Agreement:
==========

The author of this document will not be responsible for any damage and/or
license violation that may occur. The information within this document is
provided "as is" without warranty of any kind...
This information was "collected" during sleepless nights, and is NOT
officially released by Microsoft! It shall give you a peek at the Windows(tm)
internals to give you a chance to recover from corrupted data.

The author has nothing to do with Microsoft, except that he uses their
products...

If you don't agree with this, stop reading this document, and delete it at
once!

History:
========

What is the registry? Where did it came from? Two questions, which I will try to
answer here. The registry is a database (at least microsoft thinks so:)
which contains configuration information about the system.
It mainly is a memory dump which is saved to one or more files on the windows
host drive. It is loaded every system-boot and remains resident until
shutdown. Since parts of it are not used during normal operation it will be
swapped out very soon. The registry appeared with windows 3.?? (sorry, I can't
remember any earlier version :-), where it was used for file associations and
the "OLE" functions (the conection between ole-id's and the applications).
This is a critical information and since the registry has (almost) NO
CHECKSUM information (!), it sometimes gets corrupted. This is the main
reason for this doc.

Using windows 3.x, almost every configuration was done using good old ".INI"-
files, which were readable but slow and limited in size (64k). In windows 95
(and NT), the registry was used instead of these files. So, to edit a
particular setting, you would have to run the application which manages these
settings. :(but what if this app won't start? MS included a tool named
REGEDIT in windows 3.?? and 95, and a REGEDT32 in windows NT. You can use
these apps to edit ALL contents of the registry (in windows NT the registry
supports security, as well as it provides the security for the whole system!)

An application can open a "key", write values (variables) to it and fill them
with data. Each key represents also a value called "default" and can contain
any number of sub-keys. This will form a tree-structure as you can see at
the left half of REGEDIT. (note: REGEDIT from windows 3.?? has to be started
with /V or /Y, I can't remember now)

Where can I find the registry???
================================

That differs for each windows-version:

Version File(s)   Contents
3.1x   REG.DAT   Complete windows 3.?? Registry

```
95        SYSTEM.DAT              System-values (HKEY_LOCAL_MACHINE)
          USER.DAT               User-values (HKEY_USERS)

NT        SYSTEM32\CONFIG\SAM     SAM-part of the registry (=NT Security)
          SYSTEM32\CONFIG\SOFTWARE Software-Specific part
                                   (HKEY_LOCAL_MACHINE\SOFTWARE)
          SYSTEM32\CONFIG\SYSTEM  System-specific part
                                   (HKEY_LOCAL_MACHINE\System)
          PROFILES\%USERNAME%\NTUSER.DAT  User-Specific part
                                   (HKEY_CURRENT_USER\{S-1-xxx...})
          PROFILES\%USERNAME%\NTUSER.MAN  like NTUSER.DAT but a
                                   MANDATORY-profile
```

If you are using a ROAMING-profile with windows NT, NTUSER.xxx can be on
a network-share as well...

Terms
=====

The registry consists of the following elements:

> Hive: strating point of the structure. The name of an hive starts
>   with the "HKEY_"-prefix. Can be seen as a "drive" in a file
>   system.

Hive name	Beschreibung	3.1	95	NT4
HKEY_CLASSES_ROOT	Points to the "class" key in the "HKEY_LOCAL_MACHINE" hive, the only hive in windows 3.??	X	X	X
HKEY_CURRENT_USER	Information and settings valid for the currently logged in user. (Points to the correct key under "HKEY_USERS")		X	X
HKEY_CURRENT_CONFIG	Settings for the currently active hardware profile. Points to "HKEY_LOCAL_MACHINE\ CONTROL\CONTROLSETxxx		X	X
HKEY_USERS	Contains all currently active user settings. Since NT is a single user system, there will be only one key (the S-ID of the active user), and a ".DEFUALT" key (The settings for the CTRL-ALT-DEL environment)		X	X
HKEY_LOCALMACHINE	All local settings		X	X
HKEY_DYN_DATA	As the name says, here you'll find dynamic data (CPU-usage,...)		X	

