TeraByte OS Deployment Tool

User Manual

TeraByte Unlimited
Las Vegas, Nevada, USA
http://www.terabyteunlimited.com

Revision: 2024.02.19

Table of Contents

Introduction & Requirements

The TeraByte OS Deployment Tool (TBOSDT) is a specialized command shell that supports an advanced set of commands to perform various tasks related to OS deployment, as well as more generalized types of tasks, such as file management. All commands can be executed manually from the interactive command shell, as well as automatically from a script. The command line syntax for the TBOSDT program itself is as follows:

```
tbosdt [script]
```

If a script filename is given, TBOSDT will execute the script. If not, it will start up in interactive mode where commands can be executed manually, one at a time. TBOSDT is supplied in several OS versions, including Windows, DOS, Linux, and UEFI.

Requirements

Operating System:

- Windows (all NT-based versions 4.0 or later).
- Linux
- DOS

Using TBOSDT Interactively

To run TBOSDT interactively, start the program without any command line parameters. (For help extracting and running TBOSDT, please see the **Running Scripts Using TBOSDT Professional** section.) At the TBOSDT prompt, you can type `HELP` to get the list and descriptions of all available top-level commands:

```
EXIT        - Exit program
CLS         - Clears the TBOSDT window or screen
MD          - Create a folder
CD          - Change current folder
RD          - Remove a folder
VOL         - List volume
TYPE        - List contents of a file
PWD         - Print working folder
STATS       - Display stats of a file system
DIR         - List files
LS          - List files
EXTRACT     - Extract a file or files from a cab archive
COPY        - Copy command
```

```
OPEN        - Open command
CLOSE       - Close command
LIST        - List command
ADD         - Add command
DEL         - Delete command
REN         - Rename command
RESIZE      - Resize command (Pro for BootIt version only)
SET         - Set command
VER         - Program version
REBOOT      - Reboot or shutdown system
RUN         - Run a TBOSDT script (or TBScript if extension is .TBS)
EXEC        - Run a TBScript or native program (Pro version only)
RUNTBS      - Run a TBScript program (Pro version only)
SLIDE       - Slide command (Pro for BootIt version only)
MOUNT       - Associate a file system with a drive
UMOUNT      - Remove file system association with a drive
REGEXPORT   - Export registry key branch
REGIMPORT   - Import registry key branch
```

Specific help for each of the top-level commands can be displayed by typing `HELP <command>` (e.g. `HELP COPY`). Many commands also have sub-commands, and these will be displayed in the help text for the top level command. For example, the `COPY` command has 5 sub-commands, so the command `HELP COPY` will display the following:

```
HELP COPY

FILE        - Copy a file.
CAB         - Copy a file out of a CAB.
KEY         - Copy a registry key.
PARTITION   - Copy partition to a file.
INF         - Copy INF file and/or install driver.
```

`COPY FILE` would be used when copying files, while `COPY KEY` would be used when copying registry keys, etc. The specific help for `COPY FILE` and `COPY KEY` can then be displayed by typing `HELP COPY FILE` and `HELP COPY KEY` respectively.

To exit from the TBOSDT shell, type `EXIT` at the prompt.

Using TBOSDT Commands from a Script

A TBOSDT script consists of a text file that contains one or more commands, listed in the order to be executed. Comments can be included in the script by putting a semi-colon (;) at the beginning of the line. Blank lines can also be used to improve readability when viewing the file. A TBOSDT script can be executed in two ways:

1. From the OS command line: Start TBOSDT with the name of the script file as the command line parameter (e.g. `tbosdt myscript`). When all commands have been executed, TBOSDT will return to the OS command prompt, and all mounted drives and open files will be automatically unmounted or closed. The exception to this is when the `INTERACTIVE` or `STAY` command is used within the script. The `INTERACTIVE` command will cause the script to stop executing at that point, and leave you in interactive mode within TBOSDT. Any drives mounted at that point will remain mounted, any registry files opened will remain open, and commands can be executed manually from the TBOSDT prompt. To leave interactive mode, type `EXIT` at the prompt. This will cause the running script to continue executing the remaining commands (if any) in the script, and then exit normally to the OS command prompt. The `STAY` command also switches to interactive mode but only once the script has completed.

2. By using the `RUN` command: The `RUN` command can be used to execute a script from interactive mode or from another script. The `RUN` command syntax is simply `RUN <script>`. The `RUN` command will execute the script, and then return to interactive mode, or to the calling script. When using the `RUN` command from interactive mode, any drives mounted, registry files opened, or registry keys opened will remain that way when the script terminates, unless they have been explicitly unmounted or closed by the script. Starting with TBOSDT version 1.42, if the script to be run has a .TBS (case insensitive) extension, it will be launched as a TBScript (runtbs) instead of a normal TBOSDT script.

If an error is encountered while executing a script, an error message will be displayed on the console, and the script will terminate. The error message will indicate the line number of the script where the error was encountered, along with some specifics about the nature of the error.

TBOSDT Command Reference

This following sections divide the TBOSDT command set into 6 functional groups:

1. Commands for Working with Files and Folders
2. Commands for Working with Drives and File Systems
3. Commands for Working with Registry Files (Hive Files)
4. Commands for Working with the Registry on a Running Windows System
5. Commands Associated with Windows Driver Installation
6. Additional Commands

Each section will include a table summarizing the syntax and options for each command in the group, as well as some notes and examples of usage. While the majority of the commands are available in all 3 OS versions, there are some that are only available in the Windows version. These will be noted in the command reference tables.

General Usage Notes

- When using the Linux version of TBOSDT, options are specified with a "-" instead of a "/".
- Where applicable, TBOSDT commands support the use of wildcard characters "*" and "?".
- Some commands, such as those that directly access hard drives, will require root/admin privileges.
- When running from a .run script, confirmation options, such as `/y` for the `DEL FILE` and `COPY FILE` commands, are not required to avoid a confirmation prompt.
- Many of the commands can be executed by using one or more alternate syntax variations. All available syntax variations for each command are listed in the TBOSDT help text, as well as in the command reference tables in this document.
- Paths to files and directories should use the same separator character that is native to the OS in use; "\" for DOS and Windows, and "/" for Linux.

1. Commands for Working with Files and Folders

Important: When using the Linux version of TBOSDT, all command line options are specified by using a "-" instead of a "/" for a single character and "--" when more. **Example:** `LIST FILES -s --AD`

Command Function	Command Syntax
List files	DIR pathmask [/w][/p][/s][/f][/xj][/x][/ad][/ah][/as][/ar][/aa] LS pathmask [/w][/p][/s][/f][/xj][/x][/ad][/ah][/as][/ar][/aa] LIST FILES pathmask [/w][/p][/s][/f][/xj][/x][/ad][/ah][/as][/ar][/aa] /w - wide display (sfn only). /p - pause on each full screen of information. /s - include sub-folders. /f - follow symlinks (local file system only) /xj - don't follow junctions/symlinks (Windows) /x - display short names only. /ad – list directories only. /ah – list hidden files only. /as – list system files only. /ar – list read-only files only. /aa – list archive files only.
List contents of a file	LIST FILE filename [/p][/r] TYPE filename [/p][/r] /p – pause after a full screen of information. /r – raw output (no UTF16 translation). /u8 – process UTF8 sequences.
List alternate data stream names of a file	LIST ADS filename
Add/Delete a line of text in a text file	SET TEXTLINE filename text [/d][/n][[/b][/a] [searchtext]] /d - delete line. /n - use only newline instead of cr/lf.

	/b - insert before searchtext. /a - insert after searchtext.
Copy a file	COPY sourcefile [targetfile] [/s][/y][/n][/q][/nr][/tm=n] [/tmzp=n] /s - span across folders /f - follow symlinks (local file system only) /xj - don't follow junctions/symlinks (Windows) /tm - Split file across multiple files with max size of n /tmzp - Zero padding to use for /tm option /y - overwrite all /n - overwrite none /c - ignore errors /q - quiet mode /nr - no protective recurse checking.
Extract a file or files from a cab archive	COPY CAB cabfile sourcefile targetdir [/y][/n][/l][/q] EXTRACT cabfile sourcefile targetdir [/y][/n][/l][/q] /y - overwrite all /n - overwrite none /l - list only /q - quiet mode
Rename a file	REN FILE orgfile newfile REN orgfile newfile /q – quiet mode
Delete a file	DEL FILE path [/y][/q] /y - do not confirm /q – quiet mode
Change directory	OPEN DIR path CD path
Create a directory	MD path [/s] ADD DIR path [/s] /s - create entire path as needed. /q – quiet mode
Remove a directory	DEL DIR path [/y][/s] RD path [/y][/s] /y - do not confirm /s - delete all files and sub-folders /q – quite mode (no error messages)
Print working directory	LIST DIR PWD

The `SET TEXTLINE` command can be used to add or delete a line in an existing text file, or even to create a text file containing a specified line of text. The following are some examples of usage:

`SET TEXTLINE file.txt "line 1"`	create file.txt, add “line 1” as the first line
`SET TEXTLINE file.txt "line 2"`	add “line 2” at end of file.txt
`SET TEXTLINE file.txt "line 2" /a`	same as above
`SET TEXTLINE file.txt "line 0" /b`	add “line 0” at beginning of file.txt
`SET TEXTLINE file.txt "line 1a" /a "line 1"`	add “line 1a” after “line 1”
`SET TEXTLINE file.txt "line 1b" /b "line 2"`	add “line 1b” before “line 2”
`SET TEXTLINE file.txt "line 2" /d`	delete “line 2”
`SET TEXTLINE file.txt "last line"`	add “last line” at end of file
`SET TEXTLINE file.txt "las*" /d`	delete line starting with “las” (as in “last line”)

The `EXTRACT` (or `COPY CAB`) command can extract one or more files from a cab file. This command can be useful in conjunction with the `COPY INF` command if the driver files (inf, sys, cat) are available only within a cab file, rather than as separate files.

`EXTRACT file.cab file.inf c:\thisdir`	extract file.inf from file.cab and place in c:\thisdir
`EXTRACT file.cab file.inf .`	extract file.inf from file.cab and place in current directory (note the period as the targetdir parameter)

The remainder of the commands work the same as (or very similar to) their operating system equivalents. Following are some basic examples of usage:

`DIR`	list contents of current directory
`DIR *.txt`	list all files with a .txt extension in current directory
`PWD`	display the current working directory
`COPY FILE filename a:`	copy filename to drive a:
`COPY c:\test\* c:\test2 /s`	copy contents of folder c:\test and all sub folders to c:\test2
`MD newdirectory`	create a directory named newdirectory in current directory
`REN FILE filename newfilename`	rename filename to newfilename
`REN FILE *.txt *.doc`	change all .txt extensions to .doc extensions
`DEL FILE filename /y`	delete filename, and do not confirm the deletion

2. Commands for Working with Drives and File Systems

Important: When using the Linux version of TBOSDT, all command line options are specified by using a “`-`” instead of a “`/`”. **Example:** `LIST HD 0 -f`

Command Function	Command Syntax
Open a file system (associate a file system with a drive). The drive must be a number 0 through 9.	OPEN FS drive: phydrivenum [partitionid] [/r][/n8d3][/q] OPEN FS drive: filename [partitionid] [/r][/n8d3][/q] MOUNT drive: filename [partitionid] [/r][/n8d3][/q] /r - open read-only (NTFS only) /n8d3 - disable SFN creation (NTFS only) /q - use volume sequence as volume id
Close a file system	CLOSE FS drive:

	UMOUNT drive:
Assign additional UEFI file systems a drive letter. UEFI version only.	OPEN UEFIFS
Close the additional UEFI file system opened. UEFI version only.	CLOSE UEFIFS
Display statistics for a file system	STATS d:
Display volume label for a file system	LIST VOL [d:]
Set the volume label for a file system. (Currently supported for mounted drives 0: through 9: and drive letters using the Windows version)	SET VOL "volume name" [d:]
List partitions on a hard drive	LIST HD hdnum [/f][/u][/q][/w][/s][/t][/a=n] /f - include free space in listing /u - include unallocated areas in listing /q - use volume sequence as volume id /w - use wide format /s - list disk signature(s) /t – list partition type instead of file system /a - alignment value in sectors (0=cylinder).
List partition details (1.45+ version)	LIST PART hdnum partid [/q] /q - use volume sequence as volume id
List a volume serial number for FAT/FAT32/NTFS partitions or UUID for Ext2/3/4 partitions (Pro Versions 1.50+ Only)	LIST VOLSN phyhdnum partid [/q] /q – use sequence for volume id
Open hard drive level software cache.	OPEN CACHE hdnum [size] [/l=readlimit] [/w=writelimit] [/p] [/z] /l - max io read/write request size to cache. /w - max io write request size to cache. (override /l) /p - enable partial lazy writing. /z - enable full lazy writing. Enabling lazy writing can result in unknown data loss. Always attempt to CLOSE CACHE when lazy writing is enabled to get a report of any write failures. The cache is automatically closed when the device it's no longer being used. When closed automatically, no write failure errors are reported.
Close hard drive level software cache or force it to flush to disk.	CLOSE CACHE hd [/f][/w] /f - force close (ignore lazy write errors). /w - flush lazy writes to disk only.
Copy a partition to a file	COPY PARTITION phyhdnum partid targetfile /b][/q] /b - setup for use with bootfile application. /q - use sequence for volume id.
Add a virtual FAT drive	ADD VIRTDRV filename sizeinmb [volname] [/b] [/c=n] [/h=n] [/s=n][/l][/w][/fat32][/blocksize=n][/w][/bs=filename] [/bsraw][/empty]

	/b - setup for use with bootfile application /c - last cylinder value. /h - last head value. /s - sectors per track value. /l - limit virtual drive file size. /w - treat warning as error. /empty – create an empty file. /fat32 – try using FAT32 instead of FAT as file system. /blocksize – size of a sector in bytes. /bs – name of file containing boot sector code to use. (note: fat/fat32 only) /bsraw – overlay raw boot sector (/bs) with existing BPB. The c h s values all need to be provided for them to be used. They can be used to create floppy sized virtual drives by making sizeinmb zero and providing the c, h, and s values (remember for c and h the last value is one less than the total count of cylinders or heads. e.g. h=254 means total of 255 heads).
Add a virtual PC VHD drive	ADD VHD filename sizeinmb [/f] /f – create a fixed type VHD drive.
Expand a virtual PC VHD drive	SET VHDSIZE filename sizeinmb
Add a VHDX drive	ADD VHDX filename sizeinmb [/f][/s4] /f – create a fixed type VHD drive. /s4 – Use sector size of 4KiB (4096 bytes). For Windows caching reasons, it is highly recommended to create partitions within a VHDX file that are aligned on at least a 1MiB boundary. That is either 2048 sectors for 512 byte sized sectors or 256 sectors for 4096 byte sized sectors.
Expand a VHDX drive	SET VHDXSIZE filename sizeinmb
Add a vmware VMDK drive	ADD VMDK filename sizeinmb [/s] /s – create a SCSI type VMDK drive.
Expand a vmware VMDK drive	SET VMDKSIZE filename sizeinmb
Add a VirtualBox VDI drive	ADD VDI filename sizeinmb [/f] /f – create a fixed type VDI drive.
Expand a VirtualBox VDI drive	SET VDISIZE filename sizeinmb
Change size of GPT area	SET GPTSIZE [sizeinmb]
Add a partition (Pro for BootIt Version) (fsid can be “fat12”, “fat16”, “fat”, “fat32”, “ntfs” or “exfat” starting in 1.82)	ADD PARTITION hdnum partid sizeinmib fsid name [fsb] [/q] [/n] [/f] [/e] [/a=n] [/c=n] [/h=n] [/s=n] [/b=n] [/u] [/w] [/bs=filename] [/bsraw] fsid - file system identifier (6=FAT,7=NTFS,11=FAT32,…) name - name for new partition (gpt/embr). fsb - free space before size in MiB (omit for end of block). /q - use sequence for volume id.

	/n - use \"normal\" geometry. /f - force ending head/sector in MBR. /e - align on end. /a - alignment value in sectors (0=cylinder). /c - last cylinder value. /h - last head value. /s - sectors per track value. /b - bytes per cluster /u - leave unformatted (default when fsid format not supported) /w - treat warning as error. /bs - name of file containing boot sector code to use. (note: fat/f32 only) /bsraw – overlay raw boot sector (/bs) with existing BPB.
Delete a partition (Pro for BootIt Version)	DEL PARTITION hdnum partid [/q][/y][/g] /q - use sequence for volume id. /y - no confirmation to perform delete. /g – leave MS partition gaps in GPT.
Slide a partition (Pro for BootIt Version)	SLIDE hd id [fsb] [/q] [/n] [/f] [/e] [/r] [/a=n] [/c=n] [/h=n] [/s=n] [/z] hd - hard drive number. id - partition id. fsb - free space before size in MiB (omit for end of block). /q - use sequence for volume id. /n - use "normal" geometry. /f - force ending head/sector in MBR. /e - align on end. /r - raw slide (move unused sectors). /a - alignment value in sectors (0=cylinder). /c - last cylinder value. /h - last head value. /s - sectors per track value. /z - no progress display.
Copy a partition (Pro for BootIt Version)	COPY PARTITION hd id thd tid [fsb] [/q] [/n] [/f] [/e] [/r] [/a=n] [/c=n] [/h=n] [/s=n] [/b] [/z] [/i] hd - source hard drive number. id - source partition id. thd - target hard drive number. tid - target partition id. fsb - free space before size in MiB (omit for end of block). /q - use sequence for volume id. /n - use "normal" geometry. /f - force ending head/sector in MBR. /e - align on end. /r - raw copy (copy unused sectors).

	/a - alignment value in sectors (0=cylinder). /c - last cylinder value. /h - last head value. /s - sectors per track value. /b - assume original hd for boot. /z - no progress display. /i - ignore lock failures.
Resize a partition (Pro for BootIt Version)	RESIZE hdnum partid [newsizemb] [/q] [/n] [/f] [/e] [/b] [/a=n] [/c=n] [/h=n] [/s=n] [/z] /q - use sequence for volume id. /n - use "normal" geometry. /f - force ending head/sector in MBR. /e - align on end. /b - resize beginning (extended partitions only) /a - alignment value in sectors (0=cylinder). /c - last cylinder value. /h - last head value. /s - sectors per track value. /z - no progress display.
Set a volume serial number for FAT/FAT32/NTFS partitions or UUID for Ext2/3/4 partitions (Pro Versions 1.50+ Only)	SET VOLSN phyhdnum partid serialnumber [/q] /q – use sequence for volume id.
Set a partition active	SET PART ACTIVE phyhdnum partid [/q] /q - use sequence for volume id.
Set the CHS values for a partition entry	SET PART GEO phyhdnum partid [/C=lastcyl][/H=lasthead][/S=sectorspertrack][/q] /q - use sequence for volume id.
Set the partition type (Pro for BootIt 1.45+ version)	SET PART TYPE phyhdnum partid fsid|{guid}[/q] /q - use sequence for volume id.
Set the partition attribute flag (Pro for BootIt 1.45+ version)	SET PART ATTR phyhdnum partid attrvalue[/q] /q - use sequence for volume id.
Set the partition ID (Pro for BootIt 1.45+ version)	SET PART ID phyhdnum partid id|{guid}[/q] /q - use sequence for volume id.
Set the partition name in the GPT/EMBR (Pro for BootIt 2.20+ version)	SET PART NAME phyhdnum partid name [/q] /q - use sequence for volume id.
Change the partition entry number/slot (Pro for BootIt 1.84+ version)	SET PART NUM phyhdnum partid entrynum entrynum is a value 0-n
Set a drives partitioning type (Pro for BootIt 1.45+ version)	SET DISKTYPE phyhdnum mbr | embr | gpt [/n] [/f] [/c=n] [/h=n] [/s=n] [/u] /n - use "normal" geometry. /f - force ending head/sector in MBR. /c - last cylinder value. /h - last head value. /s - sectors per track value.

	/u - unlimit primaries (EMBR related)
Delete EMBR (keeping MBR)	DEL EMBR phyhdnum
Set the Disk ID (the two commands are equivalent. "newid" can be either the disk GPT {guid} or MBR signature. The "set diskid" command was added in version 1.72)	SET DISKID phyhdnum newid SET MBR SIG phyhdnum newid
Change Windows BCD or MountedDevices references. (Pro Versions 1.57+ Only) (/b added in 1.82)	SET REF [BCD|MountedDevices|All] phyhdnum partid orgdiskref orgpartref [newdiskref newpartref] [/oss] [/dno] [/b] [/q] orgdiskref – original disk guid or MBR signature. orgpartref – original partition guid or starting lba. newdiskref – optional new disk guid or MBR signature. newpartref – optional new partition guid or starting lba. /oss – original sector size /dno – disable changing only new to original. /b – change BCD {boot} references. /q – use sequence for volume id. If a GUID is used for a partition or disk reference then both must reference a GUID.
Install MBR code on a hard drive	SET MBR CODE phyhdnum [type] type can be "standard" or "win7"
Copy sectors to/from a file.	COPY SECTORS phyhdnum lba numsectors file [/w][/y] /w – write sectors from file (otherwise read sectors to file) /y – skip confirmation when /w used. When /w is not specified this command reads sectors from a drive and saves them to a file. When /w is specified it reads data from a file and writes it to sectors on the drive. The /w parameter is intended for use by those who know exactly what they are doing. You must be very careful when using the /w parameter. It is only available in interactive mode and will abort in script mode.

