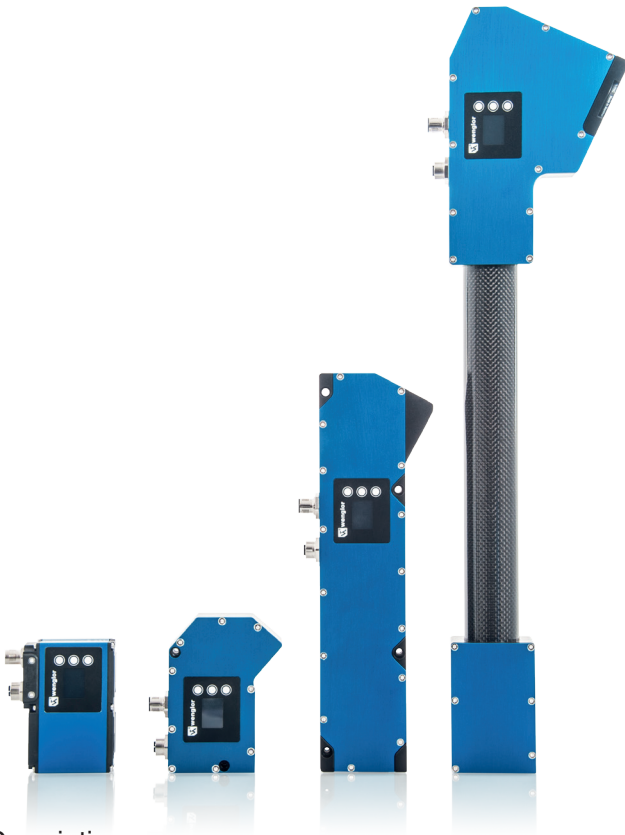


weCat3D MLSL/MLWL

2D/3D Profile Sensors



DLL Interface Description

Table of Contents

- 1. **Change Index** 4
- 2. **Document Information** 4
 - 2.1 References 4
- 3. **Introduction** 5
 - 3.1 System Requirements 5
- 4. **Application Example** 5
- 5. **SDK Functions** 8
 - 5.1 Connecting weCat3D sensor 8
 - 5.2 Closing connection 8
 - 5.3 Check connection 9
 - 5.4 Get general sensor information 9
 - 5.5 Get scanned profile 10
 - 5.6 Check DLL FiFo state 11
 - 5.7 Reset DLL FiFo 12
 - 5.8 Setup profile sensor 12
 - 5.9 Read DLL version 12
 - 5.10 Read property value (experimental) 13
- 6. **List of ASCII Commands** 14
 - 6.1 Setup profile sensor 14
 - 6.1.1 Initiate reboot 14
 - 6.1.2 Setup exposure time 14
 - 6.1.3 Setup acquisition line time 14
 - 6.1.4 Deactivate laser 14
 - 6.1.5 Set user LED 15
 - 6.1.6 Enable signal (Z) 15
 - 6.1.7 Enable signal (strength) 15
 - 6.1.8 Enable signal (width) 15
 - 6.1.9 Setup socket connection timeout 16
 - 6.1.10 Setup the heartbeat signal 16
 - 6.1.11 Start acquisition 16
 - 6.1.12 Stop acquisition 16
 - 6.1.13 Reset settings 16

6.1.14	Reset encoder	16
6.1.15	Reset picture counter	17
6.1.16	Save settings	17
6.1.17	Load settings	17
6.1.18	Setup trigger source	17
6.1.19	Setup trigger divider	18
6.1.20	Software trigger command	18
6.1.21	Setup encoder trigger function	18
6.1.22	Enable fixed frame mode	19
6.1.23	Setup number of profiles in fixed frame mode	19
6.1.24	Setup sync out	19
6.1.25	Setup delay of sync out	19
6.1.26	Enable signal	19
6.1.27	Setup signal minimum width	20
6.1.28	Setup signal maximum width	20
6.1.29	Setup signal selection	20
6.1.30	Setup encoder count direction	20
6.1.31	Region of interest (ROI)	20
6.1.32	E/A functions	21
6.2	Read properties of weCat3D sensor (experimental)	24
7.	Data Structure	25
7.1	General	25
7.1.1	Buffer structure (one selected signal)	25
7.1.2	Buffer structure (two selected signals)	25
8.	Appendix	26
8.1	GetInfo (XML mode)	26
8.2	GetInfo (Text mode)	27

1. Change Index

Rev.	Date	Author	Approval	Description
1				Initial document
2		BE	CHW	<ul style="list-style-type: none">• New document structure• Additional ASCII commands• New experimental functions• Changes for ASCII commands for DLL version 1.7.0 or later

Table 1: Changes to the Document

2. Document Information

2.1 References

Document	Version	Reference
Operating_Instructions_MLSL-MLWL.pdf	1.00	

Table 2: References to external documents

3. Introduction

This document describes the functions and the commands for using the DLL to realize custom application development for the weCat3D product series. The DLL is for users who want to create their own 2D/3D applications using the weCat3D Sensor series.

3.1 System Requirements

Applications development with the DLL/shared library requires a Microsoft operating system (WIN7, WIN10)/Linux (Ubuntu 14.04, OpenSuSe).

The weCat3D product series requires a GigabitEthernet interface.

Windows QT SDK requires the Microsoft VC++ 2013 Redistributable Package (already delivered with the SDK).

The SDKs are available for download at www.wenglor.com in the product's separate download area.

The SDK is distributed through different packages, each package provides an example project on how to use the SDK functions. Each project is written under different IDEs using different programming language.

SDK Windows demo project comes with both DLL types (x32 and x64). To compile the project with x32 compiler, copy the content of EthernetScanner_x32 into the EthernetScanner folder. To compile the demo project with x64 compiler, copy the content of EthernetScanner_x64 into the folder EthernetScanner folder.