> Key: A key to the registry can be seen as a directory in a file

```
                        system.
            Value:  can be seen as the registrys "file"
            Data:   is the actual setting, can be seen as the contents of a
                    file


Windows 3.x
===========

This registry is the easiest one. It consists of 3 blocks, which are not
"signed" at all:

Block                      Position        Size
Header                     0               32 Bytes
Navigation-Info            0x00000020      ???
Data-Block                 ???             ???

The "???" marked values can be read from the header.

Header
======

Offset  Size    Description
0x0000  8 Byte  ASCII-Text: "SHCC3.10"
0x0008  D-Word  ?
0x000C  D-Word  ? (always equal the D-Word at 0x0008)
0x0010  D-Word  Number of entrys in the navigation-block
0x0014  D-Word  Offset of the data-block
0x0018  D-Word  Size of the data-block
0x001C  Word    ?
0x001E  Word    ?

Values marked "?" are not important for a read-access, and therefore unknown
to me...

Navigation-Block
================

This is where chaos rules! It consists of two different, 8 byte long blocks:

        * Navigation-Info-Record,
        * Text-Info-Record

The first record in the navigation block is a navigation info record.

Navigation-Info-Record

Offset  Size    Contents
0x00    Word    Next Key (same level)
0x02    Word    First Sub-Key (one level deeper)
0x04    Word    Text-Info-Record Key-Namens
0x06    Word    Text-Info-Record Key-Value (default)

The values are the locical number of the block inside the file:

      offset=blocksize*blocknumber+headersize
```

since 2 of this values are constant:

        offset=8*blocknumber+0x20

Text-Info-Record
================

Offset Size   Contents
0x00   Word   ?
0x02   Word   number of references to this text
0x04   Word   Text-length
0x06   Word   Offset of the text-string inside the data-block

To get the text-offset inside the file you have to add this offset to the
data-offset inside the header.

Data-Block
==========

The data-block only consists of a collection of text-strings. Right in front
of every text is a word which may or may not have a meaning. The offset in
the text-info record points directly to the text, the text-size has to be
defined in the text-info record too.

Windows 95
==========

the Windows95-Registry Files:

inside the windows-directory (default: C:\WINDOWS) are 2 files which are
loaded to form the registry:

        SYSTEM.DAT

and

        USER.DAT

This files are mapped to the following hives:

        HKEY_LOCAL_MACHINE in SYSTEM.DAT

and

        HKEY_USERS in USER.DAT

The file structure:
===================

Both files have the same structure. Each of them consists of 3 blocks where
1 of these blocks can be repeated.

Every block has a 4 byte long signature to help identify its contents.

```
ID      Block-contents          Max. size
CREG    Header                  32 Bytes @ Offset 0
RGKN    Directory information
        (Tree-structure)        ??? @ Offset 32
RGDB    The real data
        (Values and data)       max. 65535 Bytes an Offset ??
```

these blocks are "sticked together" with no space between them, but always
a multiple of 16 in size.

the CREG-Block
==============

```
Offset          Size            Inhalt
0x00000000      D-Word          ASCII-"CREG" = 0x47455243
0x00000008      D-Word          Offset of 1st RGDB-block
0x00000010      D-Word          # of RGDB-blocks
```

all other values are not needed to read the registry...

the RGKN-Block
==============

I assume that RGKN stands for ReGistry-Key-Navigation. This block contains
the information needed to built the tree-structure of the registry. This
block will be larger then 65536 bytes (0xFFFF)!

All offset-values are RELATIVE to the RGKN-block!

```
Offset          Size    Contents
0x00000000      D-Word  ASCII-"RGKN" = 0x4E4B4752
0x00000004      D-Word  Size of the RGKN-block in bytes
0x00000008      D-Word  Rel. Offset of the root-record
0x00000020      ????    Tree-Records (often the 1st Record)
```

the Tree-Record
===============

The tree-record is a "complete" registry-key. It contains the "hash"-info
for the real data stored in this key.

```
Offset  Size    Contents
0x0000  D-Word  Always 0
0x0004  D-Word  Hash of the key-name
0x0008  D-Word  Always -1 (0xFFFFFFFF)
0x000C  D-Word  Offset of the owner (parent)-records
0x0010  D-Word  Offset of the 1st sub-sey record
0x0014  D-Word  Offset of the next record in this level
0x0018  D-Word  ID-number of the real key
```

the 1st entry in a "usual" registry file is a nul-entry with subkeys: the
hive itself. It looks the same like other keys. Even the ID-number can
be any value.