TBOSDT has the ability to `OPEN` (or `MOUNT`) partitions on physical hard drives, as well as file systems contained within a file – such as an ISO file, or a file containing an image of a floppy diskette or hard drive. These drives are assigned a number **from 0 through 9** when opened, and then referenced in subsequent commands by using that number plus a colon. The following are some example sequences showing how these commands can be used:

Mounting and accessing a partition on physical hard drive 0 (requires admin privileges)

```
LIST HD                 list partitions on HD0 (includes partition IDs)
OPEN FS 4: 0 1          open partition with ID=1 on HD0 and assign it to drive 4:
DIR 4:                  list files on drive 4:
```

`STATS 4:`	display statistics for file system mounted at 4:
`DEL FILE 4:filename /y`	delete filename on drive 4: and do not confirm the deletion
`CLOSE FS 4:`	close drive 4:

Mounting and accessing a floppy diskette image file named floppy.img

`OPEN FS 2: floppy.img`	open (or mount) the floppy image as drive 2:
`COPY FILE 2:`	copy filename from drive 2: to current directory
`CLOSE FS 2:`	close drive 2: (close the image file)

Mounting and accessing an ISO file named terabyte.iso

`OPEN FS 3: terabyte.iso`	open (or mount) the ISO file as drive 3:
`COPY FILE 3:filename filename1`	copy filename on drive 3: to filename1 in current directory
`CLOSE FS 3:`	close drive 3: (close the ISO file)

Mounting and accessing a partition contained within a hard drive image file. In this case, the file is a VPC virtual hard drive file named disk1.vhd

`OPEN FS 5: disk1.vhd 1`	open (or mount) partition with ID=1 as drive 5:
`OPEN FS 5: disk1.vhd 0x1`	same as above, but using 0x1 format to specify the ID
`CLOSE FS 5:`	close drive 5: (close the virtual hard disk file)
`UMOUNT 5:`	same as above, but using the UMOUNT syntax

Note that all open file systems will be **closed automatically** when exiting TBOSDT (or when a script completes), without having to explicitly unmount or close them.

Copying a partition to a file

The `COPY PARTITION` command can copy an existing partition to a file. The resulting file will be the same size in bytes as the partition copied. The file can then be worked with in the same way as other files containing a file system (such as ISO files and floppy diskette images):

`COPY PARTITION 0 0xd3 part.img`	copy partition with ID=d3 on HD0 to the file part.img
`COPY PARTITION 0 0xd3 part.img /b`	same as above but setup for use with bootfile (adds 1 sector)
`MOUNT 2: part.img`	mount part.img and assign to drive 2:
`UMOUNT 2:`	unmount part.img

Creating a virtual FAT drive

The `ADD VIRTDRV` command will create a file containing a FAT file system. The resulting file can then be worked with in the same way as other files containing file systems:

`ADD VIRTDRV fat.img`	create 32 MB FAT file system in file fat.img
`ADD VIRTDRV fat.img 32 TEST`	same as above but specify “TEST” as the volume name

`ADD VIRTDRV fat.img 32 TEST /b`	same as above but setup for use with bootfile (adds 1 sector)
`OPEN FS 3: fat.img`	open (mount) fat.img and assign to drive 3:

When working with hard drives and hard drive image files, there are some additional commands (some have already been used above) that can accomplish tasks such as listing partitions on a drive, setting a partition active, and installing MBR code on the drive. These commands can be used without having to first `OPEN` or `MOUNT` the drive or drive image file. The following are some examples of these commands (these typically require admin/root privileges):

Creating a Virtual PC VHD drive

The `ADD VHD` command will create a file in the format compatible with the Virtual PC/Server software products.

`ADD VHD my.vhd 32`	create 32 MB VHD dynamic virtual drive.
`ADD VHD my.vhd 32 /f`	create 32 MB VHD fixed virtual drive.

Note that TBOSDT currently supports only dynamic and fixed drive types that reside in a single .VHD file. If the files are split (e.g. my.vhd, my.v01, my.v02) then the virtual hard drive will not be supported by TBOSDT. Splitting can occur when the VHD virtual drive is located on a FAT volume and exceeds 4GiB in size. In addition, the current Virtual PC software only supports VHD drives up to 128GiB. Hyper-V supports drives larger than 128GiB.

Expanding the size of a virtual PC VHD drive

Use `SET VHDSIZE` command to extend the size of a VHD virtual drive. The maximum size for a VHD drive that will work in Virtual PC is 128GiB. Hyper-V supports VHD files that are larger.

`SET VHDSIZE my.vhd 64`	extend the VHD drive to be 64 MB in size.

Creating a VMWare VMDK drive

The `ADD VMDK` command will create a file in the format compatible with the VMWare software products.

`ADD VMDK my.vmdk 32`	create 32 MB VMDK MonolithicSparse virtual IDE drive.
`ADD VMDK my.vmdk 32 /s`	create 32 MB VMDK MonolithicSparse virtual SCSI drive.

Note that TBOSDT currently supports only MonolithicSparse (and pre-allocated raw) drive types that reside in a single .VMDK file.

Expanding the size of a VMWare VMDK drive

Use `SET VMDKSIZE` command to extend the size of a VMDK virtual drive. The maximum size for a VMDK drive is 2TiB.

`SET VHDSIZE my.vhd 64`	extend the VHD drive to be 64 MB in size.

Listing partitions on a physical hard drive, or in a hard drive image file

`LIST HD 1`	list partitions on HD1 (includes partition IDs)
`LIST HD 1 /f`	same as above, but include free space in the listing
`LIST HD disk1.vhd`	list partitions in hard drive image file disk1.vhd

Setting a partition active on a physical hard drive, or in a hard drive image file

`SET PART ACTIVE 1 2`	set partition with ID=2 active on HD1
`SET PART ACTIVE disk1.vhd 4`	set partition with ID=4 active in hard drive image file disk1.vhd

Install MBR code to a physical hard drive, or to a hard drive image file

`SET MBR CODE 1`	install standard MBR code on physical hard drive HD1
`SET MBR CODE disk1.vhd`	install standard MBR to hard drive image file disk1.vhd

Setting the NT Disk Signature on a hard drive, or in a hard drive image file

The `SET MBR SIG` command will write the 4-byte NT disk signature to the MBR sector (1st sector) of a hard drive. The NT disk signature is located at bytes 0x1B8 through 0x1BB in the MBR sector. The disk signature can be cleared by writing zero. Here are some examples:

`SET MBR SIG 1 0x1A3C77D1F`	set the MBR sig to 0xA3C77D1F on physical hard drive 1
`SET MBR SIG 0 0x0`	set the MBR sig to 0 (clear the sig) on physical hard drive 0
`SET MBR SIG disk1.vhd 0x0`	set the MBR sig to 0 (clear the sig) in file disk1.vhd

Setting the CHS values for a partition entry

The `SET PART GEO` command will set one or more of the CHS values for a partition to specified values. The following is an example of how this command can be used:

`SET PART GEO 0 1 /h=254 /s=63`	for partid=1 on HD0, set last head to 254, sectors/track to 63

3. Commands for Working with Registry Files

Important: When using the Linux version of TBOSDT, all command line options are specified by using a "`-`" instead of a "`/`". **Example:** `DEL KEY 0 Key1 -s`

Command Function	Command Syntax
Open a registry file (0 through 4)	OPEN REG r path [/r][/f] /r – attempt recovery of dirty file. /f – force open when dirty.
Close a registry file	CLOSE REG r
Open a registry key (0 through 19)	OPEN KEY k keyname r [/k][/c] /k - r references a key instead of registry file. /c - create key if not found
Open a registry key (0 through 19) (Pro version only)	OPEN KEYORD k keyord r [/k] /k - r references a key instead of registry file.
Close a registry key	CLOSE KEY k

Associate the CurrentControlSet key with a numeric id k	OPEN KEYCCS k r
Import a registry key branch (REGEDIT4)	REGIMPORT filename r
Export a registry key branch (REGEDIT4)	REGEXPORT filename k [subkeyname]
List keys in a registry key	LIST KEYS keynum [subkeyname]
Copy a registry key	COPY KEY sourcekeynum targetkeynum [copyname]
Rename a registry key	REN KEY k newname
Add a registry key	ADD KEY keyname r [/k] /k - treat r as keynum instead of registry file.
Delete a registry key	DEL KEY k keyname [/s] /s – delete all subkeys
List values in a registry key	LIST VALUES keynum [subkeyname] [/d] /d - list value data
Display a value in a registry key	LIST VALUE keynum [subkeyname] valuename
Change or add a value in a registry key	SET VALUE k subkeyname valuename valuetype value valuetype: none sz, expandsz, hex, dword, multisz, qword
Delete a value in a registry key	DEL VALUE k valuename
Add/Remove a string from a multi-string registry value	SET MSZVALUE k subkeyname valuename string [/d][[/b][/a] [searchtext]] /d - delete string. /b - insert before searchtext. /a - insert after searchtext.

TBOSDT can work with registry files in the form of **hive files.** A hive file is a binary image of a registry key. This is as opposed to a reg file, which is a registry key saved in text format. Hive files can be created with the Windows Registry Editor by electing to export a selected key as a hive file, rather than as a reg file. The Registry Editor can similarly import hive files, as well as load and unload hives from certain registry branches.

Once a hive file is opened with TBOSDT, the registry keys within the hive can be opened, closed, listed, added, deleted, copied, and renamed. In addition, registry key values can be listed, added, deleted, and modified.

Opening and Closing a registry (hive) file

Opening a registry file requires assigning it to a number **from 0 through 4**. In the following example, the `OPEN REG` command is used to open reg.hiv and assign it to 2, and then `CLOSE REG` is used to close the file. When closing the file, the same number must be used as when opening it.

```
OPEN REG 2  reg.hiv          open registry file reg.hiv and assign it to 2
CLOSE REG 2                  close registry file assigned to 2
```

Opening, Closing, and Listing registry keys

Once a registry hive file has been opened, most of the other registry commands will require that one or more keys be opened before anything further can be done. To open a key, the `OPEN KEY` command is used to assign a number **from 0 through 19** to the key. The assigned number can then used in subsequent commands to reference that particular key in the registry file. To close the key, the `CLOSE KEY` command is used. The same number assigned when opening the key must be used to close it.

As an example, assume that a hive file named reg.hiv contains 2 keys named Key1 and Key2. The following sequence of commands can be used to open the registry file, open the entire hive as a key, list the keys in the hive, close the key, and then close the registry file.

```
OPEN REG 2 reg.hiv          open the registry file and assign it to 2
OPEN KEY 0 "" 2             open the entire hive as a key and assign it to 0
LIST KEYS 0                 list the keys in the hive (will list Key1 and Key2)
CLOSE KEY 0                 close the key assigned to 0
CLOSE REG 2                 close the registry file assigned to 2
```

When you close a registry file, **all open keys are automatically closed** at the same time. Therefore, it is not necessary to explicitly close all open keys first before closing a registry file. In addition, exiting from TBOSDT (or the completion of a script) will automatically close all open keys, as well as close all open registry files.

To open just Key1 (instead of the entire hive), the following sequence could be used instead:

```
OPEN REG 2 reg.hiv          open registry file and assign it to 2
OPEN KEY 0 Key1 2           open Key1 and assign it to 0
LIST KEYS 0                 list the keys in Key1 (will list the subkeys of Key1)
CLOSE REG 2                 close the hive file assigned to 2
```

If Key1 in the example above contained a subkey named SubKey1, and the goal was to open just SubKey1, that could be accomplished as follows:

```
OPEN REG 2 reg.hiv          open registry file and assign it to 2
OPEN KEY 0 Key1\SubKey1 2   open SubKey1 under Key1 and assign it to 0
LIST KEYS 0                 list the keys in SubKey1
CLOSE REG 2                 close the hive file assigned to 2
```

In addition, the `OPEN KEY` command's `/k` option allows for a key to be opened by referencing an already opened key, instead of a registry file. This is demonstrated in the example below:

```
OPEN REG 2 reg.hiv      open registry file and assign it to 2
OPEN KEY 0  "" 2        open the entire hive as a key and assign it to 0
OPEN KEY 1 Key1 0 /k    open Key1 and assign it to 1 (the /k means the 0 is a key, not a file)
CLOSE REG 2             close the registry file assigned to 2
```

Adding and Deleting keys

Keys can be added with the `ADD KEY` command, and can be added with or without first opening a key. The following sequence of commands will create a new key named Key3 in the root of the hive, and then a new subkey of Key3 named SubKey1.

```
OPEN REG 2 reg.hiv              open registry file and assign it to 2
ADD KEY Key3 2                  create Key3 in the root of the registry hive assigned to 2
ADD KEY Key3\SubKey1 2          create SubKey1 under Key3
CLOSE REG 2                     close the registry file assigned to 2
```

A key can also be added by using the `OPEN KEY` command with the `/c` option. The `/c` option will create the key if it doesn't already exist. The example below will create and open Key3:

```
OPEN REG 2 reg.hiv              open registry file and assign it to 2
OPEN KEY 0 Key3 2 /c            create Key3, open it, and assign it to 0
CLOSE REG 2                     close the registry file assigned to 2
```

Deleting a key is accomplished with the `DEL KEY` command, and requires that a key first be opened (opening first is optional when adding a key). If a key contains one or more values, those will be deleted automatically when the key is deleted. If a key contains one or more subkeys, those can be deleted at the same time by using the `/s` option. Note that a `DEL KEY` command will fail if there are subkeys below it and the `/s` option is not used. In the following example, Key3 that was added in the example above is now deleted, along with any subkeys it contains:

```
OPEN REG 2 reg.hiv              open registry file and assign it to 2
OPEN KEY 0 ""                   open the entire hive as a key and assign it to 0
DEL KEY 0 Key3 /s               delete Key3 and all subkeys below it (/s)
CLOSE REG 2                     close the registry file assigned to 2
```

Copying and Renaming keys

An opened key can be renamed with the `REN KEY` command. The example below renames the key opened as Key3 to Key 4.

```
OPEN REG 2 reg.hiv              open registry file and assign it to 2
OPEN KEY 0 Key3 2               open Key3 and assign it to 0
REN KEY 0 Key4                  rename Key3 to Key4
CLOSE REG 2                     close the hive file assigned to 2
```

The `COPY KEY` command can copy a registry key by first opening both the key to be copied (the source), and the key to be copied to (the target). This is best demonstrated by the example below, which copies a key named Key3 from one registry file to another. When a key is copied, all subkeys and values in the key are copied along with it:

```
OPEN REG 2 reg1.hiv             open the source registry file, assign it to 2
OPEN KEY 0 Key3 2               open Key3 in the source registry file, assign it to 0
OPEN REG 3 reg2.hiv             open the target registry file, assign it to 3
```

`OPEN KEY 1 "" 3`	open the entire hive of reg2.hiv as a key, assign it to 1
`COPY KEY 0 1 Key3`	copy Key3 in reg1.hiv to reg2.hiv, and name the copy Key3
`CLOSE REG 2`	close the source registry file
`CLOSE REG 3`	close the target registry file

In the example above, the copy of Key3 from reg1.hiv will now appear in the "root" of reg2.hiv, and it will be named Key3. To clarify, the copy of the key in reg2.hiv could have also been named Key4 (or any other name) by using the command '`COPY KEY 0 1 Key4`' instead. Keys can be copied from one location to another within the same registry file, as well as from one registry file to another.

Listing Values names and Value data in a registry key

The values contained in a key (or subkey) can be listed by using the `LIST VALUES` command. The `/d` option can be used to display the value data in addition to the value names. The following sequence demonstrates how the `LIST VALUES` command can be used:

`OPEN REG 2 reg.hiv`	open registry file and assign it to 2
`OPEN KEY 0 Key1 2`	open Key1 and assign to 0
`LIST VALUES 0`	list value names in Key1
`LIST VALUES 0 /d`	list value names and value data in Key1
`LIST VALUES 0 SubKey1 /d`	list value names and data in SubKey1 under Key1
`CLOSE REG 2`	close the hive file assigned to 2

The `LIST VALUE` command is also available, and is used to display one particular value in a key or subkey. The following commands would display the data contained in the value named Value1.

`OPEN REG 2 reg.hiv`	open registry file and assign it to 2
`OPEN KEY 0 Key1 2`	open Key1 and assign to 0
`LIST VALUE 0 Value1`	display data in Value1 in Key1
`LIST VALUE 0 SubKey1 Value1`	display data in Value1 in Subkey1
`CLOSE REG 2`	close the hive file assigned to 2

Setting a Value in a registry key

The `SET VALUE` command can be used to either modify existing values in a key, or to create new ones. The following sequence of commands will first create a new value named Value1 in Key1, and then modify the value of Value1.

`OPEN REG 2 reg.hiv`	open registry file and assign it to 2
`OPEN KEY 0 "" 2`	open the entire hive as a key and assign it to 0
`SET VALUE 0 Key1 Value1 sz string1`	create Key1/Value1 of type sz with a value of "string1"
`SET VALUE 0 Key1 Value1 sz string2`	change data in Value1 to "string2"
`CLOSE REG 2`	close the hive file assigned to 2

Deleting a Value in a registry key

To delete a value, the DEL VALUE command is used. This command will only work on values in the open key itself (does not work on subkeys). The following sequence of commands will delete Value1 in Key1.

```
OPEN REG 2 reg.hiv              open registry file and assign it to 2
OPEN KEY 0 Key1 2               open Key1 and assign it to 0
DEL VALUE 0 Value1              delete Value1
CLOSE REG 2                     close the hive file assigned to 2
```

Adding or Removing a string from a multi-string registry value

SET MSZVALUE is a special command to work with multi-string registry values. It can add/delete a string based on search criterion similar to the SET TEXTLINE command. The following examples use SET MSZVALUE to add/delete strings in a multi-string value named V1 in a key named Key1:

```
OPEN KEY 0 Key1 2                       open Key1 and assign to 0
LIST VALUES 0 /d                        list all value names and their values in Key1
SET MSZVALUE 0 "" V1 "xyz"              adds string “xyz” as the last string
SET MSZVALUE 0 "" V1 "xyz" /a           same as above
SET MSZVALUE 0 "" V1 "abc" /b           adds string “abc” as the first string
SET MSZVALUE 0 "" V1 "abc" /b "def"     adds string “abc” before string “def”
SET MSZVALUE 0 "" V1 "abc" /a "def"     adds string “abc” after string “def”
SET MSZVALUE 0 "" V1 "abc" /d           deletes string “abc”
SET MSZVALUE 0 "" V1 "Hello World" /d   deletes string “Hello World”
```

Importing and Exporting registry keys

A registry key branch can be exported from an open registry hive file by using the REGEXPORT command. The exported file will be in reg file format (text). Similarly, reg files can be imported into a registry hive file by using the REGIMPORT command. In the following example, the registry key branch named Key3 is exported to a file named key3.reg, and then that file is imported into another registry file:

```
OPEN REG 2 reg.hiv              open registry file reg.hiv and assign it to 2
OPEN KEY 0 Key3 2               open Key3 and assign it to 0
REGEXPORT key3.reg 0            export Key3 to a file named key3.reg
OPEN REG 3 reg2.hiv             open registry file reg2.hiv and assign it to 3
REGIMPORT key3.reg 3            import file key3.reg to registry file reg2.hiv
CLOSE REG 2                     close the hive file assigned to 2
CLOSE REG 3                     close the hive file assigned to 3
```

Note that a key must first be opened before exporting it, while keys are imported into a registry file by just opening the hive file, and then importing the key branch into the hive. The REGIMPORT and REGEXPORT commands support the **REGEDIT4** (ANSI) format for registry files, but not REGEDIT5 (Unicode). Files imported with REGIMPORT must be in REGEDIT4 format.

4. Commands for Working with the Registry on a Running Windows System

NOTE: All commands in this section are available **only in the Windows version** of TBOSDT

Command Function	Command Syntax
Associate a registry key with a numeric id k.	OPEN WINKEY k keyname base [/c][/r] /r – open read-only. /c - create key if not found. base - HKLM, HKCR, HKU, HKCU.
Remove the numeric id associated with a registry key.	CLOSE WINKEY k
Display keys found in another key.	LIST WINKEYS keynum [subkeyname]
Delete a registry key.	DEL WINKEY k keyname [/s] /s - delete all subkeys.
List value names located in a key.	LIST WINVALUES keynum [subkeyname] [/d] /d - list value data
List the contents of a single value in a registry key.	LIST WINVALUE keynum [subkeyname] valuename
Set a registry value.	SET WINVALUE k valuename valuetype value ... valuetype: none, sz, expandsz, hex, dword, multisz, qword
Delete a registry value.	DEL WINVALUE k valuename

Opening a registry key, listing keys, and closing the registry key (Windows version only)

The `OPEN WINKEY`, `LIST WINKEYS`, and `CLOSE WINKEY` commands are similar to the `OPEN KEY`, `LIST KEYS`, and `CLOSE KEY` commands described in section 3, except that they are working with the registry of a running Windows system (as are all commands in this section). The following example opens the System key, assigns the key to 0, lists any subkeys contained within the key, and then closes the key. The System key is located in HKLM branch of the registry (HKEY_LOCAL_MACHINE). Note that the branch of the registry under which the key exists must be specified when using the `OPEN WINKEY` command.

`OPEN WINKEY 0 System HKLM`	Open the System key in HKLM branch and assign to 0
`LIST WINKEYS 0 System`	List subkeys in the System key
`CLOSE WINKEY 0`	Close the System key assigned to 0

Creating and deleting a registry key (Windows version only)

The following example uses the `OPEN WINKEY` command with the `/c` option to create and open a new key under the System key in the HKLM branch of the registry. It then creates a new subkey under the new key. It then opens the System key itself and deletes both the new key and the subkey under it with the `DEL WINKEY` command using the `/s` option.