4. Application Example

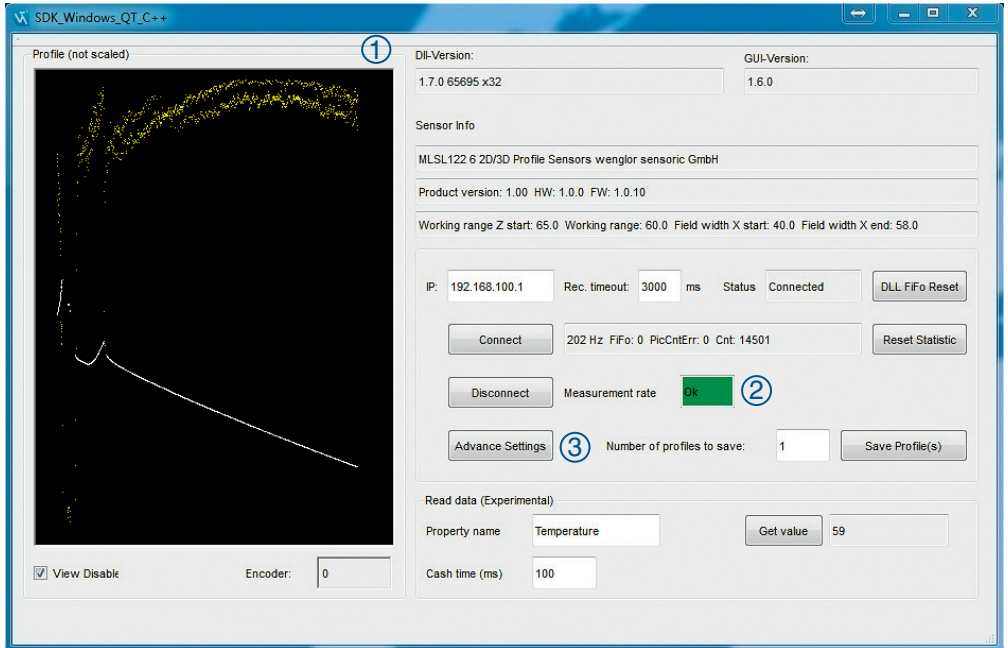
Each SDK has a demo project with the source code. The demo application is given as a mean to demonstrate the data transmission from the sensor to the application using the SDK functions.

Below is a screenshot from the SDK_Windows_QT_C++.



NOTE!

Please check functionality with the demo client before start programming.



Main window of the demo application delivered with the QT SDK for windows (may differ in other demo projects)

- ① The demo project builds a connection to the profile sensor and shows a 2D representation of the scanned profile.
The white points in the representation show the scanned profile, while the yellow dots display the intensity (signal strength) of each point.
- ② The main window in the demo project shows also the state of the measurement rate.
If the measurement rate is within the allowed limits, the display field will show "Ok" (green background).
If the measurement rate is too fast, the display field will show "too fast" (red background).
- ③ Click on the button "Advance Settings" in order to check the ROI settings and the corresponding max scan request value.
If the demo project fails to build a connection to the profile sensor, it will display the error message "Ethernet-Scanner_Connect: Error in connection":



NOTE!

Please check the IP address of your profile sensor and your network settings (check the reachability of the profile sensor in the network by pinging the profile sensor's IP address using the ping command in the operating system console, e.g. "ping 192.168.100.1").

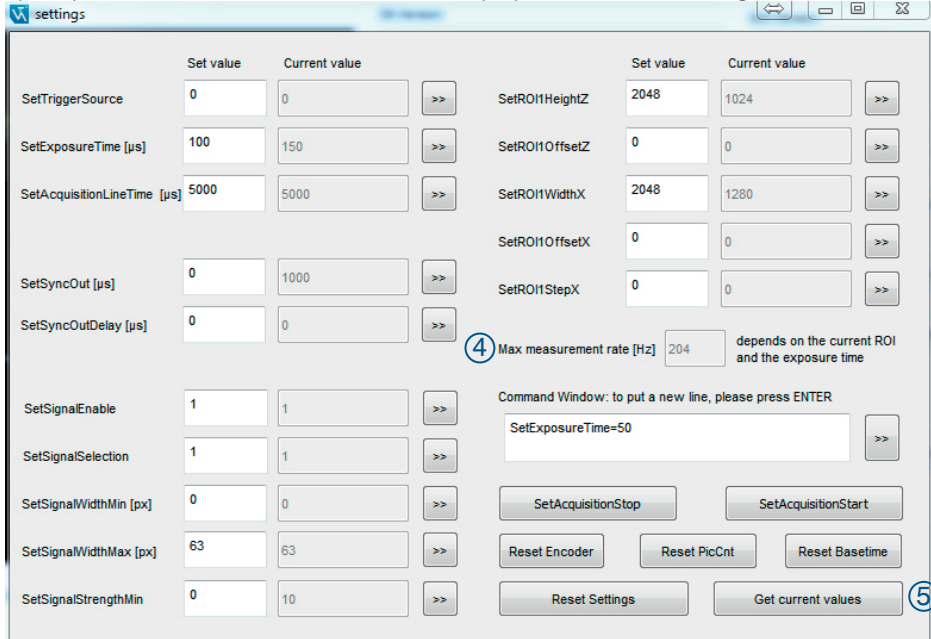
The demo project shows the error message "Error in linearisation data" if the profile sensor is already connected to another app/computer



NOTE!

You can check the connection state of the profile sensor through the web interface. Just type the IP address of the profile sensor in a web browser and look at the "connected to" field on the right side of the web interface.

The advance settings window (opens only if the connection to the profile sensor is established) allows to set-up the profile sensor and read the values of basic properties. It allows sending raw ASCII commands as well.



