

The terminology is somewhat misleading, as each registry key is similar to an associative array, where standard terminology would refer to the name part of each registry value as a "key". The terms are a holdout from the 16-bit registry in Windows 3, in which registry keys could not contain arbitrary name/data pairs, but rather contained only one unnamed value (which had to be a string). In this sense, the Windows 3 registry was like a single associative array, in which the keys (in the sense of both 'registry key' and 'associative array key') formed a hierarchy, and the registry values were all strings. When the 32-bit registry was created, so was the additional capability of creating multiple named values per key, and the meanings of the names were somewhat distorted.[6]

For compatibility with the previous behavior, each registry key may have a "default" value, whose name is the empty string.

Each value can store arbitrary data with variable length and encoding, but which is associated with a symbolic type (defined as a numeric constant) defining how to parse this data. The standard types are:[7]

List of standard registry value types

Type ID	Symbolic type name	Meaning and encoding of the data stored in the registry value
0	REG_NONE	No type (the stored value, if any)
1	REG_SZ	A string value, normally stored and exposed in UTF-16LE (when using the Unicode version of Win32 API functions), usually terminated by a NUL character
2	REG_EXPAND_SZ	An "expandable" string value that can contain environment variables, normally stored and exposed in UTF-16LE, usually terminated by a NUL character
3	REG_BINARY	Binary data (any arbitrary data)
4	REG_DWORD / REG_DWORD_LITTLE_ENDIAN	A DWORD value, a 32-bit unsigned integer (numbers between 0 and 4,294,967,295 [$2^{32} - 1$]) (little-endian)
5	REG_DWORD_BIG_ENDIAN	A DWORD value, a 32-bit unsigned integer (numbers between 0 and 4,294,967,295 [$2^{32} - 1$]) (big-endian)
6	REG_LINK	A symbolic link (UNICODE) to another registry key, specifying a root key and the path to the target key
7	REG_MULTI_SZ	A multi-string value, which is an ordered list of non-empty strings, normally stored and exposed in Unicode, each one terminated by a null character, the list being normally terminated by a second null character.[8]
8	REG_RESOURCE_LIST	A resource list (used by the *Plug-n-Play* hardware enumeration and configuration)

9	REG_FULL_RESOURCE_DESCRIPTOR	A resource descriptor (used by the *Plug-n-Play* hardware enumeration and configuration)
10	REG_RESOURCE_REQUIREMENTS_LIST	A resource requirements list (used by the *Plug-n-Play* hardware enumeration and configuration)
11	REG_QWORD / REG_QWORD_LITTLE_ENDIAN	A QWORD value, a 64-bit integer (either big- or little-endian, or unspecified) (introduced in Windows 2000)[9]

The keys at the root level of the hierarchical database are generally named by their Windows API definitions, which all begin with "HKEY".[2] They are frequently abbreviated to a three- or four-letter short name starting with "HK" (e.g. HKCU and HKLM). Technically, they are predefined handles (with known constant values) to specific keys that are either maintained in memory, or stored in hive files stored in the local filesystem and loaded by the system kernel at boot time and then shared (with various access rights) between all processes running on the local system, or loaded and mapped in all processes started in a user session when the user logs on the system.

The HKEY_LOCAL_MACHINE (local machine-specific configuration data) and HKEY_CURRENT_USER (user-specific configuration data) nodes have a similar structure to each other; user applications typically look up their settings by first checking for them in `HKEY_CURRENT_USER\Software\Vendor's name\Application's name\Version\Setting name`, and if the setting is not found, look instead in the same location under the HKEY_LOCAL_MACHINE key. However, the converse may apply for administrator-enforced policy settings where HKLM may take precedence over HKCU. The Windows Logo Program has specific requirements for where different types of user data may be stored, and that the concept of least privilege be followed so that administrator-level access is not required to use an application.[a][10]

HKEY_LOCAL_MACHINE (HKLM)

[edit]

Abbreviated HKLM, HKEY_LOCAL_MACHINE stores settings that are specific to the local computer.[11]

The key located by HKLM is actually not stored on disk, but maintained in memory by the system kernel in order to map all the other subkeys. Applications cannot create any additional subkeys. On Windows NT, this key contains four subkeys, "SAM", "SECURITY", "SYSTEM", and "SOFTWARE", that are loaded at boot time within their respective files located in the `%SystemRoot%\System32\config\` folder. A fifth subkey, "HARDWARE", is volatile and is created dynamically, and as such is not stored in a file (it exposes a view of all the currently detected Plug-and-Play devices). On Windows Vista and above, a sixth and seventh subkey, "COMPONENTS" and "BCD", are mapped in memory by the kernel on-demand and loaded from `%SystemRoot%\System32\config\COMPONENTS` or from boot configuration data, `\boot\BCD` on the system partition.

- The "`HKLM\SAM`" key usually appears as empty for most users (unless they are granted access by administrators of the local system or administrators of domains managing the local system). It is used to

reference all "Security Accounts Manager" (SAM) databases for all domains into which the local system has been administratively authorized or configured (including the local domain of the running system, whose SAM database is stored in a subkey also named "SAM": other subkeys will be created as needed, one for each supplementary domain). Each SAM database contains all built in accounts (mostly group aliases) and configured accounts (users, groups and their aliases, including guest accounts and administrator accounts) created and configured on the respective domain, for each account in that domain, it notably contains the user name which can be used to log on that domain, the internal unique user identifier in the domain, a cryptographic hash of each user's password for each enabled authentication protocol, the location of storage of their user registry hive, various status flags (for example if the account can be enumerated and be visible in the logon prompt screen), and the list of domains (including the local domain) into which the account was configured.

