OXApi

Intro

The OX-SDK is a set of software libraries and tools to support the integration of the OX into third party applications. It has branches for C#, C++ and Python. For detailed installation and compatibility instructions please read the specific sections, C# for C# and Python and C++ for C++ . The section Ox describes the commands individually.

Contents

SDK for C#

Summary

The OX-SDK is a set of software libraries and tools to support the integration of the OX into third party applications.
The SDK is based on the .NET Library 'OXApi.dll' which can be easily integrated into own C# or VB programs or standard software tools like Matlab or TestStand.
In addition there is a python wrapper (oxapi.py) available for the 'OXApi.dll' to use all their features from a python environment.

The OX SDK consists of the following parts:

- A .NET assembly (OXApi.dll) which provides access to all OX configuration and measurement possibilities.
- A python wrapper which allows the use of OxAPI.dll in a python environment.
- A set of examples in C# and Python which demonstrate the OXApi usage.

Folder structure:
\API\OXApi.dll   The OX SDK main assembly.
\API\oxapi.py   Python wrapper for OXApi.dll.
\API\Newtonsoft.Json.dll   Third party Json parser (MIT license, see newtonsoft.json.txt).
\API\websocket-sharp.dll   Third party Websocket library (MIT license, see websocket-sharp.txt).

\OxApiExamples   A C# project which demonstrates the usage of the OXApi.
\OxPythonExamples   Python examples which demonstrate the usage of the OXApi python wrapper.

Notes:

OXApi:
- The OXApi.dll requires Microsoft .NET Framework 4.6.1 https://www.microsoft.com/de-ch/download/details.aspx?id=49982
- Only one configuration connection can be established at one time, so if the Webinterface is active, the OXApi will not be able to connect.

Python:
- You have to add the path of the OXApi.dll and oxapi.py to your python path (e.g. sys.path.append(r"C:\Program Files\Baumer\OXSDK\API").
- In order to use the wrapper, pythonnet (>= 2.4) has to be installed in your python environment. (https://pypi.org/project/pythonnet/)

UDP Streaming:
- To use UDP Streaming, the desired streams should be activated by the OXApi or the Webinterface.
- Ensure that the streaming target IP-Address is configured correctly.
- If more than one OX should stream to the same computer, different ports should be used.
- Keep in mind that the windows firewall may block an application from opening an UDP port.

Version history:

- V1.0.0 First release.
- V1.0.2 Changed measurement data layout, this is the minimum SDK version for OX firmware version V1-0-7 or greater.

SDK for C++

Summary

The OX-CPP-SDK is a set of software libraries and tools to support the integration of the OX into third party applications.
The SDK is based on the C++ Library “libOXApi.so" which can be easily integrated into own C++ programs.

The OX C++ SDK consists of the following parts:

- A C++ shared library (libOXApi.so) which provides access to all OX configuration and measurement possibilities.
- The C++ Header files to use the library.
- A example in C++ which demonstrate the OXApi usage.

Linux x86 - LibOxApi_XX-XX-XX.tar.gz:

Folder structure:

/usr/lib/libOXApi.so	The OX C++ SDK main shared library.
/usr/include	The OX C++ SDK header files.
/example	A C++ cmake example which demonstrates the usage of the OXApi.

Notes:

OXApi:
 - libOXApi.so requires Ubuntu 18.04 LTS with Boost 1.70
 - Only one configuration connection can be established at one time, so if the Webinterface is active, the OXApi will not be able to connect.

Setup Development System:
 - Get Ubuntu 18.04 LTS (AMD64 desktop image) from https://releases.ubuntu.com/18.04/
   Note: If you only get a black screen while installation, press ESC while Grub menu and then F6 and mark "nomodeset". The start install.
 - Setup Network to get access to the internet
 - Run following command in terminal:
```
sudo apt update
sudo apt install build-essential cmake
```
 - Install / Upgrade libboost 1.70
```
sudo add-apt-repository ppa:mhier/libboost-latest
sudo apt update
sudo apt install libboost1.70-dev
```

Use SDK example:
 - The SDK files are copied to ~/OxApi/
 - Create build directory and run build
```
mkdir ~/OxApi/build
cd ~/OxApi/build
cmake ../example
cmake --build .
```
 - Run example
```
./oxapiexamples
```

Windows Win32 and x64 - LibOxApi_XX-XX-XX.zip:

Folder structure:

/libs	The OX C++ SDK main shared library.
/include/OXApi	The OX C++ SDK header files.
/example	A C++ cmake example which demonstrates the usage of the OXApi.