The screenshot shows a 'settings' window with two columns of parameters. Each parameter has a 'Set value' and a 'Current value' field, with a '>>' button to update the current value. The parameters are:

Parameter	Set value	Current value
SetTriggerSource	0	0
SetExposureTime [μs]	100	150
SetAcquisitionLineTime [μs]	5000	5000
SetSyncOut [μs]	0	1000
SetSyncOutDelay [μs]	0	0
SetSignalEnable	1	1
SetSignalSelection	1	1
SetSignalWidthMin [px]	0	0
SetSignalWidthMax [px]	63	63
SetSignalStrengthMin	0	10
SetROIHeightZ	2048	1024
SetROIOffsetZ	0	0
SetROIWidthX	2048	1280
SetROIOffsetX	0	0
SetROIStepX	0	0
Max measurement rate [Hz]	204	depends on the current ROI and the exposure time

Below the parameters, there is a 'Command Window' section with a text input field containing 'SetExposureTime=50' and a '>>' button. At the bottom, there are several buttons: 'SetAcquisitionStop', 'SetAcquisitionStart', 'Reset Encoder', 'Reset PicCnt', 'Reset Basetime', 'Reset Settings', and 'Get current values' (marked with a circled 5).

The settings window of the demo project delivered with the QT SDK for windows (may differ in other demo projects)

④ The max measurement rate field computes the maximum measurement rate for triggering the profile sensor from the current ROI settings. The equation for computing the maximum value is available in the source code.

⑤ “Get current values” button updates the values of some basic properties by calling and parsing the XML data description from the profile sensor.



NOTE!

The computed max measurement rate value is only an approximate value.

In order to get scanned profiles from the sensor in a reliable way, the host application should send the following commands in the given sequence to the sensor to build a connection:


1. Build a connection to the sensor (EthernetScanner_Connect).
2. Check the connection status (EthernetScanner_GetConnectStatus).
3. Make sure that the DLL is initialized (EthernetScanner_GetInfo).



NOTE!

Make sure that the connection to the profile sensor has already been established.

- 4. (Optional) Set up the sensor according to application needs through ASCII commands (EthernetScanner_WriteData).
- 5. Read the scanned profiles from the sensor (EthernetScanner_GetXZI) and process the data accordingly.
- 6. Disconnect from sensor before ending the application (EthernetScanner_Disconnect).




NOTE!

In DLL version 1.7.0 or higher, there is no need to send the ASCII command “SetInitializeAcquisition” to the profile sensor after each connection. The DLL sends this command automatically. If your program sends this command, the DLL (1.7.0 or higher) will ignore it. Sending the command “SetInitializeAcquisition” from the DLL has brought a lot of performance improvements to the DLL.

5. SDK Functions

All the SDK functions are based on C function standard calls (_stdcall) and are compatible with all compilers that support C programming language. In fact, since the functions are based on C standard call, they can be deployed in a wide range of IDEs (QT, Visual studio C++, Visual Basic, C#, Delphi, Matlab, Labview, Embarcadero, etc.)

5.1 Connecting weCat3D sensor

Command	void* EthernetScanner_Connect(char *chIP, char *chPort, int iTimeout)
Parameter 1	char *chIP: the IP address of the sensor: „192.168.100.1“ with \0 at the end
Parameter 2	char *chPort: the port number of the sensor: „32001“ with \0 at the end
Parameter 3	int iTimeout: Timeout in [ms] for the receive-function to close the connection, if no data are received. 0 to keep the connection alive
Response	void* a handle to the profile sensor. A NULL pointer is returned in case of failure
Description	<div>This function will create a connection to the weCat3D sensor. The function will return a handle to the profile sensor, which will be passed to other functions.</div> <div>NOTE! For checking the connection status with the profile sensor enter “EthernetScanner_GetConnectStatus”.</div>

5.2 Closing connection

Command	void* _stdcall EthernetScanner_Disconnect(void *pEthernetScanner)
Parameter 1	void*: the handle of the profile sensor (returned by the function EthernetScanner_Connect) will be disconnected
Response	void* a NULL pointer
Description	Close the connection between the DLL and the weCat3D sensor. The handle of the profile sensor will not be valid anymore

5.3 Check connection

Command	<code>void EthernetScanner_GetConnectStatus(void *pEthernetScanner, int *uiConnectStatus)</code>
Parameter 1	<code>void*</code> : a handle to the profile sensor returned by the function <code>EthernetScanner_Connect</code>
Parameter 2	<p><code>int*</code> : a pointer to an integer variable, through which the connection status is returned. There are two states for the connection:</p> <p><code>ETHERNETSCANNER_TCPSCANNERCONNECTED</code> (3) meaning that the profile sensor is successfully connected to the DLL and the given IP and PORT in the function <code>EthernetScanner_Connect</code> are valid. This status does not tell, if a valid measurement data from the profile sensor can be obtained, please refer to <code>EthernetScanner_GetInfo</code>.</p> <p><code>ETHERNETSCANNER_TCPSCANNERDISCONNECTED</code> (0) meaning that the profile sensor is disconnected or the given IP and PORT in the function <code>EthernetScanner_Connect</code> are not valid.</p>
Response	---
Description	This function checks the connection status to the profile sensor. The function is a non-blocking function.

