Command	Description	Example
Ata	Execute a custom ATA command without data transfer according to the ATA standard. Works for SATA/IDE drives only.	`// Perform STANDBY IMMEDIATE command` `Ata 0xE0 0 0 0`

AtaIn	Execute a custom ATA command with data direction IN according to the ATA standard. Works for SATA/IDE drives only.	`// Perform IDENTIFY DEVICE command.`
		`AtaIn 0xEC 0 0 0 512`
		`Print LastResult.Bytes`

AtaOut	Execute a custom ATA command with data direction OUT according to the ATA standard. Works for SATA/IDE drives only.	`// Perform WRITE DMA EXT Command: LBA=1200, write 1 sector`
		`bytes = new byte[512]`
		`bytes[0] = 0x37`
		`AtaOut 0x35 0 1200 1 bytes`

Compare	Compare device sectors with the specified pattern.	`c = Compare 00 0 1000000`
		`maxBlockSize = 0`
		`foreach (block in c.Blocks)`
		`  if (maxBlockSize < block.Count)`
		`  maxBlockSize = block.Count`
		`Print "Maximum interval of different sectors: " maxBlockSize`

Contains	Indicates whether the soughtString occurs within this text.	`res = SmartTable`
		`Contains res.Text "Head flying hours"`
		`if (LastResult.OK)`
		`  Print "Name of SMART attribute 240 is found."`

DeviceFirmware	Reads firmware revision of the hard drive.	`DeviceFirmware`
		`c = Contains LastResult.Text "30"`
		`if (c.OK)`
		`{`
		`  Print "Serial number contains '30' at position: " c.Number`
		`}`

DeviceModel	Reads model number of the hard drive.	`DeviceModel`
		`c = Contains LastResult.Text "FUJITSU"`
		`if (c.OK)`
		`{`
		`  Print "The device was manufactured by Fujitsu"`
		`}`

DeviceSerial	Reads the serial number of the device.	`DeviceSerial`
		`hasZero = Contains LastResult.Text "0"`
		`if (hasZero.OK)`
		`{`
		`  Print "Serial number contains zero at position: " hasZero.Number`
		`}`

DeviceSize	Reads capacity of the device in bytes.	`DeviceSize`
		`sectorCount = LastResult.Number / 512`
		`if (sectorCount > 268435455)`
		`{`
		`  Print "The device has more than 28 bit LBA addressable sectors: " sectorCount`
		`}`

Entropy	Calculates how much random data is on the entire media, or the specified region. 100 - totally random data, 0 - fully ordered data.	`// Analyse entropy of 1M sectors`
		`Entropy 0 1000000`
		`if (LastResult.Number > 90)`
		`{`
		`  Print "This device possibly contains encrypted data"`
		`}`

EntropyBytes	Calculates how much random data is in byte array. 100 - totally random data, 0 - fully ordered data.	`// Let's find out entropy of device ID sector`
		`idRes = Identify`
		`EntropyBytes idRes.Bytes`

Erase	Performs wiping with specified pattern of the entire media or the interval defined by startLBA and endLBA parameters.	`// Wipes sectors 0-200 with HEX pattern 55AA` `Erase 55AA 0 200`

Exit	Stops script execution and writes message to log if one is specified.	`last = LastLBA`
		`for (startLba = 0; startLba < last.Number; startLba += 1024)`
		`{`
		`  endLba = startLba + 1023`
		`  ReadSectors startLba endLba`
		`  if (LastResult.Error)`
		`  Exit "Script is stopped after the first read error"`
		`}`

Find	Performs regular expression search over specified disk region.	`//Searches for hexadecimal numbers` `Find "[0-9a-fA-F]+" 0 1000000`
FindHEX	Performs a search of a specified HEX pattern over a specified disk region.	`// Search for NTFS file records` `FindHEX "46 49 4C 45" 0 1000000`

FindWords	Performs a search of words or phrases over a specified disk region.	`FindWords "Volume Directory Folder" 0 10000`
		`foreach (t in LastResult.Strings)`
		`  if (t == "Folder")`
		`  {`
		`  Print "Folder is found!"`
		`  break`
		`  }`

Hash	Performs hash calculation of the entire media or specified range. Supports simultaneous	`// Run double-hash calculation` `Hash "MD5 SHA1"`

	calculation of two hashes. Treats unreadable (bad) sectors as zeros.	