`OPEN WINKEY 0 System\Test HKLM /c`	Create & open Test key under System and assign it to 0

`OPEN WINKEY 1 System\Test\SubKey HKLM /c`	Create & open SubKey key under the Test key
`OPEN WINKEY 2 System HKLM`	Open the System key itself
`DEL WINKEY 2 Test /s`	Delete the Test key and the SubKey key under it

Listing, Setting, and Deleting Registry Values (Windows version only)

The `LIST WINVALUE`, `LIST WINVALUES`, `SET WINVALUE`, and `DEL WINVALUE` commands work in similar fashion to the `LIST VALUE`, `LIST VALUES`, `SET VALUE`, and `DEL VALUE` covered in section 3. The following example commands demonstrate their use.

`OPEN WINKEY 0 System\Setup HKLM /c`	Open the System\Setup key and assign it to 0
`LIST WINVALUE 0 CmdLine`	List the value of CmdLine in Setup key
`LIST WINVALUES 0`	List the names of all values in Setup key
`LIST WINVALUES 0 /d`	List the names and values of all values in Setup key
`SET WINVALUE 0 SetupType dword 0x1`	Set the value of SetupType value in Setup key to 0x1
`SET WINVALUE 0 NewValue sz "Hello World"`	Create NewValue of type sz, set to “Hello World”
`DEL WINVALUE 0 NewValue`	Delete NewValue in Setup key

5. Commands Associated with Windows Driver Installation

Important: When using the Linux version of TBOSDT, all command line options are specified by using a “`-`” instead of a “`/`”. All commands in this section except `COPY INF` are available only in the Windows version.

Command Function	Command Syntax
Copy an INF file and optionally install a driver	COPY INF inffile targetwindir windir r [section r] [/ia64] [/x64] [/i] [/b] [/r] [/n] [/f] [/e] [/in] [/nr] [/ver][/pt][/sm] *inffile* –path to the INF file. *targetwinddir* – path to the target windows folder (for COPY INF). *windir* – path of the windows folder from within the target itself. In other words the path from the point of view of the target once booted. *r* – open registry id of the system registry. *section* – extension to use on the end of the Manufacturer section to allow specialized INF file sections. e.g. [Manufacturer.section] *r* – open registry id of the software registry. /ia64 - install for IA64 architecture (default x86) /x64 - install for amd64 architecture (default x86) /i - install the driver /b - driver being installed is required for system bootstrap /r - remove existing hardware reference to device /n - do not copy inf to target /f – filter on the hardware id /e – error if section is missing otherwise just warning message /in – process include and needs directives

	/nr – disable DirID redirection (i.e. 13 to 12) /ver – target os version in format of 0xMMMMmmmm where MMMM is the major version and mmmm is the minor version. This is needed when the INF file uses versioning information. The target version information is as follows: 0x60002 = Windows Server 2012 and Windows 8 0x60001 = Windows Server 2008 R2 and Windows 7 0x60000 = Windows Server 2008 and Windows Vista 0x50002 = Windows Server 2003 0x50001 = Windows XP 0x50000 = Windows 2000 /pt – target os product type: 1 = Workstation 2 = Domain Controller 3 = Server /sm – target os suite mask (not normally used): 0x001 = Small Business 0x002 = Enterprise 0x004 = Back Office 0x008 = Communications 0x010 = Terminal 0x020 = Small Business Restricted 0x040 = Embedded NT 0x080 = Data Center 0x100 = Single User TS 0x200 = Personal 0x400 = Service Appliance
Add an INF and associated files to the system. (Windows version only)	ADD INF d:\filename [sourcemediapath] [/r][/d][/c][/n] /r - refresh (replace-only). /d - delete source after copy. /c - copy CAT file only. /n - no overwrite.
Remove the INF and associated files from the system. (Windows version only)	DEL INF filename [/f] /f - force removal.
Add a CAT file to the catalog database. (Windows version only)	ADD CATALOG d:\catfilename [newname]
Remove catalog file reference from catalog database. (Windows version only)	DEL CATALOG catfilename

Installing a Windows driver with COPY INF (any version)

The COPY INF command can be used to install a Windows driver to an inactive (not running) Windows installation by mounting the partition from TBOSDT, opening the system hive file, and then running COPY INF. The INF file and its associated driver files (typically a SYS file and a CAT file) must be in the same directory. Note that while the COPY INF command will install the drivers and allow the system to boot, it will be necessary to use the additional ADD INF command to automate completion of the driver installation. See example 2 in the TBOST Application Examples section for additional information on installing a Windows driver.

```
MOUNT 0: 0 1                                    mount partition with ID =1 on drive HD0
OPEN REG 0 0:\windows\system32\config\system     open the system registry hive
COPY INF vmscsi.inf 0:\windows c:\windows 0 /i /b  Install the driver, /b indicates driver is
                                                   required for boot
CLOSE REG  0                                    close the system registry hive
UMOUNT  0:                                      unmount the partition
```

Using ADD INF to complete a Windows driver installation (Windows version only)

The ADD INF command will add an INF and associated files to the Windows operating system. You must supply the full path to the INF file. A practical use of this command is to aid in the automated completion of a new driver installation after a COPY INF command has been executed for the same driver. This example assumes the full set of driver files is located in c:\drivers\intel

```
ADD INF c:\drivers\intel\iaahci.inf c:\drivers\intel     add the driver files (if necessary)
```

6. Additional Commands

Important: When using the Linux version of TBOSDT, all command line options are specified by using a "-" instead of a "/".

Command Function	Command Syntax
Exit from TBOSDT or continue execution of a script.	EXIT
Clears the TBOSDT window or screen.	CLS
Elevate TBOSDT in windows	ELEVATE params
Run a TBOSDT script (or TBScript if .tbs extension)	RUN scriptname
Run a TBScript or native program (Pro Version Only)	RUNTBS scriptname EXEC program.exe
Boot a UEFI Kernel Loader. (UEFI Version Only)	BOOT loadername.efi
Pause script execution and go to interactive mode (type EXIT to continue)	INTERACTIVE
Go to interactive mode after the script ends (only useful from scripts started from the OS command line)	STAY
Reboot or shutdown the system	REBOOT [/f][/s] [/f] – force reboot (if available). [/s] – shutdown

Display a value from an INI file	LIST INI filename sectionname valuename
Set a value in an INI file (Omit value to delete valuename from the file)	SET INI filename section valuename value
List a UEFI Boot Variable By default only the boot items in bootorder are listed, use /a option to search for all. The effective secure boot value is return unless /a is used which lists only the actual value. Under Linux efivarfs must be mounted on the kernel x64/ia32 matching the UEFI type (x64/ia32). The Linux command is: mount -t efivarfs none /sys/firmware/efi/efivars	LIST UEFI BOOTITEMS [/a] [/a] – Search for all items. LIST UEFI BOOTORDER LIST UEFI CURRENT LIST UEFI TIMEOUT LIST UEFI SECUREBOOT [/a] – Actual setting
Set a UEFI Boot Variable bootnum = 0x#### as listed in *list uefi bootitems* index = 0-n where to place in the list.	SET UEFI BOOTORDER bootnum index SET UEFI BOOTNEXT bootnum SET UEFI TIMEOUT num_seconds
List UEFI Boot Parameter String for Program	LIST UEFI STARTPARAMS
Set UEFI Boot Parameter String for Program or delete it if no parameter provided. To ignore the set startparams on startup, press / hold the “end” key just as the black screen prior to booting appears. (Example with embedded quotes: “””c:\path with space\startup.tbs”””	SET UEFI STARTPARAMS “param string”
Add UEFI Boot Item (Pro for BootIt Version)	ADD UEFI BOOTITEM hdnum partid description [bootnum] [filepath] [/u] [bootnum] – default 0 to auto find open entry [filepath] – default \efi\boot\bootx64.efi [/u] – update existing entry.
Delete UEFI Boot Item (Pro for BootIt Version)	DEL UEFI BOOTITEM bootnum [-k] [-k] – delete hot key if exists
Copy UEFI Boot Item (Pro for BootIt Version)	COPY UEFI BOOTITEM frombootnum tobootnum [--del] [--del] – Move instead of copy.
Default to using sequence for volume id	SET OPTION SEQ b
Prevent lockup when accessing certain USB controllers	SET OPTION USBLIO b
Change how a quote around switches is handled. (e.g. when enabled “-x” will not be considered a switch)	SET OPTION NOQUOTESWITCH b
Enable/Disable most status and error messages	SET OPTION QUIET b
Use volume labels for partition names (on by default)	SET OPTION USEVOLLAB b
Use volume locking during mount (Windows/TBOS Only).	SET OPTION LOCKING b b=0=No Locking b=1=Normal Locking (default)

	b=2=Lock with Dismount.
Flush volumes during mount (Linux/DOS)	SET OPTION LOCKING b b=0=No Flushing (default) b>0=Flush.
Set the default sector alignment for partitions.	SET OPTION SECTORALIGN n n=0 (Cylinder),1,2,4,8,16,32,…,2048
Change code page for console and conversions. (Console code page is only accurate under Windows. It's recommended that you only use conversion code pages of 65001 (UTF8) or -1 (Auto))	SET OPTION CODEPAGE w n w=1=Console w=2=Conversions n=Code Page
Change locale. (To use the OS locale Date/Time under OS environment use the locale 6 option. Use setting 0 returns to using default)	SET OPTION LOCALE l l=0=Reset to defaults l=1=Use ISO8601 for Date/Time l=2=Default Date/Time – No zero prefix l=3=Default Time – Remove AM/PM space l=4=Default Time – Use lower case l=5=Pull in locale from environment l=6=Use OS locale for Date/Time

Commands for working with INI files

There are two special commands that can be used with Windows INI type files. The `LIST INI` command can display a value defined in the file, while the `SET INI` command can change or delete a value. The following are examples of how these commands can be used:

```
LIST INI system.ini drivers wave          display value of "wave" in "drivers" section
SET INI system.ini drivers wave file.dll  change value of "wave" to "file.dll"
SET INI system.ini drivers wave           delete the "wave" value name from the file
```

The difference between BOOT and EXEC in the UEFI Version

The `BOOT` and `EXEC` commands are very similar in that both will start a UEFI application (which typically has an .EFI extension). However, the `BOOT` command differs in the following ways:

1) It does not allow parameters
2) It will unload and remove all drive assignments (0: to 9: as well as C: to Z:)
3) It makes all UEFI file systems available to the UEFI application
4) It expects to be launching a kernel loader that will not return.

If the kernel loader returns control to the program, the C: drive will be made available again; however, all other drive assignments would need to be reestablished.

Executing a script from interactive mode, or from another script

The `RUN` command will execute a script, either from the command prompt, or from another script. If executed from the command prompt, you will be returned to the prompt when execution completes. If a `RUN` command is executed from a script, it will return to the calling script when

execution completes, and continue where it left off. Note that when executing a script with the `RUN` command, all open file systems, registry files, and registry keys **will remain open** when the script completes, unless explicitly closed by the script. This differs from running a script from the OS command line, where all open items will be closed automatically when the script completes.

Executing TBScript or native programs

The Professional (paid) version of TBOSDT includes the TBScript™ engine which allows you to build scripts that include logical and mathematical operations. For complete details see the TBScript manual and the **TBOSDT Professional - TBScript Extensions** section of this manual.

Pausing execution of a script

The `INTERACTIVE` command can be used to pause execution of a script to examine the results to that point, or to manually execute some commands for testing or debugging purposes. All mounted file systems, open registry files, and open registry keys will remain as they were before the pause. To continue execution of the script from where it left off, type `EXIT`.

Rebooting or shutting down the system

The `REBOOT` command can be used to reboot or shut down the system. Using `REBOOT` without any parameters will reboot the system, while `REBOOT /s` will shut it down. The `/f` option can be used to force a reboot if necessary, and if available.

Exiting interactive mode

The `EXIT` command will exit TBOSDT when running in interactive mode. All mounted (open) file systems, registry files, and registry keys will be closed. In the case where the `INTERACTIVE` command is used to pause script execution (as explained above), the `EXIT` command will cause TBOSDT to continue script execution at the point where it left off.

TBOSDT Application Examples

Installing all IDE Controller Drivers Included in Windows XP

This example covers on a situation where an existing image of Windows XP needs to be restored and booted from a target system with a different IDE controller for the boot drive. The IDE driver required for the new controller is included with Windows, but is not installed in the backup image. An example of this would be if the motherboard failed on a system, and the replacement motherboard used a different IDE controller. Since the driver required for the new controller is not installed in the existing backup image, restoring the image and attempting to boot will result in a blue screen and Stop error 0x0000007B (unrecognized boot device).

A procedure for installing all IDE drivers included with Windows XP is outlined in MS KB article 314082 (http://support.microsoft.com/?kbid=314082). The KB article procedure extracts all IDE controller drivers included with Windows from %SystemRoot%\DriverCache\I386\Driver.cab to the %SystemRoot%\System32\Drivers directory, and then does a registry merge to create the necessary registry keys for each driver.

The same result can be achieved with TBOSDT, and it can be done on a restored partition that is currently unable to boot. To accomplish this, the **xp_ide.run** script has been included in the scripts directory. The script uses the `EXTRACT` command to extract the required driver files, and then uses a series of registry commands to create the necessary keys and values in the system hive.

This script is intended to be run with the target Windows XP partition in an **inactive** (not running) state. The script assumes that the OS is Windows XP SP2, and can be executed as is from either the DOS or Windows version of TBOSDT. To run it from the Linux version, all instances of "`/`" for the command options used would have to be changed to "`-`".

Before the script can be executed, **2 lines must be edited** near the beginning of the script. The first is the `MOUNT` command, which must be edited to reflect the correct drive and partition ID for the target partition. The second is the precautionary `EXIT` command, which must be commented out or deleted to allow the remainder of the script to be executed. Both lines to be edited are near the beginning of the script, and are commented.

The **xp_ide.run** script makes the following assumptions:

1. The OS is Windows XP SP2. It extracts the required driver files first from the driver.cab file, and then from the sp2.cab file. Both files are located in the Driver Cache folder. This ensures that all driver files are installed, and also that all are the latest versions as of Service Pack 2.

2. The Windows directory on the target partition is named “Windows” (i.e. c:\windows).

To execute the script: `tbosdt xp_ide.run`

Installing a Mass Storage Driver not included with Windows

This example covers a situation similar to the example above, except that the driver required for the new mass storage controller is not included with Windows. The starting point for this example is that an existing image has been restored to the target partition, but attempting to boot from the restored partition results in a blue screen (unrecognized boot device), or the system automatically reboots shortly after the Windows logo is shown. In this case, the vendor-supplied driver required can be installed in a **two-stage process** requiring 2 TBOSDT scripts. The first stage (Stage 1) is needed to install the driver and make the partition bootable. It’s also used to automate running of the second stage script. The second stage (Stage 2) is required to automate the completion of the driver installation once Windows has booted.

Stage 1: The two essential functions of the Stage 1 script are to install the driver and make the target partition bootable from the new controller, and then to set it up so that the Windows version of TBOSDT will execute the Stage 2 script the first time the partition is booted from. The following is a summary of how it accomplishes this:

- Copy (or extract from a CAB file) the vendor-supplied driver files to a folder on the target partition. This step is optional, but since both Stage 1 and Stage 2 need to have the full set of driver files available, it may be more convenient to do this rather than (for example) accessing them from a floppy drive or CD drive both times. A set of driver files typically includes an INF file, a SYS file, and a CAT file.
- Run the `COPY INF` command to install the driver. This command makes the target partition bootable from the new controller, although the full driver installation is not yet complete. Note that the full set of driver files must be in the current directory when the `COPY INF` command is executed.
- Setup Windows to run the Windows version of TBOSDT and the Stage 2 script on first boot.
- Copy the Windows version of TBOSDT and the Stage 2 script to a folder on the target partition.

The requirements for running the Stage 1 script are as follows:

- A boot disk (or another OS on the same system) to run TBOSDT from. Any of the TBOSDT OS versions can be used to execute the Stage 1 script – whichever is most convenient.
- The vendor-supplied driver files for the new controller must be available at a location accessible from TBOSDT. They can either be included on the boot disk, or can be on a drive or partition that TBOSDT can access (floppy drive, CD drive, hard drive partition, etc.)

- The Windows version of TBOSDT must be available to copy to the target partition.
- The Stage 2 script must be available to copy to the target partition.

Stage 2: The essential function of the Stage 2 script is to automate completion of the driver installation. When the Stage 2 script completes, Windows will reboot automatically, and then boot up in normal mode. The following is a summary of how it accomplishes this:

- Run the `ADD INF` command to add the INF and associated files to Windows.
- When the script completes, Windows will reboot the system automatically.

The requirements for running the Stage 2 script are as follows:

- It must be executed while Windows (the target partition) is running, and therefore must be executed by the Windows version of TBOSDT.
- The full set of vendor-supplied driver files must be available, and all together in one directory. As mentioned above in the description of Stage 1, it may be most convenient to have the Stage 1 script copy (or extract from a CAB file) the files to a directory on the target partition, and have Stage 2 access them from that location.

When the Stage 2 script completes, Windows will reboot automatically. On the next boot, Windows will boot up normally to the desktop. Typically, you will see one or more notifications that new hardware has been installed, and then be prompted to reboot. On reboot, the driver installation should be complete. Note that in some cases, Windows Product Activation (WPA) will be tripped because of the hardware changes, and will require reactivation.

To better demonstrate this process, two scripts have been included in the scripts directory. These scripts are named **drvins_1.run** for Stage 1 and **drvins_2.run** for Stage 2. The scripts include the actual TBOSDT commands needed to install the driver for the *Intel 82801GR/GH SATA AHCI Controller*, which is being used in the scripts as an example. The scripts are commented to describe each step taken.

Most of the commands in the scripts are commented out, because they are specific to the Intel SATA driver. The scripts can be used either as examples to develop other scripts from, or can be used after editing and uncommenting the appropriate lines to accommodate another driver.

Disabling a CPU specific service.

Disabling a CPU specific service such as INTELPPM or AMDK7 can easily be accomplished by setting that services `start` value to `4` (disable).

Here's an example of disabling INTELPPM with the assumption that the target windows partition is located on HD0 with a partition ID of 1 and Windows is installed in \Windows.

```
mount 0: 0 0x1
open reg 0 0:\windows\system32\config\system
open keyccs 0 0
```

```
set value 0 "services\intelppm" "Start" dword 4
```

TBEXPLO.TBS Script (Pro Version Only)

This script is a mini file manager that allows you to view files and directories; copy, delete, rename, manage attributes of files and directories; copy windows protected files; search for files and directories; explore Windows registries, including delete, rename, create, and copy keys and values; search for keys and values, etc. from DOS, TBOS, Linux, Windows, or UEFI environment.

This script also allows you to manage disks and partitions (delete, resize, slide, copy, change disk type, change disk IDs, change GUIDs, etc.).

SETWINDL.TBS Script (Pro Version Only)

This script is used to assign drive letters to partitions from outside of Windows. This can come in handy where a system won't boot or you can't sign in due to incorrect or changed drive lettering. Some common situations are unable to boot up a server because directory services won't start, or unable to sign in (logs in then back out right away) because the paging file can't be found. This can also be used to change the drive letter Windows is using.

INISTART.TBS Script (Pro Version Only)

This is a beta script to enable resetting Windows default startup values. This may be useful when the system has been modified by a virus, spyware, or other application. Please contribute your knowledge and submit script enhancements to TeraByte support.

SAVEREG.TBS/RESTOREG.TBS Scripts (Pro Version Only)

These scripts are used to backup and restore Windows registry files. Please contribute your knowledge and submit script enhancements to TeraByte support.

COPYWIN.TBS (Pro Version Only)

This script is used to perform a file-based copy of a Windows installation from a physical drive or a backup image file to another physical drive. It can be used to:

- Copy Windows from a MBR type disk (or GPT) to a GPT disk (or MBR), while changing (or not) the disk type of destination, and configuring Windows to boot.
- Copy Windows between disks of different sector size (for example, from a disk with 512b sector to a disk with 4k sector, or vice-versa).

- Copy Windows from a larger disk to a smaller disk. This type of operation is detailed in "Method 4" of the following KB article: https://kb.terabyteunlimited.com/kb-articles/restoring-to-a-smaller-drive-or-partition/

This script can only be run in a Windows environment.

CHGDTYPE.TBS (Pro Version Only)

This script is used to convert disk types, including disks that contain a Windows installation.

- Convert from GPT to MBR or EMBR with unlimited primaries (depending on the number of partitions).
- Convert from MBR to GPT.
- Convert from EMBR to GPT or MBR (if less than 4 partitions).
- On MBR type disk, it can also be used to remove the *System Reserved* partition and configure the main system partition to be active.

Details on using this script can be found in the following KB article: https://kb.terabyteunlimited.com/kb-articles/convert-a-disk-from-gpt-to-mbr-or-mbr-to-gpt-using-the-chgdtype-tbs-script/

COPYP2V.TBS (Pro Version Only)

Use this helpful script to automate converting a physical machine to a virtual machine. Requires the appropriate version of the imaging program (Image for Windows, Image for DOS, Image for Linux, Image for UEFI) be installed or available in the current directory along with OSDTOOL.TBS.

USBBOOT.TBS (Pro Version Only)

This script is used to enable Windows to boot from USB drives. The version included with the TBOSDT package supports Windows Vista or later. For Windows 2000 and XP, a version with additional files can be downloaded from the TBOSDT web page at the www.terabyteunlimited.com website.

TBIBCKUP.TBS (Pro Version Only)

This script is used to back up all partitions of all drives. It can be run under Windows, DOS, Linux, or UEFI. If no command line options are specified when the script is run you will be prompted to select the destination drive (if not automatically found) and asked to confirm the backup operation.

The following command line parameters can be used:

- `/d` Used to specify the partition to store the backup files. This is optional if backups were already done (either by the script or by Simple Operations mode of one of the imaging programs) as the script will search for the partition containing the backup folder.