5.4 Get general sensor information

Command	<code>int EthernetScanner_GetInfo(void *pEthernetScanner, char *chInfo, int iBuffer, char *ch-Mode)</code>
Parameter 1	<code>void*</code> : a handle to the profile sensor returned by the function <code>EthernetScanner_Connect</code>
Parameter 2	<code>char*</code> : a pointer to a raw buffer (of type <code>char</code>), where the profile sensor information will be written.
Parameter 3	<code>int</code> : the length of the raw buffer. The programmer should make sure that the length of the raw buffer is larger than the length of the return sensor information.
Parameter 4	<p><code>char*</code> : the type of the information to be returned through the argument 2. There are two types:</p> <p>“text”: returns basic information about the profile sensor as a text such as sensor name, working ranges, MAC, etc. (see appendix 1 for example).</p> <p>“xml”: returns a full description of the profile sensor in xml format. The xml contains general information about the profile sensor, the current values of all features as well as all ASCII commands supported by the profile sensor. See appendix 2 for an example.</p>

Response	<p>The function will return ETHERNETSCANNER_GETINFOSMALLBUFFER (-2), if the length of the raw buffer(argument 3) is less than the length of the information to be returned through the raw buffer (argument 2).</p> <p>The function will return ETHERNETSCANNER_GETINFOINVALIDXML (-4), if it failes to get the xml description from the profile sensor.</p> <p>The function will return ETHERNETSCANNER_GETINFONOVALIDINFO (-3), if it failes to get general info from the the profile sensor.</p> <p>Else the function will return the length of the data [in bytes] written in the raw buffer (argument 2).</p> <p>Normally the function will return ETHERNETSCANNER_GETINFONOVALIDINFO only in the case when the profile sensor is already connected to another app/computer, or when the function EthernetScanner_Connect within the same app is called a second time before calling the EthernetScanner_Disconnect function. In that case, the handle to the profile sensor returned in the first EthernetScanner_Connect call will be overwritten by the next call of the EtherntnetScanner_Connect function. Thus, the profile sensor can not be disconnected from the DLL anymore. A workaround is to restart the profile sensor.</p> <p>Only when the function returns a value larger than 0, then it is possible to get measurment data from the profile sensor (see EthernetScanner_GetXZExtended).</p>
Description	<p>The function is used to get general information about the connected profile sensor. The function is also used to make sure that the profile sensor is not connected to another app/computer. Please note that the profile sensor supports onyl 1:1 connection.</p>

5.5 Get scanned profile

Command	int EthernetScanner_GetXZExtended(void *pEthernetScanner, double *pdoX, double *pdoZ, int *piIntensity, int *piPeakWidth, int iBuffer, unsigned int *puiEncoder, unsigned char *pucUSRIO, int dwTimeOut, unsigned char *ucBufferRaw, int iBufferRaw, int *iPicCnt)
Parameter 1	void *: a handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	double*: pointer to a raw buffer (of type double) used by the function to write in the X coordinates [in mm] of the measured profile. Pass NULL, if the buffer is not used
Parameter 3	double*: pointer to a raw buffer (of type double) used by the function to write in the Z coordinates [in mm] of the measured profile. Pass NULL, if the buffer is not used
Parameter 4	int* a pointer to a raw buffer (of type int) used by the function to write the intensity [10 bit] of the measured profile. Pass NULL, if this buffer is not used
Parameter 5	int* a pointer to a raw buffer (of type int) used by the function to write the peek width [in pixel < 32 pxl] . Pass NULL, if this buffer is not used
Parameter 6	int: the length of the raw buffers passed in argument 2 to 5. The length of the raw buffers should be larger than the number of measured points returned by the profile sensor
Parameter 7	int*: a pointer to a variable (of type int) which returns the encoder value of the current measured profile

Parameter 8	int*: a pointer to a variable (of type int) which returns the IO status of the current measured profile. The IO status is decoded as follows: bit0: EA1 bit1: EA2 bit2: EA3 bit3: EA4 bit4: TTL Encoder A bit5: TTL Encoder B bit6: TTL Encoder C Pass NULL, if this value is not used
Parameter 9	The value of the blocking time to wait for a new measured profile, until the function times out. The value 0 makes the function non-blocking.
Parameter 10	Deprecated. Pass NULL
Parameter 11	Deprecated. Pass NULL
Parameter 12	int* : a pointer to variable (of type int) which returns the picture counter of current measured profile. This value is used to control the sequence of the received profiles.
Response	In the case of a success call, the function will return the total number of points of the measured profile written to the raw buffer (in argument 2 to 5). The function will return ETHERNETSCANNER_GETXZINONEWSCAN (-1) if no new profile is available, ETHERNETSCANNER_GETXZINVALIDLINDATA (-2) if the profile sensor is already connected to another app/computer (see EthernetScanner_GetInfo to check if the profile sensor is connected to other app/computer), ETHERNETSCANNER_READD-ATASMALLBUFFER (-3) if the length of the buffer given in argument 1 to 5 is shorter than the data to be written.
Description	The function pulls one scan from the internal FiFo in the DLL, if a new scan is available. The DLL saves all the scanned profiles received from the profile sensor in an internal FiFo buffer. The programer is responsible to pull the scanned profiles using this function as fast as possible to prevent overflow of the FiFo. If the program can not pull the scanned profiles fast enough, then it is recommended to decrease the output rate of the profile sensor. The function could be set to be blocking or non-blocking depending on the value of argument 9. To check the status of the FiFo see EthernetScanner_GetDllFiFoState function. To know how to set up the output rate of the profile sensor see the ASCII commands SetTrigger-Mode and SetAcquisitionLineTime.

5.6 Check DLL FiFo state

Command	int EthernetScanner_GetDllFiFoState(void *pEthernetScanner)
Parameter 1	void* : the handle to the profile sensor returned by the function EthernetScanner_Connect
Response	int: the status of the FiFo in the DLL in % (0 – 99)
Description	the function is used to check the status of the internal FiFo in the DLL to prevent the overflow and hence, to prevent the loss of unpulled scanned profiles.

