

X-Ways Software Technology AG

# *WinHex/ X-Ways Forensics*

*Integrated Computer Forensics Environment.*

*Data Recovery & IT Security Tool.*

*Hexadecimal Editor for Files, Disks & RAM.*

Manual

# Contents

<b>1</b>	<b>Preface .....</b>	<b>1</b>
1.1	About WinHex and X-Ways Forensics .....	1
1.2	Legalities .....	2
1.3	License Types.....	3
1.4	Differences between WinHex and X-Ways Forensics.....	4
<b>2</b>	<b>Technical Background.....</b>	<b>4</b>
2.1	Using a Hex Editor.....	4
2.2	Endian-ness .....	5
2.3	Integer Data Types .....	5
2.4	Floating-Point Data Types .....	6
2.5	Date Types .....	6
2.6	ANSI ASCII/IBM ASCII .....	8
2.7	Checksums .....	8
2.8	Digests.....	9
2.9	Technical Hints .....	9
<b>3</b>	<b>Forensic Features.....</b>	<b>10</b>
3.1	Case Management .....	10
3.2	Evidence Objects.....	11
3.3	Log & Report Feature .....	12
3.4	Internal Viewer.....	13
3.5	Registry Report .....	14
3.6	Drive Contents Table .....	15
3.7	Mode Buttons .....	17
3.8	Logical Search.....	18
3.9	Hash Database.....	19
<b>4</b>	<b>Working with the Hex Editor .....</b>	<b>20</b>
4.1	Start Center .....	20
4.2	Entering Characters.....	20
4.3	Edit Modes .....	20
4.4	Status Bar .....	21
4.5	Scripts.....	22
4.6	WinHex API.....	22
4.7	Disk Editor .....	23
4.8	Directory Browser.....	24
4.9	Directory Browser Context Menu.....	26
4.10	RAM Editor.....	28
4.11	Template Editing .....	29
4.12	Useful Hints .....	30
<b>5</b>	<b>Data Recovery .....</b>	<b>31</b>
5.1	File Recovery with the Directory Browser .....	31
5.2	File Recovery by <i>Name</i> .....	31

5.3	File Recovery by <i>Type</i> .....	32
5.4	File Type Definitions .....	33
5.5	Manual Data Recovery.....	34
<b>6</b>	<b>Menu Reference .....</b>	<b>35</b>
6.1	File Menu .....	35
6.2	Edit Menu.....	36
6.3	Search Menu.....	37
6.4	Position Menu .....	38
6.5	View Menu.....	39
6.6	Tools Menu .....	40
6.7	File Tools .....	42
6.8	Specialist Menu.....	43
6.9	Options Menu.....	45
6.10	Window Menu.....	46
6.11	Help Menu.....	46
6.12	Windows Context Menu .....	47
<b>7</b>	<b>Options.....</b>	<b>47</b>
7.1	General Options .....	47
7.2	Directory Browser Options .....	50
7.3	Undo Options .....	51
7.4	Security & Safety Options .....	51
7.5	Search Options .....	52
7.6	Replace Options .....	53
<b>8</b>	<b>Miscellaneous .....</b>	<b>54</b>
8.1	Block .....	54
8.2	Modify Data .....	54
8.3	Conversions.....	55
8.4	Wiping and Initializing .....	56
8.5	Disk Cloning .....	57
8.6	Images and Backups.....	59
8.7	Backup Manager .....	60
8.8	Position Manager .....	60
8.9	Data Interpreter .....	61
<b>Appendix A:</b>	<b>Template Definition.....</b>	<b>62</b>
1	Header.....	62
2	Body: Variable Declarations .....	63
3	Body: Advanced Commands .....	64
4	Body: Flexible Integer Variables.....	66
<b>Appendix B:</b>	<b>Script Commands .....</b>	<b>67</b>
<b>Appendix C:</b>	<b>Disk Editor Q&amp;A .....</b>	<b>74</b>
<b>Appendix D:</b>	<b>Master Boot Record.....</b>	<b>75</b>
<b>Appendix E:</b>	<b>Surplus Sectors.....</b>	<b>76</b>

# 1 Preface

## 1.1 About WinHex and X-Ways Forensics

Copyright © 1995-2005 Stefan Fleischmann. All rights reserved.

Marketed by

X-Ways Software Technology AG

Carl-Diem-Str. 32

32257 Bünde

Germany

Fax: +49 721-151 322 561

Web: <http://www.x-ways.net>

Product homepage: <http://www.x-ways.net/winhex/>

Ordering: <http://www.x-ways.net/winhex/order.html>

Support forum: <http://www.winhex.net>

E-mail address: [mail@x-ways.com](mailto:mail@x-ways.com)

Registered in Bad Oeynhausen (HRB 7475). CEO: Stefan Fleischmann. Board of directors (chairwoman): Dr. M. Horstmeyer.

X-Ways Software Technology AG is a stock corporation incorporated under the laws of the Federal Republic of Germany. WinHex was first released in 1995. This manual was compiled from the online help of WinHex v12.2, released May 2005. It is available in English, German, French, and Spanish.

French translation by Jérôme Broutin (user interface) and Henri Pouzoullic (program help & manual) in January 2000. Revised and updated since by Bernard Leprêtre. Entire Spanish translation by José María Tagarro Martí. Italian translation by Fabrizio Degni (user interface). Brazilian Portuguese translation by Dr. Heyder Lino Ferreira (user interface).

Cryptography consulting: Alexandre Pukall

The following operating systems are supported:

- Windows 95/98/Me

- Windows 2000

- Windows XP

Professional users around the world include...

U.S. and German federal law enforcement agencies, ministries such as the Australian Department of Defence, U.S. national institutes (e.g. the Oak Ridge National Laboratory in Tennessee), the Technical University of Vienna, the Technical University of Munich (Institute of Computer Science), the German Aerospace Center, the German federal bureau of aviation accident investigation, Microsoft Corp., Hewlett Packard, Toshiba Europe, Siemens AG, Siemens Business Services, Siemens VDO AG, Infineon Technologies Flash GmbH & Co. KG, Ontrack Data International Inc., KPMG Forensic, Ernst & Young, Ericsson, National Semiconductor, Lockheed Martin, BAE Systems, TDK Corporation, Seoul Mobile Telecom, Visa International, DePfa Deutsche Pfandbriefbank AG, Analytik Jena AG, and many other companies and scientific institutes. Please visit the web site to find out how to order the full version!

## 1.2 Legalities

Copyright © 1995-2005 Stefan Fleischmann. All rights reserved. Marketed by X-Ways Software Technology AG. No part of this publication may be reproduced, or stored in a database or retrieval system without the prior permission of the author. Any brand names and trademarks mentioned in the program or in this manual are properties of their respective holders and are generally protected by laws.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. However, the author neither offers any warranties or representations nor does he accept any liability with respect to the program or the manual.

### License Agreement

Your use, distribution, or installation of a software product published by X-Ways Software Technology AG indicates your acceptance of this license agreement. If you do not agree to any of the terms, then do not install, distribute or use the product.

A trial version may be only used for evaluation purposes. Purchasing one license authorizes you to install one copy of the full version of the software on a single machine. Additional licenses authorize you to install additional copies of the full version at the same time.

This software, and all accompanying files, data, and materials, are distributed “as is” and with no warranties of any kind, whether express or implied, to the maximum extent permitted by applicable law. The user must assume the entire risk of using the program, knowing in particular that this software is not designed or intended for use in hazardous environments requiring fail-safe performance, where its failure to perform, misuse or inability to use adequately can reasonably be expected to lead to death, personal injury, or severe physical or environmental damage. In no event shall X-Ways Software Technology AG, or its officers, directors, employees, affiliates, contractors, or subsidiaries be liable for any direct, indirect, incidental, consequential, or punitive damages whatsoever arising out of the use or inability to use the software, to the maximum extent permitted by applicable law. Any liability will be limited exclusively to refund of purchase price by X-Ways Software Technology AG.

You may not rent, lease, modify, translate, reverse-engineer, decompile or disassemble the software or create derivative works based on it without prior explicit permission. All rights of any kind in the software product which are not expressly granted in this license agreement are entirely and exclusively reserved to and by X-Ways Software Technology AG.

No component of the software (except the WinHex API) must be accessed by other applications or processes.

### Acknowledgements

The algorithms “Pukall Cipher 1” (PC 1) and “Pukall Stream Cipher Hash Function” are copyright by Alexandre Pukall. Source code available from <http://www.multimania.com/pc1>,

<http://www.multimania.com/cuisinons/progs/>, and <http://www.freecode.com>.

The MD5 message digest is copyright by RSA Data Security Inc.

The “zlib” compression library is copyright by Jean-loup Gailly and Mark Adler. Homepage: <ftp://ftp.cdrom.com/pub/infozip/zlib/zlib.html>

X-Ways Forensics contains software by Igor Pavlov, [www.7-zip.com](http://www.7-zip.com).

Outside In® Viewer Technology © 1991-2005 Stellant Chicago, Inc. All rights reserved.

Parts of the registry viewer are copyright by Markus Stephany, [dumphive\\_\[at\]mirkes\\_\[dot\]de](mailto:dumphive_[at]mirkes_[dot]de). All rights reserved. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Markus Stephany nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

## 1.3 License Types

To use WinHex as a full version, you need a base license for personal, professional, or specialist use. If you are going to use WinHex on multiple machines, you will also need additional licenses. The full version will save files larger than 200 KB, write disk sectors, edit virtual memory and show no evaluation version reminders. It will reveal its license status on start-up and in the About box.

- Personal licenses are available at a reduced price for non-commercial purposes only, in a non-business, non-institutional, and non-government environment.
- Professional licenses allow usage of the software in any environment (at home, in a company, in an organization, or in public administration). Professional licenses provide the ability to execute scripts and to use the WinHex API.
- Specialist licenses in addition to this allow to use the Specialist Tools menu section, to fully interpret Ext2, Ext3, CDFS/ISO9660, and UDF media, and enable support for RAID and dynamic disks. Particularly useful for IT security specialists. Plus X-Ways Replica 1.3, a DOS-based forensically sound disk cloning and imaging software is included.
- Forensic licenses in addition to this allow to use the powerful case managing and report generating capabilities of WinHex, the internal viewer and additionally a powerful viewer component, the gallery view and advanced features of the drive contents table, plus ReiserFS support. Furthermore, they allow to read and write evidence files (.e01). Particularly useful for

computer forensic examiners. The forensic edition of WinHex is called X-Ways Forensics. Also includes X-Ways Replica 2.33, with advanced disk cloning and imaging capabilities.

Please see the Order page of the web site (<http://www.x-ways.net/winhex/order.html>) on how to order your licenses.

## 1.4 Differences between WinHex and X-Ways Forensics

WinHex and X-Ways Forensics are identical (and work with the same forensic license), notably except for the following:

- WinHex (winhex.exe) always identifies itself as WinHex in the user interface, X-Ways Forensics (xwforensics.exe) as X-Ways Forensics. The program help and the manual, however, statically refer to “WinHex” in most cases.
- X-Ways Forensics only allows to open those files in an editable mode that have been created in the current case's folder or its subfolders or in the general folder for temporary files, for decoding/decryption/conversion purposes, etc. All other files, image files, virtual memory, and disks in general, are strictly opened in view mode (read-only), to enforce forensic procedures, where no evidence must be altered in the slightest. Similarly, only the current case's folders and the general folder for temporary files are considered legitimate output folders where files can be saved. This strict write protection of X-Ways Forensics ensures that no original evidence can possibly be altered accidentally, which is a crucial aspect in court proceedings.
- Certain files (see <http://www.x-ways.net/winhex/setup.html> for details) are not part of the WinHex download, but owners of forensic licenses can copy them from X-Ways Forensics to enable the full feature set known from X-Ways Forensics in WinHex as well. Using WinHex instead of X-Ways Forensics can be desirable when not bound by strict forensic procedures and when in need to work more aggressively on files, disks, or images, e.g. repairing boot sectors etc., or when working with multiple clones where one clone is declared a working copy and cleared for write access.

## 2 Technical Background

### 2.1 Using a Hex Editor

A hex editor is capable of completely displaying the contents of each file type. Unlike a text editor, a hex editor even displays control codes (e.g. linefeed and carriage-return characters) and executable code, using a two-digit number based on the hexadecimal system.

Consider one byte to be a sequence of 8 bits. Each bit is either 0 or 1, it assumes one of two possible states. Therefore one byte can have one of  $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^8 = 256$  different values.

Since 256 is the square of 16, a byte value can be defined by a two-digit number based on the hexadecimal system, where each digit represents a tetrad or nibble of a byte, i.e. 4 bits. The sixteen digits used in the hexadecimal system are 0-9, A-F.

You can change the value of a byte by changing these digits in the hexadecimal mode. It is also possible to enter the character that is assigned to a certain byte value by a character set (cf. Entering Characters). All kinds of characters are allowed (e.g. letters and punctuation marks). Example: A byte whose decimal value is 65 is displayed as 41 in hexadecimal notation ( $4 \cdot 16 + 1 = 65$ ) and as the letter A in text mode. The ASCII character set defines the capital letter A to have the decimal value of 65.

When editing files of a certain type (for instance executable files), it is essential not to change the file *size*. Moving the addresses of executable code and included data results in severely damaging such files. Please note that changing the contents of a file generally may be the reason for the corresponding application to behave anomalously. It is quite safe to edit text passages in a file. At any rate, it is recommendable to create backup files before editing.

The command “Combined Search” was especially designed for editing files created by computer games to save the game state. If you know the value of a variable in two of such files, you can find out the offset, i.e. the position, at which this data is saved. Example: If two files hold the information that you have 5 resp. 7 points/lives/..., search simultaneously for the hex value 05 in the first and 07 in the second file.

## 2.2 Endian-ness

Microprocessors differ in the position of the least significant byte: Intel®, MIPS®, National Semiconductor, and VAX processors have the least significant byte first. A multi-byte value is stored in memory from the lowest byte (the “little end”) to the highest byte. For example, the hexadecimal number 12345678 is stored as 78 56 34 12. This is called the *little-endian* format.

Motorola and Sparc processors have the least significant byte last. A multi-byte value is stored in memory from the highest byte (the “big end”) to the lowest byte. For example, the hexadecimal number 12345678 is stored as 12 34 56 78. This is called the *big-endian* format.

## 2.3 Integer Data Types

Format/Type	Range	Example
signed 8 bit	-128...127	FF = -1
unsigned 8 bit	0...255	FF = 255
signed 16 bit	-32,768...32,767	00 80 = -32,768
unsigned 16 bit	0...65,535	00 80 = 32,768
signed 24 bit	-8,388,608...8,388,607	00 00 80 = -8,388,608
unsigned 24 bit	0...16,777,215	00 00 80 = 8,388,608
signed 32 bit	-2,147,483,648...2,147,483,647	00 00 00 80 = -2,147,483,648
unsigned 32 bit	0...4,294,967,295	00 00 00 80 = 2,147,483,648
signed 64 bit	$-2^{63}$ ( $\approx -9 \cdot 10^{18}$ )... $2^{63}-1$ ( $\approx 9 \cdot 10^{18}$ )	00 00 00 00 00 00 00 80 = $-2^{63}$

Unless stated otherwise, multi-byte numbers are stored in little-endian format, meaning that the first byte of a number is the least significant and the last byte is the most significant. This is the common format for computers running Microsoft Windows. Following the little-endian paradigm, the hexadecimal values 10 27 can be interpreted as the hexadecimal number 2710 (decimal: 10,000).

The Data Interpreter is capable of interpreting data as all of the aforementioned integer types.

## 2.4 Floating-Point Data Types

Type	Range	Precision [Digits]	Bytes
Float (Single)	$\pm 1.5^{-45}$ ... $3.4^{38}$	7-8	4
Real	$\pm 2.9^{-39}$ ... $1.7^{38}$	11-12	6
Double (Double)	$\pm 5.0^{-324}$ ... $1.7^{308}$	15-16	8
Long Double (Extended)	$\pm 3.4^{-4932}$ ... $1.1^{4932}$	19-20	10

The type names originate from the C programming language. The corresponding Pascal names are specified in brackets. The Real type exists only in Pascal. The Data Interpreter is capable of translating hex values in an editor window into floating-point numbers of all four types and vice-versa.

In the computer, a floating-point number  $F$  is represented by a mantissa  $M$  and an exponent  $E$ , where  $M \times 2^E = F$ . Both  $M$  and  $E$  are signed integer values themselves. The four data types differ in their value ranges (i.e. the number of bits reserved for the exponent) and in their precision (i.e. the number of bits reserved for the mantissa).

On Intel®-based systems, calculations upon floating-point numbers are carried out by a math coprocessor while the main processor waits. The Intel® 80x87 uses 80-bit precision for calculations, whereas RISC processors often use 64-bit precision.

## 2.5 Date Types

The following date formats are supported by the Data Interpreter:

- **MS-DOS Date & Time (4 bytes)**

The lower word determines the time, the upper word the date. Used by several DOS function calls, by the FAT file systems and many system utilities such as file archivers.

Bits	Contents
0-4	Second divided by 2
5-10	Minute (0-59)
11-15	Hour (0-23 on a 24-hour clock)
16-20	Day of the month (1-31)
21-24	Month (1 = January, 2 = February, etc.)
25-31	Year offset from 1980

- **Win32 FILETIME (8 bytes)**

The FILETIME structure is a 64-bit integer value representing the number of 100-nanosecond intervals since January 1, 1601. Used by the Win32 API.

- **OLE 2.0 Date & Time (8 bytes)**

A floating-point value (more exactly: a double) whose integral part determines the number of days passed since December 30, 1899. The fractional part is interpreted as the day time (e.g. 1/4 = 6:00 a.m.). This is the OLE 2.0 standard date type, e.g. it is used by MS Excel.

- **ANSI SQL Date & Time (8 bytes)**

Two consecutive 32-bit integer values. The first one determines the number of days since November 17, 1858. The second one is the number of 100-microsecond intervals since midnight. This is the ANSI SQL standard and used in many databases (e.g. InterBase 6.0).

- **UNIX, C, FORTRAN Date & Time (4 bytes)**

A 32-bit integer value that determines the number of seconds since January 1, 1970. This data type was used in UNIX, by C and C++ (“time\_t”), and by FORTRAN programs since the 80's. Sporadically defined as the number of *minutes* since January 1, 1970. The Data Interpreter options let you switch between both sub-types.

- **Java Date & Time (8 bytes)**

A 64-bit integer value that specifies the number of milliseconds since January 1, 1970. Principally stored in big endian, which is the typical byte order in Java.

## 2.6 ANSI ASCII/IBM ASCII

ANSI ASCII is the character set used in Windows applications. It is standardized by the American National Standards Institute. MS-DOS uses the IBM ASCII character set (also called OEM character set). These character sets differ in the second half, containing characters with a ASCII values greater than 127.

It is reasonable to switch the menu option “Use ANSI ASCII” off when viewing or editing files originating from a DOS program.

Use the “Convert” command of the Edit menu to convert text files from one character set into the other.