Identify	Runs native device identification command.	`Identify`
		`if (LastResult.Bytes[434] == 1)`
		`{`
		`  Print "This device is SSD"`
		`}`

LastLBA	Reads the number of the last device LBA.	`lastLbaRes = LastLBA`
		`prevSector = 0`
		`prevSector = lastLbaRes.Number - 1`
		`Scan linear prevSector lastLbaRes.Number`

Length	Count all elements in an array or string.	`matched = FindWords "NTFS exFAT HFS"`
		`Length matched.Strings`
		`Print LastResult`

PercentLBA	Calculates LBA number based on percent value. percent of 50 will return a middle LBA.	`// Calculate LBA number corresponding`
		`// to one third of Max LBA`
		`PercentLBA 33`

PowerCycle	Powers off and on the device and waits for it to become ready.	`Identify`
		`PowerCycle`
		`Identify`

PowerOff	Powers off the device.	`PowerOff`
		`Sleep 1000`
		`PowerOn`

PowerOn	Powers on the device and waits for it to become ready.	`PowerOff`
		`Sleep 1000`
		`PowerOn`

Print	Prints specified string/variable value to the report.	`LastLBA`
		`if (LastResult.Number == 0)`
		`  Print "Firmware error or PUIS mode is active. Last LBA: " LastResult`

RandomNumber	Generates a random integer value from 0 up to maxValue.	`// Read one of sectors from 0 to 1000 selecting it randomly`
		`RandomNumber 1000`
		`ReadSectors LastResult.Number LastResult.Number`

ReadDCO	Reads raw Device Overlay Configuration sector of the device.	`dco = ReadDCO`
		`if (dco.Error)`
		`  Exit`
		
		`isErrorLogEnabled = (dco.Bytes[14] >> 2) & 1`
		`if (isErrorLogEnabled == 1)`
		`  Print "SMART error log is enabled"`

		else Print "SMART error log is disabled"

ReadMaxAddressDCO	Reads maximum LBA according to Device Configuration Overlay.	dco = ReadMaxAddressDCO
		hpa = ReadNativeMaxAddress
		if (dco.Number > hpa.Number)
		{
		Print "Device capacity is limited by DCO."
		Print "To be able to access the entire surface you need to reset the DCO to factory settings."
		}

ReadNativeMaxAddress	Reads native maximum LBA number according to Host Protected Area.	hpa = ReadNativeMaxAddress endLba = LastLBA if (hpa.Number > endLba.Number) Print "HPA is active. To be able to access the entire surface you need to disable HPA."
ReadSectors	Reads raw sector data from a range of sectors. Max allowed range is 1024 sectors.	ReadSectors 10 1000 if (LastResult.OK) Print LastResult
Reset	Performs hard device reset.	Reset Identify
RestoreDCO	Restores factory settings of Device Overlay Configuration (DCO).	hpaAddress = ReadNativeMaxAddress dcoAddress = ReadMaxAddressDCO if (dcoAddress.Number > hpaAddress.Number) RestoreDCO
Scan	Performs scan of the entire media, or of the specified region.	Scan linear 0 0xF00000 if (LastResult.OK) Print "Media surface up to LBA 0xF00000 is OK."

SeekTest	Performs seek test	// Perform random seek test within the first 600K sectors;
		// Timeout = 50 seconds
		SeekTest random 50 0 600000
		// Perform backward seek test within the last 600K sectors;
		// Timeout = 50 seconds. We will need to perform some math to
		// calculate the range we need to test:
		lastSector = LastLBA
		firstSector = 0
		firstSector = lastSector.Number - 600000
		SeekTest backward 50 firstSector lastSector.Number

SetMaxAddress	Limits the hard drive's capacity to the specified maximum LBA via Host Protected Area.	`// Clips ATA device size to 10000 sectors (5 Mbytes)`
		`// by means of HPA`
		`// Since the very first sector on the drive is 0 and not 1,`
		`// we need to set Max LBA to 9999 and not to 10000.`
		`SetMaxAddress 9999`
		`PowerCycle`
		`Identify`
		`LastLba`