  You can specify the destination partition using any of the following formats:
  - The hd num and partition id: `/d:hd@pid`
  - The hd sig (starting with 0x or #) and the partition id: `/d:#hdsig@pid`
  - The hd GUID and the partition GUID: `/d:hdguid@guid`
  - The partition GUID (only): `/d:partguid`
  - The hd num, hd sig, or hd GUID only if Simple Operations backups were already done on the disk: `/d:hdref`

  For example:

  ```
  runtbs tbibckup.tbs /d:3@0x05
  tbosdtw.exe tbibckup.tbs /d:{FE173FDD-1BA9-D14B-49A9-CD5869B69A19}
  tbosdt tbibckup.tbs --d:0x54AF25B1
  ```

- `/uy` Use this parameter to have the script not ask for confirmation before backing up all the partitions.

  If the `/d` parameter is not specified and no backups were already done, the script will ask for the destination partition, but won't ask for confirmation.

  If the `/d` parameter is not specified, but backups were already done in Simple Operation mode (`\TeraByte_TBI_Backups` already exists on a partition), the script won't ask for the destination (it will use the partition containing the previous backups) and it won't ask for confirmation.

  If the `/d` parameter is specified in conjunction with `/uy`, then the script will create the backups directly to the specified destination partition.

  For example:

  ```
  tbosdtw.exe tbibckup.tbs /uy /d:{FE173FDD-1BA9-D14B-49A9-CD5869B69A19}
  tbosdtw.exe tbibckup.tbs /uy
  tbosdt tbibckup.tbs -uy
  ```

The FIXBOOT.TBS Script

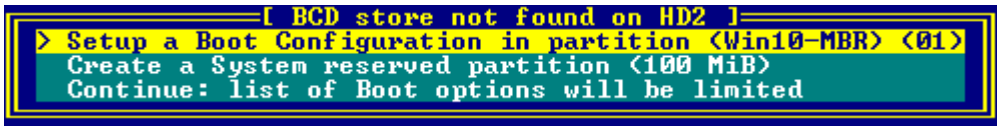
(Pro Version Only)

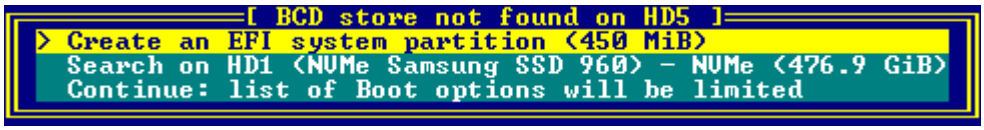
This script allows the Windows boot options to be changed as needed to aid in repairing or troubleshooting booting issues for the selected Windows system. If the BCD store for the Windows system isn't found, you'll be given the choice of continuing with limited options or setting up a Boot Configuration. If it is found, you'll also be given the option to repair it.

If the script is run directly you will be presented with a menu from which to select the desired Windows system. If the script is called from the osdtool.tbs script, this step will be skipped and the selected Windows system for osdtool will be used.

Setup a Boot Configuration

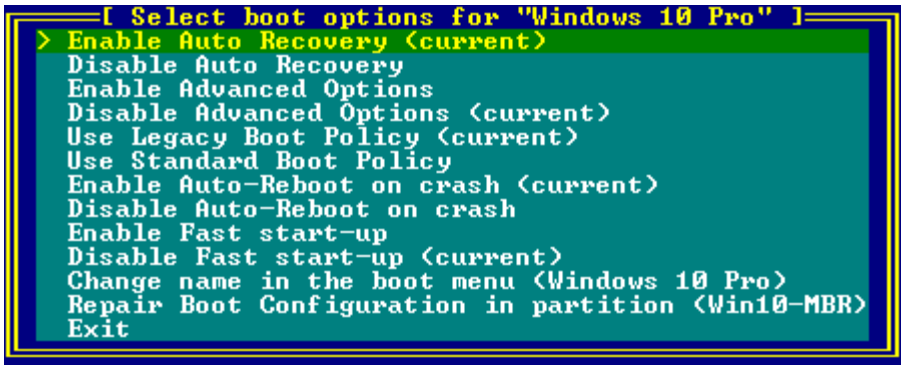
If the BCD store for the selected Windows system wasn't found, you'll be given the choice to setup a boot configuration, create a boot partition, or search other drives (available options vary depending on the disk type and how the system is configured). Several examples are shown below:





Selecting to setup a boot configuration will create the boot folders and required files (including the BCD). For MBR drives, if you select to setup the boot configuration in the Windows partition, you'll be given a choice to set the partition active if it isn't. For GPT drives, an EFI System partition is required (either on the Windows drive or found after searching). If BootIt UEFI is installed and that drive is selected, the corresponding BCD will be selected, or, if it doesn't exist, a new \EFI\Microsoft.### folder can be created for use with a new boot menu item (the menu item is not created). Using this option provides an easy method of creating the BCD and boot files when you have a Windows partition without its booting files (e.g. you copied a Windows partition to an empty drive).

If the BCD store for the selected Windows system was found, the following options will be available. If the BCD isn't found (and wasn't created), the options will be limited to those that don't require a BCD store. The currently enabled/active settings are indicated in the menu.



Enable/Disable Auto Recovery

When enabled Windows will automatically boot into recovery mode after several failed boot attempts and try to repair the system. Disabling this option can be helpful when troubleshooting by keeping the system booting in normal mode.

Enable/Disable Advanced Options

Will enable or disable booting to the advanced options of the boot menu. These options include Repair Your Computer, Safe Mode, Debugging Mode, Disable Driver Signature Enforcement, etc.

Use Legacy/Standard Boot Policy

Controls the boot policy method used. By default, Standard mode is used by Windows 8.x/10. In Legacy mode, F8 can be pressed during boot-up to access the *Safe Mode* / *Repair Your Computer* option menu.

Enable/Disable Auto-Reboot on crash

This option is normally enabled by default, which can make viewing BSOD errors difficult or impossible. Disabling this option will allow the error to remain on the screen.

Enable/Disable Fast start-up

Will enable or disable the Windows *Fast startup* option. Warning: You must disable the Windows *Fast startup* option or you risk corruption of your partitions and data when the partitions are used outside of Windows (e.g. you boot into Image for Linux after shutdown and save an image to a data drive).

Change name in the boot menu

Select this option to change the name shown in the Windows boot menu.

Repair the Boot Configuration

If the BCD store for the Windows system exists and is found, this option will be available. Selecting it will repair the boot configuration. This is useful when the BCD is corrupt or needs references updated/fixed.

The OSDTOOL.TBS Script

(Pro Version Only)

Introduction & Requirements

OSDTOOL is an interactive script, included with the professional version of the TeraByte OS Deployment Tool (TBOSDT), that provides a convenient way to use TBOSDT for the purpose of performing deployment-related tasks on Windows NT-based operating systems such as:

- Installing drivers
- Removing installed drivers
- Changing the HAL (hardware abstraction layer) file
- Changing the Computer Name
- Configuring services

A typical scenario for using OSDTOOL would be to first restore an image of Windows to new/different hardware, and then run the script to make the necessary changes so that the restored OS partition will boot and run successfully. These changes are all made while the target Windows OS is not running.

Changes needed to get a restored Windows OS to boot on new hardware can include:

- Installing a storage driver for the boot device (hard drive controller)
- Changing the HAL (hardware abstraction layer)

In some cases it may be necessary to first remove all installed drivers, which OSDTOOL can also do.

NOTE: OSDTOOL cannot reconfigure a 64-bit OS to function on a 32-bit system.

Requirements

- Target OS: Windows XP or later
- A fully licensed copy of TeraByte OS Deployment Tool (TBOSDT) Professional

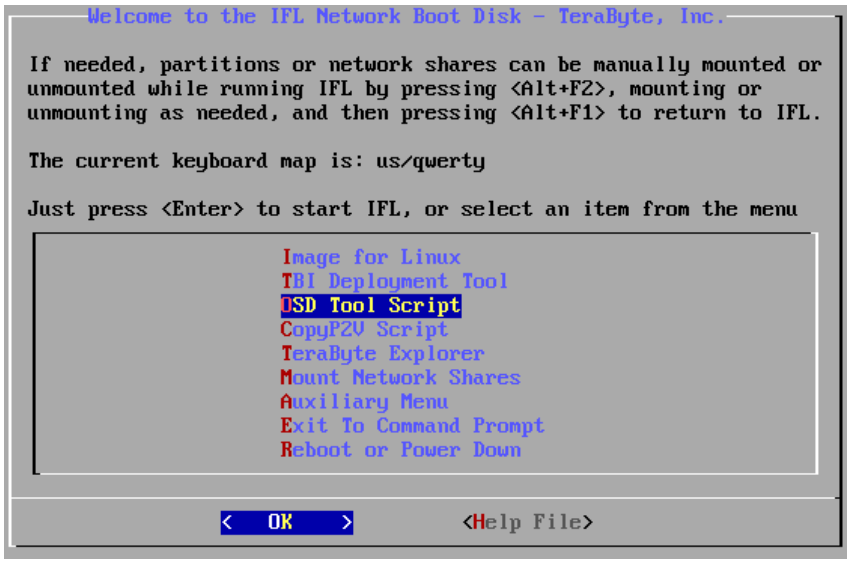
Using OSDTOOL – Manual Mode

In this mode, the script is menu driven. Operations are manually chosen from the displayed menus. This mode is entered by default when the script is started with no command line parameters.

The script can be run from Windows, Linux, DOS, TBOS, or UEFI. Use the appropriate version of TBOSDT for your OS. Here are some examples:

`runtbs ..\scripts\osdtool.tbs`	Launch OSDTOOL.TBS from TBOSDT in the default folder
`tbosdtw.exe osdtool.tbs`	Launch from Windows command line
`tbosdt osdtool.tbs`	Launch from Linux command line
`tbosdt.exe osdtool.tbs`	Launch from DOS command line

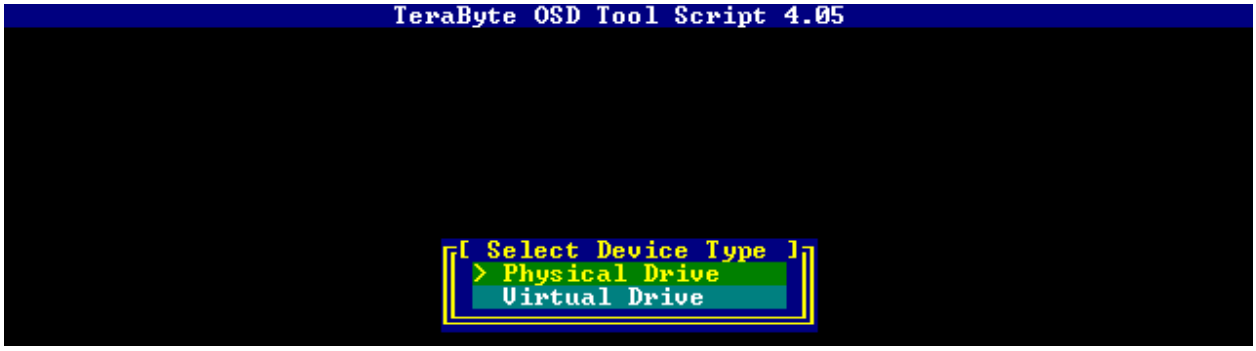
You can also run the script directly from the Image for Linux boot media menu by selecting the **OSD Tool Script** option (shown below for Image for Linux CUI).



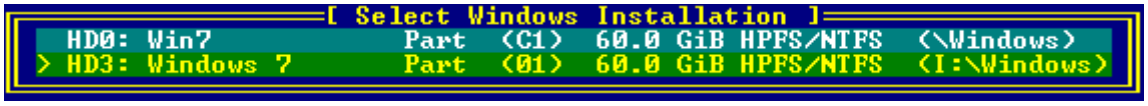
Please refer to the **Running Scripts Using TBOSDT Professional** section if you need more help using TBOSDT to run a script.

Selecting the Windows Installation

The **Select Device Type** menu will be displayed when the script starts.

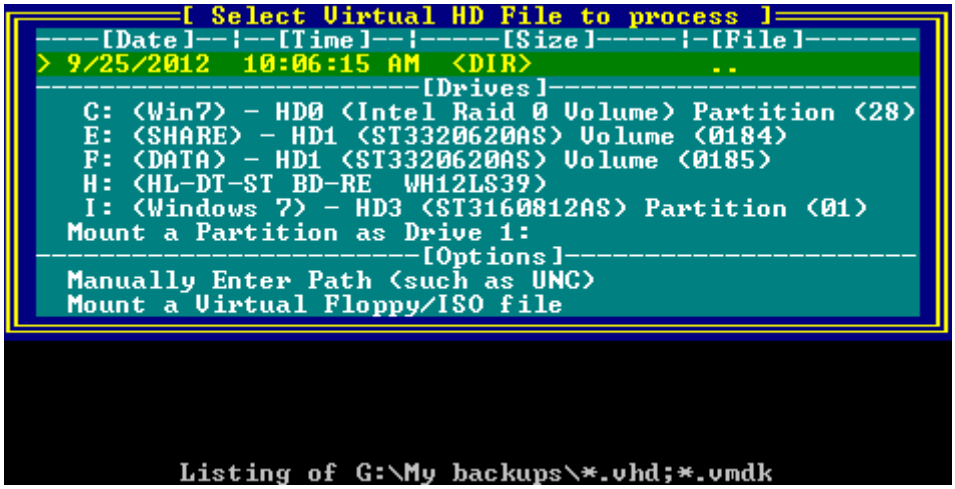


Selecting **Physical Drive** will allow you to choose from the partitions on the drives available to the system. Here, there is only one drive:



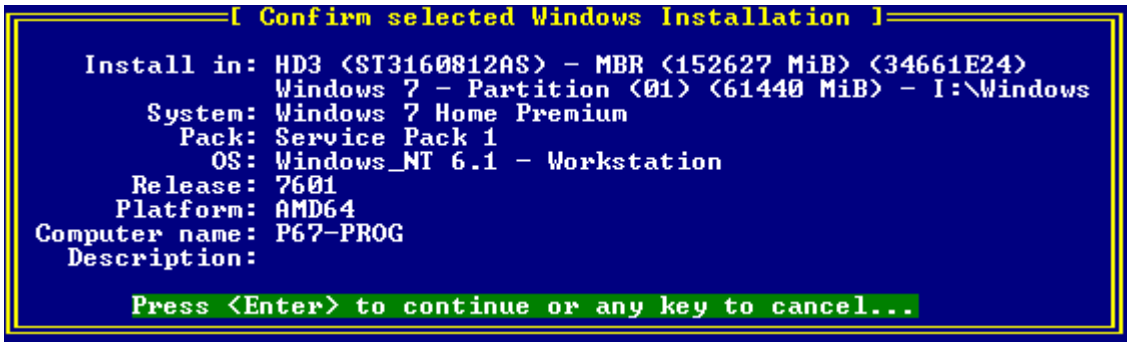
NOTE: If running OSDTOOL in Windows, you will not be able to select or make changes to the running Windows partition.

Selecting **Virtual Drive** will allow you to browse for the virtual disk file and mount it for use. Browsing options (such as mounting a CD/DVD drive) vary depending on the OS being run.



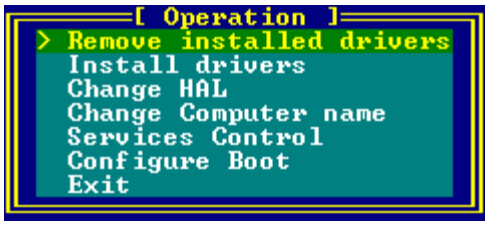
NOTE: Only single-file VHD and VMDK files can be accessed.

OSDTOOL will display the Windows OS information for the selected partition. Verify this information is correct before proceeding.



OSDTOOL automatically creates a backup of the Windows registry when you make changes. The following menu will be displayed if this backup is available.

Restoring the backup after a failure from previously applied changes allows you to start over easily. Make your selection to go to the **Operation** menu.



NOTE: Depending on the version of Windows some options may be limited or not available.

Remove Installed Drivers

Selecting this option will allow you to remove drivers installed in the selected Windows system.

Remove drivers keeping storage and printer drivers

Remove drivers keeping printer drivers

Remove all drivers

These options control which type of drivers you want removed. You can select to leave the existing storage and printer drivers installed, leave existing printer drivers installed, or remove all drivers.

After selecting one of these options, a warning will be displayed before proceeding.

OSDTOOL will now remove the drivers. The progress of the operation will be displayed on-screen.

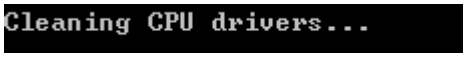
Press ENTER when it finishes.

Once driver removal is complete, you can now install a driver, change the HAL, or return to the **Operation** menu.

Remove CPU drivers only

This will allow you to just remove the CPU drivers. All other drivers installed will remain installed.

NOTE: There is no warning displayed when you select this option. The CPU drivers will be removed immediately.

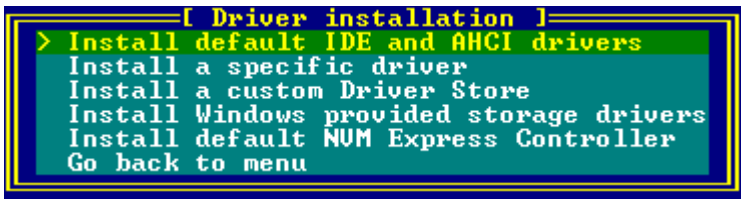


When finished, press ENTER and you will be at the same sub-menu as above.

Install Drivers

This option allows you to easily install drivers into the selected Windows system. Drivers must be standard Windows INF type drivers and can usually be obtained from the device's manufacturer if not available in the standard Windows drivers.

Select the desired option from the **Driver installation** menu.



Install default IDE [and AHCI] drivers

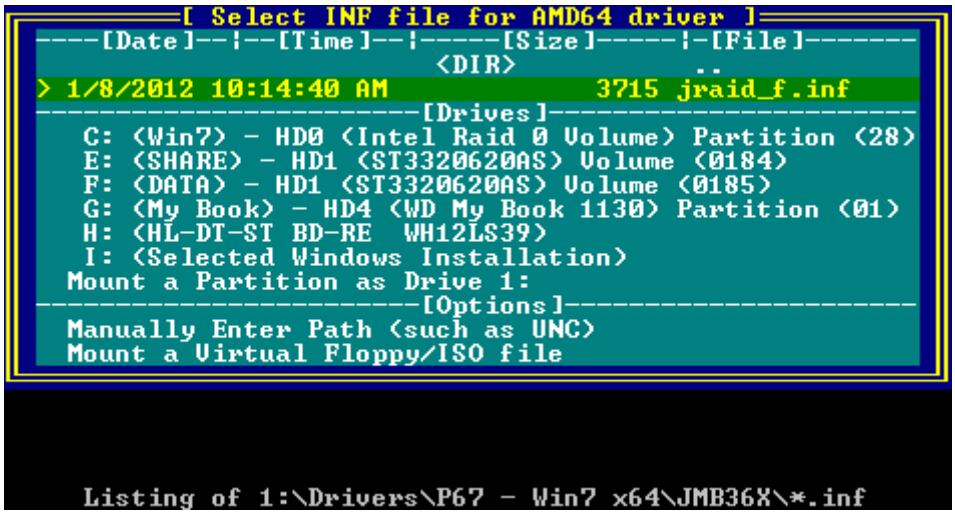
The default IDE (and AHCI drivers, if applicable) for the selected Windows OS will be installed. Progress will be shown on-screen. Upon successful completion, a message will be displayed:

Press ENTER to return to the **Driver installation** menu.

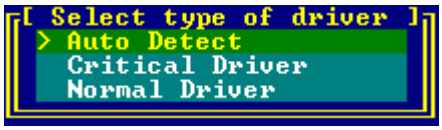
Install a specific driver

Browse to the INF file for the driver to be installed and select it. Multiple options are available to access the desired folder and vary by the OS. You may need to mount a partition or access a CD, for example.

Below is an example of browsing to a JMicron RAID driver. The partition containing the driver was mounted as Drive 1:.



After the driver is selected you will need to select the driver type. You can choose **Auto Detect**, **Critical driver** or **Normal driver**.



Auto Detect should be selected in most cases. OSDTOOL will attempt to determine the type of driver being installed and select the appropriate method. Using either of the options below will force installation of the driver using the selected type.

Critical Driver - select for any mass storage device driver (IDE, SATA, eSATA, RAID, etc.). required for Windows to boot.

Normal driver - select for non-storage devices and non-critical storage devices.

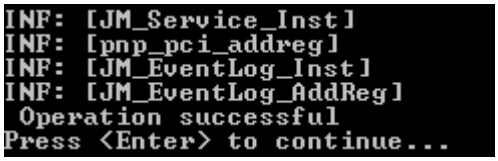
Next, select the desired filter option.

Filter on Hardware ID can be helpful if the drive with the Windows system being modified is installed in the destination computer and OSDTOOL is being run on that computer. Enabling this option will allow only those drivers that match the detected hardware to be installed. In most cases, you will not want to select this option if the drive is being modified on a system with dissimilar hardware (physical or virtual).

Selecting **Do not filter on Hardware ID** will allow the installation of any driver. No checking is performed to determine if the driver matches the current running system.

Finally, select to start the installation.

The progress on the installation will be displayed on-screen (if there are any errors, please refer to: **Error When Installing a Driver**). Press ENTER when finished.

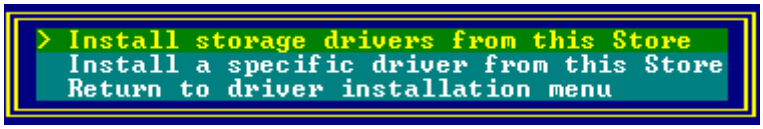


You will be returned to the **Driver installation** menu.