The first 32 ASCII values do not define printable characters, but control codes:

Hex	Control Code	Hex	Control Code
00	Null	10	Data Link Escape
01	Start of Header	11	Device Control 1
02	Start of Text	12	Device Control 2
03	End of Text	13	Device Control 3
04	End of Transmission	14	Device Control 4
05	Enquiry	15	Negative Acknowledge
06	Acknowledge	16	Synchronous Idle
07	Bell	17	End of Transmission Block
08	Backspace	18	Cancel
09	Horizontal Tab	19	End of Medium
0A	Line Feed	1A	Substitute
0B	Vertical Tab	1B	Escape
0C	Form Feed	1C	File Separator
0D	Carriage Return	1D	Group Separator
0E	Shift Out	1E	Record Separator
0F	Shift In	1F	Unit Separator

## 2.7 Checksums

A checksum is a characteristic number used for verification of data authenticity. Two files with equal checksums are highly likely to be equal themselves (byte by byte). Calculating and comparing the checksums of a file *before* and *after* a possibly inaccurate transmission may reveal transmission errors. An unaffected checksum indicates that the files are (in all likelihood) still identical. However, a file can be manipulated on purpose in such a way that its checksum remains unaffected. Digests are used instead of checksums in such a case, where malicious (i.e. not mere random) modifications to the original data are to be detected.

In WinHex, checksums are calculated when opening (optional, cf. Security Options) or analyzing

(cf. Tools menu) a file. After modifying files, checksums can be re-calculated by pressing **ALT+F2**.

The standard checksum is simply the sum of all bytes in a file, calculated on an 8-bit, 16-bit, 32-bit, or 64-bit accumulator. The CRC (cyclic redundancy code) is based on more sophisticated algorithms, which are safer.

Example: If a transmission alters two bytes of a file in such a way that the modifications are countervailing (for instance byte one +1, byte two -1), the standard checksum remains unaffected, whereas the CRC changes.

## 2.8 Digests

A so-called digest is, similar to a checksum, a characteristic number used for verification of data authenticity. But digests are more than that: digests are *strong one-way hash codes*.

It is computationally feasible to manipulate any data in such a way that its checksum remains unaffected. Verifying the checksum in such a case would lead to the assumption that the data has not been changed, although it has. Therefore, digests are used instead of checksums if malicious (i.e. not mere random) modifications to the original data are to be detected. It is computationally infeasible to find any data that corresponds to a given digest. It is even computationally infeasible to find two pieces of data that correspond to the same digest.

Of course, random modifications, e.g. caused by an inaccurate transmission, can also be detected when using digests, but checksums are sufficient and serve better for this purpose, because they can be calculated much faster.

WinHex incorporates the widely known 128-bit MD5 message digest, SHA-1, SHA-256, and PSCHF ("Pukall Stream Cipher" hash function).

## 2.9 Technical Hints

- Technical specifications

Maximum number of windows:.....	1000 (WinNT/2000), 500 (Win9x/Me)
Maximum disk & file size: .....	≈2000 GB
Maximum number of parallel program instances: .....	99
Maximum number of positions:.....	limited by RAM only
Maximum number of reversible keyboard inputs: .....	65535
Encryption depth:.....	128 bit
Digest length in backups:.....	128/256 bit
Character sets supported: .....	ANSI ASCII, IBM ASCII, EBCDIC, Unicode (limited)
Offset presentation:.....	hexadecimal/decimal

- In most cases, the progress display shows the completed percentage of an operation. However, during search and replace operations it indicates the relative position in the current file or disk.
- The user interface looks best if *no* extra large font is used in your Windows system.
- WinHex expects your computer to be running in little-endian mode.
- Keys you specify for encryption/decryption are not saved on the hard disk. Provided that the corresponding security option is enabled, the key is stored in an encrypted state within the RAM, as long as WinHex is running.
- Search and replace operations generally run fastest with case sensitivity switched on and without wildcards enabled.
- When searching with the option “count occurrences” activated or when replacing without prompting, for a search algorithm there are generally two ways to behave when an occurrence has been found, which in some cases may have different results. This is explained by the following example:

The letters *ana* are searched in the word “banana”. The first occurrence has already been found at the second character.

1<sup>st</sup> alternative: The algorithm continues the search at the third character. So *ana* is found again at the fourth character.

2<sup>nd</sup> alternative: The three letters *ana* found in the word “banana” are skipped. The remaining letters *na* do not contain *ana* any more.

WinHex is programmed in the second manner, because this delivers the more reasonable results when counting or replacing occurrences. However, if you continue a search using the **F3** key or you choose the replace option “prompt when found”, the algorithm follows the first paradigm.

## 3 Forensic Features

### 3.1 Case Management

The integrated computer forensics environment in WinHex can be used with a forensic license of WinHex only. It offers complete case management, automated log and report file generation, and various additional features such as gallery view, category view, filename/file type mismatch detection, HPA detection, and skin color detection in pictures.

When starting up WinHex for the first time, you are asked whether to run it with the forensic interface. This means the “Case Data” window is displayed, WinHex is run in read-only mode, and you are asked to make sure the folders for temporary files and for case data are set correctly,

in order to prevent WinHex from writing files to the wrong drive.

In order to work with a case, make sure the “Case Data” window is visible on the left of the main window. If not, enable View | Show | Case Data.

From the File menu, you may create a new case (start from scratch), open an existing case, close the active case, save the active case, back up the case in a ZIP archive (including all associated files such as contents tables, but without image files and without recovered files), or automatically generate a case report. A case is stored in a .xfc file (xfc stands for X-Ways Forensics Case) and in a subfolder of the same name, just without the .xfc extension. This subfolder and its child folders are created automatically when the case is created. You may select the base folder for your cases in General Options. It is not necessary to explicitly save a case, unless you need to be sure it is saved at a given time. A case is saved automatically at latest when you close it or exit the program.

In the case properties window, you may name a case according to your own conventions (e.g. title or number). The date and time you create a case is recorded and displayed. The internal case filename is displayed as well. You may enter a description of the case (of arbitrary length) and the examiner's name, the examiner's organization's name and address. You may enable or disable the automated log feature for the whole case. Optionally, the evidence object subfolders in the case folder are always suggested as default output folders for files recovered/copied off a file system. You may wish to disable that feature if your preference is to copy files from various evidence objects into the same output folder.

The most powerful concept in X-Ways Forensics, that allows to systematically review files on computer media, is the so-called *drive contents table*. It is possible to create such contents tables for all evidence objects of a case in one step, and it is even possible to unify contents tables from various evidence objects to a single, global, case-wide contents table, a flat overview of all existing and deleted files from all subdirectories, from all partitions, from all media and image files that belong to a case, sortable by various criteria and searchable with a logical simultaneous search algorithm.

In order to completely *delete* a case, you need to delete its .xfc file and the corresponding directory with the same name and all its subdirectories.

## 3.2 Evidence Objects

You may add any currently attached computer medium (such as hard disk, memory card, USB stick, CD-ROM, DVD, ...), any image file, or ordinary file to the active case. It will then be permanently associated with this case (unless you remove it from the case later), displayed in the tree-like case structure, and designated as an *evidence object* or *source of evidence*. A subfolder is created in the case folder for each evidence object, where by default files will be saved that you recover from that evidence object, so it will always be obvious from which object exactly (and from which case) recovered files originate.

In the evidence object properties window, you may enter a title or number for that evidence object according to your own conventions. The date and time it was associated with the active case is recorded and displayed. The internal designation of the evidence object is displayed as well as its original size in bytes. You may enter comments of arbitrary length that apply to the evidence objects, and a technical description of it is added by WinHex automatically (as known from the Medial Details Report command in the Specialist menu). You may have WinHex calculate a hash (checksum or digest) on the evidence object and verify it later, so that you can be sure that data authenticity has not been compromised in between. Hashes stored in evidence files are imported automatically when added to a case. You may disable the automated log feature for a specific evidence object if the log feature is enabled for the case as a whole.

Ways how to add files or media to a case: The "Add" commands in the case data window's File menu. The "Add" command in the edit window's tab's context menu. The "Add" command of a directory browser's item's context menu.

### **Sub-elements**

All evidence objects in turn have further elements associated with them. There is a list of annotations/bookmarks, initially blank, where you may specially mark and comment an unlimited number of positions of interest, specifically for the evidence object. See Position Manager. Up to 32 contents tables can be associated with an evidence object. They are created by the Specialist menu commands Create Drive Contents Table and Create Directory Contents Table. They show the files of a volume including those in subdirectories in a single flat view, optionally grouped by file categories. From an evidence object's context menu you can also create special report contents tables, which are initially blank and to which you can add notable files via the directory browser's context menu (Position section). There is an item in the report table's context menu that allows you to toggle inclusion in the report.

Finally, if you extract free space, slack space, or text from a volume (using Specialist menu commands), the resulting files will show up in the case tree below the corresponding evidence object as well.

## **3.3 Log & Report Feature**

### **Logs**

When enabled in the case and the evidence properties window, WinHex obstinately logs all activities performed when the case is open. That allows you to easily track, reproduce, and document the steps you have followed to reach a certain result, for your own information and for the court room.

The following is recorded:

- when you select a menu item, the command title (or at least an ID), and the name of the active edit window, if not an evidence object, preceded by the keyword "Menu",
- when a message box is displayed, the message text and what button you pressed (OK,

- Yes, No, or Cancel), preceded by the keyword "MsgBox",
- when a small progress indicator window is displayed, its title (like "Recovering files...") and whether the operation was completed or aborted, preceded by the keyword "Operation",
- a screenshot of each displayed dialog window with all selected options, e.g. for a complex operation that follows, preceded by the window's title,
- original source path of each recovered file,
- destination path of each recovered file when recovered with the directory browser or the Access button menu,
- the extensive log produced by Clone Disk and File Recovery by Type,
- your own entries (free text) that you add with the Add Log Entry command, either to the case as a whole or to a certain evidence object.

All activities are logged with their exact date and time, internally in FILETIME format with 100-nanosecond interval precision. Logs are by default associated with the case as a whole. However, logs of activities that apply to a certain evidence object are directly associated with that evidence object. This determines where they appear in a report. Screenshots are saved as .png files in the "log" subfolder of a case folder. They can optionally be converted to black & white images, which allows to print them in a cost-effective way along with the report.

## Reports

You may create a report from the File menu of the Case Data window. The report is saved as an HTML file and can thus be displayed and opened in a variety of applications. For example, you may view it in your favorite Internet browser and open and further process it in MS Word.

The report starts with the general case title and details, followed by a list of hyperlinks to the individual evidence object sections. For each evidence object, the report specifies its title, details, and description, your comments, your annotations, and the evidence object related log. The report ends with the general log.

## 3.4 Internal Viewer

The internal viewer can be invoked with the "View" command in the Tools menu and in the directory browser's context menu. It shows picture files of various file formats (see Gallery View) and the internal structure of Windows registry files. If you try to view a file that is not supported by the internal viewer, the first defined external viewer is invoked instead.

There is an additional viewer component that integrates seamlessly and allows to conveniently view more than 200 (!) file formats (such as MS Word/Excel/PowerPoint/Access/Works/Outlook, HTML, PDF, CorelDraw, StarOffice, OpenOffice, ...) directly in WinHex and X-Ways Forensics. This component is provided to all owners of forensic licenses issued for v12.05 and later. [More information online.](#)

## Registry Viewer

MS Windows maintains an internal database called registry which contains all important settings for the local system and installed software in a tree-like structure. The data is persistently stored in files called registry hives. You can open and view hives without importing them into your own active registry. Supported formats are Win9x/Me/NT/2k/XP hives. Win9x and WinMe hives are located in the files "user.dat", "system.dat", and their backups. WinNT, Win2k, and WinXP hives are located in the file "ntuser.dat" in a user profile and in the directory \system32\config.

Up to 16 hives can be opened in the registry viewer at the same time. Since Win9x/Me and WinNT/2k/XP registries have different internal structures, their hives cannot be opened and viewed at the same time. If a different format is encountered, only the hive that was opened last will be displayed in the window.

With a right-click a pop-up menu can be opened anywhere in the window, which lets you invoke the commands "Search" and "Continue Search". Clicking "Search" invokes a dialog that lets you specify a search expression and where you want to search. You can browse either keys or names or values or all of them. The search starts at the topmost root and spans all opened hives. "Continue Search" finds the next match after at least one match has been found. (The currently selected element is not relevant for where the search continues). In the right-hand window the pop-up menu also contains the command "Copy" which lets you copy the value of the selected element to the clipboard.

## 3.5 Registry Report

From within the registry viewer, WinHex can create a HTML-based report, listing values of possibly relevant registry keys, when you invoke the command "Create Registry Report" in the pop-up menu. The registry keys that are to be reported in all open hives are specified in a text file called "Reg Report Keys.txt". The registry files you view must have their original names, or else the report may fail. You may edit the list of registry keys in this files to tailor the report to your own needs.

### Format of entries in "Reg Report Keys.txt"

*(operating system shortcut) (tabstop) (registry path) (tabstop) (description) (linefeed)*

*operating system shortcuts:*

9x: Windows 9x/Me

NT: Windows NT/2000/XP

*registry path:*

Full path of registry keys

HKLM: HKEY\_LOCAL\_MACHINE

HKCU: HKEY\_CURRENT\_USER

If an asterisk ("\*") is provided as the last key, all keys on the same level and deeper and their values will be included in the report.

example:

NT HKLM\Software\Microsoft\Windows\CurrentVersion\\* report whole Windows branch

If you wish to report a particular value that exists in all subkeys of a certain key, you can as well write an "\*" for all subkeys and include the value after that.

example:

9x HKCU\Identities\\*\UserID UserID value of every identity

The generated report contains the registry path with its timestamp (Windows NT/2000/XP only), the filename of the registry hive that the key was found in, the description that was provided in the "Reg Report Keys" file, and the value.

## 3.6 Drive Contents Table

Requires a specialist or forensic license. Creates a disk “catalog” of existing and non-existent (deleted or orphaned) files and directories on a logical drive or partition, with user-configurable information such as attributes, all available date & time stamps, size, allocated clusters, hash (checksum or digest), alternate data streams (which contain hidden data, on NTFS drives only), etc. Extremely useful to systematically examine the contents of a disk. Allows to limit the search for files whose names correspond to certain file masks (e.g. \*.jpg;\*.gif). Up to two asterisks allowed if they are located at the beginning and the end of a filename mask. You may *exclude* files using filename patterns that begin with a colon (:). For example, you may include all files except NTFS system files by providing the following patterns: \*;:\$\*. All files with names that start with the letter "A", but do not contain the word "garden": A\*:\*garden\*.

FAT: The “**particularly thorough file system search**” option searches for orphaned subdirectories (subdirectories that are no longer referenced by any other directory).

NTFS: The “particularly thorough file system search” option searches for file records in sectors that do not belong to the current MFT. Such file records can be found e.g. after a partition has been recreated, reformatted, moved, resized, or defragmented.

The “**File header search in unallocated clusters**” option causes files to be included in the list that can still be found in free drive space based on their file header signature. You are asked to select certain file types for detection, specify output filenames etc. as known from File Recovery by Type. If these files are also found based on file system data structures, they will be actually listed twice, once with their correct name and size, and once with a generic filename and a size as detected by the File Recovery by Type mechanism. However, if they are no longer referenced by file system data structures, this option is the only way to get them listed.

The result can be **output** to a tab-delimited **text file**, which is useful for import and further processed in databases or MS Excel. If MS Excel cannot be found on your computer or if you hold the Shift key on your keyboard, you may select a different program. Sorting by date & time

stamps will result in a good overview of what a disk has been used for at a certain time. E.g. the NTFS attribute “encrypted” might quickly reveal what files may turn out to be the most important ones in a forensic analysis.

The result can also be **output** directly to the **directory browser**. That means you have an “all files view” of a logical drive or partition in WinHex and can sort by date, filename extension, etc., move to the clusters where a file is stored, recover a file, delete irrelevant items from the list, etc. However, while the directory browser offers a more interactive approach, it shows no hash values. Blue items in the directory browser indicated compressed files (“C” in the Attr. column = files compressed at the NTFS file system level, “c” in the Attr. column = files inside ZIP archives etc.), green items indicate encrypted files (“E” in the Attr. column = files at the NTFS file system level, “e” in the Attr. column = files encrypted in an archive). The reason why items are highlighted in red is explained in the Attr. column, too: Either these are detected filename/file type mismatches, or alternate data streams (ADS) or non-directory index streams (INDX) in NTFS, which are listed separately. Only a forensic license allows to separately list and examine files in **ZIP, RAR, ARJ, GZ, TAR, and BZIP archives**, as long as the archives are not encrypted. The contents of archives in archives can be included as well, but no further level. When manually deleting irrelevant items from a contents table that is associated with an evidence object, WinHex can save your changes to the contents table file.

A forensic license allows you to detect **filename/file type mismatches** in files. For example, if someone has concealed an incriminating JPEG picture by naming it “invoice.xls” (wrong filename extension), the recognized file type “JPEG!” is stated in the Mismatch column of the output file and the file is listed in the directory browser a second time with the presumed correct extension, highlighted in red. A negative match (e.g. file with extension .jpg and no known file header signature) is signaled by the word “unknown” in the Mismatch column. The file signatures and extensions used for mismatch detection are defined in the accompanying tab-delimited text file “File Type Signatures.txt”, which you may fully customize. Please note that the link between the current data in unallocated clusters and deleted files and their filenames is weak, so false alerts might be displayed if a deleted file's clusters have been re-allocated to another file of a different type in the meantime.

A forensic license also allows to display all the files in a “**category**” **view** in the directory browser (available only if the active medium is assigned to a case). That means the files will be grouped by file categories such as “documents”, “Internet”, “images”, “multimedia”, etc. This is useful for example if you need a list (or gallery view) of all images on a hard disk, disregarding the directory structure. Files are assigned to a category because of their extension. The extension-category relation is defined in the accompanying text file “File Type Categories.txt”, which can be fully customized. It is loaded at start-up. All the files that do not belong to any defined category end up in the folder “Miscellaneous”.

A forensic license additionally allows to calculate the **percentage of skin colors** in pictures. This can be done for the same file types also supported by the gallery view, both for output to the directory browser and to a file. For example, if a forensic examiner is looking for traces of child pornography, sorting images by skin color percentage in category view may accelerate your work immensely because it renders checking the mass of 0%..9% skin color percentage pictures

obsolete (e.g. thousands of browser cache garbage files). Please note that there may be false positives, i.e. skin-like colors of a non-skin surface. Pictures that cannot be correctly scanned for skin colors (e.g. too large, corrupt file or black-and-white) will be listed with a question mark instead of the skin color percentage.

A forensic license allows to optionally perform an **entropy test** on each existing file to check whether it is encrypted or compressed. If the test is positive (the entropy exceeds a certain threshold), the file is flagged with “e?” in the attribute column, to indicate that it might deserve special attention. The entropy test is not applied to ZIP, RAR, TAR, GZ, BZ, 7Z, ARJ, JPG, PNG, GIF, TIF, MP3 and MPG files, which are well-known to be compressed internally. This test is not needed to detect that files are encrypted at the NTFS file system level or inside archives.