The "hash"-value is a value representing the key's name. Windows will not
search for the name, but for a matching hash-value. if it finds one, it
will compare the actual string info, otherwise continue with the next key.

End of list-pointers are filled with -1 (0xFFFFFFFF)

The ID-field has the following format:

        Bits 31..16:   Number of the corresponding RGDB-blocks
        Bits 15..0:   continuous number inside this RGDB-block.

The hash-method:
================

you are looking for the key: Software\Microsoft

first you take the first part of the string and convert it to upper case

        SOFTWARE

The "\" is used as a seperator only and has no meaning here.
Next you initialize a D-Word with 0 and add all ASCII-values of the string
which are smaller than 0x80 (128) to this D-Word.

        SOFTWARE = 0x0000026B

Now you can start looking for this hash-value in the tree-record.
If you want to modify key names, also modify the hash-values, since they
cannot be found again (although they would be displayed in REGEDIT)

the RGDB-Block
==============

Header:

Offset Size   Contents
0x0000 D-Word ASCII-"RGDB" = 0x42444752
0x0004 D-Word Size of this RGDB-block
0x0020 ????   RGDB Records

RGDB-Record (Key-Information)
============================

Offset Size   Contents
0x0000 D-Word record length in bytes
0x0004 D-Word ID-number
0x0008 D-Word ??? Size ???
0x000C Word   text length of key name
0x000E Word   Number of values inside this key
0x0010 D-Word always 0
0x0014 ????   Key-name
0x???? ????   Values

The first size (record length) can be used to find the next record.
The second size value is only correct if the key has at least one value,
otherwise it is a little lower.

The key-name is not 0-terminated, its length is defined by the key-
text length field. The values are stored as records.

Value-Record
============

Offset   Size Contents
0x0000   D-Word   Type of data
0x0004   D-Word   always 0
0x0008   Word length of value-name
0x000A   Word length of value-data
0x000C   ???? value-name
0x????   ???? data

Data-Types
==========

value   Contents
0x00000001 RegSZ - 0-terminated string (sometimes without the 0!)
0x00000003 RegBin - binary value (a simple data-block)
0x00000004 RegDWord - D-Word (always 4 bytes in size)

Windows NT (Version 4.0)
========================

Whoever thought that the registry of windows 95 and windows nt are similar
will be surprised! They only look much the same, but have completely other
structures!
Since the RGDB-blocks in the windows 95 registry are not larger than
0xFFFF, we can see that it is optimized for a 16-bit OS...
Windows NT stores its registry in a page-oriented format with blocks
of 4kb (4096 = 0x1000 bytes)

The windows NT registry has 2 different blocks, where one can occure many
times...

the "regf"-Block
================

"regf" is obviosly the abbreviation for "Registry file". "regf" is the
signature of the header-block which is always 4kb in size, although only
the first 64 bytes seem to be used and a checksum is calculated over
the first 0x200 bytes only!

Offset   Size   Contents
0x00000000 D-Word   ID: ASCII-"regf" = 0x66676572
0x00000004 D-Word   ????
0x00000008 D-Word   ???? Always the same value as at 0x00000004
0x0000000C Q-Word   last modify date in WinNT date-format
0x00000014 D-Word   1

```
0x00000018  D-Word      3
0x0000001C  D-Word      0
0x00000020  D-Word      1
0x00000024  D-Word      Offset of 1st key record
0x00000028  D-Word      Size of the data-blocks (Filesize-4kb)
0x0000002C  D-Word      1
0x000001FC  D-Word      Sum of all D-Words from 0x00000000 to 0x000001FB
```

I have analyzed more registry files (from multiple machines running
NT 4.0 german version) and could not find an explanation for the values
marked with ???? the rest of the first 4kb page is not important...

the "hbin"-Block
================

I don't know what "hbin" stands for, but this block is always a multiple
of 4kb in size.

