DIEBOLD NIXDORF

DieboldNixdorf.com

(ENG) User Manual

WN BIOS Tool 1.6

We are interested in your opinion about this brochure.
Please send us a message if you would like to provide constructive recommendations. For this, thank you in advance.

Best regards,

Your opinion:

Contact
Christoph Annemueller
DE SyPLRetail1

Diebold Nixdorf
Wohlrabedamm 31
13629 Berlin, Germany

c.annemueller@dieboldnixdorf.com
DieboldNixdorf.com

WN BIOS Tool
(Windows 1.6.0.0 / Linux 1.6.0-5)

User Manual 1.6

Edition October 2017

Disclaimer

Important: Read before copying, installing or using!

The software is provided “as is” without any express or implied warranty of any kind of including warranties of merchantability, non-infringement or fitness for a particular purpose. Diebold Nixdorf does not warrant or assume responsibility for the accuracy or completeness of any information, text, graphics, links or other items contained within the software. Diebold Nixdorf also does not assume responsibility for errors or omissions in this document.

Diebold Nixdorf may make changes to the software, or to items referenced therein, at any time without notice, but is not obligated to support, update or provide training for the software. You agree to be solely responsible to your end users for any update or support obligation or other liability, which may arise from the distribution of the software.

Contents

About this User Manual

This documentation is intended to help you to work with WN BIOS Tool and to serve as a reference work. The detailed table of contents help you find the desired information quickly and easily.

Notes are marked by this symbol.

This symbol is used for warnings.

Style	Definition
<...>	Variables or placeholders, which have to be named, are written in Courier New and angled brackets.
[...]	Variables or other options are written in Courier New and square brackets and are facultative.
Text ref./links	Text references are underlined and written in italic.

About WN BIOS Tool

The WN BIOS Tool is a command line application for specific BEETLE motherboards and BIOS (see *System Requirements*), which is able to read, store and write BIOS parameters under Windows or Linux. Furthermore, the tool allows transmitting BIOS settings from a source system to other identical systems (target systems).

System Requirements

Identical Source and Target System

When transferring a complete BIOS setup file, the source and target system must have the same BIOS – therefore the same BIOS version - and identical system configurations [such as motherboard, CPU, Mass Storage, RAM (total amount of GB) and type / number of PCI cards]!

Password-protected BIOS

Changing boot parameters are applicable only, when the **BIOS is not password protected** (see *Clear BIOS Password*). Otherwise, every change of the BIOS setup will be denied!
Restriction: Although the BIOS is password protected, the NVRAM, User Space and User Variables can be changed anyway, because these areas are non-setup information.

Current WN BIOS Tool and Driver Version

(Status: October 2017)

Windows	Linux
WN BIOS Tool **1.6.0.0**	**1.6.0-5** **WNLPOS4:** wn-biostool-1.6.0-5.x86_64.rpm **WNLPOS3 / Suse SLES 11 SP3:** wn-biostool-1.6.0-5.el5.i386.rpm
+ Windows Driver package WN BIOS Tool Driver **1.2.0.0-3**	

Motherboards and BIOS

Motherboard	BEETLE /	BIOS Version
I1	M-II plus	01/06 or higher
J1.0	X plus	02/09 or higher
J1.1	X plus	43/09 or higher
K1 / K2	M-III	04/08 or higher
M1 / M2	M-III	R.1.2 or higher
L1 / L2	EPC 5G	03/06 or higher
D611 (H81)	iPOS plus Advanced	00/56 or higher
D611 (Q87)	iPOS plus Advanced	01/63 or higher
D746	iPOS plus	01/01 or higher

Operating Systems

Admin Rights on Source and Target Systems

To prevent unauthorized changes to the BIOS, the WN BIOS Tool runs under Windows or Linux (root) with admin rights only.

Windows (x86 and x64)
10 IoT Enterprise (LTSB 2016), 8.1 Professional, Embedded 8.1 Industry Pro, 7 Professional, POSReady 7, XP, POSReady 2009

Linux (x32 und x64)
WNLPOS 4 (64bit), WNLPOS 3 (32bit), Suse SLES 11 SP3

Further Information

Any changes to the BIOS made by the WN BIOS Tool take effect after a system restart!

If the BIOS Update Tools provided by Wincor Nixdorf are used unchanged, there is no risk that data will be deleted!

Installation

Windows Operating Systems

1. **Install** the **Windows Driver "WN BIOS Driver" for the WN BIOS Tool**. Run the setup as administrator. Follow the instructions of the installation wizard.
The default directory is *C:\Retail\Software\Wnbios*

If an older version of WN BIOS driver is installed, the installation of the driver will be prevented with an error message. Use the Windows tools to identify already installed or older versions in order to uninstall them. After a system restart, the current driver can be installed.

2. **Run** the **setup** of the **WN BIOS Tool** as **administrator** and follow the instructions of the installation wizard. The default directory is *C:\Retail\Software\BiosTool*

3. When the installation is completed, a **reboot** of the **system** is **necessary**.

For script-guided Installation, use the command line options
/verysilent /suppressmsgboxes

Linux Operating Systems

To install the RPM package, change to the directory where the package is located. The installation must be run as administrator (root)!

Perform the installation with the command rpm -Uvh <RPM Package> .rpm. The default directory is /opt/wn/biostool

Under Linux, an additional driver for the WS BIOS tool is not necessary!

Uninstallation

Windows Operating Systems

The driver for the WN BIOS Tool and the tool itself must be uninstalled separately.
Both uninstallers are located in the Windows Start Menu *„Wincor Nixdorf Bios Tool“.*
With execution of the uninstaller, the appropriate tool / driver will be removed completely.
When the uninstallation completes, a **restart** of the system is necessary.

Linux Operating Systems

The RPM package is uninstalled with the command rpm -e <RPM_PackageName>.

The uninstall process starts immediately as soon as the command rpm -e <RPM_Paketname> has been confirmed with “Enter”. No confirmation message appears.