**Hash values** can be calculated for all files listed in a contents table. In addition to this, a forensic license allows to match the hash values against individually selected (or simply all) hash sets in the internal hash database, at three possible levels:

- 1) The hash values are matched against the database, but nothing is filtered yet. Matches can be seen in the directory browser columns “hash set” and “category”. It is possible to sort by these columns and manually filter out irrelevant files or address notable files specifically.
- 2) Only those files are output that are either unknown to the hash database or known as notable. Known irrelevant files are filtered out.
- 3) Only those files are output that are known to the hash database as notable. Known irrelevant files and files unknown to the database are filtered out.

## 3.7 Mode Buttons

When examining a logical drive, partition, or image file with a file system supported by WinHex, there are four buttons that determine the display in the lower half of the window, below the directory browser.

### Sectors

The default view that shows the binary data in all sectors as hexadecimal code, ASCII text, or both, along with an offset column.

### Preview

Checks the file signature of the file currently selected in the directory browser. If found to be a picture (supported file types see below), the picture is displayed, otherwise an ASCII text extract from the beginning of the file. The result of the signature check (whether it matches the filename extension or not) is displayed in the status bar. By double-clicking the preview, you get a full-size view of a picture, where you may zoom in and out using the keys + and -. Even incomplete pictures (e.g. files incompletely recovered because of fragmentation) can usually be displayed partially.

### Gallery

Checks the file signature of all the files in the currently visible portion of the directory browser. If found to be a picture, a thumbnail is displayed, otherwise a brief summary (filename, size, signature). By scrolling in the directory browser, the gallery view scrolls as well. You may switch the directory even while the thumbnails are still loading. By double-clicking a thumbnail, you get a full-size view of a picture, where you may zoom in and out using the keys + and -. Even incomplete pictures (e.g. file incompletely recovered because of fragmentation) can usually be displayed partially.

Supported picture file types: BMP, JPG/JPEG, JPEG 2000, PNG, GIF, TIF, TGA, PCX, WMF, EMF, MNG, JBG

### **Calendar** (timeline view)

Gives a convenient overview of when the files/directories selected in the directory browser were created in a file system (red), last modified (blue), and last accessed (green), in the form of a calendar. Each day with a time stamp for at least one file or directory is filled in the calendar with the corresponding color. Weekends (Saturdays and Sundays) are specially marked. Hover the mouse over a day to find out which files exactly are represented and to see the corresponding times. If the list for a certain day is too lengthy to be displayed completely, you can still sort the directory browser in a suitable way and find out there.

Example: During which period of time were JPEG files created on a volume? Either right-click the root directory in the directory tree (case data window) to recursively list all existing files or create a contents table, then sort by filename extension, select all JPEG files, then enable the calendar view, and watch out for red bars.

## **3.8 Logical Search**

The directory browser's context menu allows logical simultaneous search operations in files and folders that are selected in the directory browser (specialist or forensic license only).

Advantages:

- + The search scope can be limited to certain files and folders, also certain files and folders that are part of a contents table.

- + Searching in files (usually = in the cluster chains allocated to files) as a "logical" kind of search will find search term occurrences even if the search term happens to be physically split in a fragmented file (occurs at the end and the beginning of discontinuous clusters) and even if the file is compressed at the NTFS file system level and optionally even if it is part of an archive (ZIP, RAR, GZ, TAR, BZ2, 7Z, and ARJ, if not encrypted, forensic license only).

- + The text contained in PDF (Adobe), WPD (Corel WordPerfect), CDR (Corel Draw), and VSD

(Visio) files can automatically be extracted and decoded prior to search, such that their plaintext will be searched as well. Potential search hits in such files would otherwise be missed because these file types typically store text in an encoded, encrypted or otherwise garbled way. This feature requires the separate viewer component to be active for the decoding and text extraction part.

+ Files in which the search term occurs can be automatically opened or added to a dedicated contents table.

Unallocated space can be included in a logical search by including the fictitious file "Free Space" in the root directory, file system areas by including the fictitious file of the same name. Slack space is included depending on the directory browser options.

- Only a physical search can cover the transition from slack space to directly following free space.

### 3.9 Hash Database

Only available with a forensic license. The internal hash database, once created, consists of 257 binary files with the extension .xhd (X-Ways Hash Database). The storage folder is selected in the General Options dialog. The hash database is organized in a very efficient way, which maximizes performance when matching hash values. It is up to you to decide, around what hash type the database is built (MD5, SHA-1, SHA-256, ...).

Each hash value in the hash database belongs to one or more hash sets. Each hash set belongs to either the category "known good"/"harmless"/"irrelevant" or "known bad"/"malicious"/"relevant"/"notable".

Hash values of files can be calculated and matched against the hash database when creating a contents table. The directory browser's optional columns "Hash Set" and "Category" will then reveal for each file to which hash set and category it belongs, if any (which allows you to sort by these aspects and filter out irrelevant files easily).

The Tools menu allows you to

- manage the active hash database: create a new (empty) one, view the list of hash sets, rename and delete hash sets
- import a single hash set text file (NSRL RDS 2.x, HashKeeper, and ILook text files are supported)
- import all the hash set text files in a certain folder and all its subfolders (ditto), optionally into a single internal hash set whose name you have to specify
- delete the active hash database, e.g. to start a fresh one with new hash sets and/or a new hash type.

The Create Hash Set command in the directory browser's context menu allows you to create your own hash sets in the internal hash database. The hash database supports up to 65,535 hash sets. Future versions will allow you to export hash sets in the hash database to the NSRL RDS format.

# 4 Working with the Hex Editor

## 4.1 Start Center

The so-called Start Center is a dialog window that is optionally displayed at startup and is meant as a simplified control panel for beginning your work. It allows to quickly open files, disks, memory modules, and folders as well as up to 255 recently edited documents (16 by default, left-hand list). These may be files, folders, logical drives or physical disks. When opened again, WinHex restores the last cursor position, the scrolling position, and the block (if defined) of each document, unless the corresponding option is disabled.

From the Start Center you are also able to access *projects* and *cases* (right-hand top list). A project consists of one or more documents to edit (files or disks). It remembers the editing positions, the window sizes and positions and some display options. By saving a window arrangement as a project you can continue to work in several documents right where you left them, with a single click only. This is especially useful for recurring tasks. When you load a project, all currently opened windows are automatically closed first.

Besides, WinHex automatically saves the window arrangement from the end of a WinHex session as a project, and can re-create it next time at startup. Each project is stored in a .prj file. It can be deleted or renamed right within the Start Center (context menu or DELETE/F2 key).

Last not least, the Start Center is the place where to manage *scripts*. You may check, edit, create, rename, and delete scripts using the context menu. To execute a script, double-click it or single-click it and click the OK button.

## 4.2 Entering Characters

In hex mode only hexadecimal characters are to be entered ('0'...'9', 'A'...'F'). In text mode you can enter all kinds of characters: letters, numbers, punctuation marks and special characters (e.g. '»', ']' and '^'). Please use the Windows program charmap.exe to find out key combinations for such characters (e.g. Alt-1-7-5 for '»'). The "WinHex" font even supports the Euro symbol (€).

## 4.3 Edit Modes

The details panel displays for each file/disk, in which mode it was opened. The details panel's context menu allows to selectively change the edit mode of the active window.

**Read-only/View mode:** Recommended for computer forensic examinations. In order to enforce strict forensic procedures, the only mode available in X-Ways Forensics, except for files in the current case's directory and in the general folder for temporary files, to allow to decode, decrypt, and convert them, etc. Files or disks that are opened in view mode cannot be (intentionally or

accidentally) edited/alterd, only viewed. In other words, they are opened write-protected = read-only.

**Default edit mode:** Modifications to files or disks opened in default edit mode are stored in temporary files. Those temporary files are created and maintained dynamically when needed. Only when you close the edit window or use the Save menu command the File Menu, the modifications are flushed and the original file or disk is updated, after prompting the user.

**In-place edit mode:** Please use caution when opening files or disks in in-place edit mode. All kinds of modifications (keyboard input, filling/removing the block, writing clipboard data, replacements, ...) are written *to the original file or disk* (“in-place”) *without prompting!* It is not necessary to save the file manually after having modified it. Instead, the modifications are saved lazily and automatically, at latest when closing the edit window. However, you may use the Save command to ensure the buffer is flushed at a given time.

The in-place edit mode is preferable if the data transfer from the original to the temporary file and vice-versa, which is obligatory in default edit mode for certain operations, consumed too much time or disk space. This may be the case when opening very large files or when modifying huge amounts of data. Since usually no temporary files are needed in in-place edit mode, this edit mode is generally faster than the default edit mode. The in-place edit mode is the only mode available when using the RAM editor.

Even in in-place edit mode the creation of a temporary file is unavoidable when altering the file size.

## 4.4 Status Bar

The status bar displays the following information about a file:

1. Number of current page and total number of pages (disk editor: sectors)
2. Current position (offset)
3. Decimal translation of the hex values at the current position
4. Beginning and end of the current block (if currently defined)
5. Size of current block in bytes (ditto)

Click the status bar cells in order to...

1. Move to another page/sector,
2. Move to another offset,
3. Define the integer type for decimal translation and
4. Define the block.

Right-click the status bar in order to copy pieces of information from the status bar into the clipboard.

Right-clicking the 2<sup>nd</sup> status bar field allows switching between absolute (default) and relative

offset presentation. This is useful when examining data that consists of records of a fixed length. After specifying the record length in bytes, the status bar displays the current record number and the relative offset therein.

Right-clicking the 3<sup>rd</sup> status bar field allows copying the four hex values at the current position in reverse order into the clipboard. This is useful for following pointers.

## 4.5 Scripts

Most of the functionality of WinHex can be used in an automated way, e.g. to speed up recurring routine tasks or to perform certain tasks on unattended remote computers. The ability to execute scripts other than the supplied sample scripts is limited to owners of professional licenses or higher. Scripts can be run from the Start Center or the command line. While a script is executed, you may press Esc to abort. Because of their superior possibilities, scripts supersede routines, which were the only method of automation in previous versions of WinHex.

WinHex scripts are text files with the filename extension ".whs". They can be edited using any text editor and simply consist of a sequence of commands. It is recommended to enter one command per line only, for reasons of visual clarity. Depending on the command, you may need to specify parameters next to a command. Most commands affect the file or disk presented in the currently active window.

See Appendix B for a description of currently supported script commands.

## 4.6 WinHex API

The WinHex API (application programming interface) allows to use the advanced capabilities of the WinHex Hex Editor programmatically from your own C++, Delphi, or Visual Basic programs. In particular, it provides a convenient and simple interface for random access to files and disks.

Developing software that uses the WinHex API requires a valid *professional* or *specialist* WinHex license. Additionally, you need import declarations for your programming language of choice, the library file "whxapi.dll", and the API documentation. Please find those files and more detailed information online at <http://www.x-ways.net/winhex/api/>.

You may also *distribute* both any software that makes use of the WinHex API and WinHex itself. There are two ways how to distribute WinHex:

1. Distribute the unlicensed WinHex version. For the API to work, your customer has to purchase professional or specialist licenses according to the number of WinHex installations needed.
- or-
2. Recommended: distribute a special API version of WinHex that is configured to only provide the API functionality and that is available at a reduced price. You may place your order online at

<http://www.x-ways.net/winhex/api/>. Volume discount available on request (please specify the number of licenses you are interested in). One WinHex API license needed per end user computer. The product will be licensed to you, you will be the actual owner of the licenses, but any of your customers may use them. The end user does not have to take care of anything related to WinHex.

## 4.7 Disk Editor

The disk editor, that is part of the Tools menu, allows you to access floppy and hard disks below the file-system level. Disks consist of sectors (commonly units of 512 bytes). You may access a disk either logically (i.e. controlled by the operating system) or physically (controlled by the BIOS). On most computer systems you can even access CD-ROM and DVD media. There is an optional raw mode for optical drives that allows to read from audio CDs and also the complete 2352-byte sectors on data CDs (CD-ROM and Video CDs) that contain error correction codes.

Opening a *logical drive* means opening a contiguous formatted part of a disk (a partition) that is accessible under Windows as a drive letter. It's also called a "volume". WinHex relies on Windows being able to access the drive. Opening a *physical disk* means opening the entire medium, as it is attached to the computer, e.g. a hard disk including *all* partitions. It could also be called the "raw device". The disk normally does not need to be properly formatted in order to open it that way.

Usually it is preferable to open a logical drive instead of a physical disk, because more features are provided in this case. For example, "clusters" are defined by the file system, the allocation of clusters to files (and vice versa) is known to WinHex, "free space" and "slack space" have a meaning. Only if you need to edit sectors outside a logical drive (e.g. the master boot record), if you wish to search something on several partitions of a hard disk at the same time, or if a partition is damaged or formatted with a file system unknown to Windows, so Windows is unable to make it accessible as a drive letter, you would open the physical disk instead. Via the menu that appears when clicking the "Access" button, you may also open individual partitions from within a physical disk. WinHex understands both conventional MBR partitioning and Windows 2000's dynamic disks as organized by the LDM (Logical Disk Manager, specialist and forensic licenses only). All dynamic volume types are supported: simple, spanned, striped, and RAID 5. Holding the Ctrl key when opening hard disks disables detection and special handling of dynamic volumes and ensures the hard disk is treated like it has been partitioned in the conventional way.

Please note the following limitations:

- Under Windows NT and its successors administrator rights are needed to access hard disks.
- Under Windows 9x, certain requirements must be met to access CD-ROM and DVD media (see Appendix C).
- Replace functions are not available.
- WinHex cannot *write* to CD-ROM or DVD.
- The disk editor cannot operate on remote (network) drives.

The appendix E of this manual provides you with specifications of the master boot record, that

can be edited using the disk editor.

#### Edit free space on drive (Windows 95/98/Me)

Under Windows 95/98/Me, it is possible to edit the currently unused space on a logical drive. The aforementioned limitations do not apply in this case. WinHex creates a file which uses the complete free space on the selected drive. You can edit this file in in-place mode. The integrity of data in the used parts of the drive cannot be affected hereby.

You can use this function to recover unintentionally deleted data which has not yet been overwritten by new files. Search for the data, mark it as the current block and copy it. Of course, data that has been deleted by WinHex using the Wipe Securely command cannot be found in unused parts of a drive any more.

**Save Sectors:** To be used analogously to the Save command for files. Part of the File menu. Writes all modifications to the disk. Please note that, depending on your changes, this may severely damage the integrity of the disk data. If the corresponding undo option is enabled, a backup of the concerned sectors is created, before they are overwritten. *This command is only available in the full version.*

## 4.8 Directory Browser

On logical drives and partitions formatted with FAT12, FAT16, FAT32, NTFS, Ext2, Ext3, CDFS/ISO 9660/Joliet, or UDF, WinHex offers a *directory browser*, which resembles the Windows Explorer's right-hand list. It can be disabled or enabled by clicking the checkbox next to the Access button. The directory browser lists existing files and directories first, then deleted files and directories. Compressed files are displayed in blue, encrypted files in green (NTFS only). Right-clicking any item in the directory browser brings up a context menu with commands for opening a file or directory, exploring a directory, locating the beginning of a file or directory on the disk, locating the corresponding directory entry (FAT) or file record (NTFS), listing the allocated clusters in a separate window, and for easily recovering a lost or existing file or directory. The latter can recreate entire directory structures. Double-clicking executes the default action (locating the data, listing clusters, and exploring in the case of a directory). Clicking items in the separate list window allows to navigate to the listed clusters in the disk editor.

Deleted files and directories are represented in the directory browser with lighter icons. Question mark icons indicate that the original file or directory contents may be still available. Deleted objects that WinHex knows are no longer accessible (either because their first cluster has been reallocated or because they have a size of 0 bytes) have icons crossed out in red.

The directory browser can sort files and directories in ascending or descending order, and still reveals the previous sort criterion with a lighter arrow. For example, if you first click the filename column and then the filename extension column, files with the same extension will internally still be sorted by name.

On NTFS drives, the directory browser lists all deleted files on the drive of which traces can still be found in a single dedicated virtual folder “Deleted Objects” for reasons of convenience. These are the same deleted files that are also recoverable using File Recovery by Name with the “thorough search” option disabled.

## Columns

Filename	Name of the listed file or directory.
Ext.	Filename extension. The part of the filename that follows the last dot, if any, if less than 6 characters.
Category	File type category corresponding to the extension, according to the definition in “File Type Categories.txt”. (forensic license only)
Path	Path of the file or directory, starting with a backward slash if the path is known, based on a volume's root, or starting with a question mark if the exact path is unknown.
Size	Size of the file or directory, without slack. On NTFS, the size of directories is listed only for contents tables.
Created	The date and time the file or directory was created on the volume it resides on. Not available on Linux filesystems.
Modified	The date and time the file or directory was last modified on the volume it resides on. On FAT, time precision is 2-second intervals only. On CDFS, the only available date and time stamp is listed in this column although it does not necessarily indicate last modification.
Accessed	The date and time the file or directory was last read or otherwise accessed on the volume it resides on. On FAT, only the date is recorded.
Record update	The date and time the file's or directory's FILE record (on NTFS) or inode (Linux filesystems) was last modified. These are filesystem data structures that contain the file's meta data.
Deletion	The date and time the file or directory was deleted. Available on Linux filesystems only.
Attr.	DOS/Windows attributes on FAT/NTFS filesystems, Unix/Linux permissions on Ext2/Ext3/Reiser filesystems, plus some proprietary symbols. See below.
SC%	Skin color percentage. Optionally available for contents tables. Indicates the degree pictures are composed of skin tones. Sorting by this column is the most efficient way to discover traces of e.g. child pornography.
Hash	The file's hash value, available in contents tables if calculated.
Hash set	In the internal hash database, the name of the hash set that the file's hash value, if available, belongs to.
Category	The category of the hash set that the file's hash value, if available, belongs to. Either

"irrelevant", "notable", or blank.

## Attributes

A = to be archived

R = read-only

H = hidden

S = system

P = junction point

C = compressed at file system level

c = compressed in an archive (ZIP, RAR, ...)

E = encrypted at file system level

e = encrypted in an archive (ZIP, RAR)

e? = possibly encrypted or compressed, according to the entropy test

The built-in priority when sorting by the Attr. column in descending order is as follows:

- 1) NTFS alternate data streams, Linux SUID
- 2) NTFS \$EFS streams, Linux symlinks
- 3) NTFS non-directory INDX streams, Linux special files
- 4) NTFS filesystem encryption
- 5) Positive mismatches (e.g. JPEG misnamed as .dll)
- 6) Negative mismatches (e.g. text file with no known signature named .jpg)
- 7) User-level encryption (e.g. in a ZIP archive)
- 8) User-level encryption supposed (flagged by the entropy test)
- 9) User-level compression (e.g. in a ZIP archive)
- 10) NTFS filesystem compression
- 11) NTFS reparse points
- 12) ordinary Windows attributes and Linux permissions

The directory browser can also be filled by Create Drive Contents Table.

