Registry Overview

- A hierarchical database containing system, machine, and user specific
- Replacement for INI
- Used in all aspects of Windows development from COM to Service
- Cornerstone of inter-process communication

Initialization Files

Traditionally, initialization data has been stored in INI files. This data consisted of user profiling, device driver, machine related information and much more. The biggest drawbacks of INI files were polar opposites.

- Toward the end of the evolution of 16 bit windows, INI files began to proliferate. After installing several products over a period of time, INI files would begin to accumulate on a hard drive. INI files might still be present for applications that had been removed from the system. A cottage industry emerged for utilities that cleaned up hard files of INI files and other vestiges of programs remaining on a system when application was removed.

- Many developers use windows.ini and system.ini to write product initialization data. This solved the problem of INI proliferation; but caused another. A defective application would write to the system initialization files incorrectly and corrupt the Windows environment.

- The registry is an extension of the registration database of 16 bit Windows. It is a hierarchical database of initialization, system related, machine specific, and user profiling data. Instead of maintaining a plethora of INI files on a machine, every application writes its data to this database. Importantly, the system or critical data of the registry is protected from inadvertent corruption by errant applications. This registry resolves the two major problems of INI files.

The registry improves upon the concept of initialization files and many ways:

- Data can be written to the registry in a variety of formats. INI files contained text. In many circumstances, text is the least efficient storage medium for data. The registry accepts binary, hex, DWORD, UNICODE, and many more data formats.

- The registry is accessible through specialized utilities provided with the operating system and registry APIs. INIs were accessible by any user via a basic text processor. This provided no protection for sensitive data in the INI file. A registry tool is provided with every installation of a Win32 operating system. However, only the more sophisticated user will be aware of the existence of these tools. In addition, the registry has security and certain data is protected from user interaction.

- The registry tools can be used to monitor and maintain the registry of a remote machine. This is an important feature for administrators. Remote access of the registry allows the administrator to fine-tune performance, add users, and delete users from a target machine.

Registry and Windows Explorer

If you can navigate the DOS file system or Windows Explorer, many parallels can be drawn. Both the file system and the registry are *tree based*. Instead of partitions, the registry offers hives. The equivalent of a directory in the registry is a key. Subkeys are akin to sub-directories. Values in the registry contain data like files in a file system.

File System Notation	Registry Equivalent
Partitions	Hives
Directories	Keys
Subdirectories	Subkeys
Files	Values

The delimiter used in a file system path is a slash "\". The registry path requires the same delimited to separate various levels of registry hierarchy.

```
File System: C:\program files\devstudio\vc\bin\uuidgen.exe

Registry: HKEY_CURRENT_USER\Microsoft\Software\data
```

Registry Utilities

- RegEdt32
- RegEdit

There are two utilities for inspecting the registry. The utilities are similar but present a slightly different GUI. RegEdit is available with Windows NT and Windows 95. RegEdt32 is a Windows NT specific tool. Neither the Win32 SDK nor Visual C++ compiler are needed to use these programs.

In Windows NT the path is:

```
drive:\winnt\system32\regedt32.exe
```

and

```
drive:\winnt\regedit.exe
```

Both executable should be immediately available through the default path of the Windows NT environment. Also, both present a GUI similar to Windows Explorer or the old File Manager application.

Careful: Registry modifications are permanent and immediate. There is no *undo* command in RegEdt32 or RegEdit .

The Registry Hierarchy

The Registry is structured as a set of subtrees of keys that contain per-computer and per-user databases. The per-computer information includes information about hardware and software installed on the computer. The per-user information includes the information in user profiles, such as desktop settings, individual preferences for certain software, and personal printer and network settings. In versions of Windows for MS-DOS, per-computer information was saved in the Win.ini and System.ini files, but it was not possible to save separate information for individual users.
In the Windows NT Registry, each individual key can contain data items called *value entries* and can also contain additional *subkeys.* In the Registry structure, keys are analogous to directories, and the value entries are analogous to files.