Inside these hbin-blocks the different records are placed. The memory-
management looks like a C-compiler heap management to me...

hbin-Header
===========

```
Offset        Size  Contents
0x0000        D-Word      ID: ASCII-"hbin" = 0x6E696268
0x0004        D-Word      Offset from the 1st hbin-Block
0x0008        D-Word      Offset to the next hbin-Block
0x001C        D-Word      Block-size
```

The values in 0x0008 and 0x001C should be the same, so I don't know
if they are correct or swapped...

From offset 0x0020 inside a hbin-block data is stored with the following
format:


```
Offset        Size  Contents
0x0000        D-Word      Data-block size
0x0004        ????  Data
```

If the size field is negative (bit 31 set), the corresponding block
is free and has a size of -blocksize!
The data is stored as one record per block. Block size is a multiple
of 4 and the last block reaches the next hbin-block, leaving no room.

Records in the hbin-blocks
==========================

nk-Record

        The nk-record can be treated as a kombination of tree-record and
        key-record of the win 95 registry.

lf-Record

        The lf-record is the counterpart to the RGKN-record (the hash-function)

vk-Record

        The vk-record consists information to a single value.

sk-Record

        sk (? Security Key ?) is the ACL of the registry.

Value-Lists

        The value-lists contain information about which values are inside a
        sub-key and don't have a header.

Datas

        The datas of the registry are (like the value-list) stored without a
        header.

All offset-values are relative to the first hbin-block and point to the block-
size field of the record-entry. to get the file offset, you have to add
the header size (4kb) and the size field (4 bytes)...

the nk-Record
=============

```
Offset       Size  Contents
0x0000       Word  ID: ASCII-"nk" = 0x6B6E
0x0002       Word  for the root-key: 0x2C, otherwise 0x20
0x0004       Q-Word      write-date/time in windows nt notation
0x0010       D-Word      Offset of Owner/Parent key
0x0014       D-Word      number of sub-Keys
0x001C       D-Word      Offset of the sub-key lf-Records
0x0024       D-Word      number of values
0x0028       D-Word      Offset of the Value-List
0x002C       D-Word      Offset of the sk-Record
0x0030       D-Word      Offset of the Class-Name
0x0044       D-Word      Unused (data-trash)
0x0048       Word  name-length
0x004A       Word  class-name length
0x004C       ????  key-name
```

the Value-List
==============

```
Offset       Size  Contents
0x0000       D-Word      Offset 1st Value
0x0004       D-Word      Offset 2nd Value
0x????       D-Word      Offset nth Value
```

To determine the number of values, you have to look at the
owner-nk-record!

```
Der vk-Record
=============

Offset       Size  Contents
0x0000       Word  ID: ASCII-"vk" = 0x6B76
0x0002       Word  name length
0x0004       D-Word      length of the data
0x0008       D-Word      Offset of Data
0x000C       D-Word      Type of value
0x0010       Word  Flag
0x0012       Word  Unused (data-trash)
0x0014       ????  Name

If bit 0 of the flag-word is set, a name is present, otherwise the
value has no name (=default)
If the data-size is lower 5, the data-offset value is used to store
the data itself!


The data-types
==============

Wert  Beteutung
0x0001       RegSZ:              character string (in UNICODE!)
0x0002       ExpandSZ:   string with "%var%" expanding (UNICODE!)
0x0003       RegBin:            raw-binary value
0x0004       RegDWord:   Dword
0x0007       RegMultiSZ: multiple strings, seperated with 0
                  (UNICODE!)

The "lf"-record
===============

Offset       Size  Contents
0x0000       Word  ID: ASCII-"lf" = 0x666C
0x0002       Word  number of keys
0x0004       ????  Hash-Records

Hash-Record
===========

Offset       Size  Contents
0x0000       D-Word      Offset of corresponding "nk"-Record
0x0004       D-Word      ASCII: the first 4 characters of the key-name,
             padded with 0's. Case sensitiv!

Keep in mind, that the value at 0x0004 is used for checking the
data-consistency! If you change the key-name you have to change the
hash-value too!

The "sk"-block
==============

(due to the complexity of the SAM-info, not clear jet)

Offset       Size  Contents
```

```
0x0000      Word  ID: ASCII-"sk" = 0x6B73
0x0002      Word  Unused
0x0004      D-Word      Offset of previous "sk"-Record
0x0008      D-Word      Offset of next "sk"-Record
0x000C      D-Word      usage-counter
0x0010      D-Word      Size of "sk"-record in bytes
????
????  ????  Security and auditing settings...
????
```

The usage counter counts the number of references to this
"sk"-record. You can use one "sk"-record for the entire registry!