## 4.9 Directory Browser Context Menu

The directory browser context menu allows the user to directly interact with the currently selected file(s). There are a number of menu commands which are available depending on the selected items. Double clicking files and directories will, depending on the circumstances, either call "View", "Explore" or the associated external program.

### View

This command allows viewing the selected file with WinHex' internal viewers for Windows Registry files and various graphical file formats. For other files, the mode of operation depends on the installed components: If X-Ways Trace is installed, and the file is either an "info2" file used by the Windows Recycle Bin or an "index.dat" used by Internet Explorer, X-Ways Trace is

called for these files. If the X-Ways Forensics external viewer component is active, all other files are sent to that viewer. If it is not, the first installed external program will be called instead.

Exceptions to all of the above are files beyond 2 GB in size and NTFS system files. These are always opened as data windows.

## **Explore**

Only available for directories and archives (ZIP, RAR, TAR...), this command allows navigating into them within the directory browser. Double-clicking archives or directories does the same. A command that allows listing the contents of directories as well as their subdirectories at the same time can be found in the directory tree's context menu instead (in the Case Data window, "Explore recursively").

## **External Programs**

Allows sending the selected file(s) to one of the external programs currently configured or the file's associated program in the current Windows installation. This association is determined based on file extension as is usual within Windows.

## **Position**

The Position group of commands allows interactions with the currently selected file on a generally more technical level. It allows accessing the file's (or directory's) first cluster on the disk in the sectors view, accessing its related information like MFT record in NTFS or Inode in Ext2/Ext3 and also sorting the files by their physical order on disk: "Sort by directory entry location" (FAT), "Sort by Inode Offset" (Ext2/Ext3) or "Sort by MFT ID" (NTFS), respectively, allow to see files and folders in the order in which they physically appear in file system data structures (directory entries, the MFT, or Inode tables).

The Position menu also allows calling for a file's or directory's cluster list, i.e. the cluster list window will be opened and filled with the selected item's cluster list, and it allows deleting the selection from the list, which is of particular interest when dealing with contents tables. The deletion of items from a contents table can be made permanent by clicking the floppy disk icon that will appear in the directory browser's caption line. The menu also allows to add or move files to special report contents tables.

## **Recover/Copy**

Allows copying the selected files from their current location to a location available for a standard Windows file dialog, e.g. out of an interpreted image file or from a local disk. This can be applied to both existing and deleted files and directories. When working with an active case and if logging is enabled, the copy/recovery process is documented in the case log. Both the source and the target paths are recorded.

## **Mark as Noteworthy/Unmark Item**

In the directory browser of an evidence object, you mark files as noteworthy, thereby adding them to a dedicated contents table and highlighting them. These files will then also be listed in the case report. Having them in a dedicated contents table allows to conveniently copy/recover them in a single step at a later point of time or get a gallery overview of these files specifically.

The visual highlighting can be undone with the context menu command "Unmark item". In order to also remove them from the dedicated contents table for noteworthy files, use the "Delete from list" command in the directory browser context menu when that contents table is loaded or press the Del key on your keyboard. Then click the floppy disk icon to save.

### **Add to Active Case**

Performs the same operation as Recover/Copy but at the same time, the resulting file(s) will be added to the current case as evidence objects.

### **Logical Search**

See chapter of that name.

### **Create Hash Set**

Creates a hash set of the currently selected files and directories and their subdirectories directly within the internal hash database.

### **Create Directory Contents Table**

Creates a contents table just like a drive contents table except that it exclusively focuses on files located within the directory currently selected and its subdirectories.

### **Open**

Opens currently selected files or directories in separate data windows. In the case of a directory, the directory's data structures will be opened.

## **4.10 RAM Editor**

The RAM editor allows to examine the physical RAM/main memory (under Windows 2000/XP, with administrator rights only) and the virtual memory of a process (i.e. a program that is being executed). All memory pages committed by a process are presented in a continuous block. Unused (free or reserved) pages are ignored by default, but optionally included and displayed with “?” characters. With no such gaps, you may compare memory dumps to files exactly with one another (absolute and virtual addresses are identical), e.g. to examine stack and heap states or observe viruses.

Select one of the listed processes. You may access either the so-called primary memory or the entire memory of this process, or one of the loaded modules. The primary memory is used by programs for nearly all purposes. Usually it also contains the main module of a process (the EXE file), the stack, and the heap. The “entire memory” contains the whole virtual memory of a process including the part of memory that is shared among all processes, except system modules. Under Windows 95/98/Me, system modules are listed optionally in the process tree. System modules are defined as modules that are loaded above the 2 GB barrier (such as kernel32.dll, gdi32.dll). They are shared among all running processes.

Please note the following limitations:

- Caution: Only keyboard input can be undone!
- Virtual memory of 16-bit processes is *partially* accessible under Windows 95/98/Me only.
- Editing is possible in in-place mode only.
- System modules of Windows 95/98/Me can only be *examined* in view mode, *not manipulated*.

The options relevant for the RAM editor are “Check for virtual memory alteration” and “Virtual Addresses”.

## 4.11 Template Editing

A template is a dialog box that provides means for editing custom data structures in a more comfortable and error-preventing way than raw hex editing does. Editing is done in separate edit boxes. Changes take effect when pressing the **ENTER** key or when quitting the template after being prompted. The data may originate from a file, from disk sectors, or from virtual memory. Especially when editing databases, you may prefer to define a custom template for ease of access to the records. You will find the command to print a template in the system menu.

A *template definition* is stored in a text file. The *template editor* enables you to write template definitions and offers syntax checking. A template definition mainly contains variable declarations, that are similar to those in source code of programming languages. The syntax is explained in detail in Appendix A. The supported data types include all the common integer, floating-point and boolean variants, date types, hex values, binary, characters, and strings type. Arrays of both single variables and groups of variables can be used.

The ability to move freely forwards and backwards within the data makes using templates particularly flexible:

- The same variable may be interpreted and manipulated in several ways.
- Irrelevant data sections can be skipped.

The *template manager* lists all text files in the WinHex directory that contain template definitions. The title of the template along with a description, the filename, and the date and time of the last modification are shown. Click the Apply button to display a template using the selected template definition for the data in the current editor window at the current position. You may also create a new template definition, delete or edit an existing one.

WinHex comes with several sample templates.

## 4.12 Useful Hints

- Use the mouse buttons as follows to define the block (if the context menu is switched off):
  - Double-clicking left sets the block beginning.
  - Single-clicking right sets the block end.
  - Double-clicking the right button clears the block.
- You may want to define the block using the keyboard (**SHIFT**+arrow keys or **ALT+1** and **ALT+2**).
- Use the **TAB** key to switch between hexadecimal and text mode.
- Use the **INS** key to switch between insert and overwrite mode.
- **CTRL+Q** closes all windows.
- **ENTER** displays the Start Center.
- **CTRL+ENTER** displays the Window Manager.
- **ESC** aborts the current operation if any, otherwise clears the block, dismisses an active dialog or template window.
- **PAUSE** stops or continues the current operation.
- **F11** repeats the last Go To Offset command. **CTRL+F11** works in the opposite direction (from the current position).
- **ALT++** is a variant of the Go To Offset command specifically to jump a certain number of sectors *down*.
- **ALT+-** is another variant specifically to jump a certain number of sectors *up*.
- **SHIFT+F7** switches between three character sets.
- **(SHIFT+)ALT+F11** repeats the last Move Block command.
- **CTRL+SHIFT+M** invokes an open evidence object's annotations.
- **ALT+F2** recalculates the auto-hash (checksum or digest) after a file was modified.
- **ALT+LEFT** and **ALT+RIGHT** allow for switching between records within a template (just as the "<" and ">" buttons). **ALT+HOME** and **ALT+END** access the first and the last record, respectively.
- **ALT+G** moves the cursor in the edit window to the current template position and closes the template window.
- **CTRL+F9** opens the Access button menu (disk edit windows only).
- WinHex accepts filenames specified in the command line and is drag-and-drop capable.
- Use scripts to make your work with WinHex more efficient.
- You can specify the name of a script as a command line parameter.
- “Invalid input”: After dismissing this error message box, the blinking cursor indicates what parameter provided by you is invalid and needs to be corrected.
- Switch from hexadecimal to decimal offset presentation by clicking the offset numbers.
- Try clicking the status bar cells (left and right mouse button).

# 5 Data Recovery

## 5.1 File Recovery with the Directory Browser

Most obviously, deleted files and directories that are listed in the directory browser can be recovered easily and selectively with the directory browser's context menu. See chapter "directory browser".

## 5.2 File Recovery by *Name*

This is a very easy to use mass file recovery function, part of the the Disk Tools menu. Requires that you have opened a logical drive or a single partition of a physical disk with the disk editor. Works on FAT12, FAT16, FAT32, and NTFS drives. You may specify one or more filename patterns that cover all the files you wish to retrieve, e.g.:

- Letter to Mr. Smith.doc
- Invoice\*.pdf
- m\*.xls
- Image\*.gif
- \*.tif

You may exclude files using filename patterns that begin with a colon (:). For example, you may include all files except NTFS system (which always start with a dollar sign) files by providing the following patterns:

- \*
- :\$\*

Please note that files that were moved to the recycle bin prior to permanent deletion are internally renamed by Windows, where only the filename extension remains the same, so using wildcards will often come handy (e.g. \*.jpg instead of abc.jpg). Unlike File Recovery by Type, this function will also restore the file date & time and its attributes.

Optionally this function recovers/copies only those files that currently exist in the file system (from a user's pointer of view) or that are considered non-existent (deleted or otherwise lost).

Alternatively to using the file allocation table of a FAT drive, WinHex can optionally also rely on files not being fragmented, recovering them as a continuous stream of consecutive clusters.

Check "Intercept invalid filenames" to prevent a failure of the recovery because of filenames with characters considered as invalid by the file system. Useful for example if you wish to recover files that had filenames in a non-western language with a western-language Windows version. This option will rename such a file if necessary to ensure that it can be recreated.

On an NTFS drive, if the file you are looking for cannot be found, it may help to enable the

“thorough” search. It is not enabled by default because it takes significantly more time.

You must also specify an output folder where to recreate the original file(s). Important: make sure this folder is on a different drive. Specifying a folder on the same drive where you are recovering from could easily overwrite disk space where deleted files reside that you still wish to recover! That way they would be lost forever. It might also lead to a loop, if WinHex repeatedly "recovers" files that it has just recreated.

## 5.3 File Recovery by *Type*

Another data recovery function in the Disk Tools menu. This recovery method is also referred to as "file carving". It searches for files that can be recognized by a characteristic file header signature (a certain sequence of byte values). Because of this approach, File Recovery by Type does not depend on the existence of functional file system structures. When found based on the signature, the files are saved to the output folder that is specified by the user. Optionally, recovered files of each type are put into their own subfolder (...\\JPEG, ...\\HTML, etc.). Note that File Recovery by Type assumes contiguous file clusters, so produces corrupt files in case the files were originally stored in a fragmented way. A log file "File Recovery by Type.log" about the selected parameters and the recovery results is written to the output folder for verification purposes.

Since no use is made of a possible presence of a (functional or non-functional) file system, the original *file sizes* are principally *unknown* to this algorithm, and so are the original *filenames*. That is why the resulting files are named according to the following pattern: Prefix[X]id0000.ext. "Prefix" is an optional prefix you provide. "id" is a unique character combination that identifies an entry in the file type definition database (aa = 1st entry, ab = 2nd entry, ...). "0000" is an incrementing number per file type. "ext" is the filename extension that corresponds to the file header signature according to the file definition database. WinHex can often detect if recovered JPEG, GIF, and files of some other types, are corrupt or incomplete (caused e.g. by file fragmentation). If this is the case, it will mark these files as corrupt in the log file and insert a capital X in the output filename. If the user-supplied file size limit is found to be too small for certain files, this will be noted in the log file as well.

The algorithm tries to determine the original size of JPEG, GIF, PNG, BMP, TIFF, CDR, AVI, WAV, ZIP, MS Word, MS Excel, MS PowerPoint, RTF, and HTML files by examining their data structure, roughly limited by the user-supplied maximum size. The corresponding entries in the file type definition database must not be altered in order for the size and type detection to work for these file types. For other types, the files are recovered at the exact size specified by the user as the maximum (in KB). Be generous when specifying this size because whereas files recovered "too large" can still be opened by their associated application, truncated files often can't be and are obviously incomplete.

Technically it is possible to select as many file types for simultaneous recovery as you like. However, if you e.g. recover MS Office and AVI files at the same time and the MS Office files you expect are around a few KB and the AVI files around a 1 GB in size, using a single global

maximum file size would not be a good idea. That's why optionally you can define an individual default size for each file type in the file type definition database.

By default, file headers are only searched at *cluster* boundaries because the beginning of a cluster is the only place where a file can start in a cluster-based file system. However, you may also select to search for sector-aligned file headers. This is useful to find files from a previously existing volume with a different cluster layout. If performed on a physical medium or raw file with no cluster layout defined, WinHex searches at sector boundaries anyway if cluster boundaries are selected. There is yet another possibility, a thorough byte-level search. This is necessary when recovering files from backup files or tapes, or JPEG files from within MS Word documents, where they are not aligned at cluster or sector boundaries. This comes at the cost of a possibly increased number of false positives, though, misidentified file signatures occurring randomly on a media, not indicating the beginning of a file.

You may limit the scope of the recovery to a currently selected block if necessary and/or to allocated or unallocated space (option available on a logical drive or volume). E.g. in order to recover files that were deleted, you select to recover from unallocated space only. Files that are not accessible any more because of file system errors may still be stored in clusters that are considered as in use.

The option "Ext2/Ext3 block logic" causes this recovery method to deviate from the standard assumption of no fragmentation in that it will follow the typical Ext block pattern, where e.g. the 13th block from the header of the file is considered an indirect block that references the following data blocks. This option has no effect when applied to partitions that WinHex knows have a file system other than Ext2 and Ext3 or when a header is found that is not block-aligned.

If you enable the option "No actual recovery, just list found files", no files are actually output and no log file is written. Files are only listed in the directory browser, e.g. for inspection with the gallery viewer and for selective recovery from within the directory browser.

## 5.4 File Type Definitions

"File Type Signatures.txt" is a tab-delimited text file that serves as a file type definition database for contents tables and for the File Recovery by Type command. Up to 255 entries supported.

WinHex comes with various preset file type signatures. You may fully customize the file type definitions and add your own ones. When you click the Customize button to edit the file "File Type Signatures.txt", by default WinHex opens the file in MS Excel. This is convenient because the file consists of columns separated by tabs. If you edit the file with a text editor, be sure to retain these tabs, as WinHex relies on their presence to properly interpret the file type definitions. MS Excel retains them automatically. After editing the file type definitions, you need to exit the dialog window and invoke the File Recovery by Type or Create Drive Contents Table menu command again to see the changes in the file type list.

### **1st column: File Type**

A human-readable designation of the file type, e.g. "JPEG". Everything beyond the first 19 characters is ignored.

### **2nd column: Extensions**

One or more file type extensions typically used for this file type. E.g. "jpg;jpeg;jpe". Specify the most common extension first because that one will be used by default for naming recovered files. Everything beyond the first 45 characters is ignored.

### **3rd column: Header**

A unique header signature by which files of this file type can be recognized. May be specified in either ASCII or hex (e.g. 0xFFD8FF). To find out characteristic file header signatures in the first place, open several existing files of a certain type in WinHex and look for common byte values near the beginning of the file at identical offsets. Everything beyond the first 16 characters is ignored.

### **4th column: Offset**

The relative offset within a file at which the signature occurs. Often simply 0.

### **5th column: Footer**

Optional. A signature (constant byte sequence) that reliably indicates the end of a file. May be specified in either ASCII or hex. A footer signature may help to force a recovery with the correct file size. Still, the recovery algorithm does not search for the footer further than the number of bytes specified as the maximum file size, starting from the header. Everything beyond the first 16 characters is ignored.

### **6th column: Default in KB**

Optional. A file type specific default maximum file size in KB that can override the global maximum file size specified in the File Recovery by Type dialog window. Useful because e.g. an MPEG video could be more around 1 GB in size, where a Windows icon file (.ico) could be around 1 KB in size.

## **5.5 Manual Data Recovery**

It is possible to restore lost or logically deleted files (or more general: data) that are merely marked as deleted in the file system, but have not been *physically* erased (or overwritten).

Open the logical drive where the deleted file resided on using the disk editor. Principally you can recreate such a file by selecting the disk sectors, that were allocated to the file, as the current block and saving them using the menu command Edit | Copy Block | Into New File. But it may

prove difficult to *find* the sectors where the file is still stored. There are two general ways to accomplish this:

1. In case you know a snippet of the file you are looking for (e.g. the characteristic signature in the header of a JPEG file or the words “Dear Mr. Smith” in a MS Word document), search it on the disk using the common search commands (“Find Text” or “Find Hex Values”). This is a very simple and safe way, and can be recommended to anyone.
2. In case you only know the filename, you will need some knowledge about the filesystem on the disk (FAT16, FAT32, NTFS, ...) to find traces of former directory entries of the file and thereby determine the number of the first cluster that was allocated to the file. Detailed information on file systems is available on the WinHex web site. The following applies to all FAT variants:

If the directory that *contained* the file (let's call that directory “D”) still exists, you can find D on the disk using Tools | Disk Tools | List Directory Clusters. The factory template for FAT directory entries that comes with WinHex will then be helpful to find out the number of the first cluster that was allocated to the deleted file in that directory. Otherwise, if D has been deleted as well, you need to find the contents of D (using the directory entry template) starting with the directory that contained D.

Deleted files and directories are marked with the character “à” (hexadecimal: E5) as the first letter in their name.

You may encounter the problem that the file to recover is fragmented, i. e. not stored in subsequent contiguous clusters. On FAT drives, the next cluster of a file can be looked up in the file allocation table at the beginning of the drive (simple templates to do this can be found on the web site), but this information is erased when a file is deleted.

## 6 Menu Reference

### 6.1 File Menu

**New:** This command is used to create a file. The file is initialized with zero bytes and principally opened in default edit mode. You have to specify the desired file size.

**Open:** Lets you open one or more files. You may choose an edit mode in case it is not predetermined in the Options menu.

**Save:** Saves the currently displayed file to the disk. In in-place edit mode, using this command is not necessary. When using the disk editor, this command is named “Save Sectors”.

**Save As:** Saves the currently displayed file under a different name.

**Make Backup Copy/Make Disk Backup:** cf. “Backups”

**Restore Backup:** Select an image or backup file (.whx file) that you would like to restore.

**Backup Manager:** cf. “Backups”

**Execute:** Executes the current file if executable, or otherwise the associated program.

**Print:** Use this command to print a file, disk sectors or RAM contents. Define the printing range via offsets. You may select and set up a printer. Choose the character set for printing and accept or change the suggested font size. The recommended font size is calculated as follows: print resolution (e.g. 720 dpi) / 6 (e.g. = 120). If desired you may enter a comment which will be printed at the end.