Each of these subtrees is described in detail later in this chapter. Each of the root key names begins with "HKEY_" to indicate to software developers that this is a *handle* that can be used by a program. A handle is a value used to uniquely identify a resource so that a program can access it.

HKEY_LOCAL_MACHINE

Contains information about the local computer system, including hardware and operating system data such as bus type, system memory, device drivers, and startup control data.

HKEY_CLASSES_ROOT

Contains the associations between applications and file types (by filename extension). It also contains object linking and embedding (OLE) Registry information associated with COM objects, and file-class association data (equivalent to the Registry in Windows for MS-DOS). The entries in this subtree are the same as in HKEY_LOCAL_MACHINE\Software\Classes.
For detailed information on HKEY_CLASSES_ROOT, see the *OLE Programmer's Reference* in the Windows NT 4.0 Software Developer's Kit.

HKEY_CURRENT_CONFIG

Contains configuration data for the current hardware profile. Hardware profiles are sets of changes to the standard configuration of services and devices established by data in the Software and System keys under HKEY_LOCAL_MACHINE. Only the changes appear in HKEY_CURRENT_CONFIG.
The entries in this subtree also appear in:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\HardwareProfiles\Current.

HKEY_CURRENT_USER

Contains the user profile for the user who is logged on, including environment variables, desktop settings, network connections, printers, and application preferences.

HKEY_USERS

Contains all actively loaded user profiles, including HKEY_CURRENT_USER, and the default profile. Users who are accessing a server remotely do not have profiles under this key on the server; their profiles are loaded into the Registry on their own computers.

Hives and Files

The Registry is divided into parts called *hives*. A hive is a discrete body of keys, subkeys, and values rooted at the top of the Registry hierarchy. Hives are distinguished from other groups of keys in that they are permanent components of the Registry; they are not created dynamically when the system starts and deleted when it stops. Thus, HKEY_LOCAL_MACHINE\Hardware, which is built dynamically by the Hardware Recognizer when Windows NT starts, is not a hive.
Data in the hives is supported by files in the *Systemroot*\System32\Config and *Systemroot*\Profiles*Username* subdirectories. Figure 23.7 shows the relationship between the hives and their supporting files.

Each hive in the Windows NT Registry is associated with a set of standard files

Registry hive	Filenames
HKEY_LOCAL_MACHINE\SAM	**Sam, Sam.log, Sam.sav**
HKEY_LOCAL_MACHINE\Security	**Security, Security.log, Security.sav**
HKEY_LOCAL_MACHINE\Software	**Software, Software.log, Software.sav**
HKEY_LOCAL_MACHINE\System	**System, System.alt, System.log, System.sav**
HKEY_CURRENT_CONFIG	**System, System.alt, System.log, System.sav**
HKEY_USERS\.DEFAULT	**Default, Default.log, Default.sav**
(Not associated with a hive)	**Userdiff, Userdiff.log**
HKEY_CURRENT_USER	**Ntuser.dat, Ntuser.dat.log**

By default, the supporting files for all hives except HKEY_CURRENT_USER are in *Systemroot*\System32\Config.
The HKEY_CURRENT_USER support files are stored in all subdirectories of *Systemroot*\Profiles, except for the All Users subdirectory.
The Ntuser.dat files store user profiles; the Ntuser.dat.log files track changes to Ntuser.dat.
The Ntuser and Userdiff files are new to Windows NT 4.0:

- The Ntuser.dat file, which stores the user profile, replaces the *usernamexxx* and *adminxxx* files in previous versions of Windows NT.

- The Ntuser.dat file in *Systemroot*\Profiles\DefaultUser replaces the Userdef file in previous versions of Windows NT. This profile is used to create the HKEY_CURRENT_USER hive when a new user logs on to Windows NT for the first time.

- The Userdiff files, which are only in *Systemroot*\System32\Config, are not associated with any hive. They are used to upgrade existing user profiles from previous versions of Windows NT to Windows NT 4.0. The user profiles are upgraded the first time the user logs on to Windows NT 4.0.