- The " `HKLM\SECURITY` " key usually appears empty for most users (unless they are granted access by users with administrative privileges) and is linked to the Security database of the domain into which the current user is logged on (if the user is logged on the local system domain, this key will be linked to the registry hive stored by the local machine and managed by local system administrators or by the built in "System" account and Windows installers). The kernel will access it to read and enforce the security policy applicable to the current user and all applications or operations executed by this user. It also contains a "SAM" subkey which is dynamically linked to the SAM database of the domain onto which the current user is logged on.
- The " `HKLM\SYSTEM` " key is normally only writable by users with administrative privileges on the local system. It contains information about the Windows system setup, data for the secure random number generator (RNG), the list of currently mounted devices containing a filesystem, several numbered Control Sets (such as " `HKLM\SYSTEM\ControlSet001` ") containing alternative configurations for system hardware drivers and services running on the local system (including the currently used one and a backup), a " `HKLM\SYSTEM\Select` " subkey containing the status of these Control Sets, and a " `HKLM\SYSTEM\CurrentControlSet` " which is dynamically linked at boot time to the Control Set which is currently used on the local system. Each configured Control Set contains:
  - an "Enum" subkey enumerating all known Plug-and-Play devices and associating them with installed system drivers (and storing the device-specific configurations of these drivers),
  - a "Services" subkey listing all installed system drivers (with non device-specific configuration, and the enumeration of devices for which they are instantiated) and all programs running as services (how and when they can be automatically started),
  - a "Control" subkey organizing the various hardware drivers and programs running as services and all other system-wide configuration,
  - a "Hardware Profiles" subkey enumerating the various profiles that have been tuned (each one with "System" or "Software" settings used to modify the default profile, either in system drivers and services or in the applications) as well as the `Hardware Profiles\Current` subkey which is dynamically linked to one of these profiles.
- The " `HKLM\SOFTWARE` " subkey contains software and Windows settings (in the default hardware profile). It is mostly modified by application and system installers. It is organized by software vendor (with a subkey for each), but also contains a "Windows" subkey for some settings of the Windows user interface, a "Classes" subkey containing all registered associations from file extensions, MIME types, Object Classes

IDs and interfaces IDs (for OLE, COM/DCOM and ActiveX), to the installed applications or DLLs that may be handling these types on the local machine (however these associations are configurable for each user, see below), and a "Policies" subkey (also organized by vendor) for enforcing general usage policies on applications and system services (including the central certificates store used for authenticating, authorizing or disallowing remote systems or services running outside the local network domain).

- The "`HKLM\SOFTWARE\Wow6432Node`" key is used by 32-bit applications on a 64-bit Windows OS, and is equivalent to but separate from "`HKLM\SOFTWARE`". The key path is transparently presented to 32-bit applications by WoW64 as `HKLM\SOFTWARE` [12] (in a similar way that 64-bit applications see `%SystemRoot%\Syswow64` as `%SystemRoot%\System32`)

HKEY_CLASSES_ROOT (HKCR)

[edit]

Abbreviated HKCR, HKEY_CLASSES_ROOT contains information about registered applications, such as file associations and OLE Object Class IDs, tying them to the applications used to handle these items. On Windows 2000 and above, HKCR is a compilation of user-based `HKCU\Software\Classes` and machine-based `HKLM\Software\Classes`. If a given value exists in both of the subkeys above, the one in `HKCU\Software\Classes` takes precedence.[13] The design allows for either machine- or user-specific registration of COM objects.

Abbreviated HKU, HKEY_USERS contains subkeys corresponding to the HKEY_CURRENT_USER keys for each user profile actively loaded on the machine, though user hives are usually only loaded for currently logged-in users.

HKEY_CURRENT_USER (HKCU)

[edit]

Abbreviated HKCU, HKEY_CURRENT_USER stores settings that are specific to the currently logged-in user.[14] The HKEY_CURRENT_USER key is a link to the subkey of HKEY_USERS that corresponds to the user; the same information is accessible in both locations. The specific subkey referenced is `(HKU)\(SID)\...` where (SID) corresponds to the Windows SID; if the "(HKCU)" key has the following suffix `(HKCU)\Software\Classes\...` then it corresponds to `(HKU)\(SID)_CLASSES\...` i.e. the suffix string "_CLASSES" is appended to the (SID).

On Windows NT systems, each user's settings are stored in their own files called NTUSER.DAT and USRCLASS.DAT inside their own Documents and Settings subfolder (or their own Users sub folder in Windows Vista and above). Settings in this hive follow users with a roaming profile from machine to machine.

HKEY_PERFORMANCE_DATA

[edit]

> This key provides runtime information into performance data provided by either the NT kernel itself, or running system drivers, programs and services that provide performance data. This key is not stored in any hive and not displayed in the Registry Editor, but it is visible through the registry functions in the Windows API, or in a simplified view via the Performance tab of the Task Manager (only for a few performance data on the local system) or via more advanced control panels (such as the Performances Monitor or the Performances Analyzer which allows collecting and logging these data, including from remote systems).
>
> This key is used only in Windows 95, Windows 98 and Windows ME.[15] It contains information about hardware devices, including Plug and Play and network performance statistics. The information in this hive is also not stored on the hard drive; the Plug and Play information is gathered and configured at startup and is stored in memory.[16]

Even though the registry presents itself as an integrated hierarchical database, branches of the registry are actually stored in a number of disk files called *hives*.[17] (The word hive constitutes an in-joke.)[18]

Some hives are volatile and are not stored on disk at all. An example of this is the hive of the branch starting at `HKLM\HARDWARE`. This hive records information about system hardware and is created each time the system boots and performs hardware detection.

Individual settings for users on a system are stored in a hive (disk file) per user. During user login, the system loads the user hive under the HKEY_USERS key and sets the HKCU (HKEY_CURRENT_USER) symbolic reference to point to the current user. This allows applications to store/retrieve settings for the current user implicitly under the HKCU key.

Not all hives are loaded at any one time. At boot time, only a minimal set of hives are loaded, and after that, hives are loaded as the operating system initializes and as users log in or whenever a hive is explicitly loaded by an application.