Sleep	Pause the script execution for specified number of milliseconds.	`Sleep 10000` `Print "10 seconds elapsed."`
SmartAttributeRaw	Reads raw SMART value by attribute ID.	`res = SmartAttributeRaw 5` `Print "Reallocated Sectors Count value is " res`
SmartAttributeThreshold	Reads threshold SMART value by attribute ID.	`res = SmartAttributeThreshold 5` `Print "Reallocated Sectors Count threshold value is " res`
SmartAttributeValue	Reads normalized SMART value by attribute ID.	`res = SmartAttributeValue 5` `Print "Reallocated Sectors Count normalized value is " res`
SmartAttributeWorst	Reads worst SMART value by attribute ID.	`res = SmartAttributeWorst 5` `Print "Reallocated Sectors Count worst value is " res`
SmartTable	Reads the entire SMART attribute table of the device.	`SmartTable`
SwitchAutoExitOnError	Toggles automatic exit after the first error on or off. By default, auto-exit is off.	`SwitchAutoExitOnError on` `Print "From now on, script will stop if it encounters any error during execution."`
SwitchAutolog	Toggles automatic script command logging on or off. By default, autologging is on for any script.	`SwitchAutolog off` `Print "All scripts commands have begun to work in a silent mode."`

TimeFromStart	Returns the time in seconds since the moment the current script has started.	`Scan linear 0 1000000`
		`t = TimeFromStart`
		`Print "Linear scan time: " t.Number`

ToInteger	Converts byte array into an equivalent 32-bit integer value. Default value is little-endian.	`id = Identify`
		`bytes = new byte[4]`
		`for (i = 0; i < 4; i++)`
		`{`
		`  bytes[i] = id.Bytes[120 + i]`
		`}`
		`ToInteger bytes`

		`Print "Total sector count = " LastResult.Number`

ToLong	Converts byte array into an equivalent 64-bit long value. Default value is little-endian.	`id = Identify`
		`bytes = new byte[8]`
		`for (i = 0; i < 8; i++)`
		`{`
		`  bytes[i] = id.Bytes[200 + i]`
		`}`
		`ToLong bytes`
		`Print "Number of sectors = " LastResult.Number`

ToNumber	Converts the specified string to Number. Decimal and hexadecimal formats supported.	`jcbCards = "((?:2131|1800|35[0-9]{3})[0-9]{11})"`
		`matches = Find jcbCards`
		`foreach (m in matches.Strings)`
		`{`
		`  numeric = ToNumber m`
		`  // Validate numeric card number`
		`  // ....`
		`}`

ToShort	Converts byte array into an equivalent 16-bit short value. Default value is little-endian.	`id = Identify`
		`bytes = new byte[2]`
		`for (i = 0; i < 2; i++)`
		`{`
		`  bytes[i] = id.Bytes[434 + i]`
		`}`
		`ToShort bytes`
		`Print "Nominal media rotation rate = " LastResult.Number`

TransferReadRateTest	Performs transfer rate test executing read operations. The benchmark is performed for the inner, middle and outer tracks. For more precise results allow 30-60 seconds for testing. 20 seconds is considered sufficient for most applications.	`// Measure read transfer rate for 20 seconds` `TransferReadRateTest 20`
TransferWriteRateTest	Performs transfer rate test executing write operations. The benchmark is performed for the inner, middle and outer tracks. For more precise results allow 30-60 seconds for testing. 20 seconds is considered sufficient for most applications. Binary zeroes by default or specified pattern are used for writing.	`// Measure write transfer rate for 40 seconds` `TransferWriteRateTest 40`

WriteDCO	Writes raw Device Configuration Overlay (DCO) sector.	`dco = ReadDCO`
		`bytes = dco.Bytes`
		`// Decrease one of DCO Max LBA bytes`
		`bytes[6] -= 1`
		`// Write change to DCO`
		`WriteDCO bytes`

WriteSectors	Overwrites a specified sector range with the predefined data (pattern). Max allowed range is 1024 sectors. To fill/wipe entire drive, use Erase command instead.	`data = new byte[512]`
		`data[0] = 'H'`
		`data[1] = 'I'`
		`data[2] = 0xBE`
		`data[3] = 0xAF`
		`WriteSectors 0 data`
		`if (LastResult.OK)`
		`  Print  "Sector has been written"`

WriteTest	Performs write test.	`// Perform random 50-second write test`
		`// within whole media surface`
		`WriteTest random 50`
		`// Perform backward write test`
		`// within the last 600K sectors for 30 seconds`
		`lastSector = LastLBA`
		`firstSector = 0`
		`firstSector = lastSector.Number - 600000`
		`WriteTest backward 30 firstSector lastSector.Number`