Four types of files are associated with hives.

File type	Description
No extension	Contains a copy of the hive.
.alt	Contains a backup copy of the critical *HKEY_LOCAL_MACHINE\System* hive. Only the System key has an .alt file.
.log	Contains a transaction log of changes to the keys and value entries in the hive.
.sav	Contains copies of the hive files as they looked at the end of the text mode stage in Setup. There are .sav files for Software, SAM, Security, System, and .Default. A new feature of Windows NT 4.0 backs up the contents of the hives during setup. Setup has two stages: text mode and graphics mode. The hive is copied to a .sav file after the text-mode stage of setup to protect it from errors that might occur if the graphics-mode stage of setup fails. If setup fails during the graphics-mode stage, only the graphics-mode stage is repeated when the computer is restarted; the .sav file is used to rebuild the hives.

Atomicity and Hive Recovery in the Registry

The Registry ensures *atomicity* of individual actions. This means that any change made to a value (to set, delete, or save) either works or does not work: The result will not be a corrupted combination of the old and new configuration even if the system stops unexpectedly because of power failure, hardware failure, or software problems. For example, if an application sets a value for an entry and the system shuts down while this change is being made, when the system restarts, the entry will have either the old value or the new value, but not a meaningless combination of both values. In addition, the size and time data for the key containing the affected entry will be accurate whether the value was changed or not changed.

Flushing Data
In Windows NT, data is written to the Registry only when a *flush* occurs, which happens after changed data ages past a few seconds, or when an application intentionally flushes the data to the hard disk.
The system performs the following flush process for all hives (except for the System hive):

1. All changed data is written to the hive's .log file along with a map of where it is in the hive, and then a flush is performed on the .log file. All changed data has now been written in the .log file.
2. The first sector of the hive file is marked to indicate that the file is in transition.
3. The changed data is written to the hive file.
4. The hive file is marked as completed.

Note

If the system shuts down between steps 2 and 4, when the hive is next loaded at startup (unless it's a profile hive that is loaded at logon), the system sees the mark left in step 2, and proceeds to recover the hive using the changes contained in the .log file. That is, the .log files are not used if the hive is not in transition. If the hive is in transition, it cannot be loaded without the .log file.
A different flush process is used for the System hive because it is an important element during system startup and is used too early during startup to be recovered as described in the previous flush process.
The System.alt file contains a copy of the data contained in the System file. During the flush process, changes are marked, written, and then marked as done. Then the same flush process is followed for the System.alt file. If there is a power failure, hardware failure, or software problems at any point during the process, either the System or System.alt file contains the correct information.
The System.alt file is similar to a .log file except that at load time, rather than having to reapply the logged changes, the system just switches to System.alt. The System.alt file is not needed unless the System hive is in transition.

User Profile Hives

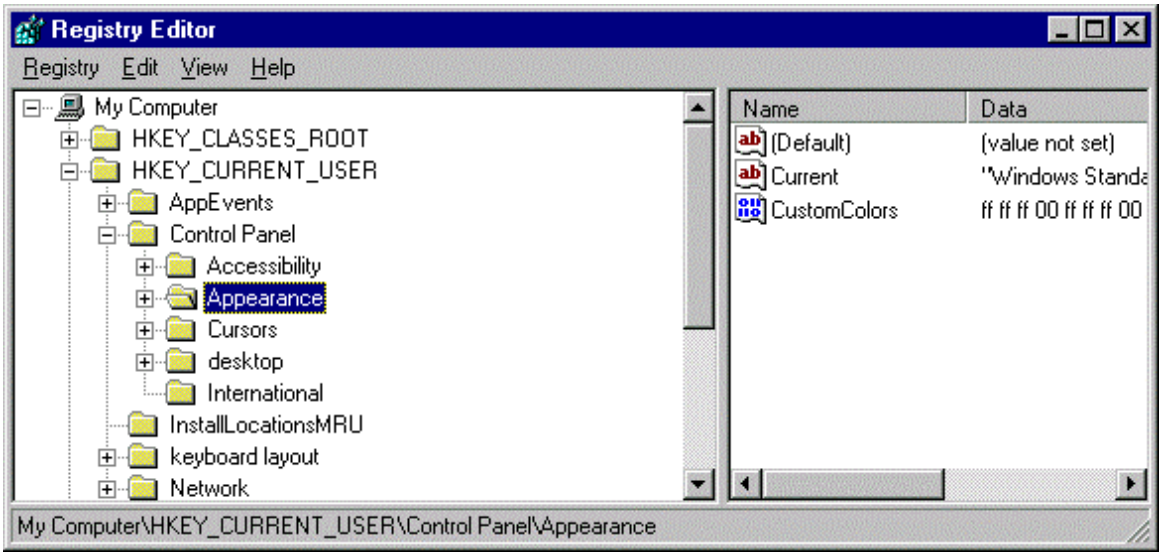
Each time a new user logs on to a computer, a new hive is created for that user with a separate file for the user profile. The system administrator can copy a user profile file to a different directory and view, repair, or copy entries to another computer by using *Registry Editor*.