The registry is physically stored in several files, which are generally obfuscated from the user-mode APIs used to manipulate the data inside the registry. Depending upon the version of Windows, there will be different files and different locations for these files, but they are all on the local machine. The location for system registry files in Windows NT is `%SystemRoot%\System32\config\`; the user-specific HKEY_CURRENT_USER user registry hive is stored in `Ntuser.dat` inside the user profile. There is one of these per user; if a user has a roaming profile, then this file will be copied to and from a server at logout and login respectively. A second user-specific registry file named UsrClass.dat contains COM registry entries and does not roam by default.

Windows NT systems store the registry in a binary file format which can be exported, loaded and unloaded by the Registry Editor in these operating systems. The following system registry files are stored in `%SystemRoot%\System32\config\`:

- `Sam` – `HKEY_LOCAL_MACHINE\SAM`
- `Security` – `HKEY_LOCAL_MACHINE\SECURITY`
- `Software` – `HKEY_LOCAL_MACHINE\SOFTWARE`
- `System` – `HKEY_LOCAL_MACHINE\SYSTEM`
- `Default` – `HKEY_USERS\.DEFAULT`

- `Userdiff` – Not associated with a hive. Used only when upgrading operating systems.[19]

The following file is stored in each user's profile folder:

- `%USERPROFILE%\Ntuser.dat` – `HKEY_USERS\<User SID>` (linked to by HKEY_CURRENT_USER)

For Windows 2000, Server 2003 and Windows XP, the following additional user-specific file is used for file associations and COM information:

- `%USERPROFILE%\Local Settings\Application Data\Microsoft\Windows\Usrclass.dat` (path is localized) – `HKEY_USERS\<User SID>_Classes` (`HKEY_CURRENT_USER\Software\Classes`)

For Windows Vista and later, the path was changed to:

- `%USERPROFILE%\AppData\Local\Microsoft\Windows\Usrclass.dat` (path is not localized) alias `%LocalAppData%\Microsoft\Windows\Usrclass.dat` – HKEY_USERS\<User SID>_Classes (`HKEY_CURRENT_USER\Software\Classes`)

Windows 2000 keeps an alternate copy of the registry hives (.ALT) and attempts to switch to it when corruption is detected.[20] Windows XP and Windows Server 2003 do not maintain a `System.alt` hive because NTLDR on those versions of Windows can process the `System.log` file to bring up to date a System hive that has become inconsistent during a shutdown or crash. In addition, the `%SystemRoot%\Repair` folder contains a copy of the system's registry hives that were created after installation and the first successful startup of Windows.

Each registry data file has an associated file with a ".log" extension that acts as a transaction log that is used to ensure that any interrupted updates can be completed upon next startup.[21] Internally, Registry files are split into 4 kB "bins" that contain collections of "cells".[21]

The registry files are stored in the `%WINDIR%` directory under the names `USER.DAT` and `SYSTEM.DAT` with the addition of `CLASSES.DAT` in Windows ME. Also, each user profile (if profiles are enabled) has its own `USER.DAT` file which is located in the user's profile directory in `%WINDIR%\Profiles\<Username>\` .

The only registry file is called `REG.DAT` and it is stored in the `%WINDIR%` directory.

To access the registry files, the device needs to be set in a special mode using either:

- WpInternals (Put the device into flash mode.)
- InterOp Tools (Mount the MainOS Partition with MTP.)

If any of the above methods worked, the device's registry files can be found in the following location:

```
{Phone}\EFIESP\Windows\System32\config
```

The registry contains important configuration information for the operating system, for installed applications as well as individual settings for each user and application. A careless change to the operating system configuration in the registry could cause irreversible damage, so it is usually only installer programs which perform changes to

the registry database during installation/configuration and removal. If a user wants to edit the registry manually, Microsoft recommends that a backup of the registry be performed before the change.[22] When a program is removed from control panel, it may not be completely removed and, in case of errors or glitches caused by references to missing programs, the user might have to manually check inside directories such as program files. After this, the user might need to manually remove any reference to the uninstalled program in the registry. This is usually done by using `RegEdit.exe`.[23] Editing the registry is sometimes necessary when working around Windows-specific issues e.g. problems when logging onto a domain can be resolved by editing the registry.[24]

Windows Registry can be edited manually using programs such as `RegEdit.exe`, although these tools do not expose some of the registry's metadata such as the last modified date.

The registry editor for the 3.1/95 series of operating systems is `RegEdit.exe` and for Windows NT it is `RegEdt32.exe`; the functionalities are merged in Windows XP. Optional and third-party tools similar to `RegEdit.exe` are available for many Windows CE versions.

Registry Editor allows users to perform the following functions:

- Creating, manipulating, renaming[25] and deleting registry keys, subkeys, values and value data
- Importing and exporting `.REG` files, exporting data in the binary hive format
- Loading, manipulating and unloading registry hive format files (Windows NT systems only)
- Setting permissions based on ACLs (Windows NT systems only)
- Bookmarking user-selected registry keys as Favorites
- Finding particular strings in key names, value names and value data
- Remotely editing the registry on another networked computer

`.REG` files (also known as Registration entries) are text-based human-readable files for exporting and importing portions of the registry using an INI-based syntax. There are two main versions of REG files:

- Windows 9x and NT 4.0 REG files are ANSI-based. They start with the string `REGEDIT4`.[26]
- Windows 2000 and later REG files are Unicode-based. They start with the string *Windows Registry Editor Version 5.00*.