Run WN BIOS Tool

Start **command prompt / command line as administrator** and change to directory

Windows: *C:\Retail\Software\biostool\wnbiostl.exe*
Linux: /opt/wn/biostool/wnbiostl.exe

Under Windows the command prompt resp. the WN BIOS Tool must be run as »administrator« and under Linux as »root«. Otherwise, the tool exits with *“Fatal Error: Tool must be called as root/administrator”*.

It is not permitted to run more than one entity of WN BIOS Tool. If more than one entity has started, it comes to conflicts within the tool! Therefore, the try to open a second entity will be closed with ***“****APP_ERR_ONLY _ONE_INSTANCE finished (EC 99)”* immediately.

Command Line Parameters

The WN BIOS Tool is called with *wnbiostl.* Commands control the tool. A parameter, parameter value, list, or file can follow a command.

Syntax			
wnbiostl	command	<list>	
		<Num.list>	
		<parametername>	<parametervalue>
		<filename>	
		<varname>	<varvalue>
		<address>	<Data>
	<parameter>	<parametervalue>	

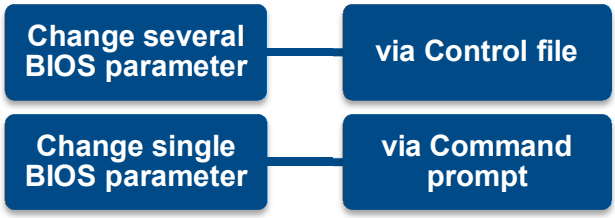
Command Overview and Help Text

Command: *wnbiostl -?*

This command shows a list of all possible commands and options (see also "*Appendix A: Command Parameter Overview (-?)*").

Modify BIOS Parameters Values

The WN BIOS Tool offers the possibility to change BIOS parameters either via a **control file** or directly via the **command prompt**. If several BIOS parameters should be changed at the same time, it is advisable to use the control file. If the setting of a single BIOS parameter shall be changed quickly, the use of the command prompt will do.



Valid BIOS Parameter Values

Command: *wnbiostl -i*

This command lists all available BIOS parameters with their valid values. Every single BIOS parameter is listed in a separate line. A line begins with *Parameter* followed by the '*name of the BIOS parameter*' in inverted commas. Behind the colon, all possible '*values of the parameter*' are listed. The comma separates all possible valid values each parameter.

```
Parameter 'WAKE_ON_LAN': 'Disabled',  'Enabled'
Parameter 'ONBOARD_COM5_6': 'Always Enabled',  'Auto Disable'
Parameter 'RESTORE_AC_POWER_LOSS': 'Power Off',  'Power On',  'Last State'
Parameter 'CONFIGURE_SATA_AS': 'IDE',  'AHCI'
```

Change BIOS Setup via Control File

Commands: **Read:** *wnbiostl -r <filename.txt>*
Write: *wnbiostl –w <filename.txt>*

With WN BIOS Tool, the BIOS setup can be saved to and changed in a control file (ASCII text file). The control file is being created with *wnbiostl -r <filename.txt>* and contains all essential system information, the current BIOS settings, and also the customized data areas.

If the modified file contains any user variables (regardless of the number), any pre-existing user variables are deleted previously, before the user variables of the file is being set! If the file does not contain user variables, existing/stored user variables remain untouched!

Entries, which **start** with **"#"** (exceptions: #! #? #:), contains control parameters for the tool and are read-only!

This procedure **does not apply** for **changing** the **boot order** and the **user space**!

The **file extension .txt is recommended** because the imported file must be an ASCII text file! Other file formats are not supported and cause unexpected malfunctions.

The commands *wnbiostl -r <filename.txt>* and *–w <filename.txt>* are denied, when the BIOS is password-protected.

If no filename and file extension is given, the output is displayed on screen! Otherwise, the file will be saved into the root directory (/biostool) of WN BIOS Tool.

Procedure

1. Save current BIOS setup in a control file
Command: *wnbiostl -r <filename.txt>*

2. Modify values in the control file
Open the file with a text editor. Search for the BIOS parameter, which should be changed. Edit the file. Afterwards, change the corresponding value of the parameter (see *wnbiostl -i*). Save the file in the same directory.

3. Transfer modified values into the BIOS
Command: *wnbiostl –w <filename.txt>*
Run this command to load the modified file into the BIOS and therefore to apply the modified settings contained in the file.

Change BIOS Setup via Command Line

Commands: **Display:** *wnbiostl <parameter>*
Change: *wnbiostl <parameter> [<parametervalue>]*

If the command does not run correctly, the error message
ERROR: x
Error Code: y appears.
x is the error message and y the error code (see *Appendix B: Error Codes (-e)*).

Procedure

Example: Set WAKE_ON_LAN from Disabled to Enabled

1. Show current value of BIOS parameter
The command *wnbiostl WAKE_ON_LAN* displays WAKE_ON_LAN=Disabled

2. Change value of BIOS parameter
The command *wnbiostl WAKE_ON_LAN Enabled* changes the value from disabled to enabled.

See *wnbiostl -i* for valid values of a parameter.

3. Verify changed value
wnbiostl WAKE_ON_LAN displays now WAKE_ON_LAN=Enabled

Transfer BIOS Setup from One to Other Systems

Commands Save in file: *wnbiostl -st <filename.bsf>*
Restore from file: *wnbiostl -sf <filename.bsf>*

These commands save the BIOS setup from a source system and restore this setup to several other systems (targets). By using this method, settings, which are not available or changeable with the tool (e.g. password setting) are considered as well.

Source and target systems must have the same BIOS version and the same system configuration, e.g. motherboard, mass storage, and I/O ports.

From Kx motherboards on, boot devices within the boot device class, which are disabled under "[…] BBS Priorities" manually, will not reactivated via a binary file, because the boot devices are not part of the BIOS setup settings. The value of the boot device classes will be restored via a binary file.

To distinguish the binary file from a possibly existing control file, it is recommended to use the file extension .bsf (**B**IOS **S**etup **F**ile).

The binary file <filename.bsf> generated with -st must be used as it is! Any manually change of the file will cause conflicts.