Registry Size Limit

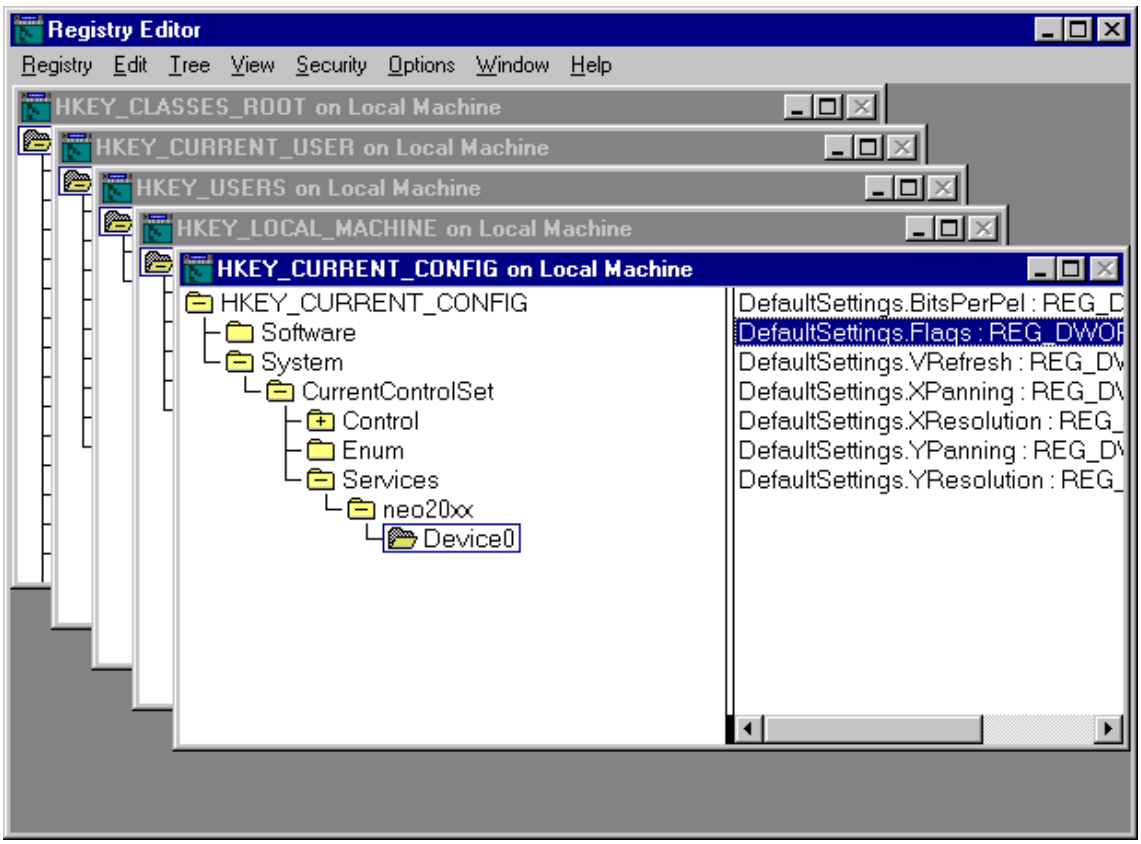
Registry data is stored in the paged pool, an area of physical memory used for system data that can be written to disk when not in use. The **RegistrySizeLimit** value establishes the maximum amount of paged pool space (and disk paging file space) that can be consumed by Registry data from all applications. It is designed to prevent the Registry from consuming space needed by processes.
The **RegistrySizeLimit** value establishes a maximum size for the Registry. It does not allocate space in the paged pool, nor does it assure that the space will be available if needed.
By default, **RegistrySizeLimit** is set to 25 percent of the size of the paged pool. When the paged pool size changes, either because Windows NT adjusts it or because an administrator changes it, the value of **RegistrySizeLimit** changes, too. (Typically, the paged pool is set at 32 MB, so the **RegistrySizeLimit** value is 8 MB.)
The system ensures that the minimum value for **RegistrySizeLimit** is 4 MB, and the maximum is approximately 80 percent of the **PagedPoolSize** value. Thus, the paged pool is limited to a maximum size of 128 MB, and the **RegistrySizeLimit** value cannot exceed 102 MB (80 percent of 128 MB).
To view or change the value of **RegistrySizeLimit**, edit the entry under the following subkey:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
RegistrySizeLimit must have a type of REG_DWORD and a data length of 4 bytes, or it will be ignored. The **RegistrySizeLimit** value is approximate.
To view or change the size of the paged pool, use the **PagedPoolSize** value entry under the following subkey:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Memory Management