Windows 9x format `.REG` files can be imported by Windows 2000 and later.[26] These later systems also allow exporting `.REG` files in Windows 9x/NT format.[*citation needed*]

Data is stored in `.REG` files using the following syntax:[26]

```
[<Hive name>\<Key name>\<Subkey name>]
"Value name"=<Value type>:<Value data>
```

The Default Value of a key can be edited by using `@` instead of "Value Name":

```
[<Hive name>\<Key name>\<Subkey name>]
@=<Value type>:<Value data>
```

String values do not require a `<Value type>` (see example), but backslashes (`\`) need to be written as a double-backslash (`\\`), and quotes (`"`) as backslash-quote (`\"`). (The requirement for escaping is not totally consistent: files containing strings with unescaped leading backslash do exist and are accepted by the system for importing.)[27]

For example, to add the values "Value A", "Value B", etc. to the `HKLM\SOFTWARE\Foobar` key:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Foobar]
"Value A"="<String value data with escape characters>"
"Value B"=hex:<Binary data (as comma-delimited list of hexadecimal values)>
"Value C"=dword:<DWORD value integer>
"Value D"=hex(0):<REG_NONE (as comma-delimited list of hexadecimal values)>
"Value E"=hex(1):<REG_SZ (as comma-delimited list of hexadecimal values representing a UTF-16LE NUL-
"Value F"=hex(2):<Expandable string value data (as comma-delimited list of hexadecimal values repres
"Value G"=hex(3):<Binary data (as comma-delimited list of hexadecimal values)> ; equal to "Value B"
"Value H"=hex(4):<DWORD value (as comma-delimited list of 4 hexadecimal values, in little endian byt
"Value I"=hex(5):<DWORD value (as comma-delimited list of 4 hexadecimal values, in big endian byte o
"Value J"=hex(7):<Multi-string value data (as comma-delimited list of hexadecimal values representin
"Value K"=hex(8):<REG_RESOURCE_LIST (as comma-delimited list of hexadecimal values)>
"Value L"=hex(a):<REG_RESOURCE_REQUIREMENTS_LIST (as comma-delimited list of hexadecimal values)>
"Value M"=hex(b):<QWORD value (as comma-delimited list of 8 hexadecimal values, in little endian byt
```

Data from `.REG` files can be added/merged with the registry by double-clicking these files or using the /s switch in the command line. `REG` files can also be used to remove registry data.

To remove a key (and all subkeys, values and data), the key name must be preceded by a minus sign (`-`).[26]

For example, to remove the `HKLM\SOFTWARE\Foobar` key (and all subkeys, values and data),

```
[-HKEY_LOCAL_MACHINE\SOFTWARE\Foobar]
```

To remove a value (and its data), the values to be removed must have a minus sign (`-`) after the equal sign (`=`). [26]

For example, to remove only the "Value A" and "Value B" values (and their data) from the `HKLM\SOFTWARE\Foobar` key:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Foobar]
"Value A"=-
"Value B"=-
```

To remove only the Default value of the key `HKLM\SOFTWARE\Foobar` (and its data):

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Foobar]
@=-
```

Lines beginning with a semicolon are considered comments:

```
; This is a comment. This can be placed in any part of a .reg file
[HKEY_LOCAL_MACHINE\SOFTWARE\Foobar]
"Value"="Example string"
```

Windows group policies can change registry keys for a number of machines or individual users based on policies. When a policy first takes effect for a machine or for an individual user of a machine, the registry settings specified as part of the policy are applied to the machine or user settings.

Windows will also look for updated policies and apply them periodically, typically every 90 minutes.[28]

Through its *scope* a policy defines to which machines and users the policy is to be applied. Whether a machine or user is within the scope of a policy or not is defined by a set of rules which can filter on the location of the machine or user account in organizational directory, specific users or machine accounts or security groups. More advanced rules can be set up using Windows Management Instrumentation expressions. Such rules can filter on properties such as computer vendor name, CPU architecture, installed software, or networks connected to.

For instance, the administrator can create a policy with one set of registry settings for machines in the accounting department and policy with another (lock-down) set of registry settings for kiosk terminals in the visitors area. When a machine is moved from one scope to another (e.g., changing its name or moving it to another organizational unit), the correct policy is automatically applied. When a policy is changed it is automatically re-applied to all machines currently in its scope.

The policy is edited through a number of administrative templates which provides a user interface for picking and changing settings. The set of administrative templates is extensible and software packages which support such remote administration can register their own templates.

Command line editing

[edit]

reg	
Developer	Microsoft
Operating system	Microsoft Windows
Type	Command
License	Proprietary commercial software

Website	docs.microsoft.com/en-us/windows-server/administration/windows-commands/reg
regini	
Developer	Microsoft
Operating system	Microsoft Windows
Type	Command
License	Proprietary commercial software
Website	docs.microsoft.com/en-us/windows-server/administration/windows-commands/regini

The registry can be manipulated in a number of ways from the command line. The `Reg.exe` and `RegIni.exe` utility tools are included in Windows XP and later versions of Windows. Alternative locations for legacy versions of Windows include the Resource Kit CDs or the original Installation CD of Windows.

Also, a `.REG` file can be imported from the command line with the following command:

```
RegEdit.exe /s file
```

The /s means the file will be *silent merged* to the registry. If the `/s` parameter is omitted the user will be asked to confirm the operation. In Windows 98, Windows 95 and at least some configurations of Windows XP the `/s` switch also causes `RegEdit.exe` to ignore the setting in the registry that allows administrators to disable it. When using the `/s` switch `RegEdit.exe` does not return an appropriate return code if the operation fails, unlike `Reg.exe` which does.

```
RegEdit.exe /e file
```

exports the whole registry in V5 format to a UNICODE `.REG` file, while any of

```
RegEdit.exe /e file HKEY_CLASSES_ROOT[\<key>]
RegEdit.exe /e file HKEY_CURRENT_CONFIG[\<key>]
RegEdit.exe /e file HKEY_CURRENT_USER[\<key>]
RegEdit.exe /e file HKEY_LOCAL_MACHINE[\<key>]
RegEdit.exe /e file HKEY_USERS[\<key>]
```

export the specified (sub)key (which has to be enclosed in quotes if it contains spaces) only.

```
RegEdit.exe /a file
```

exports the whole registry in V4 format to an ANSI `.REG` file.