Windows nt date/time format
===========================

The time-format is a 64-bit integer which is incremented every
0,0000001 seconds by 1 (I don't know how accurate it realy is!)
It starts with 0 at the 1st of january 1601 0:00! All values are
stored in GMT time! The time-zone is important to get the real
time!

Common values for win95 and win-nt
==================================

Offset values marking an "end of list", are either 0 or -1 (0xFFFFFFFF).
If a value has no name (length=0, flag(bit 0)=0), it is treated as the
"Default" entry...
If a value has no data (length=0), it is displayed as empty.

simplyfied win-3.?? registry:
=============================

```
+-----------+
| next rec. |---+              +----->     +------------+
| first sub |   |              |      | Usage cnt. |
| name      |   | +-->     +------------+   |      | length      |
| value     |   | | | next rec. |   |      | text        |------->  +-------
+
+-----------+   | | | name rec. |--+ +------------+          | xxxxx |
   +------------+ |   | value rec. |-------> +------------+          +-------
+
   v          | +------------+              | Usage cnt. |
+-----------+       |                        | length      |
| next rec. |       |                        | text        |------->  +-------+
| first sub |------+                        +------------+          | xxxxx |
| name      |                                                       +-------+
| value     |
+-----------+
```
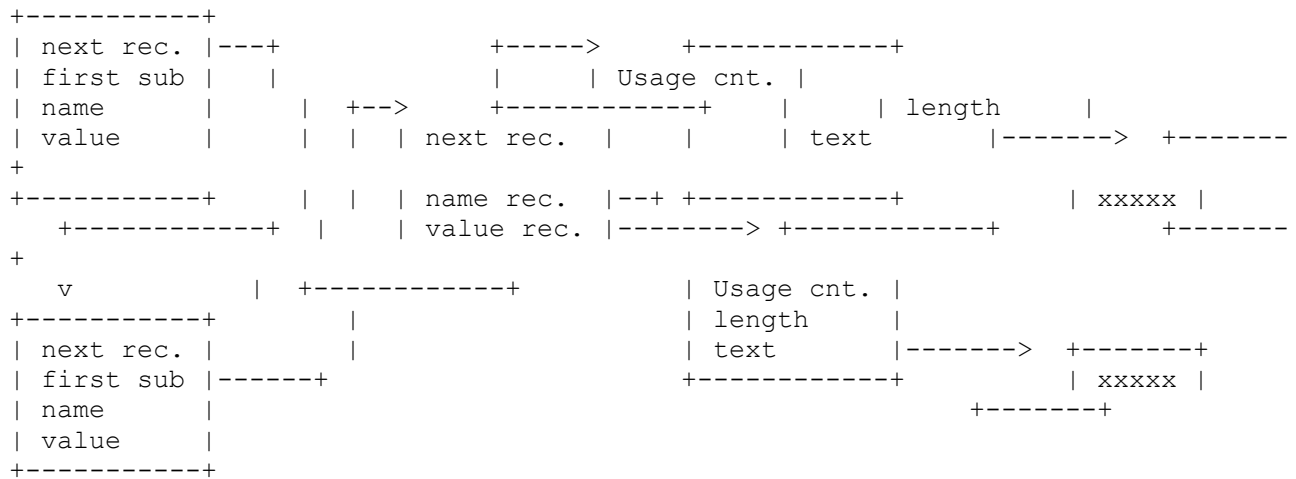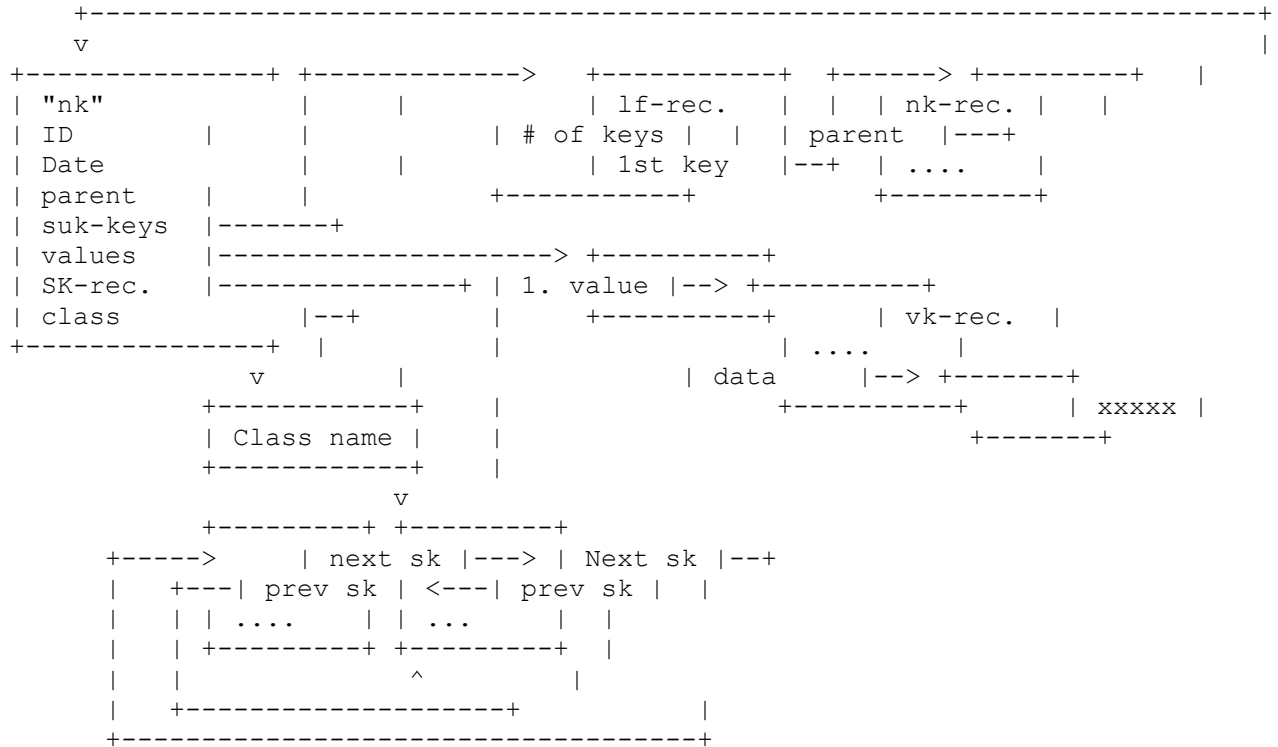
```
Greatly simplyfied structure of the nt-registry:
=================================================


        +--------------------------------------------------------------------+
      v                                                                     |
+---------------+ +------------->   +----------+  +------> +---------+    |
| "nk"          |        |               | lf-rec.  |  | | nk-rec. |    |
| ID            |        |          | # of keys |  | | parent  |---+
| Date          |        |          | 1st key   |--+ | ....    |
| parent        |        |          +----------+        +---------+
| suk-keys  |-------+
| values    |-------------------> +----------+
| SK-rec.   |--------------+ | 1. value |--> +----------+
| class         |--+        |     +----------+      | vk-rec.  |
+---------------+  |        |                  | ....     |
          v      |               | data     |--> +-------+
        +-----------+     |               +----------+     | xxxxx |
        | Class name |     |                         +-------+
        +-----------+     |
                 v        |
        +---------+ +---------+
    +----->     | next sk |---> | Next sk |--+
    |   +---| prev sk | <---| prev sk |  |
    |   | | ....     | | ...      |  |
    |   | +---------+ +---------+  |
    |   |              ^          |
    |   +-------------------+          |
    +----------------------------------+


------------------------------------------------------------------------------

Hope this helps....  (Although it was "fun" for me to uncover this things,
               it took me several sleepless nights ;)

          B.D.
```