Install a custom Driver Store

A driver store is a folder that contains multiple drivers (often one subfolder per driver/device). Populate the store with drivers Windows needs for the system (e.g. AHCI, RAID, USB 3.0, NIC, audio, video, etc.). Required storage drivers (critical drivers) will still need to be installed so Windows can boot. Other drivers in the store will be available when Windows searches for drivers while installing devices, allowing them to be installed automatically.

Select this option, then browse to the store folder and press **Ctrl-Enter** to select it. The drivers will be copied and then the following menu will appear:



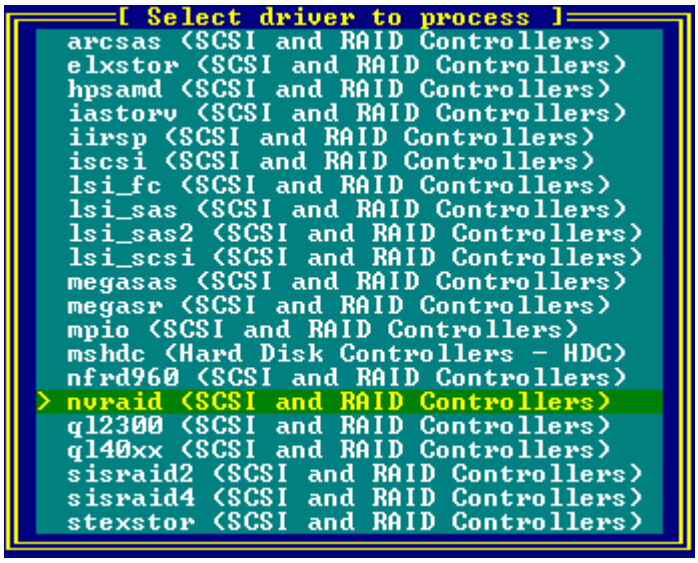
Install storage drivers from this Store will allow you to install all the storage drivers found in the driver store or select and install specific storage drivers found in the store.

Install a specific driver from this Store will let you browse the store and select and install a specific driver (INF file) from the driver store.

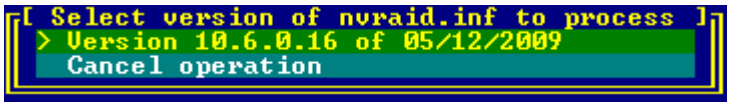
Install Windows provided storage drivers

Selecting this option will allow you to choose from a list of built-in Windows drivers. It may take several seconds for the list to be displayed.

NOTE: This option is only available for Windows Vista, Windows 7, Windows 2008 R2, and Windows 8.



Scroll through the list and select the desired driver. In this example, the *nvraid* driver has been selected. To continue with the installation select the “Version” you want from the menu.



Progress will be shown on-screen. Upon successful completion, the following will be displayed:

Press ENTER to return to the list of drivers. Either select another driver to install or press ESC to return to the **Driver installation** menu.

Install default NVM Express Controller

Select this option to install the default NVM Express (NVMe) drivers. Note that the Windows installation must have native NVMe support for this option to function (Windows 8.1+).

Change HAL

Use this option to specify the name of the hal*.dll file that will be required by the hardware on which the Windows installation will be running. It may take several seconds for the list of HAL files to be displayed. The currently selected file is marked with an asterisk. Below is an example of Windows XP.

Select the desired HAL file from the list to change to it.

The following error message will be displayed if OSDTOOL can't find the HAL files. For example, Windows 7 64-bit has only one HAL.

From either message, press ENTER to return to the menu.

Change Computer Name

Use this option when you need to change the Windows computer name. The name is limited to 15 characters and can only contain letters, numbers, or hyphens.

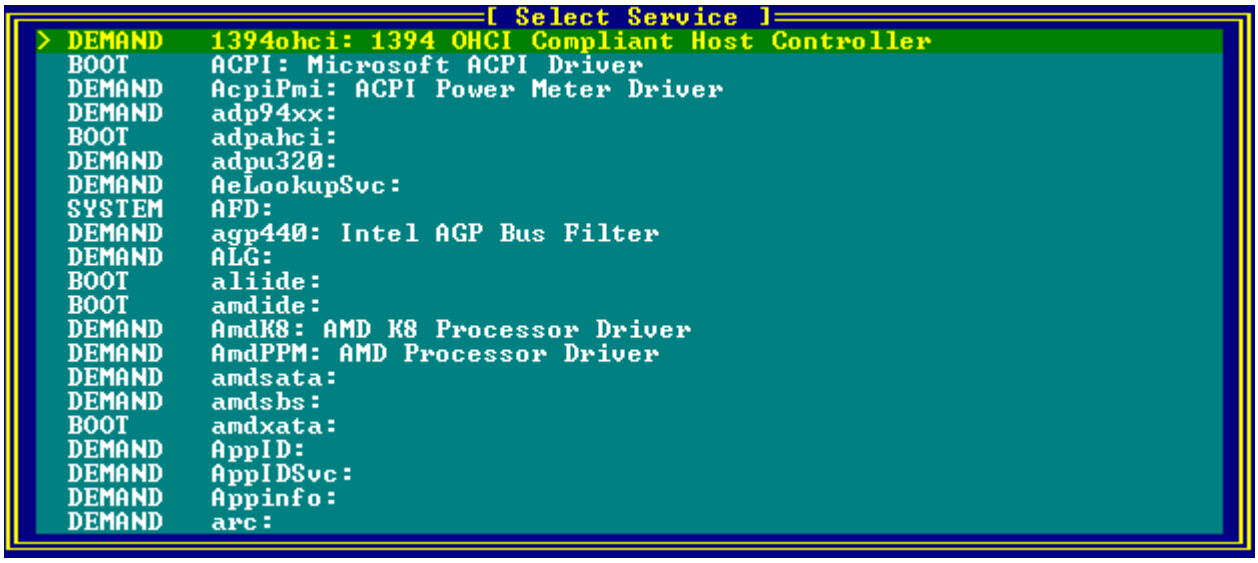
NOTE: If you select this option and don't wish to change the name, leave the new name blank and press ENTER. Pressing ESC will not return to the menu.

Type in the new name and press ENTER.

Press ENTER to return to the menu.

Services Control

A list of services on the selected Windows system will be displayed. Scroll through the list and select a service that needs changed.



The service details will be displayed along with a menu allowing you to change the Start Type.

Select the new Start Value from the menu to change it or press ESC to return to the **Select Service** list without making a change.

When finished, press ESC from the **Select Service** list to return to the **Operation** menu.

Configure Boot

This option will run the `fixboot.tbs` script which allows Windows boot options to be changed as needed to aid in repairing or troubleshooting booting issues for the selected Windows system. Refer to the ***Error! Reference source not found.*** section of the manual for more details.

Using OSDTOOL – Automated Mode

In this mode, the script runs based on answers provided in an answer file. The answer filename must be given on the command line as a parameter for the `osdtool.tbs` script, as shown in these examples:

```
tbosdt osdtool.tbs answer.ini
runtbs osdtool.tbs answer.ini
TBOSDT.EXE osdtool.tbs answer.ini
tbosdtw.exe osdtool.tbs ..\answer.ini
tbosdtw.exe osdtool.tbs "C:\Terabyte Scripts\answer.ini"
```

Answer File Format

The answer file is a plain text file, typically using an `.ini` filename extension (such as `answer.ini`). Inside the file, the osdtool.tbs parameters (those that would normally be chosen from the menus) are specified by assigning values to variables. Comments can be added to the answer file by using `//` at the beginning of the comment line.

The following is a list of the parameters that can be specified, the requirements for each, and some examples.

1. WinInst

Specifies the target Windows installation (the installation you need OSDTOOL to modify). This parameter has to be the first one in the file. There are four different methods of specifying the target Windows installation. The most common method when only a single installation exists is by installation number:

a. By hard drive identifier and partition identifier (optional): `WinInst=HdPid[HD Pid]`
   `// for hard drive 0 and partition ID 0x3`
   Example: `WinInst=HdPid[0 0x3]`

   `// for hard drive with disk signature 0x12344321 and partition ID 0x3`
   Example: `WinInst=HdPid[0x12344321 0x3]`

   `// for a GPT hard drive with GUID {4ca05180-a699-450a-9a0c-de4fbe3ddd89}`
   `// and the first installation on the hard drive.`
   Example: `WinInst=HdPid[{4ca05180-a699-450a-9a0c-de4fbe3ddd89}]`

   `// for hard drive 0 and the third installation found on the drive.`
   Example: `WinInst=HdPid[0 #3]`

b. By installation number: `WinInst=Num[n]`
   Example: `WinInst=Num[1]`

c. By Windows drive letter: `WinInst=DrvLtr[X:]`
   Example: `WinInst=DrvLtr[E:]`

d. By path to a virtual drive file followed by installation number to select (default is first) or partition id.
   Examples:
   ```
   WinInst=F:\vhd\test.vhd 2
   WinInst=1:\vhd\test.vhd
   WinInst="D:\path with spaces\virtual.vhd" [0x102]\Windows
   ```

NOTE: For **(d)** above, if a TBOSDT drive is used in the path as shown in the 2nd example above, then the drive will have to be defined with the parameter "Drive" (see item 7). TBOSDT drives will be a number from 1 to 9 (e.g. 1:, 2:, etc.)

NOTE: For **(a)** and **(c)** above, if the windows directory is not `\Windows` or `\WinNT` then you can specify the custom Windows directory by adding it to the end of the parameter:

Examples:
```
WinInst=HdPid[0 0x3]\WinSys
WinInst=DrvLtr[F]\WinSys
WinInst=HdPid[0x12344321]\WinSys // first installation
```

2. RestBck

This is used for restoring or deleting the registry backups automatically created by the script. The script will make a backup copy of the software and system registry prior to modifying it, provided a backup doesn't already exist. This option allows you to determine what you want with any existing backups when the script runs. If you don't include this value any existing backups are kept. Possible values are:

- `1` - Restore the backup
- `2` - Delete the backup
- `3` - Keep the backup

Example: `RestBck=2`
This will remove the backups if they exist (allowing new ones to be created).

3. ClrDrv

Specifies if installed drivers are to be cleaned (removed). If this parameter is used, it has to be set before the `InstDrv` parameter. Possible values are:

- `1` - Remove drivers keeping storage and printer drivers
- `2` - Remove drivers keeping printer drivers
- `3` - Remove CPU drivers only
- `4` - Remove all drivers

Examples: `ClrDrv=4` *will remove all installed drivers*
`ClrDrv=1` *will remove all drivers except existing storage and printer drivers*

NOTE: The `ClrDrv` parameter must be specified before the `InstDrv` parameter in the ini file, or it will not be recognized.

4. InstDrv

When only a numeric value is given, this parameter specifies that default IDE, IDE/AHCI, or NVMe drivers are to be installed. If the selected value is not supported or needed for the

Windows installation a warning or other message will be displayed and the script will continue. Possible values are:

1 - Install IDE drivers for XP, and IDE/AHCI drivers for Vista and Win 7/8.x/10
2 - Install NVMe drivers (installation must provide native support for NVMe)

Example: `InstDrv=1`

5. InstDrv

When more than just a numeric value is given, the `InstDrv` parameter is being used to install a specific driver or specify a driver store (a folder containing multiple drivers).

Installing a Specific Driver

Specify the full path to an INF file to install. If a TBOSDT drive (such as 1: or 2:) is used in the path, then the drive used must be defined with the parameter `Drive` (see item 7 below). The path to the INF file can be followed by either the `critical` or `normal` option, and the `filter` option (if needed):

`critical` - specifies that the driver is a storage driver required for booting Windows
`normal` - specifies that the driver is normal (non-critical) driver
`filter` - specifies to filter on the hardware ID

These options must be prefaced by `-` (see examples below).

Examples:
```
InstDrv=1:\Drivers\Intel\iaStor.inf -critical
InstDrv=1:\Drivers\Intel\iaStor.inf -normal -filter
InstDrv=1:\Drivers\Intel\iaStor.inf -filter
```

NOTE: If neither `critical` or `normal` are specified, automatic detection of the driver type will be used to select the appropriate installation method.

To install a driver located on the Windows installation itself, drive 0: has to be used to specify the location of the driver (unless the script is used under Windows and the Windows installation is accessed using a Windows drive letter).

Example: `InstDrv=0:\drivers\Intel\iaStor.inf -critical`

For Vista, Win7, Windows 2008 R2, and Win8, storage drivers provided by Windows can be installed by providing only the name of the .inf file to install, followed by the `-critical` option.

Example: `InstDrv=lsi_scsi.inf -critical`

For Linux, a driver located on a CD/DVD can be referenced by appending the full path to /cdrom.

Example: `InstDrv=/cdrom/drivers/Intel/iaStor.inf`

Installing a Driver Store

A driver store is a folder that contains multiple drivers (often one subfolder per driver/device). Populate the store with drivers Windows needs for the system (e.g. AHCI, RAID, USB 3.0, NIC, audio, video, etc.). Required storage drivers (critical drivers) will still need to be installed so Windows can boot. Other drivers in the store will be available when Windows searches for drivers while installing devices, allowing them to be installed automatically.

Specify the path to the driver store folder. The path to the store can be followed by either the `inststor` or the `filter` option (if needed):

`inststor` - all storage drivers in store will be installed
`filter` - specifies to filter on the hardware ID

These options must be prefaced by `-` (see examples below).

Examples:
```
InstDrv=G:\DriverStore -inststor
InstDrv=1:\Drivers\Laptop –inststor –filter
```

You can also specify the store folder and then drivers within it to install by specifying the path and then the relative path to the .inf file. In the example below, the driver store path is `G:\DriverStore` and the `iaStor.inf` driver located in the `Intel` subfolder (full path of `G:\DriverStore\Intel\iastor.inf`). Multiple drivers can be specified (one per line). The `critical`, `normal`, and `filter` options can also be specified, if needed.

Example:
```
InstDrv=G:\DriverStore
InstDrv=Intel\iaStor.inf -critical
InstDrv=ASMedia\asahci64.inf
```

Parameter `InstDrv` can also accept storage drivers provided by Windows (by providing only the name of the .inf file to install, followed by the `-critical` option if needed (not mandatory):

Example:
```
InstDrv=lsi_scsi.inf –critical
```

6. HalFile

This parameter specifies the name of the hal*.dll file that will be required by the hardware on which Windows will be running. If the file specified is invalid for the system being restored, an alternate HAL will be used if possible and available. Otherwise, the HAL won't be changed. Just the file name is required (full path is not needed). You can also use one of the following special "auto" values to have the script try to determine the correct HAL based on the environment from which the script is running:

`auto` - use determined uniprocessor or multiprocessor HAL
`auto-uni` - use determined uniprocessor HAL
`auto-multi` - use determined multiprocessor HAL

Examples: To use a specific HAL

```
HalFile=halaacpi.dll
```

To have TBOSDT automatically select the HAL

```
HalFile=auto
```

7. Drive[n]

This parameter is used to specify a drive numbered `n`, where `n` is a TBOSDT drive from 1 to 9. A drive can be used as many times as required in the answer file, but only needs to be defined once. A drive can be defined at any location in the answer file.

Examples: To specify the primary CD/DVD drive under DOS/TBOS/UEFI as Drive[1]

```
Drive[1]=o0
```

To specify the first CD/DVD drive found under DOS/TBOS/UEFI as Drive[1]

```
Drive[1]=o
```

To specify a partition on a USB drive under DOS or TBOS as Drive[5]

```
Drive[5]=u0 0x304
```

To specify a partition 0xFE on drive 0 as Drive[2]

```
Drive[2]=0 0xFE
```

To specify partition with ID 0x5 on disk with signature 0x12344321

```
Drive[2]=0x12344321 0x5
Drive[2]=#305414945 0x5
```

To specify partition with GUID {FE173FDD-1BA9-D14B-49A9-CD5869B69A19}

```
Drive[2]= {FE173FDD-1BA9-D14B-49A9-CD5869B69A19}
```

To specify partition with GUID {FE173FDD-1BA9-D14B-49A9-CD5869B69A19} on disk {15DD9D75-8497-A707-49A9-CD5869B69A19}

```
Drive[2]= {15DD9D75-8497-A707-49A9-CD5869B69A19}  {FE173FDD-1BA9-
D14B-49A9-CD5869B69A19}
```

Virtual drives can be defined by providing the full path of the file using a TBOSDT drive from 1 to 9 as well. When doing this the drive being mounted can only access drives that have a lower number. The example below requires that drive 1: be defined before drive 2: can be mounted since it relies on drive 1:.

Examples:

```
Drive[2]=1:\drivers\FloppyWithSataDrivers.img
Drive[1]=0 0x02
```

NOTE: You can use Image for Windows to determine the drive numbers and partition ID values. For example, running `imagew /L > idvalues.txt` will save the data to the `idvalues.txt` file for easy reference.

8. ComputerName

Use this parameter when you need to change the Windows computer name. The name is limited to 15 characters and can only contain letters, numbers, or hyphens.

If you wish the script to create the computer name automatically you can specify `*auto*` as the name. The name will be determined as follows:

- If accessing a physical drive: The first six letters of the System Manufacturer followed by a hyphen and then eight characters calculated from the System UUID.
- If accessing a virtual drive: The name of the virtual drive file.

Examples: Specify "MY-COMPUTER" as the computer name:
`ComputerName=MY-COMPUTER`

Let the script set the computer name automatically:
`ComputerName=*auto*`

9. IgnoreDirty

Normally the script will abort or prompt you if the target registry is dirty and cannot be recovered. Setting this parameter to one (1) will allow the script to update the registry even if it is dirty and cannot be recovered.

Example: `IgnoreDirty=1`

10. ExitOnErr

The normal behavior will be for the script to exit on error. Setting this parameter to zero will cause the script to enter Manual mode instead (menu operation). If this parameter is set to 1, or is not there at all, the script will exit on any error condition.

Example: `ExitOnErr=0`

11. UseAlternateHal

The default behavior if the HAL file specified in parameter `HalFile` is not available, is to use an alternate HAL if possible (the alternate HAL is halmacpi.dll). To not use any alternate HAL, and thus to not change the HAL if the one specified is not available, this parameter can be used and should be set to `0`. Supported values are:

`0` - do not use alternate HAL
`1` - use alternate HAL
`2` - user (if alternate HAL is found)
`3` (or higher) - ask user, waiting the specified value in seconds (will use alternate HAL if time runs out with no response)

Examples: Don't use alternate HAL:
`UseAlternateHal=0`

Ask user and wait 5 seconds. If no response, use alternate HAL:
`UseAlternateHal=5`

Answer File Examples

Below are two answer file examples. Each line is followed by a comment to explain its function.

Example A:

```
WinInst=HdPid[0 0x18]
//Work with Windows installation on HD0, part ID 0x18

ClrDrv=2
//Remove drivers keeping printer drivers (must follow WinInst line)

InstDrv=1
//Install default IDE/AHCI drivers

InstDrv=1:\drivers\intel\iaAHCI.inf -critical
//Install iaAHCI.inf storage driver located on TBOSDT drive 1:

HalFile=halmacpi.dll
//Set HAL file to halmacpi.dll

Drive[1:]=0 0x0390
//Define drive 1: as HD0, partition ID 0x0390
```

Example B:

```
WinInst=1:\vhd\test.vhd
//Windows installation is on a virtual drive on TBOSDT drive 1:

ClrDrv=1
//Remove drivers keeping storage and printer drivers

InstDrv=2:\Drivers\Intel\iaStor.inf -critical
//Install the Intel SATA storage driver located on TBOSDT drive 2:

Drive[1]=1 0x102
//Drive 1: is partition 0x102 on HD1

Drive[2]=o0
//Drive 2: is the primary CD/DVD drive (o0 = optical drive 0)

ExitOnErr=0
//Script will enter Manual Mode (menu driven) on error
```

List of Error Messages

The following error messages are returned if there is a problem.

1 No Windows Installation found
2 Unable to Access Windows Installation
3 Mount of a drive failed
4 Error on cleaning drivers
5 Error when installing IDE drivers
6 Error when installing a specific driver
7 File tbosdtw.exe cannot be found
8 File not found (INF File or Virtual HD for example)
9 Error when removing the CPU drivers
10 No HAL files available
11 Unable to change HAL
12 HAL file of the answer file not available
13 A parameter in the answer file invalid (e.g. out of range, unknown, etc…)
14 Unable to change the computer name
15 Registry backup failed
16 Disk write failure.

Important Notes & Limitations

When using OSDTOOL, please keep the following in mind:

Using the Standard (Free) Version of TBOSDT

The standard (free) version of TBOSDT does not support running scripts like OSDTOOL. OSDTOOL requires the Professional (purchased) version to run. TBOSDTS comes with your purchase of one of the TeraByte Drive Image Backup and Restore Suite products or BootIt Collection. The Profession for BootIt version further extends the commands available to include partitioning related commands.

Running OSDTOOL in Windows Vista, Windows 7, or Windows 8

OSDTOOL will function under Windows Vista and later if run in Administrator Mode. This means you need to run `tbosdtw.exe` as an administrator. If run as a standard user, OSDTOOL won't be able to find any Windows installations on the drives.

FAT32 Partition Labels

FAT32 partition labels may not be displayed correctly. The label displayed in OSDTOOL is the same as that shown by Image for DOS and Image for Windows and may not be the same as that shown by Windows Disk Management or Explorer. For example: You create a FAT32 partition using Windows XP Disk Management and label it `XPUSB`. OSDTOOL will display `NO`

NAME as the partition's label. This is due to the fact that the volume label field in the boot sector is not correctly updated under Windows XP.

If selecting a FAT32 partition, make sure you select the correct partition.

Preparing the Windows System

The following tips may be helpful:

- Make sure the necessary Windows drivers are available for the system and have them ready.
- If possible, remove any unnecessary or incorrect drivers/programs from the source system before using OSDTOOL. This will prevent them from causing problems when the system is booted on the target computer. This will only be possible if the system being moved is currently able to run. Examples include uninstalling video drivers, sound drivers, etc. By doing this normally in Windows, you may avoid potential problems.