5.7 Reset DLL FiFo

Command	<code>int EthernetScanner_ResetDllFiFo(void *pEthernetScanner)</code>
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Response	The function returns ETHERNETSCANNER_OK if the calling was successful
Description	The function is used to reset the internal FiFo in the DLL. However, that could lead to the loss of unpulled scanned profiles. This function is useful, if the application can not pull the scanned profiles fast enough and the programmer wants to process the latest scanned profile. In that case, it is recommended to call this function just before calling the function EthernetScanner_GetXZiExtended.



5.8 Setup profile sensor

Command	<code>int EthernetScanner_WriteData(void *pEthernetScanner, char *ucBuffer, int uiBuffer)</code>
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	char*: a pointer to a raw buffer (of type char) which contains the ASCII command to be sent to the profile sensor
Parameter 3	int: the length of the raw buffer passed in argument 2
Response	The function returns the number of bytes sent to the profile sensor. Normally, it should be the same length as the ASCII command.
Description	The function is used to send ASCII commands to setup the profile sensor

5.9 Read DLL version

Command	<code>int EthernetScanner_GetVersion(unsigned char *ucBuffer, int uiBuffer)</code>
Parameter 1	char*: a pointer to a raw buffer (of type char) used by the function to write in the DLL version
Parameter 2	int: the length of the raw buffer used in argument 1
Response	The function returns the total length (in bytes) of the written data in the raw buffer. If the length of DLL version to be written in the raw buffer is larger than the length of the raw buffer given in argument 2, the function returns ETHERNETSCANNER_ERROR (-1).
Description	The function is used to check the current version of the DLL

5.10 Read property value (experimental)

Command	<code>int EthernetScanner_ReadData(void* pEthernetScanner, char * chPropertyName, char * chRetBuf, int iRetBuf, int iCacheTime)</code>
Parameter 1	void * the handle to the profile sensor returned by the function EthernetScanner_Connect
Parameter 2	char * buffer with the property name (ending with char /0)
Parameter 3	char * return buffer for the property value
Parameter 4	int the length of the return buffer
Parameter 5	int the cache time in ms (XML mode ≥ 0 or Scan mode -1)
Response	The function returns ETHERNETSCANNER_READDATAOK (1) in case of success operation, ETHERNETSCANNER_READDATASMALLBUFFER (-1) if the return buffer passed in argument 3 is shorter than the length of the data available to be written in the buffer, ETHERNETSCANNER_READDATANOTSUPPORTEDMODE (-2) in the case where the given property is not supported in the current read mode (like PicutreCounter in general data mode), ETHERNETSCANNER_READDATAFEATURENOTDEFINED (-3) if the property name is not supported.
Description	<p>The function reads the property value from the profile sensor. For experimental reasons, this DLL version supports only few properties (see chapter 6.2). The properties' values are cached in the DLL and the iCacheTime (argument 4) defines how old the property value should be before writing it in the return buffer (argument 3).</p> <p>In the case where $iCacheTime \geq 0$ (general data mode); if the properties cache is older than the given value in iCacheTime, the DLL will refresh its cache first and then write the property value in the return buffer.</p> <p>In the case where $iCacheTime = -1$ (scan data mode), the DLL will read the property value from the data delivered by the current scan (pulled using the function EthernetScanner_GetXZIEExtended). The property value in this mode will hold until the next successful call of the function EthernetScanner_GetXZIEExtended.</p> <div>  NOTE! Setting a low value for iCacheTime in general data mode (i.e. $iCacheTime = 0$) will decrease the performance of the DLL since the DLL is then forced to read the full properties from the profile sensor and parse it each time the EthernetScanner_ReadData function is called. This would be evident if the DLL is working on low resource system. </div> <div>  NOTE! Not all properties are supported on both reading modes (see chapter 6.2). </div>

6. List of ASCII Commands

6.1 Setup profile sensor

Below are the ASCII commands that are used to set up the profile sensor using the function EthernetScanner_WriteData.


6.1.1 Initiate reboot

Command	SetReboot		
Description	Reboot the system		

6.1.2 Setup exposure time

Command	SetExposureTime=x		
Parameter	Range of value x: 26...100000	Default:	150
Description	The value is in μ s.		

6.1.3 Setup acquisition line time

Command	SetAcquisitionLineTime=x		
Parameter	Range of value x: 166 ... 100000	Default:	MLWL: 5714 MLSL: 5000
Description	<p>Time between two consecutive profiles in μs. This function is only effective in internal trigger mode. 166μs = 6000 Hz</p> <p>Explanation: MLWL: 5714μs = 175 Hz MLSL: 5000μs = 200 Hz</p> <div> NOTE! It is necessary to reduce the ROI settings in the profile sensor to get a higher LineTimeRate.</div>		

6.1.4 Deactivate laser

Command	SetLaserDeactivated=x		
Parameter	Values of x: 0 Laser on 1 Laser off	Default:	0
Description	<p>Default Laser on</p> <p>Software command to control the laser as a global function. If this function is set to 1(enabled), then all other enabled signals on the E/A do not have any effects.</p>		

6.1.5 Set user LED

Command	SetUserLED=x		
Parameter	Values of x: 0 off 1 red 2 green 3 orange	Default:	0
Description	The command controls the user LED for optical display of the application status directly at the weCat3D Sensor.		

6.1.6 Enable signal (Z)

Command	SetSignalContentZ=x		
Parameter	Values of x: 0 disabled 1 enabled	Default:	1
Description	By default, the data sent from the profile sensor contains four signals: Z (the depth), X (the width), I (the intensity /signal strength) and the Peak width. This command will disable sending the Z signal with the data to save the bandwidth of the network.		

6.1.7 Enable signal (strength)

Command	SetSignalContentStrength=x		
Parameter	Values of x: 0 disabled 1 enabled	Default:	1
Description	By default, the data sent from the profile sensor contains four signals: Z (the depth), X (the width), I (the intensity /signal strength) and the Peak width. This command will disable sending the I signal with the data to save the bandwidth of the network.		