Procedure

To create, save, and deploy the BIOS setup, complete the following procedure:

1. Set up source systems **BIOS**
Adjust the desired values of the BIOS on the source system manually (press F2 during boot for BIOS setup menu).

2. Save BIOS setup in a binary file
Command: *wnbiostl -st <filename.bsf>*
This command saves the BIOS setup in a binary file.

3. Transfer BIOS setup to other systems
Command: *wnbiostl -sf <filename.bsf>*
Start WN BIOS Driver and Tool on target system first.
This command copies source systems BIOS setup to the target system from the created file.

The binary file must be created with wnbiostl -st <filename.bsf> and has to be located in the same directory as the WN BIOS Tool itself. Wrong file extension will be rejected!

Save and Restore BIOS Setup to/from File

The commands *wnbiostl -dt <file.bsf>* (save) and *wnbiostl -df <file.bsf>* (write) are for Diebold Nixdorf internal product purposes only!

Clear BIOS Password

Command: *wnbiostl –cp <filename.bsf>*

To prevent manipulations, changes to the BIOS setup are not possible when the BIOS is password protected (NVRAM, User Space and User Variables are excepted because these are non-setup information). Changes are therefore possible only, when no password has been set or the password has been deleted previously.

The binary file <filename.bsf> must be created with wnbiostl -st <filename.bsf>. Wrong file extensions are rejected!

Clear password is not available on L1 and L2 motherboards!

Procedure

1. Save BIOS setup in a binary file
Command: *wnbiostl -st <filename.bsf>*
This command saves the current BIOS setup in a binary file (incl. BIOS password).

2. Clear BIOS password
Command: *wnbiostl –cp <filename.bsf>*
To delete the password, use *wnbiostl –cp <filename.bsf>.*
WN BIOS tool reads out the encrypted BIOS password from the binary file and checks whether the password is valid before clearing. Modification on BIOS setup can be done after password has been cleared.

Boot Order

The boot order on I1 and J1.x motherboards can be changed with the command descripted in the following chapters.
From Kx motherboards onwards, boot device classes control the boot order (see chapter “Boot Device Classes”). Therefore, the commands descripted in the following chapters change the boot order within the boot device class only (see in BIOS “BBS Priority”).

Display Boot Order

Command: *wnbiostl -gb*

This command displays the current boot order. With *–gb*, the output also shows the enabled or disabled boot devices.

Example:

```
wnbiostl -gb

boot.1.type=PXELAN
boot.1.group=+Network Card
boot.1.name=+IBA GE Slot …
boot.2.type=HARDDISK
boot.2.group=+Hard Drive
boot.2.name=+<HDD-Type/-Model>
boot.3.type=USB
boot.3.group=+USB
boot.3.name=+<USB-ThumbDrive-Model>
boot.4.type=USB
boot.4.group=+USB
boot.4.name=+Ux<USB-ThumbDrive-Model>
```

The boot order can be changed by using the **unique boot device number** (BDN) which identifies each boot entry (refer to *Change Boot Order*). See example above 1-4, 1= PXE-LAN.
The boot device number consists of three entries: **„type“, „group“** and **„name“.**

type: Type of boot device
Depending on the motherboard, if applicable the following devices are available. Note that not all types must be present.

Type	Description
BEV	Bootstrap Entry Vector: (PCI)-Card with Option ROM
CDROM	CD- or DVD drive connected via DIE, SATA or USB
EMBEDDEDSHELL	Boot from UEFI-Shell
FLOPPY	Disk drive
HARDDISK	Internal hard disk connected via IDE or SATA
PCMCIA	In- or external (USB) plugged PCMCIA card
PXELAN	LAN-Bootprom (System boot via PXELAN)
UEFI	„UEFI-device“ (e.g. UEFI subdirectory on hard disk)
USB	USB mass storage (e.g., hard disk or USB flash drive)
USB_FLOPPY	Floppy disk drive connected via USB
USB_CDROM	A USB- connected CD or DVD drive
USB_HARDDISK	A USB-attached HDD / SSD
USB_KEY	A USB- connected USB stick +Ux-<*USB-ThumbDrive-Identifier*> **U** stands for USB device, **x** is a counter for each connected USB device. The counter always increases by 1 in a numerical order. The number does not indicate the USB port number! The minus sign (-) separates these counters from the device name. The existence of the prefix Ux depends on the BIOS version.

group: Name of boot group
Group conforms to boot class. Boot classes, which have no allocated device, are not listed with `wnbiostl -gb`.
A prefixed **plus sign (+)** signifies that minimum one boot device of this group is **enabled**.

A prefixed **minus sign (-)** signifies that all boot devices in this group are **disabled**. Classes with a minus sign are always at the end of the list. A disabled boot class cannot positioned in front of an enabled boot class (prefixed plus sign) with `wnbiostl -sb`. A try will aborted with APP_ERR_CMDLINE_ERR.

name: Name of boot device
A prefixed **plus sign (+)** signifies that this boot device is **enabled**.
A prefixed **minus sign (-)** signifies that this boot device is **disabled**. This boot device is excluded from the boot order.

Change Boot Order

Command: wnbiostl -sb <numlist>
This command changes the order of the boot order based on the BDN (see *Boot Order* in advanced).

When using wnbiostl -sb <numlist>, all boot devices get a new boot device number, which have to be used furthermore.

When calling -sb <numlist> again the new boot device numbers must be used! *wnbiostl –gb* shows the new boot device numbers.

- The list of BDN <numlist> must not include blanks!
- The group membership of a single boot device is defined by the BIOS itself. Same group members (e.g. USB key 1 and USB key 2) cannot interrupted (see also *Boot Device Classes*)! **Therefore, it is not possible to set up any boot order.**
- If <numlist> contains fewer entries as wnbiostl -gb shows, only the displayed devices get a new BDN. In this case, the BDN of the remaining devices cannot be foreseen.
- The order of devices within a group is influenced by the device detection during BIOS start. If a device is not plugged anymore, this device will be removed from the list. New connected devices will be placed at the end of each group.