The space controlled by **RegistrySizeLimit** includes the hive space, as well as some of the Registry's run-time structures. Other Registry run-time structures are protected by their own size limits or by other means.
To ensure that a user can always start the system and edit the Registry, the Registry is not subject to the value set in **RegistrySizeLimit** until after the first successful loading of a hive (that is, the loading of a user profile). For more details about **RegistrySizeLimit**, see Regentry.hlp, the Registry Help file on the *Windows NT Workstation Resource Kit* CD.

REGEDIT

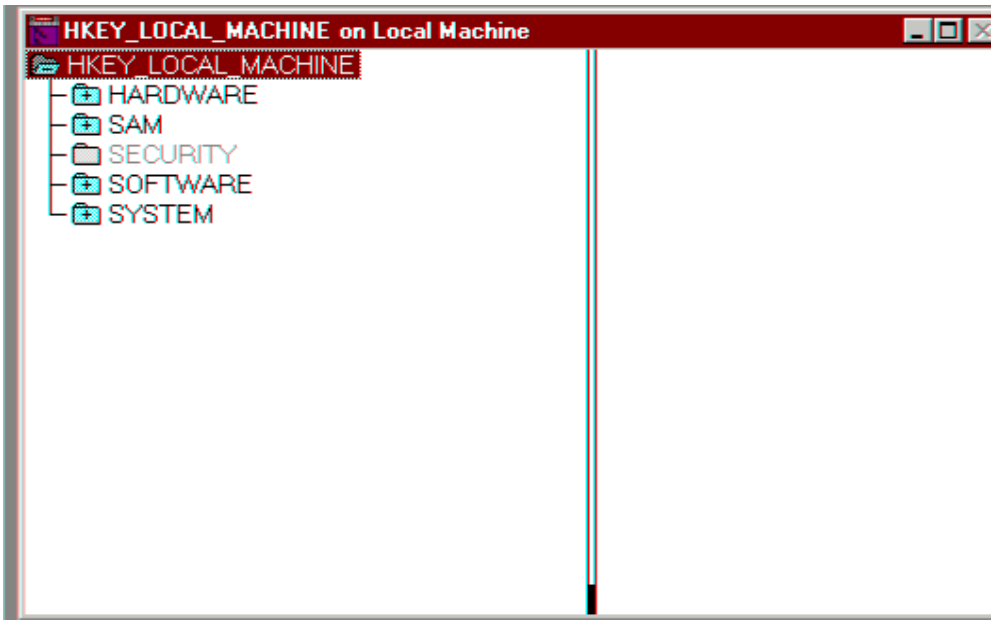


RegEdt32



HKEY_LOCAL_MACHINE

- Machine specific information
- The config.sys and system.ini type data is contained in this hive
- It contains the Security Account Manager (SAM) database
- \HKEY_LOCAL_MACHINE\software\classes is an alias for HKEY_CLASSES_ROOT



By convention vendors write machine specific data at:

```
\HKEY_LOCAL_MACHINE\software\vendor\product\version\category
```

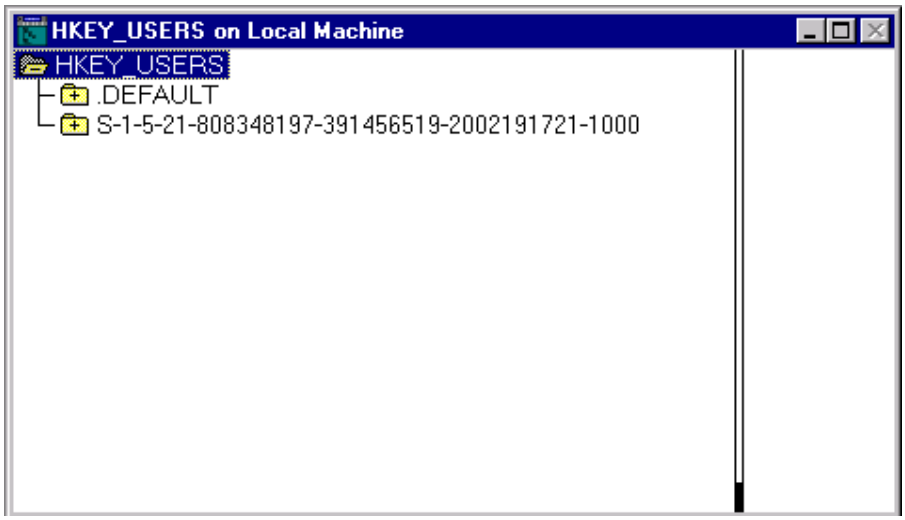
For example: assume that Vendor ABC has a product called Eclipse that writes to the registry. It writes the location of indexes and tables required for the product into the *dataloc* subkey.

```
\HKEY_LOCAL_MACHINE\software\ABC\Eclipse\1.0\dataloc
```

Warning: There is not a clearinghouse for registry names. Pick an unusual or trademarked name for the vendor key in the registry. If not, the registry entries for your product may be inadvertently deleted with the installation of another vendor's program.

HKEY_CURRENT_USERS

- Alias to the current user of HKEY_USERS
- Prevents developers from having to extrapolate the current user from HKEY_USERS
- The most popular registry key
- Modifications in HKEY_CURRENT_USER automatically affects the current user in HKEY_USERS



Every user that can access the machine has a S-X-X-XX-XXXXXXXXXX key. Information specific to this user is written into this key. When a new user is created, the *default* key is replicated and used to create the new S-X-X-XX-XXXXXXXXXX key.

Common data written into the registry is an application's window location and size. This information is written to the registry when the user stops an application. When the program is restarted the information is read to position the window at its last location. Many applications write this information into one key. The key would have four values: x, y, width and height. Any key can have one default value and many named values. Another strategy is to create a structure for the x,y, width, and height. Write the structure as binary into the registry. This requires one value and is more efficient.