Troubleshooting

Error When Installing a Driver

A driver that causes an error during installation will not be properly installed. Please verify the driver is correct for the version of Windows (XP, Vista, Windows 7, etc. / 32-bit, 64-bit) and the computer.

Some errors (like an "Unknown directive") may require a revised script from Terabyte Support to allow correct installation.

Selecting the Type of Driver

When installing the driver for the mass storage controller to which the booting drive is connected, you *must* use either the **Auto Detect** or **Critical driver** option when selecting the type of driver. Failure to do this will most likely result in a fatal Windows error (BSOD 0x7B) upon booting.

Booting the Modified Windows Installation

Once the correct storage drivers have been installed, there can still be quite a lot of other drivers and configuration changes needed before the Windows system is fully functional.

After booting Windows on target system:

- Windows may crash (BSOD 0x7B type) if drivers are not configured correctly. This is often caused by the incorrect storage driver being installed or the correct storage driver not being installed. By default, Windows will automatically reboot on these errors. To turn off this feature and allow viewing the error, press F8 when Windows is just starting

to load (immediately after the BIOS screens) and select the *Disable automatic restart on system failure* option from the menu.

- There may be a long delay (5-10 min.) on the “Welcome” screen, the log in screen, or the desktop background screen. This delay is longest on the first boot, but may remain for subsequent reboots. Check the Windows Event Log for any errors or problems. Changes may need to be made to drivers and/or services.
- “Windows Product Activation” may be triggered if using a version that requires activation.
- On the first boot-up, there may be a long delay before the mouse and/or keyboard are detected. This can happen with both USB and PS/2 devices. In most cases, Windows will eventually install the drivers and detect the devices.
- If operating the computer through a KVM switch, it’s possible for one or more connected devices to remain inactive. This will require connecting a standard USB mouse and/or keyboard directly to the computer. Once Windows has finished loading and the drivers are installed, the KVM devices should function normally.
- Log in may be different from the source system. Auto log in may be canceled, for example.
- “Found New Hardware” wizard may pop-up repeatedly due to the different hardware. Installing the correct drivers is recommended. Please note that Windows may not automatically install the drivers or not install them correctly. Some may require manual installation.
- Windows may display the taskbar/windows using the “Windows Classic Style” instead of the style selected in the source system. This problem will usually correct itself after several reboots.
- Screen resolution may be very low. Install the correct video drivers for the system.

The TBIDTOOL.TBS Script

(Pro Version Only)

Introduction & Requirements

TBIDTOOL is a script, included with the professional version of the TeraByte OS Deployment Tool (TBOSDT), that provides a convenient way to deploy backup images or systems on supported virtual disks to alternate hardware. The script can be run interactively or automated using an configuration file and perform the following tasks:

- Restoring the backup image file or copying a virtual disk to a physical drive
- Removing installed drivers
- Installing drivers
- Changing the HAL (hardware abstraction layer) file
- Changing the Computer Name
- Changing the partition volume name

A typical scenario for using TBIDTOOL would be to deploy a backup image to one or more computer systems. The process can be simplified to the point that it's only necessary to specify the configuration file to use and the script does the rest.

NOTE: TBIDTOOL cannot reconfigure a 64-bit OS to function on a 32-bit system.

Requirements

- Target OS: Windows XP or later
- A fully licensed copy of TeraByte OS Deployment Tool (TBOSDT) Professional (version 1.38 or later; version 1.50 or later to copy virtual disk)
- Image for Windows, Image for DOS, or Image for Linux (version 2.73 or later)

Using TBIDTOOL – Manual Mode

In this mode, the script is menu driven. Operations are manually chosen from the displayed menus. This mode is entered by default when the script is started with no command line parameters.

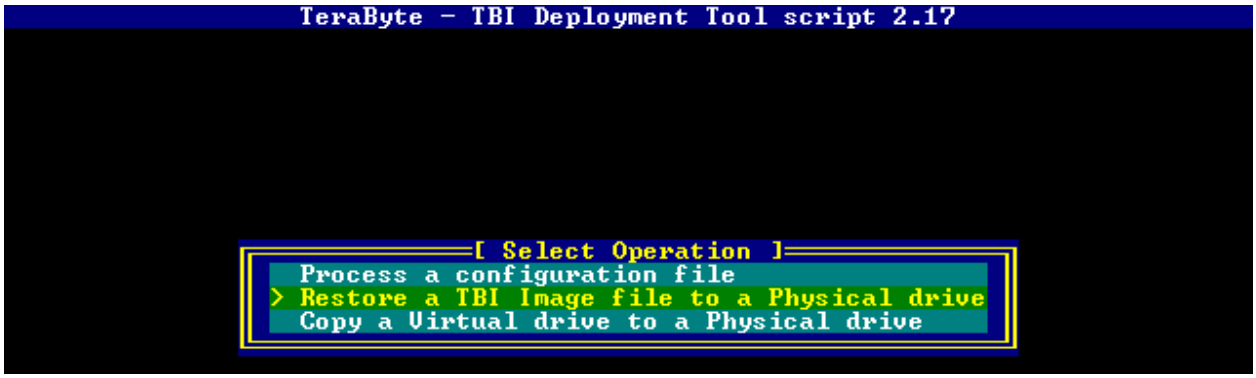
The script can be run from Windows, Linux, DOS, TBOS, or UEFI. Use the appropriate version of TBOSDT for your OS. Here are some examples:

`runtbs tbiddtool.tbs`	Launch TBIDTOOL.TBS from TBOSDT in the default folder

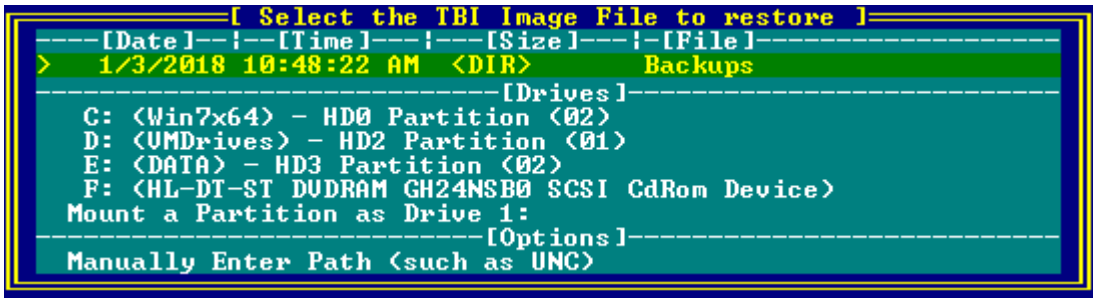
`tbosdtw.exe tbidtool.tbs`	Launch from Windows command line
`tbosdt tbidtool.tbs`	Launch from Linux command line
`tbosdt.exe tbidtool.tbs`	Launch from DOS command line

Please refer to the **Running Scripts Using TBOSDT Professional** section if you need more help using TBOSDT to run a script.

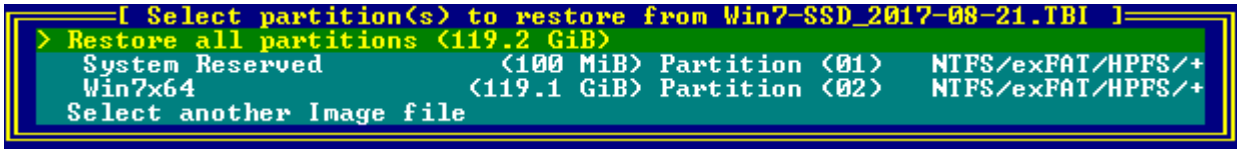
When you run the script the main menu will be displayed. You will have the choice of selecting a configuration file and letting the script proceed automatically or selecting an image file or virtual disk and proceeding interactively. (Please refer to the next section of this manual if using a configuration file.)



Select the **Restore a TBI Image file to a physical drive** option to browse for the backup file. Select the **Copy a Virtual drive to a Physical drive** option to use a virtual disk as the source instead of a backup image (single file .vhd, .vhdx, and .vmdk disks are supported). Browsing options (such as mounting a CD/DVD drive) vary depending on the OS being run. If you are using the Windows version and attempt access a network share that requires a log-in you will be prompted for the username and password.



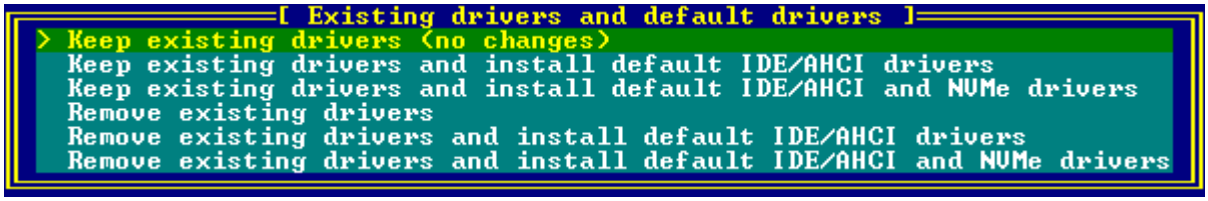
After selecting the image or virtual disk you will be prompted which partition(s) to restore or copy. You can select to restore or copy all of the partitions or just a single partition.



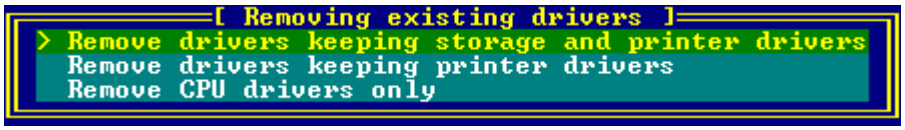
Next, select the destination for the restored image or copied virtual disk. If you selected to restore all partitions you must select an entire drive as the destination.

Select whether to keep or change the disk signature (and GUID's where applicable). When running in Windows, if you keep the signature and the source drive is present the target drive will be set to offline mode due to a signature collision. Bringing the drive online in this situation will cause Windows to modify the drive and may prevent the drive from booting properly (depends on the OS).

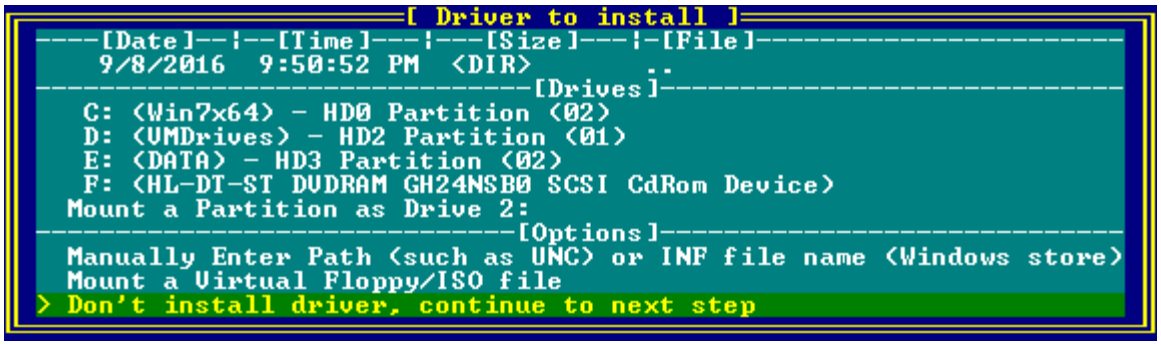
Next, select if you want existing drivers to be removed and whether or not to install the default IDE/AHCI/NVMe Windows drivers. Existing drivers are those already installed in the source Windows partition. Default IDE/AHCI/NVMe drivers are those that are the default for the source Windows system.



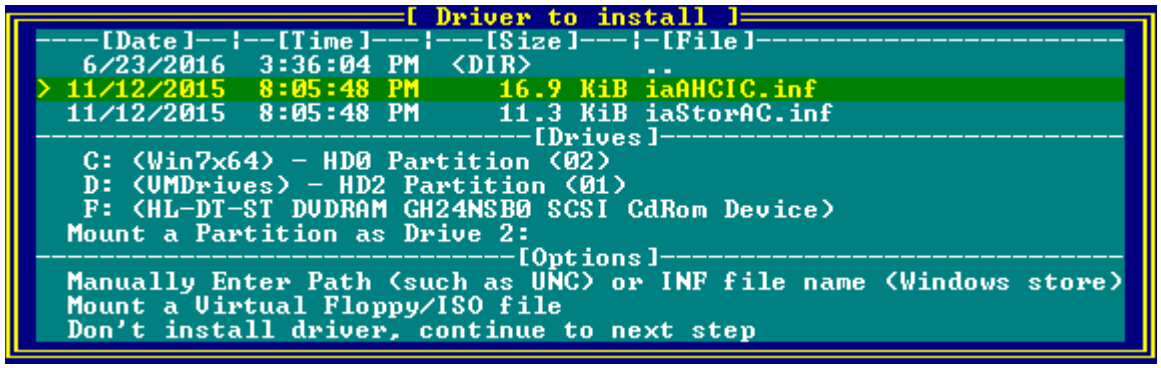
If you select to remove existing drivers you will be given the choice to remove drivers keeping storage and printer drivers, remove drivers keeping printer drivers, or remove CPU drivers only.



You can now select any drivers that need to be installed. This option allows you to easily install drivers into the Windows system. Drivers must be standard Windows INF type drivers and can usually be obtained from the device's manufacturer if not available in the standard Windows drivers. The drivers must be correct for the system being restored (e.g. 32-bit, 64-bit, version of Windows, etc.). If not installing any drivers, you can skip this step.

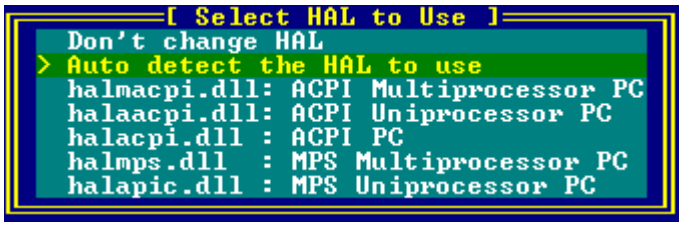


Otherwise, browse to the driver and select it.

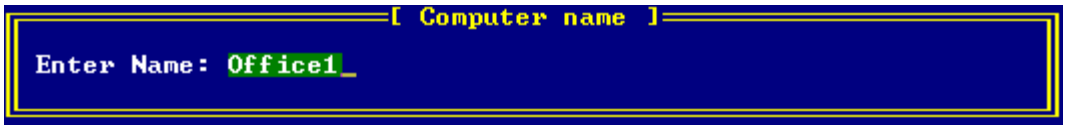


The selected driver will be displayed and you will be given the choice to select another driver or continuing to the next step. To remove a driver from the list to install, highlight it and press Enter. Then select to remove the driver from the list and press Enter to remove it or press ESC to return to the menu.

The next step is to select the HAL to use. Note that this step may not be presented (depends on the version of Windows and type of deployment). Select the appropriate option from the list to continue. If you select a HAL that isn't supported by the version of Windows being restored or copied, an alternate HAL will be used instead, if possible. Selecting "Auto" will determine the HAL to use based on the environment from which the script is running.

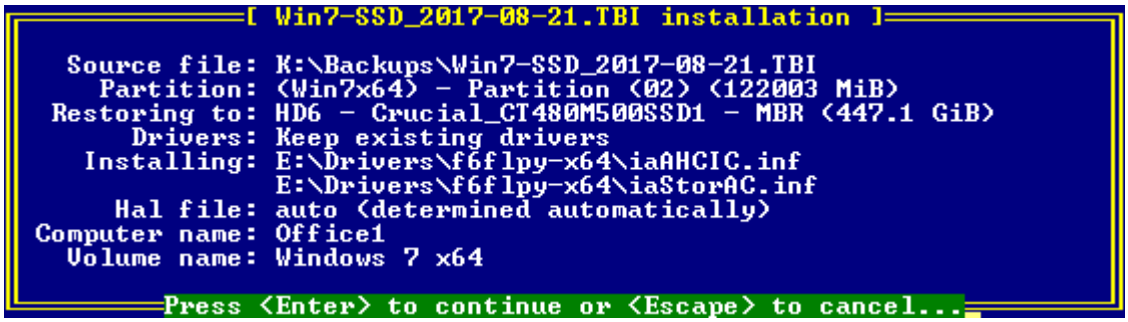


Next, enter the computer name of the target system. If there are multiple systems (e.g. dual-boot/multi-boot), you will be prompted for multiple computer names. If you don't specify a computer name the existing name in the source system will be used.



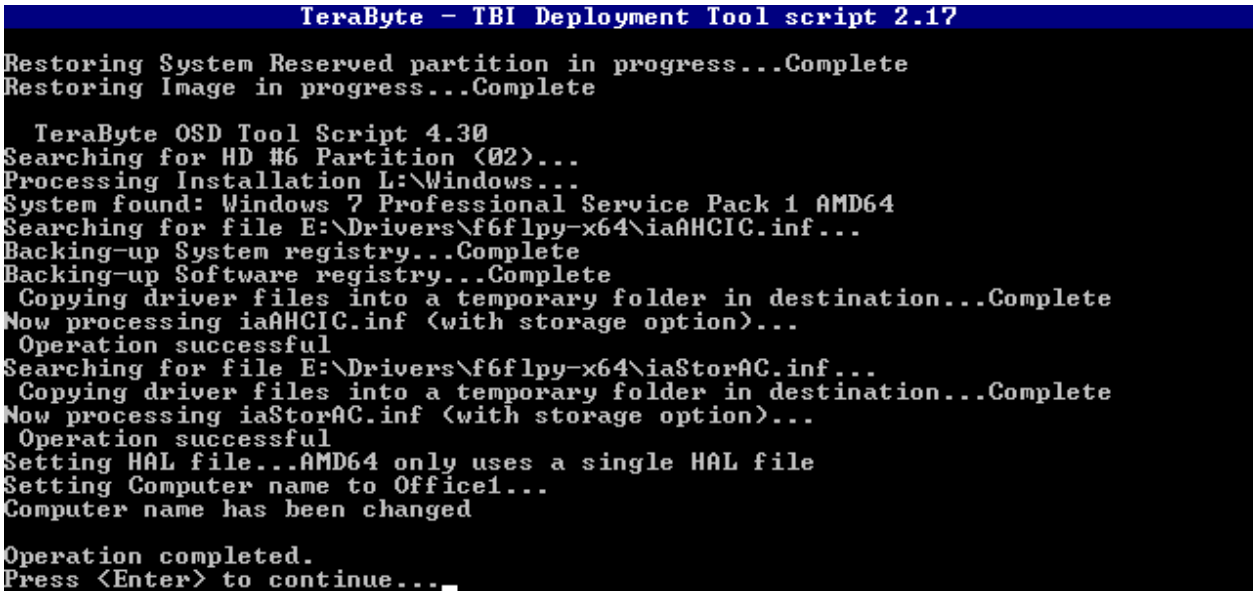
Then enter the volume name for the Windows installation. If you are restoring multiple partitions you will be prompted for multiple volume names. You will not be prompted to name partitions with a size of 500MB or less. If you don't specify a volume name the name in the source partition will be used.

The installation summary is now displayed. If everything looks correct you can press Enter to start the process. Otherwise, press ESC to cancel and return to the main menu.



The appropriate imaging program (Image for Windows, Image for Linux, or Image or DOS) will be launched to restore or copy the source system. If any existing partitions must be deleted to perform the restore, you will be prompted to continue.

When the process has finished successfully, the script will continue with configuring the Windows installation. Press Enter to finish the process.



If the configuration has been completed successfully, an *Operation completed* message will be displayed. Otherwise, an error message will be displayed. In either case, press Enter to exit the script.

Using TBIDTOOL – Automated Mode

In this mode, the script runs based on parameters provided in a configuration file. If any required parameters are missing the script will prompt for them interactively. Otherwise, for optional parameters that are not required and not specified, default values will be used. The configuration filename may be given on the command line as a parameter for the `tbidtool.tbs` script, as shown in the following examples, or it can be browsed to and selected from the main menu. You may wish to use the model names of the computer systems in the filenames of the configuration files you create.

```
tbosdt tbidtool.tbs hp_dl320_config.ini
runtbs tbidtool.tbs dell_opti320_config.ini
TBOSDT.EXE tbidtool.tbs laptop_config.ini
tbosdtw.exe tbidtool.tbs ..\win7office.ini
tbosdtw.exe tbidtool.tbs "C:\Terabyte Scripts\acer.ini"
```

If running in Windows and the configuration file is located on a network share that requires a log-in, you can specify the `/login` option on the command line:

```
runtbs tbidtool.tbs \\server\share\config.ini /login:username*password
tbosdtw.exe tbidtool.tbs \\server\share\config.ini /login:username*password
```

If the password is not specified the script will prompt for it.

Configuration File Format

The configuration file is a plain text file, typically using an `.ini` filename extension (such as `dell_opti320_config.ini`). Inside the file, the tbidtool.tbs parameters (those that would normally be chosen from the menus) are specified by assigning values to variables. Comments can be added to the configuration file by using `//` at the beginning of the comment line.

The following is a list of the parameters that can be specified, the requirements for each, and some examples.

1. TBIFile

Specifies the TBI image file to be restored. The full path can be used or it can be a relative path from the configuration file's location. If the image is a differential and the base full image is not located in the same folder, the base image must be specified using the `TBIBase` parameter or the path to it must be specified using the `SCH` parameter. If the image is an incremental and the base images are not located in the same folder, the path(s) to them must be specified using the `SCH` parameter.