In case you need more flexibility with printing, you can define a block and copy it using “Edit->Copy->Editor Display” as a hex-editor-formatted text into the clipboard. You may paste it in your favorite word processor. It should look perfect in “Courier New”, 10 pt.

**Properties:** Lets you edit the size, the time stamp and attributes of a file (under Windows NT as well as of a directory). Valid attributes are: A (archive), S (system), H (hidden), R (read-only). After entering new values in any area (size, time or attributes), simply press the **ENTER** key, so the modifications take effect.

**Open Folder:** This command is used to open several files that meet special requirements at a time. Select a folder in which to open files. Subfolders are browsed optionally. You may specify a series of file masks (like “w\*.exe;x\*.dll”). There is also a switch that permits opening only those files that contain a certain text or certain hex values. The standard search dialogs are displayed upon request for this purpose. If WinHex is not set up to work as a viewer or in-place editor (this can be done in the Tools menu), you may choose an edit mode.

**Save Modified Files:** All files which have been changed are written to the disk.

**Save All Files:** All files that have not been opened in view mode are written to the disk.

**Exit:** Use this command to end WinHex. You will be prompted to save any modifications to files and disks.

## 6.2 Edit Menu

**Undo:** Reverses the last modification, in case the corresponding undo option was activated.

**Cut:** Removes the current block from the file and puts it into the clipboard. The data following the block is pulled to the former block beginning.

**Copy Block/All/Sector:**

- **Normally:** Copies the current block/the entire file/the current sector into the clipboard. The contents of the clipboard can be pasted or written later.
- **Into New File:** Copies the data directly into a new file (not via the clipboard). For instance, this command can be used to recover a lost file from disk sectors.
- **Hex Values:** Copies the data as concatenated hex values.
- **Editor Display:** Copies the data as text, formatted as if it was displayed in the hex editor, i.e. with an offset, a hex and a text column.
- **C/Pascal Source:** Copies the data as C/Pascal-formatted source code into the clipboard.

**Paste Clipboard:** Inserts the clipboard contents at the current position of a file. The file data following this position is moved forward.

**Write Clipboard:** Copies the clipboard contents to the current file at the current position. The data at this position is overwritten. If the end of the file is encountered, the file size is increased so that the clipboard contents finds place.

**Paste Clipboard Into New File:** Creates a new file of the clipboard contents.

**Empty Clipboard:** This command is used to free the memory used by the clipboard.

**Remove:** Deletes the current block from the file. The data following the block is pulled to the former block beginning. The clipboard is not affected by this command. If the block is equally defined in all open files (i.e. it begins and ends at the same offsets), this command can even be applied to all open files at the same time.

**Paste Zero Bytes:** Use this command to insert zero bytes at the current position of a file.

**Define Block:** This function is accessible from the menu and the status bar. A dialog box lets you specify the desired block limits. This command can also be applied to all open files.

**Select All:** Defines the beginning and the end of the current file as its block limits.

**Convert:** cf. Conversions

**Modify Data:** see below

**Fill Block/File/Disk Sectors:** see below (Wiping and Initializing)

## 6.3 Search Menu

**Find Text:** This command is used to search for a specified string of up to 50 characters in the current file, disk or RAM section (cf. Search Options). Specialist and forensic licenses only: identical to Simultaneous Search, unless Shift key is pressed.

**Find Hex Values:** This command is used to search for a sequence of up to 50 two-character hex

values (cf. Search Options).

**Replace Text:** Use this command to replace occurrences of a specified string with another string (each of up to 50 characters), cf. Replace Options.

**Replace Hex Values:** Functions exactly as the Replace Text command, but is applied to a sequence of hex values (50 at max.), cf. Replace Options.

**Combined Search:** Provides a complex search mechanism. In the current and in a second file a common offset is searched, where each file contains the specified respective hex values.

**Integer Value:** Enter an integer (within the limits of the signed 64-bit integer data type). This function searches data in the current file, which can be interpreted as this integer.

**Floating-Point Value:** Enter a floating-point number (e.g.  $12.34 = 0.1234 * 10^2 = 0.1234E2$ ) and select a floating-point data type. This function searches data in the current file, which can be interpreted as this floating-point value.

**Text Passages:** Use this command to look for a sequence of letters (a-z, A-Z), digits (0-9) and/or punctuation marks. It is useful for instance if you intend to translate text passages hidden somewhere in a file with executable code.

Set the sensitivity of the search by specifying how long a character sequence must be to be recognized. Click “Tolerate Unicode characters” in order to force the algorithm to accept zero bytes between two characters.

**Continue Global Search:** This command is used to continue a global search operation (i.e. a search operation applied to all opened files) in the next file.

**Continue Search:** Lets you continue a search operation in the current file at the current position.

## 6.4 Position Menu

**Go To Offset:** Moves the current position to the specified offset. Normally this is done relative to the beginning of the file (offset 0). You can also move the cursor relative to the current position (forward or backward) or from the end of the file (backward). An offset can be specified in bytes (default), words (2 bytes), doublewords (4 bytes), records (if defined), or sectors. Press **F11** to repeat the last position movement.

**Go To Page/Sector:** Browses to the specified page, sector, or cluster. Please note that the data area on FAT drives starts with cluster #2.

**Go To FAT Entry/FILE Record:** Jump to a certain entry in the file allocation table on a FAT drive or to a certain FILE record in the master file table on an NTFS drive, respectively.

**Move Block:** Moves the current block *selection* (not the data within the block) forward or

backward. Specify the distance in bytes. Press **ALT+F11** to repeat the last block movement, press **SHIFT+ALT+F11** to reverse the movement. This command may facilitate editing a file that consists of homogeneous records of a fixed length.

WinHex keeps a history of your offset jumps within a document and allows to go **back** and **forward** in the chain later.

### **Go To...**

**Beginning Of File:** Display the first page of the current file and moves the current position to offset 0.

**End Of File:** Displays the last page of the current file and moves the current position to the last byte (offset = file size - 1).

**Beginning Of Block:** Moves the current position to the beginning of the current block.

**End Of Block:** Moves the current position to the end of the current block.

**Mark Position:** Marks the current position and thus enables you to find it again later.

**Delete Marker:** Removes the marker from the screen.

**Go To Marker:** Moves the current position to the marker set by Mark Position.

**Position Manager:** see below

## **6.5 View Menu**

**Text Display Only:** Hides the hexadecimal data display and uses the full width of the editor window for the ASCII text display.

**Hex Display Only:** Hides the ASCII text display and uses the full width of the editor window for the hexadecimal data display.

**Record Presentation:** When editing subsequent data records of the same size (for instance, table entries of a database) you may now have WinHex display every other record with a different background color, as a kind of visual aid. The color can be selected in the General Options dialog. Also, WinHex offers to display the current record number and the offset within that record (relative offset) in the status bar, based the record size and the offset of the first record as specified.

If any of the two record features is enabled, the Go To Offset command allows moving the current position in units of the current record size.

**Show:** The Case Data window is part of the forensic user interface of WinHex (X-Ways

Forensics). The **directory browser** is available for logical drives/partitions opened with the disk editor. Cluster lists can optionally be displayed for any file or folder that you double-click in the directory browser. The **Data Interpreter** is a small window that provides "translation services" for the data at the current cursor position. The **toolbar** is displayed optionally, too. A **tab control** makes each edit window accessible with a single mouse click only. The **details panel** provides in-depth information on any open object (file, disk, RAM).

## Template Manager

**Tables:** Provides four conversion tables (cf. ANSI ASCII/IBM ASCII).

## Lines & Columns

**Synchronize Scrolling:** Synchronizes up to four tiled windows on identical absolute offsets. Hold the Shift key when enabling this feature to tile the windows horizontally instead of vertically.

**Synchronize & Compare:** Synchronizes up to four windows and visually displays byte value differences. If no more than two windows are involved, WinHex maintains the initial distance between the offsets of the first shown byte in these windows when scrolling. Not synchronizing on absolute offsets is useful for example when comparing two copies of the file allocation table, which are obviously at different offsets. You may skip to the next or to the previous byte value difference by clicking the extra buttons that are provided in one of the two edit windows.

**Refresh View:** Redraws the contents of the current edit window. In case the current file was updated by an external program, WinHex offers to dismiss any changes made in WinHex and reload the file from scratch.

## 6.6 Tools Menu

**Open Disk:** See chapter "Disk Editor".

**Clone Disk:** See chapter "Disk Cloning".

**File Recovery:** See Appendix D.

**Initialize Free Space:** Confidential information is possibly stored in currently unused parts of a drive as a result of normal delete, copy and save actions. Free space on a drive can be initialized for security reasons. This effectively overwrites all data in unused parts of the disk and makes it impossible to recover this data. *Available in WinHex only, not in X-Ways Forensics.*

**Initialize Slack Space:** Overwrites slack space (the unused bytes in the respective last clusters of all cluster chains, beyond the actual end of a file) with zero bytes. This may be used in addition to "Initialize Free Space" to securely wipe confidential data on a drive or to minimize the space a compressed disk backup (like a WinHex backup) requires. Close any running or resident program

that may write to the disk prior to using this command. *Available in WinHex only, not in X-Ways Forensics.*

**Initialize MFT Records:** On NTFS drives, WinHex can clear all currently unused \$Mft (Master File Table) file records, as they may still contain names and fragments of files previously stored in them. *Available in WinHex only, not in X-Ways Forensics.*

**Scan Cluster Chains Again:** Available for partitions with one of the supported file systems. WinHex traverses all cluster chains and thereby generates a drive map. This enables WinHex to display for each sector which file or directory it is allocated to. The complete listing of deleted files on NTFS drives in the directory browser depends on this, too. It is recommended to invoke this command again after file operations on a drive to keep the information displayed by WinHex up to date. Cf. security options.

**Scan For Lost Partitions:** Formerly existing hard disk partitions that were not automatically found when opening a physical hard disk (or an image of a physical hard disk) and are not listed in the Access button menu may be found and properly identified with this command. This command searches for a master boot record and boot sector signature (0x55AA), optionally only from the first sector that follows the last (location-wise) partition that was already found, and lists newly found partitions in the Access button menu.

**Interpret as Partition Start:** When you find the start sector of a volume (e.g. lost partition) on a physical disk, this menu command allows you to make such a partition easily accessible via the Access button menu. If no known file system is detected starting at the currently displayed sector, you will be asked for the number of sectors that you wish to include in the newly defined partition.

**Set Disk Parameters:** Using this command on a physical disk, you may override the number of cylinders, heads, and sectors per track as recognized by WinHex. This can be useful to access surplus sectors at the end of the disk (in case they were not detected by WinHex), or to adjust the CHS coordinate system to your needs. Use this command on a logical drive to override the total number of clusters WinHex detects on that drive. This can prove useful when examining huge DVDs, which are detected as 2 GB media under Windows 9x.

**Open RAM:** See chapter “RAM Editor”.

**View:** Available only with a forensic license. Invokes the internal viewer.

**External Viewer:** Invokes external file viewing programs such as Quick View Plus etc., as selected in the Options menu, and opens the current file.

**Invoke X-Ways Trace:** Available only if X-Ways Trace is installed. This software can analyze the Internet Explorer's index.dat history file and the Windows recycle bin's info2 files.

**Calculator:** Runs the Windows calculator “calc.exe”. Switching to scientific mode is highly recommended.

**Hex Converter:** Enables you to convert hexadecimal numbers into decimal numbers and vice versa. Simply type in the number and press **ENTER**.

**Tables:** Provides four conversion tables (cf. ANSI-IBM-ASCII).

**Analyze Block/File:** Scans the data within the current block/the entire file and counts the occurrences of each byte value (0...255). The result is graphically displayed by proportional vertical lines. The number of occurrences and the percentage are displayed for each byte value when moving the mouse over the corresponding vertical line.

Use this command for instance to identify data of unknown type. Audio data, compressed data, executable code etc. produce characteristic graphics. The 32-bit standard checksum and the CRC32 of the selected data are also displayed.

Use the context menu of the window to switch zero byte consideration on or off, to print the analysis window, or to export the analysis to a text file.

**Calculate Hash:** Calculates one of the following checksums/digest of the entire current file, disks, or the currently selected block: 8-bit, 16-bit, 32-bit, 64-bit checksum, CRC16, CRC32, MD5, SHA-1, SHA-256, or PSCHF.

## 6.7 File Tools

**Concatenate:** Select several source files that are to be copied into one destination file. The source files are not affected.

**Split:** This command creates several destination files using the contents of a single source file. Specify a split offset for each destination file. The source file is not affected by this function.

**Unify:** Select two source files and one destination file. The bytes/words from the source files will be written alternately into the destination file. The first byte/word originates from the source file that was specified first. Use this function to create a file with odd and even bytes/words originating from separate files (e.g. in EPROM programming).

**Dissect:** Select a source file and two destination files. The bytes/words from the source files will be written alternately into the destination files. The first byte/word will be transferred to the destination file that was specified first. Use this function to create two separate files each containing either the odd or the even bytes/words of the original file (e.g. in EPROM programming).

**Compare:** This command is used to compare two edit windows (files or disks) byte by byte. Decide whether different or identical bytes shall be reported. You may indicate how many bytes to compare. If desired, the operation can abort automatically after having found a certain number of differences or identical bytes. The report is stored as a text file, whose size might otherwise grow dramatically.

The comparison starts at the respective offsets specified for each edit window. These offsets may

differ, such that e.g. the byte at offset 0 in file A is compared to the byte at offset 32 in file B, the byte at offset 1 with the one at offset 33, etc. When you select an edit window for comparison, the current cursor position will automatically be entered in the “From offset” box.

There is yet another compare function in WinHex: you may also compare edit windows visually and synchronize scrolling in these windows (see View menu).

**Wipe Securely:** This command is used to erase the contents of one or more files irrevocably, such that they cannot be restored by WinHex or other special data recover software. Each selected file is overwritten according to the current settings, shortened to a length of zero and then deleted. In the full version of WinHex the name entry of the file is erased as well. Even professional attempts to restore the file will be futile. Therefore this command should be applied to files with confidential contents, which is to be destroyed. *Available in WinHex only, not in X-Ways Forensics.*

## 6.8 Specialist Menu

*Specialist and forensic licenses only.*

**Simultaneous Search:** A parallel search facility, that lets you specify a virtually unlimited list of search terms, one per line (physical search). The search terms are either text strings or hex values (specified with a 0x prefix). They are searched simultaneously, and their occurrences can be archived in the Position Manager. WinHex will save the offset of each occurrence, the search term, the name of the file or disk searched, and in the case of a logical drive the cluster allocation as well (i.e. the name and path of the file that is stored at that particular offset, if any).

That means e.g. a forensic examiner is now able to systematically search through multiple hard drives and disk images in a single pass for words like

- drug
- cocaine
- (street synonym #1 for cocaine)
- (street synonym #2 for cocaine)
- (street synonym #3 for cocaine)
- (street synonym #3 for cocaine, alternative spelling)
- (name of dealer #1)
- (name of dealer #2)
- (name of dealer #3)

at the same time. When searching a logical drive, this will narrow down the examination to a list of files upon which to focus. If you do not have WinHex archive the occurrences, you may use the F3 key to continue the search.

**Create Drive Contents Table:** see separate chapter

**Create Directory Contents Table:** Works like Create Drive Contents Table, but for a user-selected directory and its subdirectories only. You will find this command only in the directory browser's context menu, when right-clicking a directory.

**Media Details Report:** Shows information about the currently active disk or file and lets you copy it e.g. into a report you are writing. Most extensive on physical hard disks, where details for each partition and even unallocated gaps between existing partitions are pointed out. Under Windows 2000 and XP, WinHex also reports the password protection status of ATA disks.

Forensic license only: WinHex is able to detect hidden host-protected areas (HPA, a.k.a. ATA-protected areas) on IDE hard disks under Windows 2000/XP. A message box with a warning will be displayed in case a HPA is found. At any rate, the real total number of sectors according to ATA, if it can be determined, is listed in the details report.

**Interpret Image File As Disk:** Treats a currently open and active disk image file as either a logical drive or physical disk. This is useful if you wish to closely examine the file system structure of a disk image, extract files, etc. without copying it back to a disk. If interpreted as a physical disk, WinHex can access and open the partitions contained in the image individually as known from “real” physical hard disks.

WinHex is even able to interpret *spanned* raw image files, that is, image files that consist of separate segments of any size. For WinHex to detect a spanned image file, the first segment may have an arbitrary name and a non-numeric extension or the extension “.001”. The second segment must have the same base name, but the extension “.002”, the third segment “.003”, and so forth. Both the Create Disk Image command and the DOS cloning tool X-Ways Replica are able to image disks and produce canonically named file segments. Image segmentation is useful because the maximum file size supported FAT file systems is limited.

In some rare cases WinHex may be unable to correctly determine whether the first sector in an image is the sector that contains a master boot record or already a boot sector, and consequently interprets the image structure in a wrong way. If so, hold the Shift key when invoking this command. That way WinHex will ask you and not decide on its own.

With a *forensic* license, WinHex can also interpret evidence files (.e01 images), which can be created with the Create Disk Image command.

**Assemble RAID System:** WinHex can internally destripe RAID 0 systems consisting of up to 5 components (physical hard disks or images). That way it is not necessary to use scripts that unstripe and export RAID systems to a new image, saving you time and drive space. Make sure the components are already open when you use this function. You need to select the components in the right order. WinHex lets you specify the strip size in sectors (often 128) and different RAID header sizes per component (often simply 0). You can usually tell that either the component order, the stripe size or the RAID header size is incorrect when no partitions are detected or partitions with unknown file systems or with file systems that cannot be interpreted properly.

When you add an assembled RAID system to a case (and optionally partitions opened from such a RAID system), the selected RAID configuration parameters are saved with the evidence objects, which allows to access the RAID system instantly in later sessions (forensic licenses only).

**Gather Free Space:** Traverses the currently open logical drive and gathers all unused clusters in a destination file you specify. Useful to examine data fragments from previously existing files that have not been deleted securely. Does not alter the source drive in any way. The destination file must reside on another drive.

**Gather Slack Space:** Collects slack space (the unused bytes in the respective last clusters of all cluster chains, beyond the actual end of a file) in a destination file. Otherwise similar to Gather Free Space. WinHex cannot access slack space of files that are compressed or encrypted at the file system level.

**Gather Inter-Partition Space:** Captures all space on a physical hard disk that does not belong to any partition in a destination file, for quick inspection to find out if something is hidden there or left from a prior partitioning.

**Gather Text:** Recognizes text according to the parameters you specify and captures all occurrences from a file, a disk, or a memory range in a file. This kind of filter is useful to considerably reduce the amount of data to handle e.g. if a computer forensics specialist is looking for leads in the form of text, such as e-mail messages, documents, etc. The target file can easily be split at a user-defined size. This function can also be applied to a file with collected slack space or free space, or to damaged files in a proprietary format than can no longer be opened by their native applications, like MS Word, to recover at least unformatted text.

**Bates-Number Files:** Bates-numbers all the files within a given folder and its subfolders for discovery or evidentiary use. A constant prefix (up to 13 characters long) and a unique serial number are inserted between the filename and the extension in a way attorneys traditionally label paper documents for later accurate identification and reference.