Examples

Readout the BDN with –gb.
An example shows the boot order:

Network:IBA GE Slot (BDN 1),
Hard Disk: ST500xxx (BDN 2),
USB-Key:U1-CAFlashTranscend (BDN 3),
USB-Key:U2-JetFlashQimoda (BDN 4).

A call of wnbiostl -sb 3,4,2,1 changes the boot order to that effect that the system boots from *USB-Key:U1-CAFlashTranscend (BDN 3)* first. If this device is neither bootable nor available, the system boots from *USB-Key:U2-JetFlashQimoda (BDN 4).* If the second USB drive is also not available or bootable the system boots from internal *Hard Disk: ST500xxx (BDN 2)*. If no HDD is reachable, the system starts via *Network:IBA GE Slot (BDN 1).*

wnbiostl –gb shows now
USB-Key:U1-CAFlashTranscend (BDN 1)
USB-Key:U2-JetFlashQimoda (BDN 2)
Hard Disk: ST500xxx (BDN 3)
Network:IBA GE Slot (BDN 4).

Further examples

-sb set up boot order	**Name of boot device**
Current boot order(example)	IBA GE Slot, Hard Disk: ST500xxx, USB-Key:U1-CAFlashTranscend, USB-Key:U2-JetFlashQimoda
wnbiostl –sb 4,3,2,1	USB-Key:U2-JetFlashQimoda, USB-Key:U1-CAFlashTranscend, Hard Disk: ST500xxx, IBA GE Slot
wnbiostl –sb 2,3,4,1	Hard Disk: ST500xxx, USB-Key:U1-CAFlashTranscend, USB-Key:U2-JetFlashQimoda, IBA GE Slot
Wnbiostl –sb 2,3,1,4 >> Negative example: boot order not possible when the BIOS groups boot devices <<	Rejected from WN BIOS Tool, as PXELAN-Boot is located between USB1 and USB2. Thereby the group memberships of USB keys are interrupted.
wnbiostl –sb 4	USB-Key:U2-JetFlashQimoda will be the first boot device. USB-Key:U1-CAFlashTranscend will be the 2nd boot device because this is the same group. **The order of the other devices whose BDN are not listed is not predictable (apart from the respective group members).**

Enable or Disable Boot Devices

Commands: **Enable:** *wnbiostl -eb <numlist>*
Disable: *wnbiostl -db <numlist>*

The following chapter describes how to enable or disable specific boot devices.
A prefixed **plus sign (+)** signifies that this boot device is **enabled**. A prefixed **minus sign (-)** signifies that this boot device is **disabled**. This boot device is excluded from the boot order.

By running `wnbiostl -db` on I1 and J1.x motherboards only, the boot group will be deactivated automatically, when no active device is available in this group. Conversely, by running `wnbiostl -eb`, the boot group will be activated when min. one device is available in this group.
On all other motherboards, an automatic activation or deactivation of the boot group does not take place.

If all available boot devices are disabled with *wnbiostl -db <numlist>*, the system does not boot anymore!

Procedure

Based on the example from chapter *Display Boot Order*:
Set *IBA GE Slot* from Enabled (+PXE) to Disabled (-PXE)

1. Display current boot order
Command: *wnbiostl -gb*
Get boot device number of *IBA GE Slot*:

```
boot.1.type=IBA GE Slot
boot.1.group=+Network Card
boot.1.name=+IBA GE Slot …
```

Boot device *IBA GE Slot* has boot device number 1.

2. Disable boot device
Command: *wnbiostl -db 1*
With this command, the boot device number 1 will be disabled.
wnbiostl -gb shows that *IBA GE Slot* has the prefix "-" which indicates that this boot device is disabled. The prefixed minus sign in the group entry (-Network Card) shows that there is no other enabled boot device available in this boot device group.

```
boot.1.type=IBA GE Slot
boot.1.group=-Network Card
boot.1.name=-IBA GE Slot …
```

Boot Device Classes

Some motherboards (e.g. Kx and Lx motherboards) have a BIOS that sorts the devices into classes (such as USBCDROM, USBKEY, USBHARDDISK, USBFLOPPY, CDROM, HARDDISK, and PXELAN). Each boot device belongs to a particular group, the so-called boot device class.

The order of the boot device classes can be displayed and changed on BIOS', which supports boot device classes. Every boot device class can allocate one position only.

The boot device class order is defined by the BIOS. The BIOS searches for bootable devices and sets the order automatically.

Display Boot Device Class Order

wnbiostl -gc lists the current boot device class order.

```
wnbiostl -gc

bootclass.1.type= USBCDROM
bootclass.2.type= USBKEY
bootclass.3.type= USBHARDDISK
bootclass.4.type= USBFLOPPY
bootclass.5.type= CDROM
bootclass.6.type= HARDDISK
bootclass.7.type= PXELAN
```

The command wnbiostl -gcl lists the current boot device class order as a single-line list.

```
wnbiostl -gcl
Possible BootClasses: USBCDROM,USBKEY,USBHARDDISK,USBFLOPPY,CDROM,HARDDISK,PXELAN
```

Change Boot Device Class Order

Command: *wnbiostl –sc* <list>

With *wnbiostl –sc* <list> the sequence of the order of the device classes can be changed.

Procedure

1. Display current boot class order with wnbiostl –gc.

2. Change boot class order with wnbiostl –sc <list>

When calling wnbiostl –sc HARDDISK,PXELAN,USBFLOPPY,CDROM for instance, wnbiostl –gc displays

```
wnbiostl -gc

bootclass.1.type= HARDDISK
bootclass.2.type= PXELAN
bootclass.3.type= USBFLOPPY
bootclass.4.type= CDROM
bootclass.5.type= Disabled
bootclass.6.type= Disabled
bootclass.7.type= Disabled
```

In this example, USBCDROM, USBKEY, USB HARDDISK are disabled. For instance, USBCDROM was overwritten by HARDDISK and not listed in wnbiostl –sc <list> and therefore disabled.