```
RegEdit.exe /a file <key>
```

exports the specified (sub)key (which has to be enclosed in quotes if it contains spaces) only.

It is also possible to use `Reg.exe`. Here is a sample to display the value of the registry value Version:

```
Reg.exe QUERY HKLM\Software\Microsoft\ResKit /v Version
```

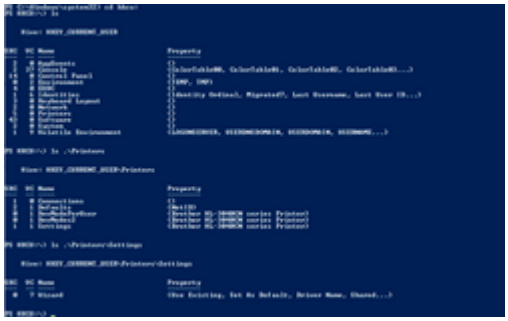
Other command line options include a VBScript or JScript together with CScript, WMI or `WMIC.exe` and Windows PowerShell.

Registry permissions can be manipulated through the command line using `RegIni.exe` and the `SubInACL.exe` tool. For example, the permissions on the `HKEY_LOCAL_MACHINE\SOFTWARE` key can be displayed using:

```
SubInACL.exe /keyreg HKEY_LOCAL_MACHINE\SOFTWARE /display
```

PowerShell commands and scripts

[edit]



Using PowerShell to navigate the registry

Windows PowerShell comes with a registry provider which presents the registry as a location type similar to the file system. The same commands used to manipulate files and directories in the file system can be used to manipulate keys and values of the registry.[29]

Also like the file system, PowerShell uses the concept of a *current location* which defines the context on which commands by default operate. The `Get-ChildItem` (also available through the aliases `ls`, `dir` or `gci`) retrieves the child keys of the current location. By using the `Set-Location` (or the alias `cd`) command the user can change the current location to another key of the registry.[29] Commands which rename items, remove items, create new items or set content of items or properties can be used to rename keys, remove keys or entire sub-trees or change values.

Through PowerShell scripts files, an administrator can prepare scripts which, when executed, make changes to the registry. Such scripts can be distributed to administrators who can execute them on individual machines. The PowerShell Registry provider supports transactions, i.e. multiple changes to the registry can be bundled into a

single atomic transaction. An atomic transaction ensures that either all of the changes are committed to the database, or if the script fails, none of the changes are committed to the database.[29][30]

Programs or scripts

[edit]

The registry can be edited through the APIs of the Advanced Windows 32 Base API Library (advapi32.dll).[31] List of registry API functions:

- RegCloseKey
- RegConnectRegistry
- RegCreateKey
- RegCreateKeyEx
- RegDeleteKey
- RegDeleteValue
- RegEnumKey
- RegEnumKeyEx
- RegEnumValue
- RegFlushKey
- RegGetKeySecurity
- RegLoadKey
- RegNotifyChangeKeyValue
- RegOpenKey
- RegOpenKeyEx
- RegQueryInfoKey
- RegQueryMultipleValues
- RegQueryValue
- RegQueryValueEx
- RegReplaceKey
- RegRestoreKey
- RegSaveKey
- RegSetKeySecurity
- RegSetValue
- RegSetValueEx
- RegUnLoadKey

Many programming languages offer built-in runtime library functions or classes that wrap the underlying Windows APIs and thereby enable programs to store settings in the registry (e.g. `Microsoft.Win32.Registry` in VB.NET and C#, or `TRegistry` in Delphi and Free Pascal). COM-enabled applications like Visual Basic 6 can use the WSH `WScript.Shell` object. Another way is to use the Windows Resource Kit Tool, `Reg.exe` by executing it from code,[32] although this is considered poor programming practice.

Similarly, scripting languages such as Perl (with `Win32::TieRegistry`), Python (with winreg), Tcl (which comes bundled with the registry package),[33] Windows Powershell and Windows Scripting Host also enable registry editing from scripts.

The offreg.dll[34] available from the Windows Driver Kit offers a set of APIs for the creation and manipulation of currently not loaded registry hives similar to those provided by advapi32.dll.

It is also possible to edit the registry (hives) of an offline system from Windows PE or Linux (in the latter case using open source tools).

COM self-registration

[edit]

Prior to the introduction of registration-free COM, developers were encouraged to add initialization code to in-process and out-of-process binaries to perform the registry configuration required for that object to work. For in-process binaries such as .DLL and .OCX files, the modules typically exported a function called DllInstall()[35] that could be called by installation programs or invoked manually with utilities like Regsvr32.exe;[36] out-of-process binaries typically support the commandline arguments /Regserver and /Unregserver that created or deleted the required registry settings.[37] COM applications that break because of DLL Hell issues can commonly be repaired with RegSvr32.exe or the /RegServer switch without having to re-invoke installation programs.[38]

Advanced functionality

[edit]

Windows exposes APIs that allows user-mode applications to register to receive a notification event if a particular registry key is changed.[39] APIs are also available to allow kernel-mode applications to filter and modify registry calls made by other applications.[40]

Windows also supports remote access to the registry of another computer via the `RegConnectRegistry` function[41] if the Remote Registry service is running, correctly configured and its network traffic is not firewalled.[42]

Each key in the registry of Windows NT versions can have an associated security descriptor. The security descriptor contains an access control list (ACL) that describes which user groups or individual users are granted or denied access permissions. The set of registry permissions include 10 rights/permissions which can be explicitly allowed or denied to a user or a group of users.

Registry permissions

Permission	Description
Query Value	The right to read the registry key value.
Set Value	The right to write a new value

Create Subkey	The right to create subkeys.
Enumerate Subkeys	Allow the enumeration of subkeys.
Notify	The right to request change notifications for registry keys or subkeys.
Create Link	Reserved by the operating system.
Delete	The right to delete a key.
Write DACL	The right to modify permissions of the container's DACL.
Write Owner	The right to modify the container's owner.
Read Control	The right to read the DACL.

As with other securable objects in the operating system, individual access control entries (ACE) on the security descriptor can be explicit or inherited from a parent object.[43]