The convention for user profiling is:

```
\HKEY_USERS\software\vendor\product\version\category
```

For Vendor ABC and product Ellipse the registry entry might be:

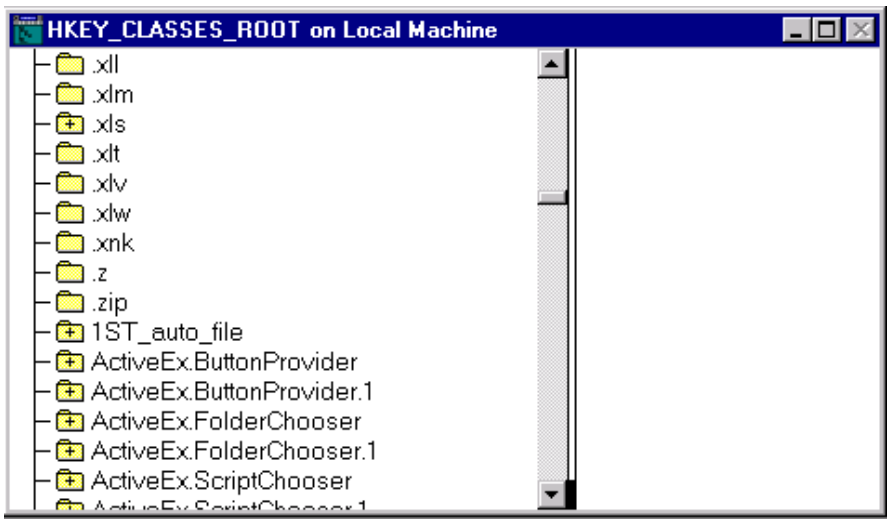
```
\HKEY_USERS\software\ABC\Eclipse\1.0\Position
```

HKEY_CLASSES_ROOT

- Contains file association
- COM information written into this hive
- ActiveX information also in this hive
- Extension of original 16-bit registration database

The original registration database from 16-bit windows contained file association and OLE registration. File association links a file extension with an executable. When a document is started its matching executable can be found through the file extension. OLE registration has been replaced with ActiveX registration.

This hive is discussed further in the last module of this course. That module introduces COM.



Registry Batch Files

- Registry batch files are useful for adding data to the registry
- The file extension should be “reg”
- Can not delete information from the registry
- “Ease of Use” is its greatest asset
- Used by many install programs to update the registry for a new product

Example Registry Batch File

```
REGEDIT

HKEY_CLASSES_ROOT\ComApp.Document = ComApp Document
HKEY_CLASSES_ROOT\ComApp.Document\protocol\StdFileEditing\server = COMAPP.EXE
HKEY_CLASSES_ROOT\ComApp.Document\protocol\StdFileEditing\verb\0 = &Edit
HKEY_CLASSES_ROOT\ComApp.Document\Insertable =
HKEY_CLASSES_ROOT\ComApp.Document\CLSID = {4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}

HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5} = ComApp Document
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\DefaultIcon = COMAPP.EXE,1
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\LocalServer32 = COMAPP.EXE
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\ProgId = ComApp.Document
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\MiscStatus = 32
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\AuxUserType\3 = ComApp
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\AuxUserType\2 = ComApp
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\Insertable =
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\verb\1 = &Open,0,2
HKEY_CLASSES_ROOT\CLSID\{4E1E7C03-0EDA-11D2-8416-9058EBAE45D5}\verb\0 = &Edit,0,2
```

The first word of a registry batch file must be ***regedit***. Afterwards, each line is an entry into the registry. The hive, key, and subkey for each entry is delimited with slashes. Conclude the registry path with an assignment statement. The value will be assigned to the default value of the subkey that concludes the path. These entries will be added to the registry. Existing entries will not be deleted or modified. For additional flexibility use registry script files or registry APIs to modify the registry.

Registry APIs

- RegCreateKeyEx
- RegSetValueEx
- RegQueryValueEx
- RegDeleteKey
- RegCloseKey
- RegOpenKey

There is a family of APIs for modifying the registry. Most of the APIs mentioned in this module have an "Ex" suffix. The older version of these registry APIs are not obsolete. If an application reads and writes text data to the registry, use the older version APIs. They were created specifically to write text data and require fewer parameters.