Enable or Disable Boot Device Classes

Command: wnbiostl –sc <list>

With wnbiostl –sc <list>, device classes can be disabled or enabled.

Every boot device class is registered one time only. The first appearance of the device class is retained!

If an invalid value is entered, the device class at this position will be disabled automatically!

The <list> of boot devices classes must not contain blanks!

An UEFI BIOS (Kx and newer) internally has 2 different list of boot device classes: Firstly, a list of “Legacy Boot Device Classes” and secondly, a list of “UEFI Boot Device Classes”. It is possible to edit one list at a time only. With BIOS setting “BOOT Mode Select”, the corresponding list can be selected.

The assigned names of boot device classes depend on the motherboard!

Enabled (active) classes must not registered at the beginning of the list.
In case the system should start from hard drive only, all other boot media can be disabled.
Command: wnbiostl –sc disabled,disabled,disabled,disabled,disabled, HARDDISK,disabled

For this example, wnbiostl -gc shows then:

```
wnbiostl -gc

bootclass.1.type= Disabled
bootclass.2.type= Disabled
bootclass.3.type= Disabled
bootclass.4.type= Disabled
bootclass.5.type= Disabled
bootclass.6.type= HARDDISK
bootclass.7.type= Disabled
```

If temporarily a USB device should be approved as the first boot device, the first entry must be set to USBKEY: wnbiostl –sc USBKEY

wnbiostl -gc shows now:

```
wnbiostl -gc

bootclass.1.type= USBKEY
bootclass.2.type= Disabled
bootclass.3.type= Disabled
bootclass.4.type= Disabled
bootclass.5.type= Disabled
bootclass.6.type= HARDDISK
bootclass.7.type= Disabled
```

If no USBKEY is connected, the boot device classes 1 to 5 are skipped. The system boots then from HARDDISK.

User Variables (Customized Variables)

The User Variables (UV) is free definable data area, in which information can be stored, read and deleted. The information are stored in the EFI-NVRAM, which is managed by the EFI-BIOS. Note that this is not a separate hardware!

This area is mainly used by the system itself. Therefore, the total size of all user variables must not exceed 32 kBytes. Otherwise, the EFI BIOS is limited in its functionality!

At the maximum, 256 free definable user variables are available in the range of 00-FF. Each user variable can store information of a maximum of 4096 Byte.
For instance, 8 user variables can store max. 4096 Byte per variable, which is in total 32 kByte.

Data, which are stored in EFI-NVRAM, are deleted unrecoverable in case of a logical damage! E. g. due to writing during a power outage or incorrect BIOS version on the wrong system platform.

If the file contains user variables (regardless of the quantity) when running wnbiostl -w <file.txt>, any pre-existing or already created user variables will be deleted before the new variables from the file will be read and written!
If the file does not contain user variables, the pre-existing/stored user variables remain untouched!

wnbiostl –ul and *–uv* are applicable only, when the **BIOS is not password-protected** (see *Clear BIOS Password*).

Display User Variables

Command: *wnbiostl –ul*

wnbiostl -ul displays all customer-specific user variables.
Every line shows information of one user variable in the following order:
name of variable, flag (0 or 1*), length, data

Length is displayed in Byte and data in hexadecimal format.

* The flag may be 0 or 1 only! 1 (delete at triggered intrusion sensor) occurs only when the intrusion detection has been activated and the intrusion sensor has been triggered.

Read, Write, and Delete

Commands: Read: *wnbiostl [-hx] -uv* <varname>
Write: *wnbiostl [-hx] -uv* <varname> <*varvalue*>
Delete: *wnbiostl -uv* <varname> ““

Use *[-hx]* -uv to read, write, or clear data in a reserved EFI area.

<*varname*> describes an area (index) wherein user-specific data <*varvalue*> can be written. This index <*varname*> should exist of exact two hexadecimal characters (00-FF).

Not displayable characters can be entered in hexadecimal with prefix *-hx* on the shell.

The hexadecimal values must not include blanks! Each data byte exists of exact 2 characters (0 … 9, a …f, A …F). Due to legibility, when readout a variable the characters are separated with a blank.

To terminate all internal strings correctly, hexadecimal values always get a “00” suffix automatically.

Read User Variable

To display the data saved in a variable, the command *–uv* with the name of the variable <*varname*> has to be used.

ASCII: *wnbiostl -uv* <varname>
Example: *wnbiostl -uv A1* displays the value of *A1 = BEETLE* in plain text.

Hexadecimal: *wnbiostl –hx -uv* <varname>
Example: *wnbiostl –hx -uv A1* displays the hex value of *A1 = 42 45 45 54 4C 45 00* with suffix 00.

Write User Variable

With the following command the user-specific value <varvalue> can be written into a selected variable <varname>.

ASCII: *wnbiostl -uv* <varname> <*varvalue*>
Example: *wnbiostl -uv A1 BEETLE* writes in variable *A1* the value *BEETLE* in plain text.

Hexadecimal: *wnbiostl –hx -uv* <varname> <*varvalue*>
Example: *wnbiostl –hx -uv A1 424545544C45* writes in variable *A1* hex value *424545544C45**00*** with suffix 00 (424545544C4500=BEETLE).

Delete Content of User Variable

To delete the content of the index the same command for write into a variable is used with no value <*varvalue*>. This is specified with two consecutive inverted commas wnbiostl -uv <VarName> ““

wnbiostl -uv <varname> *””* is valid for hex and ASCII contents!

Intrusion Sensor: Delete Content of User Variable if triggered

Command: *wnbiostl [-hx] -uv* <varName> <*Varvalue*> *i*

Some system platforms support the functionality of an intrusion sensor. If user variables are marked with the additional parameter "i" during creating or changing, their contents are deleted as soon as the intrusion sensor is triggered. Furthermore, the wnbiostl -ul supplies the value 1 in the flag field when the intrusion detection is activated in BIOS. If the intrusion detection is not implemented or disabled (wnbiostl -ul then shows the flag field with 0), no appropriately marked variable will be deleted.