Windows Resource Protection is a feature of Windows Vista and later versions of Windows that uses security to deny Administrators and the system WRITE access to some sensitive keys to protect the integrity of the system from malware and accidental modification.[44]

Special ACEs on the security descriptor can also implement mandatory integrity control for the registry key and subkeys. A process running at a lower integrity level cannot write, change or delete a registry key/value, even if the account of the process has otherwise been granted access through the ACL. For instance, Internet Explorer running in Protected Mode can *read* medium and low integrity registry keys/values of the currently logged on user, but it can only modify low integrity keys.[45]

Outside security, registry keys cannot be deleted or edited due to other causes. Registry keys containing NUL characters cannot be deleted with standard registry editors and require a special utility for deletion, such as RegDelNull.[46][47]

Backups and recovery

[edit]

Different editions of Windows have supported a number of different methods to back up and restore the registry over the years, some of which are now deprecated:

- System Restore can back up the registry and restore it as long as Windows is bootable, or from the Windows Recovery Environment (starting with Windows Vista).
- NTBackup can back up the registry as part of the *System State* and restore it. Automated System Recovery in Windows XP can also restore the registry.
- On Windows NT, the *Last Known Good Configuration* option in startup menu relinks the `HKLM\SYSTEM\CurrentControlSet` key, which stores hardware and device driver information.

- Windows 98 and Windows ME include command line (Scanreg.exe) and GUI (Scanregw.exe) registry checker tools to check and fix the integrity of the registry, create up to five automatic regular backups by default and restore them manually or whenever corruption is detected.[48] The registry checker tool backs up the registry, by default, to `%Windir%\Sysbckup` Scanreg.exe can also run from MS-DOS.[49]
- The Windows 95 CD-ROM included an Emergency Recovery Utility (ERU.exe) and a Configuration Backup Tool (Cfgback.exe) to back up and restore the registry. Additionally Windows 95 backs up the registry to the files system.da0 and user.da0 on every successful boot.
- Windows NT 4.0 included `RDISK.EXE` , a utility to back up and restore the entire registry.[50]
- Windows 2000 Resource Kit contained an unsupported pair of utilities called Regback.exe and RegRest.exe for backup and recovery of the registry.[51]
- Periodic automatic backups of the registry are now disabled by default on Windows 10 May 2019 Update (version 1903). Microsoft recommends System Restore be used instead.[52]

Windows 2000 and later versions of Windows use Group Policy to enforce registry settings through a registry-specific client extension in the Group Policy processing engine.[53] Policy may be applied locally to a single computer using `gpedit.msc` or to multiple users and computers in a domain using `gpmc.msc` .

With Windows 95, Windows 98, Windows ME and Windows NT 4.0, administrators can use a special file to be merged into the registry, called a policy file (`POLICY.POL`). The policy file allows administrators to prevent non-administrator users from changing registry settings like, for instance, the security level of Internet Explorer and the desktop background wallpaper. The policy file is primarily used in a business with a large number of computers where the business needs to be protected from rogue or careless users.

The default extension for the policy file is `.POL` . The policy file filters the settings it enforces by user and by group (a "group" is a defined set of users). To do that the policy file merges into the registry, preventing users from circumventing it by simply changing back the settings. The policy file is usually distributed through a LAN, but can be placed on the local computer.

The policy file is created by a free tool by Microsoft that goes by the filename `poledit.exe` for Windows 95/Windows 98 and with a computer management module for Windows NT. The editor requires administrative permissions to be run on systems that uses permissions. The editor can also directly change the current registry settings of the local computer and if the remote registry service is installed and started on another computer it can also change the registry on that computer. The policy editor loads the settings it can change from `.ADM` files, of which one is included, that contains the settings the Windows shell provides. The `.ADM` file is plain text and supports easy localization by allowing all the strings to be stored in one place.

INI file virtualization

[edit]

Windows NT kernels support redirection of INI file-related APIs into a virtual file in a registry location such as HKEY_CURRENT_USER using a feature called "InifileMapping".[54] This functionality was introduced to allow legacy applications written for 16-bit versions of Windows to be able to run under Windows NT platforms on which the System folder is no longer considered an appropriate location for user-specific data or configuration.

Non-compliant 32-bit applications can also be redirected in this manner, even though the feature was originally intended for 16-bit applications.

Registry virtualization

[edit]

Windows Vista introduced limited registry virtualization, whereby poorly written applications that do not respect the principle of least privilege and instead try to write user data to a read-only system location (such as the HKEY_LOCAL_MACHINE hive), are silently redirected to a more appropriate location, without changing the application itself.

Similarly, application virtualization redirects all of an application's invalid registry operations to a location such as a file. Used together with file virtualization, this allows applications to run on a machine without being installed on it.

Low integrity processes may also use registry virtualization. For example, Internet Explorer 7 or 8 running in "Protected Mode" on Windows Vista and above will automatically redirect registry writes by ActiveX controls to a sandboxed location in order to frustrate some classes of security exploits.

The Application Compatibility Toolkit[55] provides shims that can transparently redirect HKEY_LOCAL_MACHINE or HKEY_CLASSES_ROOT Registry operations to HKEY_CURRENT_USER to address "LUA" bugs that cause applications not to work for users with insufficient rights.