6.1.8 Enable signal (width)

Command	SetSignalContentWidth=x		
Parameter	Values of x: 0 disabled 1 enabled	Default:	1
Description	By default, the data sent from the profile sensor contains four signals: Z (the depth), X (the width), I (the intensity /signal strength) and the Peak width. This command will disable sending the peak width signal with the data to save the bandwidth of the network.		

6.1.9 Setup socket connection timeout

Command	SetSocketConnectionTimeout=x		
Parameter	Value Range of x: 0 ... 60000ms	Default:	0
Description	Device (sensor) ethernet connection: rx-tx-timeout in ms. Default: 0 no connection will be closed, if no ethernet data has been transferred (rx/tx).		

6.1.10 Setup the heartbeat signal

Command	SetHeartBeat=x		
Parameter	Value range of x: 0 ... 10000 ms	Default:	0
Description	<p>The command will activate the heartbeat signal in the profile sensor. The profile sends every x ms a heartbeat signal if the profile sensor does not send/receive any data. x = 0 deactivates the heartbeat signal.</p> <div>NOTE! It is recommended to activate the heartbeat signal in the profile sensor, the heart-beat signal will enable the profile sensor to detect a physical (electrical) connection drop (like in the case where the network cable is unplugged). Thus, closing the connection to the host and allowing the host to build a new connection to the profile sensor.</div>		

6.1.11 Start acquisition

Command	SetAcquisitionStart		
Description	The profile data will be sent to the host (default).		

6.1.12 Stop acquisition

Command	SetAcquisitionStop		
Description	The profile data will stop sending to the host.		

6.1.13 Reset settings

Command	SetResetSettings		
Description	Reset the sensor settings to the default values. The sensor's IP address is retained.		

6.1.14 Reset encoder

Command	SetResetEncoder		
Description	Set the encoder counter of both encoders (HTL and TTL) to 0.		

6.1.15 Reset picture counter

Command	SetResetPictureCounter
Description	Set the value of the picture counter to 0.

6.1.16 Save settings

Command	SetSettingsSave=x		
Parameter	Values of x: 0,1, 2	Default:	0
Description	0: default (this setting will be saved automatically for the load after the power supply of the sensor is switched on) 1: Set1 2: Set2		

6.1.17 Load settings

Command	SetSettingsLoad=x		
Parameter	Values of x: 0,1, 2	Default:	0
Description	0: default (this setting will be loaded after the power supply of the sensor is switched on) 1: Set1 2: Set2		

6.1.18 Setup trigger source

Command	SetTriggerSource=x		
Parameter	Values of x: 0 Internal trigger mode 1 Hardware trigger mode over SynIn function on E/A1...E/A4 2 Encoder trigger mode over HTL/TTL encoder 3 Software trigger mode	Default:	0
Description	Set 0 for internal triggering. Set 1 to trigger the profile sensor through hardware signal (useful for synchronizing multiple sensors in an application). Set 2 to trigger the profile sensor through the encoder signal (if E/A 1 and E/A 2 are defined as encoder function, the E/A encoder will be used as the trigger source, otherwise the TTL-RS422 is used). Set 3 to trigger the signal through the software command "SetTriggerSoftware".		

6.1.19 Setup trigger divider

Command	SetTriggerEncoderStep=x		
Parameter	Value range of x: 0 ... 1000	Default:	0
Description	Set a trigger divider for both hardware trigger source (Syncln input) and encoder trigger source (Encoder HTL or TTL). The profile sensor will be triggered at the x+1 signal. This property is useful, if we have a high frequency external trigger source (either Encoder or Syncln signal).		

6.1.20 Software trigger command

Command	SetTriggerSoftware		
Parameter	---		
Description	Trigger the sensor to scan a profile over a software command. The sensor should be in software trigger mode.		

6.1.21 Setup encoder trigger function

Command	SetEncoderTriggerFunction=x		
Parameter	Values of x: 0 DirectionUp 1 DirectionDown 2 Motion 3 PositionUp 4 PositionDown	Default:	2
Description	DirectionUp: The encoder will trigger the profile sensor only in one direction (counting up) DirectionDown: The encoder will trigger the profile sensor only in one direction (counting down). Motion: The encoder will trigger the profile sensor in both directions (counting up and down) PositionUp: The encoder will trigger the profile sensor in one direction (counting up) and only if the encoder position (counter value) is larger than the latest position. PositionDown: The encoder will trigger the profile sensor in one direction (counting down) and only if the encoder position (counter value) is smaller than the latest position.		

6.1.22 Enable fixed frame mode

Command	SetTriggerAmountProfilesY=x		
Parameter	Values of x: 0: Internal trigger mode 1: Hardware (Syncln) trigger mode 2: Encoder trigger mode 3: Software trigger mode	Default:	-1
Description	This command is used to activate the fixed frame mode in the profile sensor. In fixed frame mode the profile sensor sends x number of profiles (x = 0, 1, 2 or 3)(see SetAmountOfProfilesY) to the host and then stops until the profile sensor receives a new SetAcquisitionStart command or hardware signal on ProfileEnabel pin (if defined). -1 means that the fixed frame mode in the profile sensor is switched off (the profile sensor is working in normal trigger mode). The value -1 can not be used in this function. If you want to switch off the fixed frame mode, please use the command SetTriggerSource=x.		

6.1.23 Setup number of profiles in fixed frame mode

Command	SetAmountProfilesY=x		
Parameter	Value range of x: 0...10000	Default:	0
Description	The command sets up the number of profiles to be sent to the host in the fixed frame mode (see SetTriggerAmountProfilesY).		