The flag "i" must be written with every change!

The intrusion detection must be enabled in BIOS!

User Space (Customized Storage Area)

The user space is a free space, which can be used as data storage. The data of the user space are stored in a separate reserved area of the ROM chip. Contrary to the data area of the user variable, a file can be stored, readout and deleted.

The size of the user space differs from system platform to system platform.

The **maximum available of the size of the user space** must be readout with wnbiostl -ui.

The logical content of the file cannot be readout with WN BIOS tool.

The user space is stored into the firmware memory, but this area will not be modified when changing BIOS settings or updating the BIOS. Thus, this area is not related to the BIOS! Additionally, stored data are also not related to operating system dependencies! For this reasons, the data of the user space are not exposed to risks of the EFI-NVRAM risks, e.g. like user variables.

Usage example:
For instance, JPEG image, text, or an archive file can be stored in the user space.

Display User Space Information

Command: wnbiostl -ui
The command -ui reads out information into a text form, e.g. presence, size, and use of the storage area.

Read User Space

Command: wnbiostl -us <file>
The command -us <file> writes the file / content of the user space into a file.

Write User Space

Command: wnbiostl -uw <file>
The command -uw <file> deletes resp. overwrites the user space with a new <file>.

Delete Content of User Space

Command: wnbiostl -ue
The command –ue deletes the entire content of the user space.

NVRAM

Display Information of Used NVRAM

Command: *wnbiostl –vi*
wnbiostl -vi provides information whether a separately obtainable NVRAM module can be addressed on this platform. If an external NVRAM module is connected and addressable, -vi shows also the size of simultaneously transmittable data (block size) and the max. addressable memory size of the module. **The physical presence of a module is unchecked!**

The management of the NVRAM module obliged end user's responsibility.

Write on NVRAM

Command: *wnbiostl [-hx] –vw <address> <data>*
wnbiostl *[-hx]* -vw <address> <data> writes <data> starting from <address> in the NVRAM.

With *-hx* not displayable characters can be entered in hexadecimal on the shell.

The hexadecimal values must not include blanks! Each data byte exists of exact 2 characters (0 … 9, a …f, A …F). Due to legibility, when readout a variable the characters are separated with a blank.

ASCII

wnbiostl –vw <address> <data>

<address> must be entered in decimal and *<data>* in ASCII characters. The hexadecimal of each ASCII character is stored in NVRAM.

Example:
wnbiostl -vw 160 BEETLE
This command writes the data "BEETLE" in hexadecimal from address 160 (hex 0xA0) of the NVRAM in the following order:

```
0xA0 [160] -> 0x42 [B]
0xA1 [161] -> 0x45 [E]
0xA2 [162] -> 0x45 [E]
0xA3 [163] -> 0x54 [T]
0xA4 [164] -> 0x4C [L]
0xA5 [165] -> 0x45 [E]
```

Hexadecimal

wnbiostl -hx –vw <address> <data>

Both, *<address>* and *<data>* must be entered in hexadecimal. The hexadecimal of each ASCII character will be stored in NVRAM. The hexadecimal values must not include blanks! Each data byte exists of exact 2 characters (0 … 9, a …f, A …F).

Example:
wnbiostl –hx -vw A0 424545544C45
This command writes the data in hexadecimal from address 0xA00 of the NVRAM in the following order:

```
0xA0 -> 0x42
0xA1 -> 0x45
0xA2 -> 0x45
0xA3 -> 0x54
0xA4 -> 0x4C
0xA5 -> 0x45
```

Read NVRAM

Command: *wnbiostl [-hx] –vr <address> <length>*

wnbiostl [-hx] –vr <address> <length> lists from <address> the *<length>* in Bytes of the NVRAM.

ASCII

wnbiostl –vr <address> <length>

<address> and *<length>* must be given in **decimal**. The in- and output of each address is limited to 16 data Bytes. The data Bytes represent hexadecimal values.

Example:
wnbiostl –vr 160 6 lists from address 160 (hex 0xA0) the next 6 data Bytes.
In that case, address 0xA0 consists of the following data: 42 45 45 54 4c 45 [decimal BEETLE]

Hexadecimal

wnbiostl -hx –vr <address> <length>

<address> and *<length>* must be given in **hexadecimal**. The in- and output of each address is limited to 16 data Bytes. The data Bytes represent hexadecimal values.

Example:
wnbiostl –hx -vr A0 6 lists from address A0 (decimal 160) the next 6 data Bytes. In that case, address 0xA0 consists of the following data: 42 45 45 54 4c 45 [decimal BEETLE]

Show WN BIOS Tool Version

Command: *wnbiostl -v*
With this command, die used WN BIOS Tool version is shown.

For instance, on a 32bit operating system, wnbiostl -v shows the following string:

```
# Tool wnbiostl.exe: Version $Revision: 15160 $ (1.6.0.0), compiled at MM DD YYYY
```

On a 64bit operating system, the suffix „64bit“ is shown additionally.

```
# Tool wnbiostl.exe: Version $Revision: 15160 $ (1.6.0.0 (64bit)), compiled at MM DD YYYY
```

Set BIOS to Factory Settings

Command: *wnbiostl -rd*

This command sets all BIOS settings to their default values.

This command is equivalent to F3 key in the BIOS setup.

After *wnbiostl -rd* a system restart is mandatory!

Error Codes

Command: *wnbiostl -e*

This command provides a list of all possible error codes *(Appendix B: Error Codes (-e)*).