RegCreateKeyEx

The most important registry API. It creates a registry key. If the key already exists, the API will open and return a handle to the current key.

```
LONG RegCreateKeyEx(HKEY hKey, LPCTSTR lpSubKey, DWORD Reserved, LPTSTR
      lpClass, DWORD dwOptions, REGSAM samDesired, LPSECURITY_ATTRIBUTES
      lpSecurityAttributes, PHKEY phkResult, LPDWORD lpdwDisposition)
```

- hKey: the name of hive. The key will be created within this hive.
- lpSubKey: this is a symbolic string providing the path of the new subkey. Subkeys along that path that do not exist will automatically be created.
- **Reserved**: The third parameter should always be zero.
- **lpClass**: this parameter is usually ignored. Set this parameter to NULL.
- **dwOptions**: indicates the type of registry key: volatile or non-volatile. A volatile key is deleted when the current session is stopped. The key will be recreated when the session is restarted. Volatile keys can not be edited from the registry utilities: Regedit or Regedt32.
- **samDesired**: this parameter contains the access rights of the key: KEY_ALL_ACCESS, KEY_WRITE, KEY_READ, etc.
- **lpSecurityAttributes**: sets security rights of registry object.
- **phkResult**: This is an out parameter. It is the resulting handle for the registry key.

- **lpdwDisposition**: This is also an out parameter. If a new key was not created and a handle to an existing key was provided, this parameter is set to REG_OPENED_EXISTING_KEY. When a new key has been created, this parameter becomes REG_CREATED_NEW_KEY.

- If successful, RegCreateKeyEx returns ERROR_SUCCESS.

RegSetValueEx

This APIs assigns a value to a subkey. Remember a subkey can contain multiple values each with a different symbolic name. The default key is created with a NULL symbolic name.

```
LONG RegSetValueEx(HKEY hKey, LPCTSTR lpValueName, DWORD Reserved,
      DWORD dwType, CONST BYTE *lpData, DWORD cbData)
```

- hKey: Unlike RegCreateKeyEx this parameter *is not a hive*. This is a handle to a previously opened key (see RegCreateKeyEx or RegOpenKeyEx).

- Reserved: this parameter should always be zero.
- dwType: this indicates the date type to be used when writing to the registry. The most popular registry formats are:

  REG_BINARY
  REG_DWORD
  REG_MULTI_SZ
  REG_SZ

- lpData: is a pointer to the buffer. This is the data that will be written into the registry.

- cbData: the number of bytes to write to the registry.

- Like most registry APIs, if successful, this function returns ERROR_SUCCESS.

RegQueryValueEx

This API reads the value or contents of a registry key. The parameters of this API are almost identical to RegSetValueEx.

```
LONG RegQueryValueEx(HKEY hKey, LPTSTR lpValueName, LPDWORD
      lpReserved, LPDWORD lpType, LPBYTE lpData, LPDWORD lpcbData)
```

In this API, lpData is the target buffer. lpcbData is an out parameter. It indicates the number of bytes written into the cache.

RegDeleteKey

This API deletes the key identified by the provided handle. Hives can not be deleted. Depending on security, other keys may be protected from deletion.

```
LONG RegDeleteKey(HKEY hKey, LPCTSTR lpSubKey)
```

The first parameter is a hive. The last parameter is the path to the key to be deleted.

RegCloseKey

The RegCloseKey function releases the handle of a registry key.

```
LONG RegCloseKey( HKEY hKey)
```

RegOpenKeyEx

Obtains a handle to an existing key.

```
LONG RegOpenKeyEx(HKEY hKey, LPCTSTR lpSubKey, DWORD ulOptions, REGSAM
samDesired, PHKEY phkResult)
```

- hKey: hive where key is located
- lpSubKey: this is a symbolic string providing the path of the desired key.
- ulOptions: must be zero
- samDesired: this parameter contains the access rights of the key: KEY_ALL_ACCESS, KEY_WRITE, KEY_READ, etc.
- phkResult: : This is an out parameter. It is the handle to the opened registry key.