Critics labeled the registry in Windows 95 a single point of failure, because re-installation of the operating system was required if the registry became corrupt. However, Windows NT uses transaction logs to protect against corruption during updates. Current versions of Windows use two levels of log files to ensure integrity even in the case of power failure or similar catastrophic events during database updates.[56] Even in the case of a non-recoverable error, Windows can repair or re-initialize damaged registry entries during system boot.[56]

Richard WM Jones, author of libguestfs and hivex (tool to read and write the registry from systems other than the host Windows installation, including non-Windows systems), makes the following critiques:[27]

- Compared to a database with features such as indexed columns, the registry with its concepts of directories (keys), files (values), and extended attributes (permissions) is more of a file system, and an incomplete one at that. Lack of column indexing has led to the proliferation of oddly-named keys such as `\ControlSet001\Control\CriticalDeviceDatabase\pci#ven_1af4&dev_1001&subsys_00000000`.
- The hive format is based on the struct layout of the original compiler in the 1990s. The reader implementation in Windows performs insufficient validation, being easy to make to hang or crash by incorrect pointer structures (reference loops, out-of-range pointers). The writer implementation does not properly zero out unused memory, leaking kernel data structures into unused parts of the registry.
- Different versions of Windows have different ideas about what each "type" in the registry refers to. String types such as `REG_SZ` were originally stored in 7-bit ASCII, before being switched to UTF-16LE later, so to properly read and write a hive one would need to know the Windows version it is associated with.

Equivalents and alternatives

[edit]

In Windows, use of the registry for storing program data is a matter of developer's discretion. Microsoft provides programming interfaces for storing data in XML files (via MSXML) or database files (via SQL Server Compact) which developers can use instead. Developers are also free to use non-Microsoft alternatives or develop their own proprietary data stores.

In contrast to Windows Registry's binary-based database model, some other operating systems use separate plain-text files for daemon and application configuration, but group these configurations together for ease of management.

- In Unix-like operating systems (including Linux) that follow the Filesystem Hierarchy Standard, system-wide configuration files (information similar to what would appear in HKEY_LOCAL_MACHINE on Windows) are traditionally stored in files in `/etc/` and its subdirectories, or sometimes in `/usr/local/etc/`. Per-user information (information that would be roughly equivalent to that in HKEY_CURRENT_USER) is stored in hidden directories and files (that start with a period/full stop) within the user's home directory. However XDG-compliant applications should refer to the environment variables defined in the Base Directory specification.[57]
- In macOS, system-wide configuration files are typically stored in the `/Library/` folder, whereas per-user configuration files are stored in the corresponding `~/Library/` folder in the user's home directory, and configuration files set by the system are in `/System/Library/`. Within these respective directories, an application typically stores a property list file in the `Preferences/` sub-directory.
- RISC OS (not to be confused with *MIPS RISC/os*) uses directories for configuration data, which allows applications to be copied into application directories, as opposed to the separate installation process that typifies Windows applications; this approach is also used on the ROX Desktop for Linux.[58] This directory-based configuration also makes it possible to use different versions of the same application, since the configuration is done "on the fly".[59] If one wishes to remove the application, it is possible to simply delete the folder belonging to the application.[60][61] This will often not remove configuration settings which are stored independently from the application, usually within the computer's !Boot structure, in !Boot Choices or potentially anywhere on a network fileserver. It is possible to copy installed programs between computers running RISC OS by copying the application directories belonging to the programs, however some programs may require re-installing, e.g. when shared files are placed outside an application directory.[59]
- IBM AIX (a Unix variant) uses a registry component called Object Data Manager (ODM). The ODM is used to store information about system and device configuration. An extensive set of tools and utilities provides users with means of extending, checking, correcting the ODM database. The ODM stores its information in several files, default location is `/etc/objrepos`.
- The GNOME desktop environment uses a registry-like interface called dconf for storing configuration settings for the desktop and applications.
- The Elektra Initiative provides alternative back-ends for various different text configuration files.

- While not an operating system, the Wine compatibility layer, which allows Windows software to run on a Unix-like system, also employs a Windows-like registry as text files in the WINEPREFIX folder: system.reg (HKEY_LOCAL_MACHINE), user.reg (HKEY_CURRENT_USER) and userdef.reg.[62] The format is virtually the same as those of Windows `.REG` files, except that the header line is changed to "WINE REGISTRY Version 2" and paths (keys) do not start with the name of the hive.

- Registry cleaner
- Logparser – SQL-like querying of various types of log files
- List of shell icon overlay identifiers
- Ransomware attack that uses Registry

1. ^ When applications fail to execute because they request more privileges than they require (and are denied those privileges), this is known as a limited user application (LUA) bug.

1. ^ *Esposito, Dino (November 2000). "Windows 2000 Registry: Latest Features and APIs Provide the Power to Customize and Extend Your Apps". MSDN Magazine. Microsoft. Archived from the original on April 15, 2003. Retrieved July 19, 2007.*
2. ^ Jump up to: ***a*** ***b*** ***c*** *"The System Registry"*.