Known Restrictions

WNBIOS Tool-related Restrictions	Solved in version	
	Windows	WNLPOS
Version 1.1.0.0-1 supports x86-Windows operating systems (32bit) only	1.2.0.0	1.0-6
Different types of boot devices have the same name	1.2.0.1-1	1.0-6
The WN BIOS Tool shows a fifth option, PEG_PCI_IGD, for INI-ATE_GRAPHIC_ADAPTER which is not available in BIOS (J1.x and I1)	1.2.0.1-1	1.0-6
Error code always “0”, even if failure occurs	1.2.0.1-1	1.0-6
Wrong error code with “wnbiostl –w” if BIOS is password-protected (right error code: 83)	1.2.0.1-1	1.0-6
Different help text using -? and –h	1.2.0.1-1	1.0-6
The functionality of the intrusion sensor is not implemented.	1.2.0.0	1.0-6
wnbiostl -rd allows restoring defaults when the BIOS is password protected	1.2.0.1-1	1.0-6
Restrictions for Kx and Lx motherboards with BIOS 00/00 or 02/03 : Currently, some commands are not available on these hardware platforms! The affected commands are listed and marked in Appendix A.	1.2.0.32	1.2.0.32
Due to the logical complexity of WN BIOS Tool and the information exchange of programs and services different providers and Microsoft Windows operating systems, the system can be freezes after *n* procedures and *n* WN BIOS Tool iterations. Already started services can be used furthermore, but new services cannot be started. Therefore, a regular reboot of the system is mandatory.		
Linux version 1.5-38 does not support D611 (H81 and Q87) and D746 motherboards.	n/a	1.6.0

Hardware- and system-related Restrictions	
Not implemented are the following switches visible in Setup, which are not accessible to the OS runtime : ▪ Active Processor Cores ▪ Thermal Configuration Menu ▪ AMT Wait timer ▪ Parallel Port - Change Settings ▪ Parallel Port - Device Mode ▪ Serial Port Console Redirection Menu ▪ Intel® Ethernet Network Connection Menu ▪ Graphics Turbo IMON Current ▪ Setup Prompt Timeout ▪ Menu of the Secure Boot Menu ▪ Save & Exit Menu Secure boots Menu	System
Not implemented are the settings under *Chipset->Mini51/54 MCU Configuration.* This restriction is valid for D611 (H81) from BIOS 01/56, D611 (Q87) from BIOS 01/63 and D746 from BIOS 01/01.	BIOS and Tool
Changing or deleting of the BIOS password is not possible on the following motherboards: D611 (H81) from BIOS 01/56, D611 (Q87) from BIOS 01/63 and D746 from BIOS 01/01.	BIOS
Enabling *XP_GRAPHICS_SUPPORT* on J1.x motherboards and boot of the system with Windows 7 leads to a Blue Screen of Death (BSOD).	System
BIOS R.1.2 of M1 and M2 motherboards does not support changing the Boot Device Order.	BIOS

Appendix

Appendix A: Command Parameter Overview (-?)

	Description	Command	Page
Help Text	Displays the help text and the commands	*--help* *-h* */h* *-?* */?*	6
Info	Register all available BIOS parameter and their possible values	*-i* *--info*	6
Parameter	Display single BIOS parameter	*<Parameter>*	7
	Change single BIOS parameter value	*<Parameter> <Parametervalue>*	7
Setup in/from ASCII-File	Save current BIOS settings into a control file	*-r <filename.txt>*	6
	Transfer changed settings into the BIOS	*-w < filename.txt >*	6
Clear BIOS password	Clear BIOS password (consider restrictions!)	*-cp <filename >* *--clearpassword <filename>*	9
Setup in/from binary file	Save current BIOS settings in a binary file	*-st < filename>* *--copysetuptofile <filename>*	8
	Restore BIOS settings from binary file	*-sf <filename>* *-- copysetupfromfile <filename>*	8
Boot order / boot device classes	Get boot order	*-gb* *--getbootorder*	9
	Write or rather change boot order	*-sb <numlist >* *--setbootorder <numlist >*	10
	Enable boot device	*-eb <numlist >* *--enablebootdevices < numlist >*	12
	Disable boot device	*-db <numlist >* *--disablebootdevices <numlist>*	12
	Lists the current boot order of boot devices class	*-gc* *--getbootclassorder*	13
	Lists the current order of the boot device class in a line	*-gcl* *--getbootclassorderlist*	13
	1. Enable / Disable boot device class 2. Change the boot order classes	*-sc <list>* *--setbootclassorder <list>*	13 13
User Variable	Display all customer-specific variables	*-ul* *--uservarlist*	15
	Read value of customer-specific variable <varname>	*[-hx] -uv <VarName>* *--uservar <VarName>*	15
	Write customer-specific value <varvalue> into customer-specific variable <varname>	*[-hx] -uv <VarName> <VarValue>*	15
	Clear customer-specific variable	*-uv <VarName> ""*	15
	Write user variable with Intrusion Sensor Flag	*-uv <VarName> <VarValue> i*	16

User Space	Displays information about user area of the ROM's	*-ui* *--userspaceinfo*	16
	Loads and writes user data in the ROM	*-uw <filename>* *--userspaceload <filename>*	17
	Stores user data from ROM to file	*-us <Datei>* *--userspacesave <filename>*	16
	Clears the entire user data in ROM	*-ue* *--userspaceerase*	17
NVRAM Memory	Displays information about NVRAM	*-vi* *--nvinfo*	17
	Shows the size in bytes of the specified range of NVRAM	*-vw <address> <data>* *--nvwrite <address><data>*	17
	Writes data to the specified NVRAM area offset	*-vr <address> <data>* *--nvread <address> <data>*	18
Tool Version	Show WN BIOS Tool version	*-v*	18
Factory Settings	Restore Diebold Nixdorf factory settings	*-rd* *--restoredefaults*	19
Error Codes	Display error codes	*-e* *--exitvalues*	19

Appendix B: Error Codes (-e)