**Trusted Download:** Solves a security problem. When transferring unclassified material from a classified hard disk drive to unclassified media, you need to be certain that it will have no extraneous information in any cluster or sector "overhang" spuriously copied along with the actual file, since this slack space may still contain classified material from a time when it was allocated to a different file. This command copies file in their current size, and no byte more. It does not copy entire sectors or clusters, as conventional copy commands do. Multiple files in the same folder can be copied at the same time.

**Highlight Free Space/Slack Space:** Displays offsets and data in softer colors (light blue and gray, respectively). Helps to easily identify these special drive areas. Works on FAT, NTFS, and Ext2/Ext3 partitions.

## 6.9 Options Menu

**General Options:** see below

**External Programs:** Here you can specify what external file viewing programs you would like to invoke from inside WinHex using the Tools menu. Also the installation path of the viewer component that is included in forensic licenses for v12.05 and later can be specified here (by default: subdirectory ..\viewer). The viewer component can also be specifically enabled or disabled.

**Undo Options:** see below

**Security Options:** see below

**Data Interpreter Options:** cf. Data Interpreter

**Edit Mode:** Allows you to select the edit mode globally. (The details panel's context menu allows to select the edit mode specifically for an active edit window.)

**Simplified user interface:** Available when operated with a forensic license. Replaces the standard File and Edit menu with the menu that is otherwise located in the Case Data window. Like this, some menu items in the standard File menu are still available via the toolbar, and Edit menu is still available as the context menu.

**Character Set:** Lets you switch between the ANSI ASCII, IBM ASCII, and EBCDIC character for display and keyboard input. You may also use **SHIFT+F7**. EBCDIC (originating from IBM mainframes) is currently not supported by the Print command.

## 6.10 Window Menu

**Window Manager:** Displays all windows and provides “instant window switching” functionality. You may also close windows and save files.

**Save Arrangement As Project:** Writes the current window constellation into a project file. From the Start Center you will then be able to load the project and restore editing positions in each document at any time, to conveniently continue your work right where you left it or to begin your work in case of a recurring task.

**Close All:** Closes all windows and thus all open files, disks and RAM sections.

**Close All Without Prompting:** Closes all windows and thus all opened files and disks without giving you the opportunity to save your modifications.

**Cascade/Tile:** Arranges the windows in the aforementioned way.

**Minimize All:** Minimizes all windows.

**Arrange Icons:** This command arranges minimized windows.

## 6.11 Help Menu

**Contents:** Displays the contents of the program help.

**Setup:** Lets you switch between the English, the German, the French, the Spanish, the Portuguese, and the Italian user interface.

**Initialize:** Use this command to restore the default settings of this program.

**Uninstall:** Use this command to remove WinHex from your system. This works properly even if you did not install WinHex using the setup program.

**Online:** Opens the WinHex homepage, the support forum, the Knowledge Base, or the newsletter subscription page in your browser.

**About WinHex:** Displays information about WinHex (the program version, your license status, and more).

## 6.12 Windows Context Menu

The Windows shell displays the context menu when the user clicks an object with the right mouse button. WinHex is present in the context menu only if you enable the corresponding option (see “General Options”).

**Edit with WinHex:** Opens the selected file in WinHex.

**Open in WinHex:** Lets you open all files of the selected folder in WinHex, just like the Open Folder command of the File menu.

**Edit Disk:** Opens the selected disk in the disk editor of WinHex. If you hold the **SHIFT** key, instead of the selected logical drive the corresponding physical disk is opened, if any. (The latter feature is not available under Windows NT.)

WinHex provides its own context menus on the status bar, the Data Interpreter, and in the Position Manager.

# 7 Options

## 7.1 General Options

### 1st column:

- At startup, WinHex can optionally **show** the **Start Center** or **restore** the **last window arrangement** (all windows with their sizes and the positions as you left them in the precedent WinHex session).
- Specify the number of **recently** opened **documents** to remember and to **list** in the Start Center (255 at max.). Up to 9 of them are also listed at the end of the File menu.

- In addition to *which documents* (file or disk) you had recently opened, WinHex can optionally also **remember** the last editing **positions** and the block (if defined).
- You may have **WinHex** appear in the Windows **context menu**. The shell displays the context menu when the user clicks an object with the right mouse button. WinHex provides menu items for files, folders and disks. If this option is not fully selected, there is no menu item for files.
- The option **Allow multiple program instances** lets you execute WinHex more than once at a time. If it is not enabled, WinHex puts the main window of the running instance into the foreground instead of creating a new program instance.
- **Do not update file time** means that WinHex will not change the last write time when a modified file is saved.
- If **Check for surplus sectors** is disabled, WinHex will not try to access surplus sectors when a physical hard disk is opened. When additional sectors are detected, WinHex will remember them the next time you open the disk. You may enforce a new check by holding the Shift key while opening the disk. Checking for surplus sectors may cause very long delays, strange behavior or even damage to the Windows installation on *some very few* systems. Only under Windows XP surplus sectors are included automatically, which renders this option obsolete.
- If **Auto-detect deleted partitions is enabled**, WinHex tries to identify obvious deleted partitions automatically in gaps between existing partitions and in unpartitioned space directly following the last partition, when opening physical hard disks. Such additionally detected partitions will be listed in the Access button menu and marked as deleted. Please note that deleted partitions detected in gaps between existing partitions cause the partition numbering to be changed. E.g. an existing partition #3 might become partition #4 if a deleted partition is detected on the disk before it.
- The **alternative access method** for optical discs and physical hard disks under Windows NT/2000/XP may allow to access hard disks formatted with an unconventional sector size or other media that cannot be accessed otherwise, plus some sectors at the end of CD-ROMs/DVDs that may otherwise be missed. Note that it is slower than the regular access method.
- By default, **edit windows** are not **opened** in a **maximized** state.
- On a right click, **WinHex** can bring up a special **context menu**, the regular edit menu, or define the end of the current block. If this option is disabled, you can still bring up the context menu if you hold the Shift key while right-clicking.
- If you select **Show file icons**, the icons stored in a file are shown in the details panel. If a file contains no icons, the icon of the file *type* is shown if this option is “fully” selected.

- The **ENTER** key can be used to enter up to four two-digit hex values. A useful example is **0x0D0A**, which is interpreted as an end-of-line marker in the Windows world (Unix: 0x0D). The Start Center could then still be opened using **SHIFT+ENTER**.
- Decide whether you want to use the **TAB** key to switch from text to hexadecimal mode and vice versa or to enter the TAB character (0x09). In any case, **TAB+SHIFT** can be pressed to switch the current mode.

### 2nd column:

- Specify the **folder** in which to create **temporary files**.
- Specify the **folder** in which to create **backup files** (.whx).
- Specify the **folder** in which to create **project, script, and case files**.
- Specify the **folder** in which to maintain the **internal hash database**.
- If the creation of thumbnails for **pictures within** large solid RAR **archives** for **gallery** view is too slow, you may want to disable it.
- You may specify your **preferred thumbnail size** in pixels. WinHex will decrease the size automatically if needed to ensure that at least as many files are displayed in the gallery view as are displayed in the currently visible section of the directory browser.
- When gallery view is enabled, WinHex can optionally continue **loading thumbnails in the background** when the current view is full, if the number of files in the current directory is not too big.

### 3rd column:

- **Offsets** can be presented and prompted for in a decimal or **hexadecimal** notation. This setting is valid for the entire program.
- When using the RAM editor it may be reasonable to have WinHex display **virtual addresses** instead of zero-based offsets. This is always done in hexadecimal notation. The dialog window of the Goto Offset command will also prompt for virtual addresses.
- **Page** and sector **separators** may be **displayed**. If this option is enabled partially, only sector separators are displayed.
- Specify the number of **bytes per line** in an edit window. Common values are 16 or 32 (depending on the screen resolution).

- Choose how many **bytes** shall be displayed in a **group**. Powers of 2 serve best for most purposes.
- Specify how many **lines to scroll** when **rolling** the mouse **wheel** (if available).
- Use **Windows default colors** should be self-explanatory.
- Select a **color** used as the **background** of the current **block**. You can only change the color if the option “Use Windows default colors” is switched off.
- Select a **color** used as the **background** of every other fixed-length **record**, if record presentation is enabled (see Position menu).
- Select the default **color** for newly created **annotations**/positions/bookmarks.
- You may want WinHex to **hilite modified bytes**, i.e. display altered parts of a file, disk, or memory in a different color, so you can distinguish between original data and changes you have made so far. You may select the hilite color.
- You may choose a **font** for ANSI ASCII mode. The WinHex font implements the full Windows character set (even characters such as the <sup>TM</sup> and € symbols and diverse quotation marks).
- **Display Windows-styled progress bar** replaces the WinHex progress bar with the typical progress bar common to most Windows programs.
- Last not least, you may select one of several different **dialog window** and **button styles**.

Factory settings of *all* options can be restored using the Initialize command of the Help menu.

## 7.2 Directory Browser Options

- **Grouping files and directories** in the directory browser is optional.
- **Grouping existing and deleted** items in the directory browser is optional.
- The **directory browser** can optionally be displayed with a **grid**.
- \$EFS attributes and non-directory INDX streams may optionally be included in newly created contents tables for NTFS partitions.
- By default, files **recovered** via the directory browser are recreated in the output folder **including** their original **path**.

- Files can optionally be **opened** and **searched** via the directory browser including their slack (available for FAT, NTFS, Ext2/Ext3, and ReiserFS).
- **Listing deleted files and directories** in the directory browser is optional.
- In **Ext2/Ext3** file systems, **deleted files** of which only their names is still known (but no data, size, or dates), are listed optionally.
- Various columns in the directory browser are optional. They are displayed if they have a non-zero column width or hidden if their width is zero.

## 7.3 Undo Options

The availability of the “Undo” command depends on the following options:

- Specify how many sequential actions are to be reversed by the Undo command. This option does not affect the number of reversible keyboard inputs, which is only limited by the available RAM.
- In order to save time and space on your hard disk, you can specify a file size limit. If a file is larger than this limit, backups will not be created and the Undo command is not available except for keyboard input.
- Automatically created backups for the internal use with the Undo command are deleted by WinHex when closing the file, if the corresponding option is fully selected. If it is partially selected, they are deleted when WinHex terminates.
- For all kinds of editing operations you choose whether they should be reversible or not. If so, an internal backup is created before the operation takes place.

## 7.4 Security & Safety Options

The **Sector reading cache** accelerates sequential disk access by the disk editor. This option is recommended particularly when scrolling through CD-ROM and floppy disk sectors, since the number of necessary physical accesses is significantly reduced.

Enabling the option **Inspect clusters automatically** causes WinHex to automatically traverse the cluster chains of a FAT, NTFS, Ext2/3, and CDFS drive if such a drive is opened in WinHex and the required drive map does not yet exist. Using the drive map, WinHex is able to display each sector’s and cluster’s allocation (for storing which file it is used). Use the command “Inspect clusters” of the Tools menu to update the drive map.

With the option **Keep drive map for next session** enabled, all information on FAT, NTFS, Ext2/3, and CDFS drives collected by WinHex remains in the folder for temporary files even when WinHex terminates. WinHex can reuse drive maps during later program runs.

Use the option **Check for virtual memory** alteration to make sure the RAM editor inspects the structure of virtual memory every time before *reading* from or *writing* to it. If the structure has changed, a possible read error is prevented. Especially under Windows NT the checking may result in a loss of speed. When editing the “entire memory” of a process, WinHex generally *never* checks for alterations before reading, even if this option is enabled.

A **hash** can be **calculated automatically** for each file when opening it. It is then displayed in the details panel. Checksums and digests can also be calculated using the Tools menu.

Before modifications to an existing file are saved (i.e. before the **file is updated**), you are prompted for **confirmation**. To inhibit this behavior of WinHex, switch off the corresponding option.

When manually restoring backups, a **restoration report** is **shown** only in case the backup contains a digest or the backup is corrupt. Optionally, you may have WinHex always display the report always after restoration. In this case even the digest will be displayed.

The **key** that is required for encryption and decryption can be entered in a normal edit box. Optionally, you **enter** it **blindly** (asterisks are displayed instead of the actual characters). In this case you have to confirm the key in a second edit box to detect typos.

By default, the **key** is **kept in main memory** (in an encrypted state) as long as WinHex is running, so that you do not have to type it again and again if you use it several times. Possibly you prefer WinHex to erase the key after use.

Decide whether or not WinHex shall **prompt before executing a script**, or only before executing a script via the command line.

## 7.5 Search Options

**Case sensitive:** If this option is enabled, WinHex distinguishes between upper and lower case, e.g. so that “Option” is not found in the word “optionally”.

**Unicode character set:** The specified text is searched using the 256 ANSI-ASCII-equivalent Unicode characters, where the high-order byte is 0. The simultaneous search allows to search for the same text at the same time in Unicode and ASCII. For this to work, the checkbox needs to be “half” checked.

You may specify a **wildcard** (one character or a two-digit hex value), which represents one byte. For example this option can be used to find “Speck” as well as “Spock” when searching for “Sp?ck” with the question mark as the wildcard.

**Only whole words:** The searched string is recognized only, if it is separated from other words, e.g. by punctuation marks or blanks. If this option is enabled, “tomato” is not found in “automaton”.

**Search direction:** Decide whether WinHex shall search from the beginning to the end, or downwards or upwards from the current position.

**Condition: Offset modulo  $x = y$ :** The search algorithm accepts search string occurrences only at offsets that meet the given requirements. E.g. if you search for data that typically occurs at the 10<sup>th</sup> byte of a hard disk sector, you may specify  $x=512$ ,  $y=10$ . If you are looking for DWORD-aligned data, you may use  $x=4$ ,  $y=0$  to narrow down the number of hits.

**Search in block only:** The search operation is limited to the current block.

**Search in all open windows:** The search operation is applied to all open edit windows. Press F4 to continue the search in the next window. If “Search in block only” is enabled at the same time, the search operation is limited to the current block in each window.

**Count occurrences/Save occurrence positions:** Forces WinHex not to show each single occurrence, but to count them. If this option is fully enabled, WinHex will enter all occurrences into the Position Manager.

**Search for “non-matches”:** In “Find Hex Values” you may specify a single hex value with an exclamation mark as a prefix (e.g. !00) to make WinHex stop when it encounters the first byte value that *differs*.

## 7.6 Replace Options

**Prompt when found:** WinHex awaits your decision when an occurrence has been found. You may either replace it, continue or abort the search.

**Replace all occurrences:** All occurrences are replaced automatically.

**Case sensitive:** The characters that are to be replaced are searched using this option (cf. Search Options).

**Unicode character set:** The specified characters are searched and replaced in Unicode format (cf. Search Options).

You may specify one character or a two-digit hex value as a **wildcard**. This is usually done in the search string. If the *substitute* contains a wildcard, the character at the corresponding position in an occurrence will not be changed. Thus, “black” and “block” can be replaced simultaneously with “crack” and “crock” (enter “bl?ck” and “cr?ck”).

**Only whole words:** The searched string is recognized only if it is separated from other words e.g. by punctuation marks or blanks. If this option is enabled, “tomato” is not replaced in “automaton”.

**Search direction:** Decide whether WinHex shall replace from the beginning to the end, or downwards or upwards from the current position.

**Replace in block only:** The replace operation is limited to the current block.

**Replace in all opened files:** The replace operation is applied to all files not opened in view mode. If “Replace in block only” is enabled at the same time, the replace operation is limited to the current block of each file.

Hint:

WinHex is able to replace one string or hex value sequence with another one that has a different length. You will be prompted, which of the following methods shall be applied:

1st method: The data behind the occurrence is moved due to length difference. So the file size is changed. This method must not be applied to certain file types, such as executable files. It is even possible to specify nothing as the substitute, which means all occurrences will be removed from the file!

2nd method: The substitute is written into the file at the position of the occurrence. If the substitute is shorter than the searched character sequence, the exceeding characters will remain in the file. Otherwise even the bytes behind the occurrence will be overwritten (as far as the end of the file is not reached). The file size is not affected.

## 8 Miscellaneous

### 8.1 Block

You can mark a part of an open file as a “block”. This part can be manipulated by several function in the edit menu just as selections in other Windows programs. If no block is defined, these functions usually are applied to the whole file.

The current position and size of the block are displayed in the status bar. Double-clicking the right mouse button or pressing the **ESC** key clears the block.

### 8.2 Modify Data

Use this command to modify the data within the block or within the whole file, in case no block

is defined. In this version of WinHex, four types of data modifications are available. Either a fixed integer number is added to each element of the data, the bits are inverted, a constant is XORed with the data (a simple kind of encryption), ORed, or ANDed, bits are shifted logically, or bytes are swapped. By shifting bits, you can simulate inserting or removing single bits at the beginning of the block. You may also shift entire *bytes* (currently to the left only, by entering a negative number of bytes). This is useful if you wish to cut bytes from a very huge file in in-place mode, which would otherwise require the creation of a huge temporary file.

### Swap Bytes

This command assumes all data to consist of 16-bit elements (32-bit elements resp.) and swaps high-order and low-order bytes (and high-order and low-order words resp.). Use it in order to convert big-endian into little-endian data and vice versa.

### Addition

Specify a positive or negative, decimal or hexadecimal number, which is to be added to each element of the current block. An integer format defines size (1, 2 or 4 bytes) and type (signed or unsigned) of an element.

There are two ways how to proceed if the result of the addition is out of the range of the selected integer format. Either the range limit is assumed to be the new value (I) or the carry is ignored (II).

Example: unsigned 8-bit format

- I.     FF + 1 → FF   (255 + 1 → 255)
- II.    FF + 1 → 00   (255 + 1 → 0)

Example: signed 8-bit format

- I.     80 - 1 → 80   (-128 - 1 → -128)
- II.    80 - 1 → 7F   (-128 - 1 → +127)

- If you decide to use the first method, WinHex will tell you, how often the range limit has been exceeded.
- The second method makes sure the operation is reversible. Simply add -x instead of x based on the same integer format to recreate the original data.
- When using the second method it does not make a difference whether you choose a signed or an unsigned format.

## 8.3 Conversions

WinHex provides the Convert command of the Edit menu for easy conversions of different data formats and for encryption and decryption. The conversion can optionally be applied to all

opened files instead of only the currently displayed one. The formats marked with an asterisk (\*) can only be converted as a whole file, not as a block. The following formats are supported:

- ANSI ASCII, IBM ASCII (two different ASCII character sets)
- EBCDIC (an IBM mainframe character set)
- Lowercase/uppercase characters (ANSI ASCII)
- Binary\* (raw data)
- Hex ASCII\* (hexadecimal representation of raw data as ASCII text)
- Intel Hex\* (=Extended Intellec; hex ASCII data in a special format, incl. checksums etc.)
- Motorola S\* (=Extended Exorcisor; ditto)
- Base64\*
- UUCode\*

Please note:

- When converting Intel Hex or Motorola S data, the internal checksums of these formats are not checked.
- Depending on the file size, the smallest possible output subformat is chosen automatically. Intel Hex: 20-bit or 32-bit. Motorola S: S1, S2, or S3.
- When converting from binary to Intel Hex or Motorola S, only memory regions not filled with hexadecimal FFs are translated, to keep the resulting file compact.