3. ^ *"Windows 95 Architecture Components". www.microsoft.com. Archived from the original on February 7, 2008. Retrieved April 29, 2008. "The following table shows other difficulties or limitations caused by using .INI files that are overcome by using the Registry."*
4. ^ Hipson 2002, p. 5, 41–43.
5. ^ *Richter, Jeffrey; Nasarre, Christophe (2008). Windows Via C/C++ (Fifth ed.). Microsoft Press. ISBN 9780735642461. Retrieved August 28, 2021.*
6. ^ Raymond Chen, "Why do registry keys have a default value?"
7. ^ Hipson 2002, pp. 207, 513–514.
8. ^ Hipson 2002, pp. 520–521.
9. ^ Hipson 2002, p. 7.
10. ^ *"Designed for Windows XP Application Specification". Microsoft. August 20, 2002. Retrieved April 8, 2009.*
11. ^ *"HKEY_LOCAL_MACHINE". Gautam. 2009. Retrieved April 8, 2009.*
12. ^ *"Registry Keys Affected by WOW64 (Windows)". Msdn.microsoft.com. Retrieved April 10, 2014.*
13. ^ *"Description of the Microsoft Windows registry". Retrieved September 25, 2008.*
14. ^ *"HKEY_CURRENT_USER". Microsoft. 2009. Retrieved April 8, 2009.*
15. ^ *"Description of the HKEY_DYN_DATA Registry Key in Windows 95, Windows 98, and Windows 98 SE". support.microsoft.com.*
16. ^ *"A Closer Look at HKEY_DYN_DATA". rinet.ru. Archived from the original on May 9, 2008.*
17. ^ *"Registry hives". Retrieved July 19, 2007.*
18. ^ *Chen, Raymond (August 8, 2011). "Why is a registry file called a "hive"?". The Old New Thing. Retrieved July 29, 2011.*
19. ^ *"Overview of the Windows NT Registry". Retrieved December 2, 2011.*
20. ^ *"Inside the Registry". Retrieved December 28, 2007.*

21. ^ Jump up to: *a b* *Norris, Peter (February 2009). "The Internal Structure of the Windows Registry" (PDF). Cranfield University. Archived from the original (PDF) on May 29, 2009.*
22. ^ *"Incorrect Icons Displayed for .ico Files". November 15, 2009. Retrieved March 31, 2012.*
23. ^ *"How to Completely Uninstall / Remove a Software Program in Windows without using 3rd Party Software? - AskVG". www.askvg.com. August 26, 2011.*
24. ^ *"You may receive a "STOP 0x00000035 NO_MORE_IRP_STACK_LOCATIONS" error message when you try to log on to a domain". October 9, 2011. Retrieved March 31, 2012.* This page tells the user to edit the registry when resolving the issue.
25. ^ key renaming is implemented as removal and add while retaining subkeys/values, as the underlying APIs do not support the rename function directly
26. ^ Jump up to: *a b c d e* *"How to add, modify, or delete registry subkeys and values by using a .reg file". support.microsoft.com.*
27. ^ Jump up to: *a b* *"Why the Windows Registry sucks … technically". Richard WM Jones. February 18, 2010.*
28. ^ *"Applying Group Policy". Microsoft.*
29. ^ Jump up to: *a b c* *Payette, Bruce; Siddaway, Richard (2018). Windows PowerShell in Action (Third ed.). Manning Publications. pp. 7–8, 24, 608, 708–710. ISBN 9781633430297. Retrieved August 28, 2021.*
30. ^ *Warner, Timothy L. (May 2015). Windows PowerShell in 24 Hours, Sams Teach Yourself. Sams Publishing. p. 19, 211. ISBN 9780134049359. Retrieved August 28, 2021.*
31. ^ *"Reading and Writing Registry Values with Visual Basic". Retrieved July 19, 2007.*
32. ^ *"REG command in Windows XP". Retrieved July 19, 2007.*
33. ^ *"registry manual page – Tcl Bundled Packages". www.tcl.tk. Retrieved December 14, 2017.*
34. ^ *"Offline Registry Library". Retrieved June 4, 2014.*
35. ^ *"DllInstall Function". Microsoft. March 7, 2012. Retrieved March 22, 2012.*
36. ^ *"Regsvr32". Microsoft. Retrieved March 22, 2012.*
37. ^ *"How to: Register Automation Servers". Microsoft. Retrieved March 22, 2012.*
38. ^ *"How to re-register PowerPoint 2000, PowerPoint 2003, PowerPoint 2007 and PowerPoint 2010". Microsoft. January 2012. Retrieved March 22, 2012.*
39. ^ *"RegNotifyChangeKeyValue function". Microsoft.*
40. ^ *"Registering for Notifications". Microsoft.*
41. ^ *"RegConnectRegistry function". Microsoft.*
42. ^ *"How to Manage Remote Access to the Registry". Microsoft.*
43. ^ *Gibson, Darril (June 28, 2011). "Chapter 4: Securing Access with Permissions". Microsoft Windows security : essentials. Indianapolis, Ind.: Wiley. ISBN 978-1-118-01684-8.*
44. ^ *"Application Compatibility: Windows Resource Protection (WRP)". Microsoft. Retrieved August 8, 2012.*
45. ^ *Marc Silbey, Peter Brundrett. "Understanding and Working in Protected Mode Internet Explorer". Retrieved August 8, 2012.*
46. ^ *"RegDelNull v1.1". November 1, 2006. Retrieved August 8, 2012.*
47. ^ *"Unable to delete certain registry keys – Error while deleting key". March 23, 2010. Retrieved August 8, 2012.* Microsoft Support page.
48. ^ *"Description of the Windows Registry Checker Tool (Scanreg.exe)".*
49. ^ *"Command-Line Switches for the Registry Checker Tool".*

50. ^ *"How To Backup, Edit, and Restore the Registry in Windows NT 4.0". support.microsoft.com.*
51. ^ *"Technical Reference to the Registry: Related Resources". Microsoft. Retrieved September 9, 2011.*
52. ^ *Whitwam, Ryan (July 2019). "Microsoft Kills Automatic Registry Backups in Windows 10". ExtremeTech. Retrieved July 1, 2019.*
53. ^ *"How Core Group Policy Works". Microsoft. September 2, 2009. Retrieved August 13, 2012.*
54. ^ *"Chapter 26 – Initialization Files and the Registry". Microsoft. Retrieved March 3, 2008.*
55. ^ *"Microsoft Application Compatibility Toolkit 5.0". Microsoft. Retrieved July 26, 2008.*
56. ^ Jump up to: ***a*** ***b*** *Ionescu, Mark Russinovich, David A. Solomon, Alex (2012). "Registry Internals". Windows internals (6th ed.). Redmond, Wash.: Microsoft Press. ISBN 978-0-7356-4873-9.* {{cite book}} : CS1 maint: multiple names: authors list (link)
57. ^ *"XDG Base Directory Specification". standards.freedesktop.org.*
58. ^ *"Application directories". Archived from the original on May 27, 2012. Retrieved May 17, 2012.*
59. ^ Jump up to: ***a*** ***b*** *"Case Studies Of The Top 132 Annoyances With Operating Systems Other Than RISC OS". Retrieved April 3, 2012.* Page from the riscos.com website. Mentioned in points 82 and 104.
60. ^ *"RISC OS tour". Retrieved July 19, 2007.*
61. ^ *"The RISC OS Products Directory". November 2, 2006. Retrieved April 1, 2012.* {{cite web}} : CS1 maint: deprecated archival service (link)
62. ^ 3.2. Using the Registry and Regedit (Wine User Guide)

- *Hipson, Peter (2002). Mastering Windows XP Registry. Wiley. ISBN 0-7821-2987-0. Retrieved August 28, 2021.*
- *Russinovich, Mark E.; Solomon, David A. (2005). Microsoft Windows Internals (Fourth ed.). Microsoft Press. pp. 183–236. ISBN 978-0-7356-1917-3.*

- Windows Registry info & reference in the MSDN Library