EC#	Description
0	**Execution successful** (EFI_SUCCESS)
1	**Load error** (EFI_LOAD_ERROR)
2	**Invalid parameter / Invalid commandline parameter** (EFI_INVALID_PARAMETER)
3	**Not supported / NVRAM/ USERSTORAGE not supported by BIOS (Error will be generated ONLY while calling a corresponding function!)** (EFI_UNSUPPORTED)
4	**Bad buffersize** (EFI_BAD_BUFFER_SIZE) (EFI_BAD_BUFFER_SIZE)
5	**Buffer too small / Data block incomplete while transferring data from/to OS** (EFI_BUFFER_TOO_SMALL)
6	**Not ready** (EFI_NOT_READY)
7	**Device error** (EFI_DEVICE_ERROR)
8	**Write protected / It was tried to change the state of a read only SETUP switch** (EFI_WRITE_PROTECTED)
9	**No more resources available / (An internal function could not be assigned during initialization)** (EFI_OUT_OF_RESOURCES)
10	**Internal memory corrupted / May happen while using new SETUP interface** (EFI_VOLUME_CORRUPTED)
11	**Internal memory full** (EFI_VOLUME_FULL)
12	**No media** (EFI_NO_MEDIA)
13	**Media changed** (EFI_MEDIA_CHANGED)
14	**Information not found / This error is a collection for many errors where data is not found and further information is not available** (EFI_NOT_FOUND)
15	**Access denied** (EFI_ACCESS_DENIED)
16	**No response** (EFI_NO_RESPONSE)
17	**No mapping** (EFI_NO_MAPPING)
18	**Timeout** (EFI_TIMEOUT)
19	**Not started** (EFI_NOT_STARTED)
20	**Already started** (EFI_ALREADY_STARTED)
21	**Aborted** (EFI_ABORTED)
22	**Communication error** (EFI_ICMP_ERROR)
23	**Communication error** (EFI_TFTP_ERROR)
24	**Protocol error** (EFI_PROTOCOL_ERROR)
25	**Incompatible version / Will also be delivered if data file and system do not match** (EFI_INCOMPATIBLE_VERSION)
26	**Security violation** (EFI_SECURITY_VIOLATION)
27	**Checksum error** (EFI_CRC_ERROR)
28	**Unknown error / All Bios errors above 28 are collected in this error! Report this error to Wincor Nixdorf!** (EFI_UNKNOWN)
64	**The program was started on a computer that is not designed by Wincor Nixdorf!** (APP_ERR_NOT_A_WN_BIOS)
65	**The interface to the operating system does not exist or could not be started or the Bios does not support it** (APP_ERR_NO_INTERFACE)
66	**This Bios is unknown** (APP_ERR_INTERFACE_ERROR)
67	**This Bios does not support this program** (APP_ERR_APP_UNSUPPORTED)
68	**This Bios does not support any of the requested functions** (APP_ERR_NOFUNCTIONSDEFINED)
69	**Please use a newer version of this program** (APP_ERR_OUTDATED)
70	**Commandline error** (APP_ERR_CMDLINE_ERR
71	**Memory could not be obtained** (APP_ERR_MAPPING_ERROR)
72	**The BIOS interface has not been initialized until now. Report this error to Wincor Nixdorf!** (APP_ERR_INTERFACENOTINIT)
73	**Null-pointer not allowed** (APP_ERR_NULLPOINTER_PASSED)
74	**Empty string not allowed** (APP_ERR_EMPTY_STRING_PASSED)
75	**Invalid password size** (APP_ERR_PASSWORDSIZE)
76	**Invalid password** (APP_ERR_PASSWORDWRONG)
77	**The Setup size is not as expected** (APP_ERR_SETUPVARSIZEUNMATCH)
78	**Invalid parameter for this switch passed** (APP_ERR_INVALID_SWITCH_PARAM)
79	**Program memory full (should happen only under DOS)** (APP_ERR_OUT_OF_MEMORY)
80	**File not found** (APP_ERR_FILE_NOT_FOUND)
81	**Invalid BIOS-ROM-file** (APP_ERR_INVALIDBIOSFILE)
82	**The task was terminated due to missing information. Report this error to Wincor Nixdorf!** (APP_ERR_MISSING_INFORMATION)
83	**Currently this BIOS is protected by a password. Hence changes are not possible!** (APP_ERR_PASSWORDPROTECTED)
84	**Invalid BIOS-data-file** (APP_ERR_INVALIDDATAFILE)
85	**Cannot read from file** (APP_ERR_FILEREADERROR)
86	**Cannot write to file** (APP_ERR_FILEWRITEERROR)
87	**Error in data-file** (APP_ERR_FILEDATAERROR)
88	**Cannot open file** (APP_ERR_FILEOPENERROR)
89	**Checksum error in file** (APP_ERR_FILECHECKSUMERROR)
90	**File version wrong** (APP_ERR_FILEGUIDERROR)
91	**GUID of the file wrong** (APP_ERR_FILEGUIDERROR)
92	**File was not created on an identical system** (APP_ERR_FILENOTFROMSAMEBIOS)
93	**Invalid Boot order** (APP_ERR_INVALID_BOOTORDER)
94	**Function (currently) not implemented** (APP_ERR_NOT_YET_IMPLEMENTED)
95	**Application does not have administrative rights** (APP_ERR_NO_ADMIN)
96	**User data space empty** (APP_ERR_NO_DATA)
97	**User data space invalid** (APP_ERR_INVALID_DATA)
98	**File too big to fit in user data space** (APP_ERR_FILE_TOO_BIG)
99	**Only one instance of this or a similar tool is allowed to run at the same time.** (APP_ERR_ONLY_ONE_INSTANCE)
100	**Tool is unable to detect the location where default data for Setup switches is stored** (APP_ERR_UNKNOWN_DEFAULTSTORE)
101	**Tool cannot access Setup default data** (APP_ERR_NO_DEFAULTDATA)
102	**The BIOS is not protected by a password.** APP_ERR_NOTPASSWORDPROTECTED
103	**Unknown structure** (APP_ERR_UNKNOWN_BRD_INFO)
104	**An entry cannot be activated (+) because the BootClas-sOrder could not be read.** (APP_ERR_NO_BOOTORDER_CLASS_INFO)
105	**Class is not registered in the BootClassOrder!** (APP_ERR_BOOTORDER_CLASS_DISABLED)
192	**Internal error -> Please report this error to Wincor!** (APP_ERR_INTERNAL_1)
193	**Internal error! Report this error to Wincor Nixdorf!** (APP_ERR_INTERNAL_2)