Examples:

```
// image file in same folder as configuration file
TBIFile=Win7Desktop.tbi

// image file on Windows drive
TBIFile=E:\Backups\Win7Desktop.tbi

// image file on UNC path
TBIFile=\\BackupServer\Images\Win7Desktop.tbi

// image file on TBOSDT mounted partition
TBIFile=3:\Backups\Win7Desktop.tbi
```

2. TBIBase

Specifies the TBI base full image if restoring differential image. This parameter is not required if the base image is located in the same folder as the differential or the `SCH` parameter is used to specify the path (the base image will be found automatically, in this case).

Example:

```
// differential image file on Windows drive
TBIFile=E:\Backups\LaptopDiff.tbi

// base image file on alternate Windows drive
TBIBase=G:\LaptopBackups\LaptopCleanFull.tbi
```

3. VhdFile

Use this parameter when the source is a virtual disk instead of a TBI image file. Specify the path to the virtual disk (must be a single file .vhd or .vmdk). Path can be complete or relative to the

configuration file. Note that the `TBIFile` and `TBIBase` parameters do not apply when using `VhdFile`.

Example:

```
// use virtual disk as source
VhdFile=G:\VMDrives\Win7System\Win7.vmdk
```

4. SCH

Use this parameter to specify the search path(s) used when looking for base images in the backup chain when restoring the `TBIFile` image. This parameter can be used multiple times to specify multiple search paths.

Example:

```
// specify search path for base images
SCH=G:\LaptopBackups
```

5. Partition

Specifies the ID or ID's of the partition(s) being restored from a multiple partition image. Values must be entered as required by Image for Windows.

Examples:

```
// restore partition ID 0x2
Partition=0x2

// restore partitions 0x2 and 0x306
Partition=0x2,0x306

// restore volume 3
Partition=3
```

6. Login

If a log-in to a server share is necessary to access the TBI image file, this parameter is mandatory to tell the script it needs to log in. It will apply only if the TBIFile specified is on a UNC path and the script is being run in Windows. The script will prompt for the password if it's not included.

Examples:

```
// log into \\Backups\Images
Login=username*password

// prompt for password
Login=John
```

7. Destination

Specifies the hard drive (HD) number and partition ID of the restore destination. The HD number can also be the NT signature of the drive (if specified with #, 0x, or #0x) or its GUID number. When restoring to an entire disk leave the partition ID empty or set to 0. To restore to the "first free space" available on the drive, the partition ID should be set to 0x100.

Examples:

```
// restore to HD 0
Destination=0

// restore to drive with NT signature, partition ID 0x3
Destination=#0x3ccc415a 0x3

// restore to drive with GUID, use free space
Destination={4d36e97d-e325-11ce-bfc1-08002be10318} 0x80

// restore to HD 0 under Linux
Destination=l0
```

8. HalFile

Specifies the name of the HAL file (hal*.dll) to use. If this parameter is not used, the HAL used by the restored system will not be changed. If the file specified is invalid for the system being restored, an alternate HAL will be used if possible and available. Otherwise, the HAL won't be changed. Just the file name is required (full path is not needed). Possible values are:

hal.dll	Standard PC
halapic.dll	MPS Uniprocessor PC
halmps.dll	MPS Multiprocessor PC
halacpi.dll	ACPI PC
halaacpi.dll	ACPI Uniprocessor PC
halmacpi.dll	ACPI Multiprocessor PC
halsp.dll	Compaq SystemPro Multiprocessor
auto	automatically determine the HAL based on the environment in which the script is running

Example:

```
// use ACPI Multiprocessor PC HAL
HalFile=halmacpi.dll
```

9. InstDrv

When only a numeric value is given, this parameter specifies that default IDE, IDE/AHCI, or NVMe drivers are to be installed. Default value is 0. Possible values are:

- 0 - Do not install default IDE/AHCI drivers
- 1 - Install IDE drivers for XP; IDE/AHCI drivers for Vista and Windows 7/8.x/10
- 2 - Install NVMe drivers (installation must provide native support for NVMe)

NOTE: When using `VhdFile`, the default is to install IDE/AHCI drivers. To override that and not install IDE/AHCI drivers, you must use the `InitDrivers` parameter.

Example:

```
// install default IDE/AHCI drivers
InstDrv=1
```

Note: If the `InitDrivers` parameter is used after `InstDrv` in the configuration file it will override the setting for `InstDrv`.

When more than just a numeric value is given, the `InstDrv` parameter is being used to install a specific driver or specify a driver store (a folder containing multiple drivers).

Installing a Specific Driver

Specify the full path to an INF file to install. If a TBOSDT drive (such as 1: or 2:) is used in the path, then the drive used must be defined with the parameter `Drive` (see item 8 below).

Examples:

```
// install driver provided by Windows
InstDrv=usbstor.inf

// install driver from mounted drive
InstDrv=1:\Drivers\Intel\iaStor.inf

// install driver from Windows drive
InstDrv=E:\IntelDrivers\iaStor.inf
```

For Linux, a driver located on a CD/DVD can be referenced by appending the full path to /cdrom.

Example:

```
// install driver from CD in Linux
InstDrv=/cdrom/drivers/Intel/iaStor.inf
```

This parameter can be set many times (use once per driver to install).

Installing a Driver Store

A driver store is a folder that contains multiple drivers (often one subfolder per driver/device). Populate the store with drivers Windows needs for the system (e.g. AHCI, RAID, USB 3.0, NIC, audio, video, etc.). Required storage drivers (critical drivers) will still need to be installed so Windows can boot. Other drivers in the store will be available when Windows searches for drivers while installing devices, allowing them to be installed automatically.

Specify the path to the driver store folder. The path to the store can be followed by either the `inststor` or the `filter` option (if needed):

`inststor` - all storage drivers in store will be installed
`filter` - specifies to filter on the hardware ID

These options must be prefaced by `-` (see examples below).

Examples:
```
InstDrv=G:\DriverStore -inststor
InstDrv=1:\Drivers\Laptop –inststor –filter
```

You can also specify the store folder and then drivers within it to install by specifying the path and then the relative path to the .inf file. In the example below, the driver store path is `G:\DriverStore` and the `iaStor.inf` driver located in the `Intel` subfolder (full path of `G:\DriverStore\Intel\iastor.inf`). Multiple drivers can be specified (one per line). The `critical`, `normal`, and `filter` options can also be specified, if needed.

Example:
```
InstDrv=G:\DriverStore
InstDrv=Intel\iaStor.inf -critical
InstDrv=ASMedia\asahci64.inf
```

Parameter `InstDrv` can also accept storage drivers provided by Windows (by providing only the name of the .inf file to install, followed by the `-critical` option if needed (not mandatory):

Example: `InstDrv=lsi_scsi.inf –critical`

10. Drive[n]

This parameter is used to specify a drive numbered `n`, where `n` is a TBOSDT drive from 1 to 9. A drive can be used as many times as required in the configuration file, but only needs to be defined once. A drive can be defined at any location in the configuration file.

Examples:

```
// specify the primary CD/DVD drive under DOS/TBOS/UEFI as Drive[1]
Drive[1]=o0

// specify the first CD/DVD drive found under DOS/TBOS/UEFI as Drive[1]
Drive[1]=o

// specify a partition on a USB drive under DOS or TBOS as Drive[5]
Drive[5]=u0 0x304

// specify a partition 0xFE on drive 0 as Drive[2]
Drive[2]=0 0xFE

// specify partition with ID 0x5 on disk with signature 0x12344321
Drive[2]=0x12344321 0x5
Drive[2]=#305414945 0x5

// specify partition with GUID {FE173FDD-1BA9-D14B-49A9-CD5869B69A19}
Drive[2]= {FE173FDD-1BA9-D14B-49A9-CD5869B69A19}
```

```
// specify partition with GUID {FE173FDD-1BA9-D14B-49A9-CD5869B69A19} on disk
{15DD9D75-8497-A707-49A9-CD5869B69A19}
Drive[2]= {15DD9D75-8497-A707-49A9-CD5869B69A19}  {FE173FDD-1BA9-D14B-49A9-
CD5869B69A19}
```

NOTE: You can use Image for Windows to determine the drive numbers and partition ID values. For example, running `imagew /L /stdout:idvalues.txt` will save the data to the `idvalues.txt` file for easy reference.

11. Clrdrv

Specifies if installed drivers are to be removed (cleaned). Default value is `2` if `InitDrivers` parameter specifies that drivers should be removed.
Possible values are:

- `1` - Remove all drivers except storage drivers
- `2` - Remove all drivers
- `3` - Remove only CPU drivers

Examples:

```
// remove all installed drivers
ClrDrv=2

// remove all drivers except storage drivers
ClrDrv=1
```

NOTE: If the `InitDrivers` parameter is used after `ClrDrv` in the configuration file it will override the setting for `ClrDrv`.

12. ComputerName

Use this parameter when you need to change the Windows computer name. The name is limited to 15 characters and can only contain letters, numbers, or hyphens. This parameter can be set many times (once per installation) and will be used in the order of the installations found. You can also specify the number of the installation by using brackets. Omit or leave blank to keep the computer name that exists in the image file.

If you wish the script to create the computer name(s) automatically you can specify `*auto*` as the name. The existing computer name will be used if a system exists on the physical drive before it's replaced with the new installation (for multiple installations, all existing computer names are kept and will be used in the same order for the new installations). Otherwise, the computer name will be determined automatically by the OSDTOOL script.

Examples:

```
// set computer name for first Windows installation
ComputerName=Office1
```

```
// set computer names for first & second Windows installations
ComputerName=Office1
ComputerName=Office2

// set computer name for Windows installation 4
ComputerName[4]=Office4

// let the script automatically set the computer name
ComputerName=*auto*
```

13. VolumeName

Use this parameter when you need to change the volume names for the partitions being restored. This parameter can be set many times (once per unhidden partition being restored) and in the order of the partitions in the image file. Small partitions (500MiB or less) will not have their volume label changed. You can also specify the number of the volume by using brackets. Omit or leave blank to keep the volume name that exists in the image file.

Examples:

```
// set volume name for first partition
VolumeName=System1

// set volume names for first & second partitions
VolumeName=System1
VolumeName=System2

// set volume name for fifth partition
VolumeName[5]=System3
```

14. InitDrivers

Use this parameter to easily select if existing drivers are removed and if default IDE/AHCI/NVMe drivers should be installed. The default value depends on the operation. Possible values are:

- `0` - Keep existing drivers (no changes)
  (default if restoring an image file)
- `1` - Keep existing drivers and install default IDE/AHCI drivers
- `2` - Remove existing drivers
- `3` - Remove existing drivers and install default IDE/AHCI drivers
  (default if copying from a VHD drive)
- `4` - Keep existing drivers and install default NVMe drivers
- `5` - Keep existing drivers and install default IDE/AHCI and NVMe drivers
- `6` - Remove existing drivers and install default NVMe drivers
- `7` - Remove existing drivers and install default IDE/AHCI and NVMe drivers

Examples:

```
// keep existing drivers; install default IDE/AHCI
InitDrivers=1
```

```
// remove existing drivers except for storage drivers
InitDrivers=2
ClrDrv=1

// remove existing drivers except for storage drivers, and install IDE/AHCI
and NVMe
InitDrivers=7
ClrDrv=1
```

NOTE: If the `InstDrv` parameter and/or the `ClrDrv` parameter is after `InitDrivers` in the configuration file their setting will override the setting selected for `InitDrivers`.

15. ChangeDiskSigGUIDs

Set this parameter to change or keep the source disk signature and GUIDs on the target drive. When running in Windows, if you keep the signature and the source drive is present the target drive will be set to offline mode due to a signature collision. Bringing the drive online in this situation will cause Windows to modify the drive and may prevent the drive from booting properly (depends on the OS). The default is `1`. Possible values are:

- `0` – Keep existing disk signature and GUIDs
- `1` – Change the disk signature and GUIDs

Example:

```
// change disk signature and GUIDs
ChangeDiskSigGUIDs=1
```

16. Password

Use this parameter to set the password needed to access the TBI file.

Example:

```
// set password for .tbi file
Password=ThePassword
```

Configuration File Examples

Below are three configuration file examples. Options are commented to explain their function.

Example A – Running script in Windows or WinPE

```
// restore Win7Desktop.tbi located in E:
TBIFile=E:\Backups\Win7Desktop.tbi

// restore to HD 0
Destination=0

// install Intel drivers
InstDrv=E:\IntelDrivers\iastor.inf
```

```
InstDrv=E:\IntelDrivers\iaAHCI.inf

// set computer name
ComputerName=Office

// set volume name for Windows partition
VolumeName=Win7

// remove existing drivers and install default IDE/AHCI drivers
InitDrivers=3

// change disk signature and GUIDs
ChangeDiskSigGUIDs=1
```

Example B – Running script in Windows or WinPE

```
// log-in required for network share
Login=John*pswd

// restore differential image on UNC path
TBIFile=\\BackupServer\Images\LaptopCurrent.tbi

// base (full) image located on different path
TBIBase=\\BackupServer\Base\LaptopCleanInstall.tbi

// restore to HD 1
Destination=1

// install Intel drivers
InstDrv=\\BackupServer\Drivers\iastor.inf
InstDrv=\\BackupServer\Drivers\iaAHCI.inf

// set computer name
ComputerName=Win7Laptop1

// set volume name for Windows partition
VolumeName=Win7

// do not remove existing drivers
ClrDrv=0

// do not install default IDE/AHCI drivers
InstDrv=0

// do not change disk signature and GUIDs
ChangeDiskSigGUIDs=0
```

Example C – Deploying from Virtual Disk in Windows or WinPE

```
// specify virtual disk
VhdFile=G:\VMDrives\Win7x64\Drive0.vmdk

// restore to HD 2
Destination=2

// install Intel drivers
InstDrv=G:\Drivers\x64\iastor.inf
InstDrv=G:\Drivers\x64\iaAHCI.inf

// set computer name
ComputerName=Win7Office

// set volume name for Windows partition
VolumeName=Win7x64

// do not change disk signature and GUIDs
ChangeDiskSigGUIDs=0
```

Example D – Running script in Linux

```
// tbidtool.tbs configuration file used for differential restore and
// driver clean/install from IFL/Linux environment. The image
// in this case is of an entire drive.

// Specify drive/partition where TBI file is located
Drive[1]=l2 0x84

// Specify base image and differential image files
TbiBase=1:/win7.tbi
TbiFile=1:/win7-diff.tbi

// Drive 0 is the target drive
Destination=l0

// Clean all drivers and install default IDE/AHCI drivers
InitDrivers=3

// Specify drive/partition where additional INF file is located
Drive[2]=l1 0x8384
// Install iaStor.inf driver
InstDrv=2:/tbosdts/f6flpy3289/iaStor.inf

// Specify computer name and volume name
```

```
ComputerName=Win7PL
VolumeName=Win7
```

Important Notes & Limitations

When using TBIDTOOL, please keep the following in mind:

Using the Standard (Free) Version of TBOSDT

The standard (free) version of TBOSDT does not support running scripts like OSDTOOL. OSDTOOL requires the Professional (purchased) version to run. TBOSDTS comes with your purchase of one of the TeraByte Drive Image Backup and Restore Suite products or BootIt Collection. The Profession for BootIt version further extends the commands available to include partitioning related commands.

Running TBIDTOOL in Windows Vista, Windows 7, or Windows 8

TBIDTOOL will function under Windows Vista and later if run in Administrator Mode. This means you need to run `tbosdtw.exe` as an administrator. If run as a standard user TBIDTOOL won't be able to access the hard drives and message will be displayed.

Error When Installing a Driver

A driver that causes an error during installation will not be properly installed. Please verify the driver is correct for the version of Windows (XP, Vista, Windows 7, etc. / 32-bit, 64-bit) and the computer.

Some errors (like an "Unknown directive") may require a revised script from Terabyte Support to allow correct installation.

TBOSDT Professional - TBScript Extensions

The Professional version of TBOSDT includes TBScript™ for building scripts that include logical and mathematical operations. TBOSDT extends the functionality of TBScript by offering new subroutines that provide access to registry and hard drive information. The Windows version also includes support for accessing the Windows registry using the normal Win32 API interface.

This section only deals with the actual TBScript extensions and not the TBScript engine. Please refer to the TBScript reference manual included with TBOSDT Professional for a complete understanding of TBScript.

BCDBOOTMATCH Subroutine

Usage:

```
r = BCDBootMatch("path", search, diskref, partref [, sectorsize])
```

Description:

Checks if a given BCD file at “path” is setup to boot an OS based on the additional parameters provided. Set *search* to a non-zero value to search all displayed boot items or zero to only check the first displayed. For matching to GPT entries, use a GUID as the *diskref* and *partref*; for MBR/EMBR type matching use the NT signature as the *diskref* and the starting lba of the OS partition as the *partref* along with providing the sector size of the drive if it’s not 512 bytes. The return value is 1 or 0 to indicate TRUE or FALSE respectively. Added in version 1.78.

Example:

```
sub main()
   bcdmatches = BCDBootMatch("0:\boot\bcd", 1, 0x11222211, 63)
   if bcdmatches<>0 then
     printl("Match Found")
   end if
end sub
```

DELKEY Subroutine

Usage:

```
r = DELKEY(keynum, "keyname" [, "all"])
```

Description:

Deletes the registry key “keyname”. The optional “all” parameter can be used to delete a key and all subkeys. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   DelKey(0, "my key")
   ext("close reg 0")
end sub
```

DELVALUE Subroutine

Usage:

```
r = DELVALUE(keynum, "subkeyname", "valuename")
```

Description:

Deletes a registry value. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   DelValue(0, "my key", "my value")
   ext("close reg 0")
end sub
```

FSINFO Subroutine

Usage:

```
r = FSINFO("path")
```

Description:

Use this this subroutine to obtain information about a file system. The return value of 1 or 0 indicates success or failure respectively. The return value contains several members that provide information about the file system. Added in version 1.79. The member names are:

FS_SN	- Serial number of file system (may be different than expected).
FS_VOLNAME	- The volume name of the file system.
FS_NAME	- The name of the file system (may vary by environment).
FS_CSIZE	- The number of bytes in an allocation unit (cluster).
FS_SSIZE	- The number of bytes in a sector.

Example:

```
sub main()
```

```
 fsi=fsinfo("0:\")

 if fsi then
   printl("FS_SN:", hex(fsi.fs_sn))
   printl("FS_VOLNAME:", fsi.fs_volname)
   printl("FS_NAME:", fsi.fs_name)
   printl("FS_CSIZE:", fsi.fs_csize)
   printl("FS_SSIZE:", fsi.fs_ssize)
else
   printl("FSINFO Failed")
 end if

end sub
```

GETDRVLTRINFO Subroutine

Usage:

`GETDRVLTRINFO([0|1])`

Description:

Queries the system for drive letters and uses them for the `GETHDINFO` subroutine. The queried drive letters remain in system memory (including across scripts) until `GETDRVLTRINFO` is called again with a parameter of zero. This only has any affect when running the Windows version.

Example:

```
sub main()
   GetDrvLtrInfo() // same as GetDrvLtrInfo(1)
   GetHDInfo(0)
   GetDrvLtrInfo(0)
end sub
```

GETHDINFO Subroutine

Usage:

`h = GETHDINFO(num [,1|-1])`

Description:

Obtains information about a hard drive and its partitions for the given hard drive number (zero-based). The return value of 1 or 0 indicates success or failure respectively. In addition, the return value contains several members that provide information about the drive and partitions. Version 1.32 added the optional second parameter; when the second parameter is non-zero, extended information is also returned. Version 1.66 supports the second parameter being less than zero to include unpartitioned space. Version 1.51 adds a zero index to the Partition[] array member if the drive is not partitioned. The member names are:

NUM - Hard Drive Number as provided on input (num).
BIOSHDNUM - BIOS Hard Drive Number (starting at 0x80)
SIG - Disk NT Signature
GUID - Disk GUID if GPT.
SECSIZE - Sector Size
SIZE - Size of Drive in Sectors
C - Last BIOS Cylinder (Total Cylinders – 1)
H - Last BIOS Head (Total Heads – 1)
S - Number BIOS Sectors Per Track
EMBR - Indicator if EMBR exists (1) or not (0)
OutOfSync - Indicator if EMBR exists but MBR partitions are different (1)
GPT - Indicator if GPT exists (1) or not (0)
MaxEntries - Maximum number of primary partitions allowed (added in 1.76)
BUS - Bus type (SCSI,PATA,USB,ASPI,1394,SPTI,SG,SATA,VIRT,SAA,Fibre,RAID,ISCSI,SAS,SD)
NAME - Name of Device.
PARTITION[] - Array (1 based) of partition entries or Partition[0] if drive is not partitioned
PARTITION[].VOLUME[] - Array (1 based) of volume entries for a given Extended Partition

For each PARTITION or VOLUME member the following member names exist:

NAME - Name or Volume Label
STARTLBA - Starting LBA
ENDLBA - Ending LBA
ID - ID of partition as used by TeraByte products
GUID - GPT entry GUID
TYPE - Type of GPT entry
SIZE - Total Sectors
FSID - File System ID (as in partition table)
FSIDN - File System ID (for partition table based on known file systems)
FSINFOID - File System ID (specific to known file systems)
FSNAME - Friendly name of File System ID
FSIDNAME - Friendly name of File System ID (as in partition table)
MBRFLAG - MBR Bootable Flag
MBRENTRY - Entry in MBR (or EBR) Partition Table (0-3 or 255 if not in MBR)
PEFLAG - Partition Flag (EMBR/GPT and Partition Only)
DRVLTR - Drive letter assigned to partition (Windows Version Only)

For each PARTITION or VOLUME member, extended information is returned under a member named EX. The following extended members are available:

USEDSECTORS - Number of used sectors in the file system.
FREESECTORS - Number of free sectors in the file system.
LASTUSEDSECTOR - Last used sector (starting at zero) in the file system.
AACOUNT - Number of allocated areas found in the file system.