6.1.24 Setup sync out

Command	SetSyncOut=x		
Parameter	Value range of x: 0...100000	Default:	1000
Description	Defines the signal width (duration in μ s) of the SyncOut signal (high) for the E/A SyncOut.		

6.1.25 Setup delay of sync out

Command	SetSyncOutDelay=x		
Parameter	Value Range of x: 0...100000	Default:	0
Description	Defines the value of (switching) delay (in μ s) of the SyncOut trigger signal (high) for the E/A SyncOut.		

6.1.26 Enable signal

Command	SetSignalEnable=x		
Parameter	Values of x: 1 Signal 1 2 Signal 2 3 Signal 1 + Signal 2	Default:	1
Description	The command sets the number of signals which are send with each profile. See the description of signal selection.		

6.1.27 Setup signal minimum width

Command	SetSignalWidthMin=x		
Parameter	Value Range of x: 0...63	Default:	0
Description	Signal width filter: This function is a filter to define the minimum signal width (Peakwidth) in pixels. Usual values: 2 or 3		

6.1.28 Setup signal maximum width

Command	SetSignalWidthMax=x		
Parameter	Value Range of x: 0...63	Default:	63
Description	Signal width filter: This function is a filter to define the maximum signal width (Peak-width) in pixels. Usual values: 12		

6.1.29 Setup signal selection

Command	SetSignalSelection=x		
Parameter	Values of x: 0 top 1 strong 2 width 3 bottom	Default:	1
Description	Defines the signal which is to be used for the profile output.		

6.1.30 Setup encoder count direction

Command	SetEncoderCountDirection=x		
Parameter	Values of x: 0 normal 1 invert	Default:	0
Description	The count direction of the encoder values can be inverted.		

6.1.31 Region of interest (ROI)

6.1.31.1 Setup ROI width in X

Command	SetROI1WidthX=x		
Parameter	Value range of x: MLSL: 32...1280 MLWL: 32...2048	Default:	MLSL: 1280 MLWL: 2048
Description	Amount of camera rows to readout: MLWL: no effect on the measurement rate, effect on the ethernet bandwidth MLSL: in steps of 16, effect on the measurement rate, effect on the ethernet bandwidth		

6.1.31.2 Setup ROI offset in X

Command	SetROI1OffsetX=x		
Parameter	Value range of x: MSL: 0...1279 MLWL: 0...2047	Default:	0
Description	MLWL: in steps of 1 MSL: in steps of 16 Defines the offset of the ROI in X-direction in relation to the first line.		

6.1.31.3 Setup ROI step X

Command	SetROI1StepX=x		
Parameter	Value Range of x: MSL: no function MLWL: 0...2047	Default:	0
Description	MLWL: can be set with every pixel		

6.1.31.4 Setup ROI height in Z

Command	SetROI1HeightZ=x		
Parameter	Value Range of x: MSL: 35. ...1280 MLWL: 35...2048	Default:	MSL: 1280 MLWL: 2048
Description	Amount of camera lines to readout: MLWL: effect on the measurement rate MSL: effect on the measurement rate		

6.1.31.5 Setup ROI offset in Z

Command	SetROI1OffsetZ=x		
Parameter	MLWL: value range of x: 0...2046 MSL: Value range of x: 0...1022	Default:	0
Description	Defines the offset of the ROI in Z-direction in relation to the first line.		

6.1.32 E/A functions

6.1.32.1 Setup E/A functions

The sensor offers 4 separate E/A functions. The following commands relate to these E/A functions and can be used for all E/As. The encoder HTL functions are only available for E/A 1 and E/A 2. The following explanation uses the syntax to set up the E/A 1. For addressing E/A 2 to E/A 4 use the same syntax:

```
SetEA1Function=1
SetEA2Function=2
SetEA3FunctionLaserOff=0
```

Command	SetEA1Function=x		
Parameter	Values of x: 1 sync_in 2 sync_out 3 input 4 output 5 encoder_ab	Default:	5
Description	<p>Encoder_A/B (E/A 1 + E/A 2): Input function for connecting an HTL (5 to 24 V, A/B channel) rotary encoder. This function must be set for E/A 1 and E/A 2 at the same time. This function is only available for E/A 1 and E/A 2.</p> <p>If the encoder function is enabled on E/A 1/2, then the encoder value in the GetXZI function will be provided from this encoder! If no E/A 1/2 encoder function is selected, then the encoder value in the GetXZI function will be provided from TTL-RS422.</p>		

6.1.32.2 Setup E/A function laser off

Command	SetEA1FunctionLaserOff=x		
Parameter	Values of x: 0 disabled 1 enabled	Default:	0
Description	0: disabled (no function), laser is on 1: enabled E/A high state: laser is off E/A low state: laser is on The E/A should be set to input (see SetEA1Function) for this function to work		

6.1.32.3 Setup E/A function profile enable

Command	SetEA1FunctionProfileEnable=x		
Parameter	Values of x: 0 disabled 1 enabled	Default:	0
Description	0: disabled (no function) 1: enabled E/A high state: profiles will be send to the host The E/A should be set to input (see SetEA1Function) for this function to work		

6.1.32.4 Setup E/A function encoder reset

Command	SetEA3FunctionResetCounter=1\rSetEA3ResetCounterRepeat=2\rSetEA3ResetCounterSignaledge=2\rSetEA3ResetCounterEncoderHTL=1\rSetEA3ResetCounterEncoderTTLRS422=1\r		
Parameter	See the XML file description for more details on each function and its parameters		
Description	The command (set of commands) sets the E/A pin (the third pin in this case) to reset the HTL and TTL counters.		

6.1.32.5 Setup E/A 1 input function

Command	SetEA1InputFunction=x		
Parameter	Values of x: 0 Ub inactive 1 Ub active	Default:	1
Description	The input signal can be inverted as a function.		