The Convert command can also decompress raw data from any number of complete 16-cluster units compressed by the NTFS file system.

### Encryption/Decryption

Specify a string consisting of 1-16 characters as the encryption/decryption key. The more characters you enter, the safer is the encryption. The key itself is not used for encryption and decryption, instead it is digested to the actual key. The key is not saved on your hard disk. If the corresponding security option is enabled, the key is stored in an encrypted state in the RAM as long as WinHex is running.

It is recommended to specify a combination of at least 8 characters as the encryption key. Do not use words of any language, it is better to choose a random combination of letters, punctuation marks, and digits. Note that encryption keys are case sensitive. Remember that you will be unable to retrieve the encrypted data without the appropriate key. The decryption key you enter is not verified before decrypting.

The encryption algorithm is “Pukall Cipher 1” (PC 1), using a 128-bit key (=the 128-bit digest of the key you specify).

## **8.4 Wiping and Initializing**

For securely erasing (shredding) data, and also simply for filling files or disk sectors with certain byte values, WinHex offers the following options:

**Fill with hex values:** Specify either 1, 2, 3, 4, 5, 6, 12, 15, or 16 two-character hex values, which will be copied repeatedly into the current block, the entire file or all disk sectors, respectively.

**Fill with random bytes:** Specify a decimal interval (0 to 255 at max.) for random numbers, which will be copied repeatedly into the current block, the entire file or all disk sectors, respectively. The random bytes are Laplace-distributed.

In case in all open files *either a block or no block is defined*, this command can optionally be applied to all these files at the same time.

To maximize security, if you wish to totally wipe (sanitize) slack space, free space, unused NTFS records, or an entire media, you may want to apply more than one pass for overwriting disk space (up to three).

According to the Clearing and Sanitization Matrix, the standard outlined in the U.S. Department of Defense (DoD) 5220.22-M operating manual, method "c", a hard disk or floppy disk can be cleared by overwriting (once) all addressable locations with a single character. This is usually the hexadecimal value 0x00, but can be any other value. To sanitize hard disks according to method "d", overwrite all addressable locations with a character, its complement, then a random character, and verify. (This method is not approved by the DoD for sanitizing media that contain top secret information.)

The "DoD" button configures WinHex for sanitization, such that it will first overwrite with 0x55 (binary 01010101), then with its complement (0xAA = 10101010), and finally with random byte values.

The "0x00" button configures WinHex for simple initialization, wiping once with zero bytes.

## 8.5 Disk Cloning

The command "Clone Disk" is part of the Tools menu. This function copies a defined number of sectors from a source to a destination disk (or alternatively from a disk image file or to a disk image file). Both disks must have the same sector size. In order to effectively *duplicate* a drive (i.e. in order to copy all sectors of the drive), enable the appropriate option, so the correct number of sectors is entered automatically. The destination disk must not be smaller than the source disk.

Disk cloning offers options that control the behavior when bad sectors are encountered on the source disk:

- By default, you are notified of the error and prompted for either continuing or aborting the operation. "Log procedure silently" creates a complete log file of the entire operation in the folder for temporary files (filename "Cloning Log.txt"), including a report on unreadable sectors (which cannot be copied), and prevents WinHex from reporting each unreadable sector separately. This is useful e.g. for computer forensics.
- WinHex can either leave a destination sector that corresponds to a damaged source sector

unchanged or fill it with an ASCII pattern you specify (e.g. your initials, or something like "BAD "). Leave the pattern edit box blank to fill such sectors with *zero* bytes. BTW, the chosen pattern is also used to display a bad sector's contents in the disk editor.

- Bad sectors often occur in contiguous groups, and each attempt to read a bad sector usually takes a long time. You may have WinHex avoid such damaged disk areas. When a bad sector is encountered, WinHex can skip a number of subsequent sectors you specify (25 by default). This is useful if you wish to accelerate the cloning process and if you do not care about some actually readable sectors not making it to the clone.

Regular disk cloning is not an option if you want to duplicate a disk in a removable drive (e.g. a floppy disk) with only one removable drive present. The correct concept for this application is *disk imaging*, where the data is first stored in an image *file*. The image can then be copied to a different disk. The result is the same as disk cloning.

When you specify a file named "dev-null" as the destination, the data will only be read and not copied anywhere (and you will be warned of this). This is useful if you are interested in the report about bad sectors, but do not wish to actually clone or image a disk.

You may try "simultaneous I/O" if the destination is not the same physical medium as the source. Offers a chance to accelerate the cloning process by up to 30%.

There are two ways to image a disk:

- The disk cloning dialog allows copying sectors from a disk into an *uncompressed, unsplit, raw, headerless image file* and later vice-versa. Combined with the silent "log file" mode, this is preferable to creating an image in case of defective sectors on the source disk.
- For options like compression, hashing, and file splitting, please use the image and backup functionality. For easy recovery, a backup file includes information on its contents: sector numbers, source disk etc.

Cloning or imaging the drive that contains the active Windows installation can produce inconsistent copies. Make sure the source drive is not written to during the cloning/imaging/backup/restoration procedure by any other program or by Windows itself. It is recommended to move the TEMP directory to a different drive. There should no be active swap file on the drive.

Make sure no other program or service can write to the partition you are going to clone. E.g. check for resident defragmentation tools or virus scanners running in the background and deactivate them for the duration of the cloning/backup/restoration. Under Windows NT/2000/XP it is recommended to unmount the partition as a logical drive/drive letter.

After cloning a logical NTFS drive, you may need to reboot your system or run "chkdsk /f" on the target drive in order to see the new contents in Windows because only this will clear all of Windows' internal buffers.

Cloning or imaging with WinHex makes exact sector-wise, forensically sound copies, including all unused space and slack space. WinHex cannot dynamically change partition sizes or adapt to

destination disks larger or smaller than the source disks. This can be done by tools like PartitionMagic.

In order to reduce the space a backup occupies as much as possible, you can initialize unused drive space before making the backup. This is because sectors that consist but of zero values barely increase the backup size when compression is enabled.

## 8.6 Images and Backups

This command “Create Disk Image”/“Make Backup Copy” in the File menu allows to create a backup or image of the currently open logical drive, physical disk, or individual file. There are three possible output file formats, each with unique advantages.

File format:	WinHex Backup	Evidence File	Raw Image
Filename extension:	.whx	.e01	e.g. .dd
Interpretable as disk:	no	yes	yes
Splittable:	yes	yes	yes
Compressible:	yes	yes	no
Encryptable:	yes	no	no
Optional hash:	integrated	integrated	separate
Optional description:	integrated	integrated	no
Range of sectors only:	yes	no	no
Applicable to files:	yes	no	no
Automated maintenance:	Backup Manager	no	no
Compatibility:	no	(yes)	yes
Required license:	none	forensic	personal

The major advantage of evidence files and raw images is that they can be interpreted by WinHex like the original disks (with the command in the Specialist menu). This also makes them suitable for usage as evidence objects in your cases. This holds true for evidence files in particular because they can store an optional description and an integrated hash for later automated verification. Raw images have the benefit that they can be easily exchanged between various forensic tools. All output file formats support splitting into segments of a user-defined size. A segment size of 650 MB e.g. is suitable for archiving on CD-R. Evidence files are *required* to be split at 2025 MB at max.

If you have WinHex assign a filename for a WinHex backup automatically, the file will be created in the folder for backups (cf. General Options), named with the next free "slot" according to the Backup Manager's naming conventions ("xxx.whx"), and will be available in the Backup Manager. If you explicitly specify a path and a filename, you can restore the backup or image later using the Restore Backup command, and in case of split backups WinHex will automatically append the segment number to the filenames.

The encryption algorithm is “Pukall Cipher 1” (PC 1), using a 128-bit key that is digested from the 256-bit concatenation of the 128-bit digest of the key you enter and 128 bits random input. The random input is saved in the .whx file for later decryption.

Compression is based on the “Deflate” compression algorithm that is part of the popular general-purpose library *zlib*. This algorithm consists of LZ77 compression and Huffman coding. The compression ratio is the same as ZIP. The complete documentation of the .whx file format is available from the WinHex homepage at <http://www.x-ways.net/winhex/>.

## 8.7 Backup Manager

Displays a list of previously created WinHex backups. The items can be listed in a chronological or alphabetical order. Choose the backup you would like to restore. When that function completes, the original file or sector contents is shown.

You can restore the backup

- into a temporary file first such that you will still need to save it,
- directly and immediately to the disk, or
- to a new file.

In the case of disk sectors you may also wish to specify a different destination disk or a different destination sector number. It is also possible to only extract a subset of the sectors from the backup. (However, sectors at the beginning of a *compressed* backup cannot be left out during restoration.) If the backup was saved with a checksum and/or a digest, data authenticity is verified before the sectors will be directly written to the disk.

The backup manager also allows to delete backups which you do not need any longer. Backups that were created for internal use by the Undo command can be deleted by WinHex automatically (cf. Undo Options).

Backup files that are maintained by the backup manager are located in the folder specified in the General Options dialog. Their filenames are “xxx.whx” where xxx is a unique three-digit identification number. This number is displayed in the last column of the backup manager list.

## 8.8 Position Manager

The Position Manager maintains a list of file or disk offsets and corresponding descriptions, also called *annotations*, also used for search hits. Navigating from one entry to the next is easy if you press Ctrl+Left and Ctrl+Right. You may enter new positions and edit or delete existing entries. If a special offset in a file is important to you because you, you can add it to the Position Manager. This makes it a lot easier to find it again later, and you do not have to remember it. Descriptions may be up to 8192 characters in size. An appropriate description for instance could be “Data chunk begins here!”. Optionally all positions maintained by the Position Manager can be *highlighted* in the editor window in a unique color you specify, and their descriptions displayed in yellow tooltip windows when the mouse cursor is moved over them. You may also add or edit positions with the context menu of an edit window or by clicking the middle mouse button in an edit window.

Click the right mouse button in order to see a context menu in the Position Manager. The context menu provides additional commands. You may delete, load or save positions, even export the list as HTML. If the position list in the *general* Position Manager was changed, it is saved in the file *WinHex.pos* when exiting WinHex.

The complete documentation of the POS file format is available from the WinHex homepage at <http://www.x-ways.net/winhex/>.

## 8.9 Data Interpreter

The Data Interpreter is a small window that provides “translation services” for the data at the current cursor position. The options dialog lets you specify the data types to interpret. These are various integer data types (by default in decimal notation, optionally hexadecimal or octal), the binary format (8 bits of a byte), four floating-point data types, assembler opcodes (Intel®), and date types.

The Data Interpreter is also capable of translating all data types (except assembler opcodes) back into hex values. Double-click a number in the Data Interpreter window, enter a new value and press **ENTER**. The Data Interpreter will enter the corresponding hex values into the edit window at the current position.

Right-click the data interpreter to bring up a context menu. This will let you switch between big-endian and little-endian translation of integer and floating-point data. You may also choose between decimal, octal, or hexadecimal integer representation. This plus the digit grouping can also be selected in the Data Interpreter Options dialog.

### Hints:

- Some hex values cannot be translated into floating-point numbers. For these hex values the Data Interpreter displays NAN (**not a number**).
- Some hex values cannot be translated into valid dates. The value ranges of different date types are more or less narrow.
- There are redundancies in the Intel® instruction set, which show up in the Data Interpreter as duplication of both hex opcodes and mnemonics. Floating-point instructions are generally displayed as F\*\*\*.
- More detailed reference can be found in the Intel® Architecture Software Developer’s Manual Volume 2: Instruction Set Reference, available in PDF format on the Internet.

# Appendix A: Template Definition

## 1 Header

The header of a template definition has the following format:

```
template "title"  
[description "description"]  
[applies_to (file/disk/RAM)]  
[fixed_start offset]  
[sector-aligned]  
[requires offset "hex values"]  
[big-endian]  
[hexadecimal/octal]  
[read-only]  
[multiple [fixed overall size]]  
// Put any general comments to the template here.  
begin  
    variable declarations  
end
```

Tags in brackets are optional. The order of the tags is irrelevant. Expressions must only be enclosed in inverted commas if they contain space characters. Comments may appear anywhere in a template definition. Characters following a double slash are ignored by the parser.

The keyword `applies_to` must be followed by one and only one of the words `file`, `disk`, or `RAM`. WinHex issues a warning if you are going to use a template on data from a different source.

While by default templates start interpreting the data at the current cursor position when applied, an optional `fixed_start` statement ensures interpretation always starts at the specified absolute offset within the file or disk.

If the template applies to a disk, the keyword `sector-aligned` ensures the template interpretation starts at the beginning of the current sector, regardless of the exact cursor position.

Similar to the `applies_to` statement, the `requires` statement enables WinHex to prevent an erroneous application of a template definition to data that does not match. Specify an offset and a hex-value chain of an arbitrary length that identifies the data for which the template definition was intended. For example, a valid master boot record can be recognized by the hex values `55 AA` at offset `0x1FE`, an executable file by the hex values `4D 5A` (“MZ”) at offset `0x0`. There may be multiple `applies_to` statements in a template definition header, which are all considered.

The keyword `big-endian` causes all multi-byte integer and boolean variables in the template definition to be read and written in big-endian order (high-order byte first).

The keyword `hexadecimal` causes all integer variables in the template definition to be displayed in hexadecimal notation.

The keyword `read-only` ensures that the template can only be used to examine, but not to manipulate data structures. The edit controls within the template will be grayed out.

If the keyword `multiple` is specified in the header, WinHex allows browsing to neighboring data records while displaying the template. This requires that WinHex has knowledge of the record's size. If it is not specified as a parameter to the `multiple` statement, WinHex assumes the overall size of a template structure (=record) to be the current position at the end of the template interpretation less the base editing position. If this is a variable size, i.e. array sizes or move parameters are determined dynamically by the value of variables, WinHex cannot browse to precedent data records.

## 2 Body: Variable Declarations

The body of a template definition mainly consists of variable declarations, similar to those in programming languages. A declaration has the basic form

```
type "title"
```

where `type` can be one of the following:

- `int8, uint8 = byte, int16, uint16, int24, uint24, int32, uint32, int64,`
- `uint_flex`
- `binary,`
- `float = single, real, double, longdouble = extended,`
- `char, char16, string, string16,`
- `zstring, zstring16,`
- `boole8 = boolean, boole16, boole32`
- `hex,`
- `DOSDateTime, FileTime, OLEDateTime, SQLDateTime, UNIXDateTime = time_t, JavaDateTime`

`title` must only be enclosed in inverted commas if it contains space characters. `title` must not consist only of digits. WinHex does not distinguish between upper and lower case characters in titles. 41 characters are used to identify a variable at most.

`type` can be preceded by at most one member of each of the following modifier groups:

```
big-endian          little-endian
hexadecimal         decimal          octal
read-only           read-write
```

These modifiers only affect the immediately following variable. They are redundant if they appear in the header already.

The number at the end of a type name denotes the size of each variable (strings: of each character) in bits. With `char16` and `string16`, WinHex supports Unicode characters and strings. However, Unicode characters other than the first 256 ANSI-equivalent characters are not supported. The maximum string size that can be edited using a template is 8192 bytes.

The types `string`, `string16`, and `hex` require an additional parameter that specifies the number of elements. This parameter may be a constant or a previously declared variable. If it is a constant, it may be specified in hexadecimal format, which is recognized if the number is preceded by `0x`.

You may declare arrays of variables by placing the array size in square brackets next to the type or the title. Specify "unlimited" as the array size to make the template stop only when the end of file is encountered. The following two lines declare a dynamically sized ASCII string, whose length depends on the preceding variable:

```
uint8      "len"  
char[len]  "A string"
```

The same could be achieved by the following two declarations:

```
byte      "len"  
string len "A string"
```

The character “~” can be used as placeholder for later replacement with the actual array element number (see below). This does not apply to arrays of `char` variables, since they are automatically translated into a string.

Numerical parameters of `string`, `string16`, and `hex` variables as well as array size expressions may be specified in mathematical notation. They will be processed by the integrated formula parser. Such expressions need to be enclosed in brackets. They must not contain space characters. They may make use of previously declared integer variables whose names do not contain space characters either. Supported operations are addition (+), subtraction (-), multiplication (\*), integer division (/), modular division (%), bitwise AND (&), bitwise OR (|), and bitwise XOR (^). Valid mathematical expressions are for example  $(5*2+1)$  or  $(len1/(len2+4))$ . The result is always an integer and must be a positive number.

`zstring` and `zstring16` are null-terminated strings whose size is determined dynamically at run-time.

### 3 Body: Advanced Commands

When enclosed in braces, several variable declarations comprise a block that can be used

repeatedly as a whole. Note, however, that blocks must not be *nested* in the current implementation. The character ~ can be used in a variable's name as a placeholder for later replacement with the actual repetition count. The optional `numbering` statement defines where to begin counting (0 by default).

```
numbering 1
{
byte      "len"
string len "String No. ~"
}[10]
```

In this example the actual variable names in the template will be "String No. 1", "String No. 2", ..., "String No. 10". Instead of a constant number of repetitions (10 in this example), you may also specify "unlimited". In that case WinHex will repeat the block until the end of file is encountered. "ExitLoop" can be used to break out of a loop at any time.

"IfEqual" is useful for the comparison of two expressions. Operands can be either both numerical values, be it constant values, integer variables or a formulas, or byte sequences given as text or hex values which are compared byte by byte. ASCII string expressions must be enclosed in quotation marks, hex sequences must be preceded by a "0x" identifier. Formulas need to be enclosed in brackets.

```
{
byte      Value
IfEqual   Value 1
          ExitLoop
EndIf
} [10]
```

An "IfEqual" command block is terminated with an "EndIf" statement. If the compared expressions are equal, template interpretation continues after "IfEqual". Optionally, "IfEqual" can be followed by an "Else" statement. The template processor branches into the "Else" block if the expressions are not equal. "IfEqual" commands must not be nested. "IfGreater" is similar to "IfEqual". The condition is true if the first expression is greater than the second. Strings and hex values are compared lexicographically.

In order to facilitate reading and navigating the template, you may define groups of variables that are separated by empty space in the dialog box:

```
section   "...Section Title..."
...
endsection
```

The `section`, `endsection`, and `numbering` statements do not advance the current position in the data to be interpreted.

There are two commands that do not declare variables either, but are explicitly used to change the current position. This can be done to skip irrelevant data (forward movement) or to be able access certain variables more than once as different types (backward movement). Use the `move n`

statement to skip *n* bytes from the current position, where *n* may be negative. `goto n` browses to the specified absolute position from the beginning of the template interpretation (must be positive).

The following example demonstrates how to access a variable both as a 32-bit integer and as a four-part chain of hex values:

```
int32      "Disk serial number (decimal)"
move -4
hex 4      "Disk serial number (hex)"
```

## 4 Body: Flexible Integer Variables

A special variable type supported by templates is `uint_flex`. This type allows to compose an unsigned integer value from various individual bits within a 32-bit (4-byte) range in an arbitrary order and is even more flexible than a so-called bit field in the C programming language.