Note that USEDSECTORS+FREESECTORS may not equal the size of the partition when the file system area does not match the partition boundaries. The LASTUSEDSECTOR value is the last sector within the file system that is in use (the required number of sectors for restoring as reported in the TeraByte imaging utilities is LASTUSEDSECTOR+1). The AACOUNT represent the number areas within the file system that consist of some number of consecutive in-use sectors.

Example:

```
sub main()

 getdrvltrinfo()
 n=0
 h=gethdinfo(n)

 while h
   printl("HDNum:", h.num)
   printl("DiskID:0x", hex(h.sig))
   printl("secsize:", h.secsize)
   printl("size:", h.size)
   printl("embr:", h.embr)
   printl("gpt:", h.gpt)
   printl("guid:", h.guid)
   printl("bus:", h.bus)
   printl("c:", h.c)
   printl("h:", h.h)
   printl("s:", h.s)

   pn=1
   while h.partition[pn]
     printl("start lba:", h.partition[pn].startlba)
     printl("end lba:", h.partition[pn].endlba)
     printl("fs:", h.partition[pn].fsid)
     printl("name:", h.partition[pn].name)
     printl("fsname:", h.partition[pn].fsname)
     printl("size:", h.partition[pn].size)
     printl("id:", hex(h.partition[pn].id))
     printl("mbrflag:", hex(h.partition[pn].mbrflag))
     printl("mbrentry:", h.partition[pn].mbrentry)
     printl("drvltr:", h.partition[pn].drvltr)
     vn=1
     while h.partition[pn].volume[vn]
       printl("  start lba:", h.partition[pn].volume[vn].startlba)
       printl("  end lba:", h.partition[pn].volume[vn].endlba)
       printl("  fs:", h.partition[pn].volume[vn].fsid)
       printl("  name:", h.partition[pn].volume[vn].name)
       printl("  fsname:", h.partition[pn].volume[vn].fsname)
       printl("  size:", h.partition[pn].volume[vn].size)
       printl("  id:", hex(h.partition[pn].volume[vn].id))
       printl("  drvltr:", h.partition[pn].volume[vn].drvltr)
       vn=vn+1
```

```
      wend
      pn=pn+1
    wend

    printl("")
    n=n+1
    h=gethdinfo(n)
  wend
  getdrvltrinfo(0)

end sub
```

GETKEYORD Subroutine

Usage:

```
k = GETKEYORD(keynum, "subkeyname", index)
```

Description:

Obtains the key name based on a zero-based index.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   i=0
   k=getkeyord(0, "", i)
   while LEN(k)
     printl(k)
     i=i+1
     k=getkeyord(0, "", i)
   wend
   ext("close reg 0")
end sub
```

GETVALUE Subroutine

Usage:

```
r = GETVALUE(keynum, "subkeyname" "valuename")
```

Description:

Obtains a registry value. The return value is 1 or 0 to indicate success or failure respectively. The return value also contains the following members: `NAME`, `TYPE`, and `VALUE`.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   k=getvalue(0, "Software\MyApp", "MyValue")
   printl(k)
   printl(k.value)
   printl(k.name)
   printl(k.type)
   ext("close reg 0")
end sub
```

GETVALUEORD Subroutine

Usage:

```
r = GETVALUEORD(keynum, "subkeyname", index)
```

Description:

Obtains a value based on a zero-base index. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   i=0
   k=getvalueord(0, "target\key", i)
   while k=1
    printl(k.name)
    printl(k.type)
    printl(k.value)
    printl("")
    i=i+1
    k=getvalueord(0, "target\key", i)
   wend
   ext("close reg 0")
end sub
```

HDCACHEINFO Subroutine

Usage:

```
info = HDCACHEINFO(num)
```

Description:

Obtains information about a given hard drives (zero-based) cache that was opened by OPEN CACHE. A non-zero return value indicates a cache was found. The return value contains several members that provide the information. The member names are:

SIZE	- Size of the cache in bytes.
MAXIOREQ	- Largest I/O request (in bytes) to cache.
FLAGS	- Cache flags (Bit 0=Lazy Write, Bit 1=Partial Lazy Write).
READHIT	- Number of cache read hits.
READMISS	- Number of cache read misses.
WRITEHIT	- Number of cache write hits.
WRITEMISS	- Number of cache write misses.
ERRORS	- Number of cache aborts (memory or lazy write failure)
WRITEERRORS	- Number of lazy write failures.

Example:

```
sub main()
   ext("mount 0: 0 0x01")
   ext("open cache 0")

   info=hdcacheinfo(0)

   if info then
    printl("Size:", info.size)
    printl("Max IO Request Cached:", info.maxioreq)
    printl("Flags:0x", hex(info.flags))
    printl("Read Hits:", info.readhit)
    printl("Read Misses:", info.readmiss)
    printl("Write Hits:", info.writehit)
    printl("Write Misses:", info.writemiss)
    printl("Errors:", info.errors)
    printl("Lazy Write Errors:", info.writeerrors)
    printl("")
   else
    printl("No cache found")
   end if

   ext("close cache 0")
   ext("umount 0:")
end sub
```

MD5 Subroutine

Usage:

```
hash = MD5(file_or_data [, option_flags])
```

Description:

Returns a MD5 hash (or empty string on error) of a file or data provided to the subroutine. The optional option_flags parameter determines what to hash as well as how to return the hash value. The option flags are set using bits 0 and 1 as follows:

Bit 0 Clear = file_or_data contains the name of a file to hash.
Set = file_or_data contains the (string or binary) data to hash.

Bit 1 Clear = return the hash as a string value.
Set = return the hash as a binary value.

Example:

```
sub main()
   // note: to set bit 0 we use the value 1 and to set bit 1
   // we use the value 2.  To set both bits we simply add the
   // values together 1+2=3.

   printl("The MD5 hash of string ABC is: " # MD5("ABC", 1))
   printl("The MD5 hash of file FILE.EXT is: " # MD5("FILE.EXT"))

   binary_data_hash=MD5("string", 3)
   binary_file_hash=MD5("d:\path\file", 2)

end sub
```

ISELEVATED Subroutine

Usage:

```
r = ISELEVATED()
```

Description:

Use this to determine if the currently running program is elevated under Windows. The return values are:

-1 = not able to determine or elevation not applicable.
0 = not elevated.
1 = elevated.

Example:

```
sub main()
 if IsElevated()=0 then
    Ext("Elevate ^"" # arg(0) # "^"")
 end if

end sub
```

OSSTAT Subroutine

Usage:

```
r = OSSTAT("filename")
```

Description:

This subroutine is used to obtain information on files and folders directly available to the OS. It is mainly useful under Linux when you want to obtain information about special files (device, symlink, etc.). The return value of 1 or 0 indicates success or failure respectively. The return value contains several members that provide information about the file. The member names are:

ST_DEV	- The ID of the device containing the file.
ST_INO	- The inode number of the file.
ST_MODE	- The files attributes and permissions.
ST_NLINK	- The number of hard links to this file.
ST_UID	- The owners user ID.
ST_GID	- The owners group ID.
ST_RDEV	- The device ID (special files)
ST_SIZE	- The total file size in bytes.
ST_ATIME	- Last access time (number of seconds since Jan 1, 1970).
ST_MTIME	- Last modified time (number of seconds since Jan 1, 1970).
ST_CTIME	- Last status change time (number of seconds since Jan 1, 1970).

Example:

```
sub main()
 stat=osstat("testfile.txt")

 if stat then
   printl("ST_DEV:", stat.st_dev)
   printl("ST_INO:", stat.st_ino)
   printl("ST_MODE:", hex(stat.st_mode))
   printl("ST_NLINK:", stat.st_nlink)
   printl("ST_UID:", hex(stat.st_uid))
   printl("ST_GID:", hex(stat.st_gid))
   printl("ST_RDEV:", stat.st_rdev)
   printl("ST_SIZE:", stat.st_size)
   printl("ST_ATIME:", stat.st_atime)
   printl("ST_MTIME:", stat.st_mtime)
   printl("ST_CTIME:", stat.st_ctime)
 else
   printl("OSSTAT Failed")
 end if

end sub
```

REGISTER Subroutine

Usage:

```
r = REGISTER([user, key])
```

Description:

Issues TBOSDT register command. If valid user/key is provided then it is applied even if they cannot be saved to the .ini file, whereas a failure to save the name/key manually entered will not be applied. This command is not available when already registered. The return value is an error code or zero if no error.

Example:

```
sub main()
   register("name", "key")
end sub
```

SETVALUE Subroutine

Usage:

```
r = SETVALUE(keynum, "subkeyname", "valuename", "valuetype", "value")
```

Description:

Adds or changes a registry value. The value types allowed are SZ, EXPANDSZ, HEX, DWORD, MULTISZ, QWORD, and NONE. HEX is equivalent to the binary type. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open reg 0 test.hiv")
   ext("open key 0 ^"^" 0")
   setvalue(0, "", "testvalue", "dword", 0x1234)
   ext("close reg 0")
end sub
```

SHA256 Subroutine

Usage:

```
hash = SHA256(file_or_data [, option_flags])
```

Description:

Returns a SHA256 hash (or empty string on error) of a file or data provided to the subroutine. The optional option_flags parameter determines what to hash as well as how to return the hash value. The option flags are set using bits 0 and 1 as follows:

Bit 0 Clear = file_or_data contains the name of a file to hash.
Set = file_or_data contains the (string or binary) data to hash.

Bit 1 Clear = return the hash as a string value.
Set = return the hash as a binary value.

Example:

```
sub main()
   // note: to set bit 0 we use the value 1 and to set bit 1
   // we use the value 2.  To set both bits we simply add the
   // values together 1+2=3.

   printl("The SHA256 hash of string ABC is: " # SHA256("ABC", 1))
   printl("The SHA256 hash of FILE.EXT is: " # SHA256("FILE.EXT"))

   binary_data_hash=SHA256("string", 3)
   binary_file_hash=SHA256("d:\path\file", 2)

end sub
```

TBOSDT Global Variable

Description:

This global variable indicates the version of TBOSDT being used. The format of this string variable is:

V.vv – L

Where V.vv is the version number and L is the level. Level A indicates TBOSDT Professional, Level B indicates TBOSDT Professional for BootIt.

Example:

```
sub main()
   printl("Version: ", tbosdt) // outputs Version: 1.29 - A
end sub
```

TBOSDT_USER Global Variable

Description:

This global variable indicates the licensed user running TBOSDT.

Example:

```
sub main()
   printl("User: ", tbosdt_user)
end sub
```

TBOSDT_KEY Global Variable

Description:

This global variable indicates the license key running TBOSDT.

Example:

```
sub main()
   printl("Key: ", tbosdt_key)
end sub
```

WINDELKEY Subroutine (Windows Version Only)

Usage:

```
r = WINDELKEY(keynum, "keyname" [, "all"])
```

Description:

Deletes the registry key “keyname” under an open key referenced by keynum. The optional “all” parameter can be used to delete a key and all subkeys. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   WinDelKey(0, "my key")
   ext("close winkey 0")
end sub
```

WINDELVALUE Subroutine (Windows Version Only)

Usage:

```
r = WINDELVALUE(keynum, "subkeyname", "valuename")
```

Description:

Deletes a registry value. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   WinDelValue(0, "my key", "my value")
   ext("close winkey 0")
end sub
```

WINGETKEYORD Subroutine (Windows Version Only)

Usage:

```
k = WINGETKEYORD(keynum, "subkeyname", index)
```

Description:

Obtains the key name based on a zero-based index.

Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   i=0
   k=wingetkeyord(0, "", i)
   while LEN(k)
     printl(k)
     i=i+1
     k=wingetkeyord(0, "", i)
   wend
   ext("close winkey 0")
end sub
```

WINGETVALUE Subroutine (Windows Version Only)

Usage:

```
r = WINGETVALUE(keynum, "subkeyname" "valuename")
```

Description:

Obtains a registry value. The return value is 1 or 0 to indicate success or failure respectively. The return value also contains the following members: NAME, TYPE, and VALUE.

Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   k=wingetvalue(0, "Software\MyApp", "MyValue")
   printl(k)
   printl(k.value)
   printl(k.name)
   printl(k.type)
```

```
   ext("close winkey 0")
end sub
```

WINGETVALUEORD Subroutine (Windows Version Only)

Usage:

```
r = WINGETVALUEORD(keynum, "subkeyname", index)
```

Description:

Obtains a value based on a zero-base index. The return value is 1 or 0 to indicate success or failure respectively.

Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   i=0
   k=wingetvalueord(0, "target\key", i)
   while k=1
    printl(k.name)
    printl(k.type)
    printl(k.value)
    printl("")
    i=i+1
    k=wingetvalueord(0, "target\key", i)
   wend
   ext("close winkey 0")
end sub
```

WINSETVALUE Subroutine (Windows Version Only)

Usage:

```
r = WINSETVALUE(keynum, "subkeyname", "valuename", "valuetype", "value")
```

Description:

Adds or changes a registry value. The value types allowed are SZ, EXPANDSZ, HEX, DWORD, MULTISZ, QWORD, and NONE. HEX is equivalent to the binary type. The return value is 1 or 0 to indicate success or failure respectively.

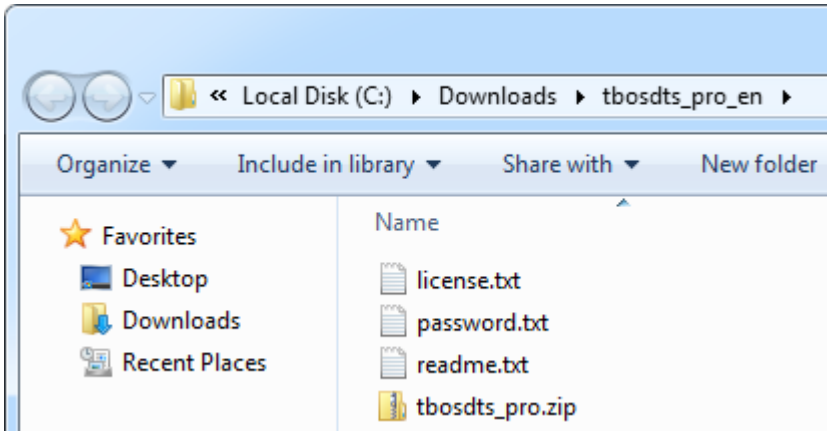
Example:

```
sub main()
   ext("open winkey ^"^" hkcu")
   winsetvalue(0, "", "testvalue", "dword", 0x1234)
   ext("close winkey 0")
end sub
```

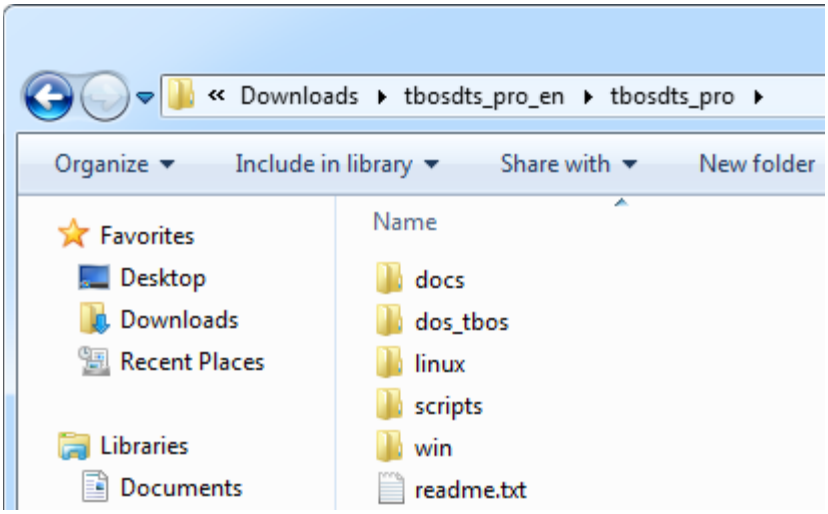
Running Scripts Using TBOSDT Professional

The Professional version of the TeraByte OS Deployment Tool (TBOSDT) is required to run scripts that utilize the TBScript™ engine (i.e. the OSDTOOL script). It comes with your purchase of one of the TeraByte Drive Image Backup and Restore Suite products or BootIt Collection. The Profession for BootIt version further extends the commands available to include partitioning related commands.

After you download the TBOSDT zip file (`tbosdts_pro_en.zip`) from the TeraByte website, extract the contents. You can use Explorer for this or any other standard file compression program.



The instructions for unzipping the extracted `tbosdts_pro.zip` file can be found in the `readme.txt` and `password.txt` files.



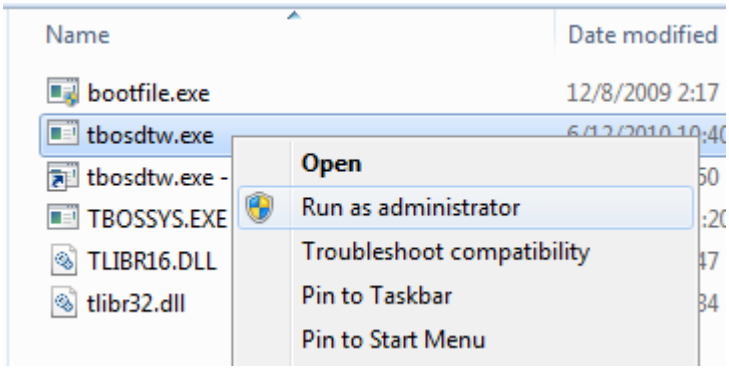
If running under Windows, the required program file (`tbosdtw.exe`) can be found in the `win` folder. If running under DOS, the required program file (`TBOSDT.EXE`) can be found in the `dos_tbos` folder. If running under Linux, the required program file (`tbosdt`) can be found in the `linux` folder.

NOTE: Once extracted, you may wish to move (or copy) the contents of the sub-folder you'll be using to a location easier to access. For example: You might move the contents of `win` to the `C:\TBOSDT` folder.

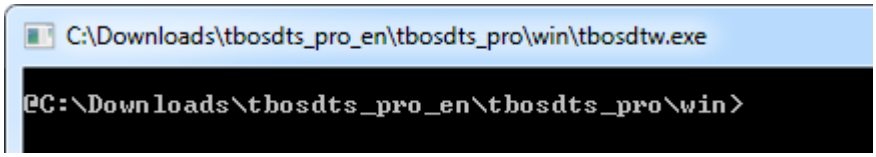
Running the script can be done from the TBOSDT command line or from a shortcut.

Running a Script from the Command Line

Run the correct version of TBOSDT for your OS version. If running in Windows Vista or later (e.g. Windows 7) with UAC enabled, right-click on the program and select the **Run as administrator** option (TBOSDT may not be able to access the drives correctly if this option is not selected).



When the program starts, it will be at the command line prompt.



Type the command to run the script and press ENTER. In this example, the `osdtool.tbs` script is in the `C:\TBOSDT` folder.

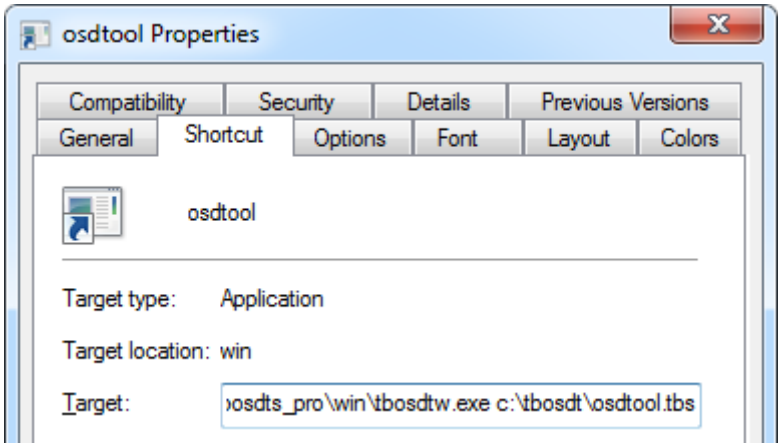
```
runtbs c:\tbosdt\osdtool.tbs
```

The script will run. When finished, the script will return to the command line prompt. To exit TBOSDT, type `exit` at the prompt and press ENTER.

Running a Script from a Windows Shortcut

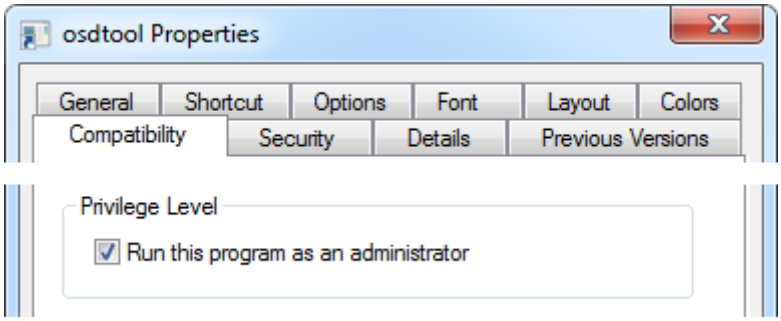
Create a standard Windows shortcut setup to run TBOSDT and the script. In this example, the `osdtool.tbs` script is in the `C:\TBOSDT` folder. The “Target” box contains the following (all on one line):

```
C:\Downloads\tbosdts_pro_en\tbosdts_pro\win\tbosdtw.exe c:\tbosdt\osdtool.tbs
```



NOTE: If the path to `tbosdtw.exe` contains any spaces, make sure to place the complete path in quotes. Do the same for `obsdtool.tbs` if its path contains any spaces.
For example: `"C:\tbosdts pro\tbosdtw.exe" "c:\tbosdt pro\scripts\osdtool.tbs"`

When running TBOSDT in Windows Vista or later (e.g. Windows 7) with UAC enabled, make sure to select the **Run this program as an administrator** option on the **Compatibility** tab of the shortcut (if not selected, TBOSDT may not be able to access the drives correctly).



Launch the shortcut to run the script. When the script is exited or finishes, the TBOSDT window will close.