6.1.32.6 Setup E/A 1 input load

Command	SetEA1InputLoad=x		
Parameter	Values of x: 0 input load disabled 0 mA 1 input load enabled 2 mA	Default:	0
Description	Enable/disable the extra load on the E/A input to get 0 level defined (Helpful for some PLC hardware).		

6.1.32.7 Setup E/A 1 output

Command	SetEA1Output=x		
Parameter	Values of x: 1 Push-Pull 2 PNP 3 NPN	Default:	1
Description	Determines the output mode for the E/A (Push-Pull, PNP or NPN).		

6.1.32.8 Setup E/A 1 output function

Command	SetEA1OutputFunction=x		
Parameter	Values of x: 0 NO 1 NC	Default:	0
Description	0: NO: high level 1: NC: low level in order of the output Push-Pull PNP NPN		

6.2 Read properties of weCat3D sensor (experimental)

The following table shows the current ASCII commands that can be used to read the properties of the profile sensor using the function EthernetScanner_ReadData.

The table shows also the availability of each command for each read mode.

See the demo project in the SDK for a ReadData example.

Property name	General data mode	Scan data mode	Remarks
ExposureTime	+	+	
SystemTime	-	+	
PictureCounter	-	+	
IOState	+	+	bit0: EA1; bit 1:EA2; bit 2: EA3; bit3: EA4
Temperature	+	+	
EncoderHTL	+	+	
EncoderTTL	+	+	
ScannerState	-	+	bit0: Profile sensor OK bit1: ExposureTime OK bit2: LaserONTime OK bit3: Internal bit4: Internal bit5: Measurement rate too fast
AcquisitionLineTime	+	+	

(+) is available; (-) is not available

7. Data Structure

7.1 General

The profile information queried by the GetXZIEExtended function are displayed separately as buffer for each value (X,Z,I). If the measured object is located outside the measuring range, the measured value is set to 0.

7.1.1 Buffer structure (one selected signal)

In case of just one selected signal (signal selection) the buffer structure appears in this order:

Buffer	X	Buffer	Z	Buffer	I	Buffer	Peakwidth	
0	double	0	double	0	int	0	int	1 st point
1	double	1	double	1	int	1	int	2 nd point
2	double	2	double	2	int	2	int	3 rd point
... *			

* to ...1280 MLSL / ...2048 MLWL

7.1.2 Buffer structure (two selected signals)

If the signal selection is set up to get signal 1 and signal 2, the buffer contains the data in the following, different order:

Buffer	X	Buffer	Z	Buffer	I	Buffer	Peakwidth	
0	double	0	double	0	int	0	int	1 st point 1 st signal
1	double	1	double	1	int	1	int	1 st point 2 nd signal
2	double	2	double	2	int	2	int	2 nd point 1 st signal
3	double	3	double	3	int	3	int	2 nd point 2 nd signal
... *			

* to ...2560 MLSL / ...4096 MLWL

8. Appendix

8.1 GetInfo (XML mode)

The following XML data description shows a part of the data returned by the function EthernetScanner_GetInfo (through parameter 2) in the xml mode:

```
<?xml version="1.0" encoding="UTF-8" ?>
<device>
  <general>
    <ordernumber>MLWL273</ordernumber>
    <productversion>1.10</productversion>
    <producer>wenglor sensoric GmbH</producer>
    <description>2D-/3D-profile sensors</description>
    <hardwareversion>
      <general>1.1.0</general>
    </hardwareversion>
    <firmwareversion>
      <general>1.0.9</general>
    </firmwareversion>
    <serialnumber>1000</serialnumber>
    <mac>54:4a:05:0a:02:1d</mac>
    <workingrange_z_start>300.0</workingrange_z_start>
    <measuringrange_z>700.0</measuringrange_z>
    <fieldwidth_x_start>280.0</fieldwidth_x_start>
    <fieldwidth_x_end>830.0</fieldwidth_x_end>
    <pixel_x_max>2048</pixel_x_max>
    <pixel_z_max>2048</pixel_z_max>
    <lininfo>1</lininfo>
    <networkload>0</networkload>
    <networkbuffer>0</networkbuffer>
    <framesize>12928</framesize>
    <encoder_htl>0</encoder_htl>
    <savencoder_htl>0</savencoder_htl>
    <encoder_ttl_rs422>0</encoder_ttl_rs422>
    <savencoder_ttl_rs422>0</savencoder_ttl_rs422>
  </general>
  <usrio>
    <ea1>
      <function>5</function>
      <state>1</state>
    </ea1>
    <ea2>
      <function>5</function>
      <state>1</state>
    </ea2>
    <ea3>
      <function>2</function>
      <state>0</state>
    </ea3>
    <ea4>
```

```

    <function>1</function>
    <state>1</state>
  </ea4>
</usrio>
<ttr_rs422>
  <en_a>
    <state>0</state>
  </en_a>
  <en_b>
    <state>0</state>
  </en_b>
  <en_0>
    <state>0</state>
  </en_0>
</ttr_rs422>
<onoffcounter>417</onoffcounter>
<ontimecounter>67158</ontimecounter>
<temperatur>43</temperatur>
<scannerstatus>95</scannerstatus>
<help></help>
</general>
<settings>
  <userstring>
    <current></current>
    <command>SetUserString</command>
    <help>describe your own laser scanner</help>
  </userstring>
  .
  .
  .
  .
</settings>
</device>

```

8.2 GetInfo (Text mode)

The following data description shows an example of the data returned by the function EthernetScanner_GetInfo (through parameter 2) in the text mode:

```

[general]
sn=6
z_start=65.000
z_range=60.000
x_range_at_start=40.000
x_range_at_end=58.000
widthX=1280
widthZ=1024

```