`uint_flex` requires an additional parameter string in inverted commas that specifies exactly which bits are used in which order, separated by commas. The bit listed first becomes the most significant bit (high value bit) in the resulting integer, and it is not interpreted as a + or - indicator. The bit listed last becomes the least significant bit in the resulting integer.

The bits are counted starting with 0. Bit 0 is the bit that is the least significant bit of the 1st byte. Bit 31 is the most significant bit of the fourth byte. Thus, the definition is based on little-endian philosophy.

For example,

```
uint_flex "15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0" "Standard 16-bit integer"
```

is exactly the same as `uint16`, the common unsigned 16-bit integer variable.

```
uint_flex "31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0" "Standard 32-bit integer"
```

is exactly the same as `uint32`, the common unsigned 32-bit integer variable.

The benefit of `uint_flex`, though, is that the number, the position, and the usage order of all bits can be chosen arbitrarily. For example,

```
uint_flex "7,15,23,31" "An unusual 4-bit integer"
```

composes a 4-bit integer out of the respective most significant bits of each of the four bytes involved. If these four bytes happen to be

F0 A0 0F 0A =

**1**1110000 **1**0100000 **0**0001111 **0**0001010,

bit 7 is **1**, bit 15 is **1**, bit 23 is **0**, and bit 31 is **0**.

So the resulting `uint_flex` is **1100** = **1**\*8 + **1**\*4 + **0**\*2 + **0**\*1 = 12.

# Appendix B: Script Commands

Script commands are case-*insensitive*. Comments may occur anywhere in a script file and must be preceded by two slashes. Parameters may be 255 characters long at most. Where in doubt because hex values, text strings (or even integer numbers) are accepted as parameters, you may use inverted commas (quotation marks) to enforce the interpretation of a parameter as text. Inverted commas are *required* if a text string or variable name contains one or more space characters, so that all characters between the inverted commas are recognized as constituting *one* parameter.

Wherever numerical parameters are expected (integer numbers), the integrated formula parser allows you to use mathematical expressions. Such expressions need to be enclosed in brackets. They must not contain space characters. They may make use of variables that can be interpreted as integer numbers. Supported operations are addition (+), subtraction (-), multiplication (\*), integer division (/), modular division (%), bitwise AND (&), bitwise OR (|), and bitwise XOR (^). Valid mathematical expressions are for example  $(5*2+1)$ ,  $(MyVar1/(MyVar2+4))$ , or  $(-MyVar)$ .

The following is a description of currently supported script commands, including example parameters.

## **Create "D:\My File.txt" 1000**

Creates the specified file with an initial file size of 1000 bytes. If the file already exists, it is overwritten.

## **Open "D:\My File.txt"**

### **Open "D:\\*.txt"**

Opens the specified file(s). Specify "?" as the parameter to let the user select the file to open.

## **Open C:**

## **Open D:**

Opens the specified logical drive. Specify ":" as the parameter to let the user select a logical drive or physical disk to open.

## **Open 80h**

## **Open 81h**

## **Open 9Eh**

Opens the specified physical media. Floppy disk numbering starts with 00h, fixed and removable drive numbering with 80h, optical media numbering with 9Eh.

Optionally, you may pass a second parameter with the Open command that defines the edit mode in which to open the file or media ("in-place" or "read-only").

## **CreateBackup**

Creates a backup of the active file in its current state.

## **CreateBackupEx 0 100000 650 true "F:\My backup.whx"**

Creates a backup of the active disk, from sector 0 through sector 1,000,000. The backup file will be split automatically at a size of 650 MB. Compression is enabled ("true"). The output file is specified as the last parameter.

If the backup file should not be split, specify 0 as the third parameter. To disable compression, specify "false". To have the Backup Manager automatically assign a filename and place the file in the folder for backup files, specify "" as the last parameter.

### **Goto 0x128**

#### **Goto MyVariable**

Moves the current cursor position to the hexadecimal offset 0x128. Alternatively, an existing variable (up to 8 bytes large) can be interpreted as a numeric value, too.

### **Move -100**

Moves the current cursor position 100 bytes back (decimal).

### **Write "Test"**

#### **Write 0x0D0A**

#### **Write MyVariable**

Writes the four ASCII characters "Test" or the two hexadecimal values "0D0A" at the current position (in overwrite mode) and moves the current position forward accordingly (i.e. by 4 bytes). Can also write the contents of a variable specified as the parameter.

### **Insert "Test"**

Functions just as the "Write" command, but in *insert* mode. Must only be used with *files*.

### **Read MyVariable 10**

Reads the 10 bytes from the current position into a variable named "MyVariable". If this variable does not yet exist, it will be created. Up to 32 different variables allowed. Another way to create a variable is the Assign command.

### **ReadLn MyVariable**

Reads from the current position into a variable named "MyVariable" until the next line break is encountered. If the variable already exists, its size will be adjusted accordingly.

### **Close**

Closes the active window without saving.

### **CloseAll**

Closes all windows without saving.

### **Save**

Saves changes to the file or disk in the active window.

### **SaveAs "C:\New Name.txt"**

Saves the file in the active window under the specified path. Specify "?" as the parameter to let the user select the destination.

**SaveAll**

Saves changes in all windows.

**Terminate**

Aborts script execution.

**Exit**

Terminates script execution and ends WinHex.

**ExitIfNoFilesOpen**

Aborts script execution if no files are already opened in WinHex.

**Block 100 200****Block "My Variable 1" "My Variable 2"**

Defines the block in the active window to run from offset 100 to offset 200 (decimal). Alternatively, existing variables (each up to 8 bytes large) can be interpreted as numeric values.

**Block1 0x100**

Defines the block beginning to be at the hexadecimal offset 0x100. A variable is allowed as the parameter as well.

**Block2 0x200**

Defines the block end to be at the hexadecimal offset 0x200. A variable is allowed as the parameter as well.

**Copy**

Copies the currently defined block into the clipboard. If no block is defined, it works as known from the Copy command in the Edit menu.

**Cut**

Cuts the currently defined block from the file and puts it into the clipboard.

**Remove**

Removes the currently defined block from the file.

**CopyIntoNewFile "D:\New File.dat"****CopyIntoNewFile "D:\File +MyVariable+.dat"**

Copies the currently defined block into the specified new file, without using the clipboard. If no block is defined, it works as known from the Copy command in the Edit menu. Can copy disk sectors as well as files. The new file will not be automatically opened in another edit window. Allows an unlimited number of "+" concatenations in the parameter. A variable name will be interpreted as an integer if not be larger than  $2^{24}$  (~16 Mio.). Useful for loops and file recovery.

**Paste**

Pastes the current clipboard contents at the current position in a file, without changing the current

position.

### **WriteClipboard**

Writes the current clipboard contents at the current position in a file or within disk sectors, without changing the current position, by overwriting the data at the current position.

### **Convert *Param1 Param2***

Converts the data in the active file from one format into another one. Valid parameters are ANSI, IBM, EBCDIC, Binary, HexASCII, IntelHex, MotorolaS, Base64, UUCode, LowerCase, and UpperCase, in combinations as known from the conventional Convert menu command.

### **Encrypt "My Password"**

Encrypts the active file or disk, or selected block thereof, with the specified key (up to 16 characters long) using the PC1 algorithm (128 bit).

### **Decrypt "My Password"**

Decrypts the active file or disk.

### **Find "John" [*MatchCase MatchWord Down Up BlockOnly SaveAllPos Unicode Wildcards*]**

#### **Find 0x1234 [*Down Up BlockOnly SaveAllPos Wildcards*]**

Searches in the active window for the name John or the hexadecimal values 0x1234, respectively, and stops at the first occurrence. Other parameters are optional. By default, WinHex searches the entire file/disk. The optional parameters work as known from usual WinHex search options.

### **ReplaceAll "Jon" "Don" [*MatchCase MatchWord Down Up BlockOnly Unicode Wildcards*]**

#### **ReplaceAll 0x0A 0x0D0A [*Down Up BlockOnly Wildcards*]**

Replaces all occurrences of either a string or hexadecimal values in the active file with something else. Can only be applied to a disk if in in-place mode.

### **IfFound**

A boolean value that depends on whether or not the last Find or ReplaceAll command was successful. Place commands that shall be executed if something was found after the IfFound command.

### **IfEqual MyVariable "Hello World"**

#### **IfEqual 0x12345678 MyVariable**

#### **IfEqual MyVariable 1000**

#### **IfEqual MyVariable MyOtherVariable**

#### **IfEqual MyVariable (10\*MyOtherVariable)**

Compares either two numerical integer values (each of them being a constant value, an integer variable or a mathematical expression) or two variables, ASCII strings, or hexadecimal values at the binary level. Comparing two objects at the binary with a different length always returns False as the result. If equal, the following commands will be executed. If conditions must not be nested.

### **IfGreater MyVariable "Hello World"**

#### **IfGreater 0x12345678 MyVariable**

**IfGreater MyVariable 1000****IfGreater MyVariable MyOtherVariable****IfGreater MyVariable (10\*MyOtherVariable)**

Accepts the same parameters as IfEqual. If the first one is greater than the second one, the following commands will be executed. If conditions must not be nested.

**Else**

May occur after IfFound or IfEqual. Place commands that shall be executed if nothing was found or if the compared objects are not equal after the Else command.

**EndIf**

Ends conditional command execution (after IfFound or IfEqual).

{...

**ExitLoop**

...}

Exits a loop. A loop is defined by braces. Closing braces may be followed by an integer number in square brackets, which determines the number of loops to execute. This is may also be a variable or the keyword "unlimited" (so the loop can only be terminated with an ExitLoop command). Loops must not be nested.

Example of a loop:

{ Write "Loop" }[10] will write the word "Loop" ten times.

**Label ContinueHere**

Creates a label named "ContinueHere"

**JumpTo ContinueHere**

Continues script execution with the command following that label.

**NextObj**

Switches cyclically to the next open window and makes it the "active" window. E.g. if 3 windows are open, and window #3 is active, NextObj will make #1 the active window.

**ForAllObjDo**

The following block of script commands (until **EndDo** occurs) will be applied to all open files and disks.

**CopyFile C:\A.dat D:\B.dat**

Copies the contents of C:\A.dat into the file D:\B.dat.

**MoveFile C:\A.dat D:\B.dat**

Moves the file C:\A.dat to D:\B.dat.

**DeleteFile C:\A.dat**

Surprisingly, deletes C:\A.dat.

**InitFreeSpace****InitSlackSpace**

Clears free space or slack on the current logical drive, respectively, using the currently set initialization settings. InitSlackSpace switches the drive temporarily to in-place mode, thus saving all pending changes.

**InitMFTRecords**

Clears unused MFT FILE records on the current logical drive if it is formatted with NTFS, using the currently set initialization settings. Simply does nothing on other file systems. The changes are written immediately to the disk.

**Assign MyVariable 12345****Assign MyVariable 0x0D0A****Assign MyVariable "I like WinHex"****Assign MyVariable MyOtherVariable**

Stores the specified integer number, binary data, ASCII text, or other variable's contents in a variable named "MyVariable". If this variable does not yet exist, it will be created. Up to 32 different variables allowed. Another way to create a variable is the Read command.

**SetVarSize MyVariable 1****SetVarSize MyVariable 4**

Explicitly sets the allocated memory size of a variable at a given time, in bytes. This can be useful e.g. for variables that hold integer values and that are the result of a calculation, if this value is to be written to a binary file with a fixed-length structure. Without SetVarSize, no assumption must be made about the size of the variable. For instance, the number 300 could be stored in any number of bytes larger than 1. If the new size set by SetVarSize is smaller than the old size, the allocated memory is truncated. If the new size is larger, the allocated memory is expanded. At any rate, the value of the persisting bytes is retained.

**GetUserInput MyVariable "Please enter your name:"**

Stores the ASCII text or binary data (0x...) specified by the user at script execution time (128 bytes at max.) in a variable named "MyVariable". The user is prompted by the message you provide as the second parameter. If the variable does not yet exist, it will be created. Other ways to create variables: Assign, Read.

**GetUserInputI MyIntegerVariable "Please enter your age:"**

Works like GetUserInput, but accepts and stores only integer numbers.

**Inc MyVariable**

Interprets the variable as an integer (if not larger than 8 bytes) and increments it by one. Useful for loops.

**Dec MyVariable**

Interprets the variable as an integer (if not larger than 8 bytes) and decrements it by one.

**IntToStr MyStr MyInt****IntToStr MyStr 12345**

Stores the decimal ASCII text representation of the integer number specified as the second parameter in a variable specified as the first parameter.

**StrToInt MyInt MyStr**

Stores the binary representation of the integer number specified as a decimal ASCII string in the second parameter in a variable specified as the first parameter.

**StrCat MyString MyString2****StrCat MyString ".txt"**

Appends one string to another. The second parameter may be a variable or a constant string. The first parameter must be a variable. The result will be saved in the variable specified by the first parameter and must not be longer than 255 characters.

**GetClusterAlloc MyStr**

May be applied to a logical drive formatted with the FAT, NTFS, Ext2/3, or CDFS file system. Retrieves a textual description of the current position's allocation, e.g. which file is stored in the current cluster, and saves that description in the specified variable.

**InterpretImageAsDisk**

Treats a raw image or evidence file like the original physical disk or partition. Requires a specialist or forensic license.

**CalcHash HashType MyVariable****CalcHashEx HashType MyVariable**

Calculates a hash as known from the command in the Tools menu and stores it in the specified variable (which will be created if it does not yet exist). The HashType parameter must be one of the following: CS8, CS16, CS32, CS64, CRC16, CRC32, MD5, SHA-1, SHA-256, PSCHF. CalcHashEx in addition displays the hash in a dialog window.

**MessageBox "Caution"**

Displays a message box with the text "Caution" and offers the user an OK and a Cancel button. Pressing the Cancel button will abort script execution.

**ExecuteScript "ScriptName"**

Executes another script from within a running script, at the current execution point, e.g. depending on a conditional statement. Calls to other scripts may be nested. When the called script is finished, execution of the original script will be resumed with the next command. This feature can help you structure your scripts more clearly.

**Turbo On****Turbo Off**

In turbo mode, most screen elements are not updated during script execution and you are not able to abort (e.g. by pressing Esc) or pause. This may accelerate script execution if a lot of simple commands such as Move and NextObj are executed in a loop.

## **Debug**

All the following commands must be confirmed individually by the user.

## **UseLogFile**

Error messages are written into the log file "Scripting.log" in the folder for temporary files. These messages are not shown in a message box that requires user interaction. Useful especially when running scripts on unattended remote computers.

## **CurrentPos**

### **GetSize**

### **unlimited**

are keywords that act as a placeholders and may be used where numeric parameters are required. On script execution, CurrentPos stands for the current offset in the active file or disk window and GetSize for its size in bytes. unlimited actually stands for the number 2,147,483,647.

# **Appendix C: Disk Editor Q&A**

## **How can I access CD-RW sectors?**

DirectCD and PacketCD must not be installed on the Windows system.

## **How can I access CD-ROM and DVD sectors under Windows 9x?**

Please make sure the following requirements are met:

1. A Windows driver of the CD-ROM/DVD drive must be installed. An MS-DOS driver is not sufficient.
2. The ASPI interface must be installed. Maybe you have to copy the file `wnaspi32.dll` manually into your `Windows\System` directory. The file is to be found on your Windows installation CD. The shareware program WinZip (available from <http://www.winzip.com>) is recommended for extracting files from CAB archives.
3. The CD-ROM/DVD drive must support the way WinHex tries to read sectors. Most of modern ATAPI and SCSI drives are suitable.

## **How can I make WinHex detect an installed PC Card ATA Flash Disk/PCMCIA Drive as a physical disk under Windows 9x?**

Windows Control Panel → System → Device Manager → Select your PCMCIA drive → Click "Properties" → Search for an option with name similar to "Int 13h device". The actual way to find this checkbox may vary on different Windows versions. If possible, *enable* this option and reboot your computer.

# Appendix D: Master Boot Record

The Master Boot Record is located at the physical beginning of a hard disk, editable using the disk editor. It consists of a master bootstrap loader code (446 bytes) and four subsequent, identically structured partition records. Finally, the hexadecimal signature 55AA completes a valid Master Boot Record.

The format of a partition record is as follows:

Offset	Size	Description
0	8 bit	A value of 80 designates an active partition.
1	8 bit	Partition start head
2	8 bit	Partition start sector (bits 0-5)
3	8 bit	Partition start track (bits 8,9 in “start sector” as bits 6,7)
4	8 bit	Operating system indicator
5	8 bit	Partition end head
6	8 bit	Partition end sector (bits 0-5)
7	8 bit	Partition end track (bits 8,9 in “end sector” as bits 6,7)
8	32 bit	Sectors preceding partition
C	32 bit	Length of partition in sectors

**Operating system indicators:**  
(hexadecimal, incomplete list)

00	Empty partition-table entry
01	DOS 12-bit FAT
04	DOS 16-bit FAT (up to 32M)
05	DOS 3.3+ extended partition
06	DOS 3.31+ Large File System (16-bit FAT, over 32M)
07	Windows NT NTFS, OS/2 HPFS, Advanced Unix
08	OS/2 v1.0-1.3, AIX bootable partition, SplitDrive
09	AIX data partition
0A	OS/2 Boot Manager
0B	Windows 95 with 32-bit FAT
0C	Windows 95 with 32-bit FAT (using LBA-mode INT 13 extensions)
0E	Logical-block-addressable VFAT (same as 06, but using LBA-mode INT 13)
0F	Logical-block-addressable VFAT (same as 05, but using LBA-mode INT 13)
17	Hidden NTFS partition
1B	Hidden Windows 95 FAT32 partition
1C	Hidden Windows 95 FAT32 partition (using LBA-mode INT 13 extensions)
1E	Hidden LBA VFAT partition
50	OnTrack Disk Manager, read-only partition
51	OnTrack Disk Manager, read/write partition

81	Linux
82	Linux Swap partition, Solaris (Unix)
83	Linux native file system (ext2fs/xiafs)
85	Linux EXT
86	FAT 16 volume/stripe set (Windows NT)
87	HPFS fault-tolerant mirrored partition, NTFS volume/stripe set
BE	Solaris boot partition
C0	DR-DOS/Novell DOS secured partition
C6	Corrupted FAT 16 volume/stripe set (Windows NT)
C7	Corrupted NTFS volume/stripe set
F2	DOS 3.3+ secondary partition

## Appendix E: Surplus Sectors

This term is used in WinHex in the following way:

Surplus sectors on a logical drive are those few sectors at the end that do not add to a full cluster and thus cannot be used by the OS (and thus by no conventional application program either).

Surplus sectors on a physical disk are those sectors at the end that are located outside the regular disk geometry scheme (because they do not add to a full cylinder/header/track entity), which is why they are usually not used by any partition or the operating system (or any conventional application program).

Surplus sectors have nothing to do with "bad" or damaged sectors or sectors a hard disk internally uses as a replacement for sectors found to be faulty.