Notes:

OXApi:
 - OXApi requires Visual Studio 2017 or higher
 - Only one configuration connection can be established at one time, so if the Webinterface is active, the OXApi will not be able to connect.

UDP Streaming:
 - To use UDP Streaming, the desired streams should be activated by the OXApi or the Webinterface.
 - Ensure that the streaming target IP-Address is configured correctly.
 - If more than one OX should stream to the same computer, different ports should be used.
 - Keep in mind that the firewall may block an application from opening an UDP port.

Version history:

- V1.0.0 First release.
- V1.1.0 Bugfix udp streaming, rework example

- V2.0.0 Small API changes, bugfixes
- V2.0.1 bugfixes (udp streamer)

Ox `type`

Namespace

Baumer.OXApi

Summary

This class provides several APIs to get profile, measurement and configuration data from the sensor as well as set the sensor configurations.

TimeoutMs `property`

Summary

Gets or sets the timeout for websocket requests in ms.

ConfigureActiveUdpStreams(streamIds) `method`

Summary

This API is used to enable/disable the UDP streams.
Everytime the function is called, only the UDP streams that are passed to the function are enabled and all other streams are disabled.
Send an empty array to disable all UDP streams. You can get the current status of the UDP streams using GetActiveUdpStreams API.

```
            Usage:
            uint[] ids = { 0, 1 };
            OX.ConfigureActiveUdpStreams(ids); //Stream with Id 0 and 1 will be
enabled and other streams will be diabled.
```

Parameters

Name	Type	Description
streamIds	System.UInt32[]	An array containing all UDP stream ids to be enabled.

ConfigureExposureTime(exposureTime) `method`

Summary

Sets the laser exposure time(typically in µs). The exposure time should be within the limits provided by GetExposureTimeLimits API.
You can read the current exposure time from the sensor using GetExposureTime API and the Exposure time units from GetExposureTimeResolution API.

```
            Usage:
            OX.ConfigureExposureTime(1000); //Sets the exposure time to 1000µs.
```

Parameters

Name	Type	Description
exposureTime	System.UInt32	Exposure time to be set(typically in μs).

ConfigureFieldOfView(limitLeft,limitRight,offset,height) `method`

Summary

Configures the sensor field of view (z-axis measured in height). Current field of view can be obtained from GetFieldOfView API.
The unit and precesion of the x and z values of the field of view can be obtained from GetFieldOfViewInfo API.
The min and max limits for x and z values of the field of view can be obtained from GetFieldOfViewLimits API.

```
Usage:
//To configure the field of view to -36mm to 25mm from left to right
//and 40mm to 10mm from top to bottom in height mode, use:
OX.ConfigureFieldOfView(-36.0, 25.0, 10.0, 30.0);
```

Parameters

Name	Type	Description
limitLeft	System.Double	The left limit.
limitRight	System.Double	The right limit.
offset	System.Double	The offset.
height	System.Double	The height.

ConfigureFieldOfView(fieldOfView) `method`

Summary

Configures the field of view (z-axis measured in height). This API is another varient of ConfigureFieldOfView.

Parameters

Name	Type	Description
fieldOfView	Baumer.OXApi.Types.FieldOfView	An object containing the field of view configuration.

ConfigureFieldOfViewDistance(limitLeft,limitRight,near,far) `method`

Summary

Configures the field of view (z-axis measured in distance from the sensor). Current field of view can be obtained from GetFieldOfViewDistance API.

Parameters

Name	Type	Description
limitLeft	System.Double	The left limit.
limitRight	System.Double	The right limit.
near	System.Double	The near distance.
far	System.Double	The far distance.

ConfigureLaserPower(factor) `method`

Summary

Configures the laser power to one of the predifined values.
Use GetLaserPowerLimits API to get the list of predefined laser power values.
Use GetLaserPower to get the current laser power.

Parameters

Name	Type	Description
factor	System.Double	The laser power factor to be set.

ConfigureNetwork(useDHCP,staticIP,subnetMask,gateway) `method`

Summary

Configures the network settings of an Ox sensor. ConfigureNetwork is an another varient of this API.
Use GetNetworkConfiguration API to read the current network configuration from the sensor.

```
            Usage:
            OX.ConfigureNetwork(false, "192.168.0.250", "255.255.255.0",
"192.168.0.1")
```

Parameters

Name	Type	Description
useDHCP	System.Boolean	Enables or disables DHCP.
staticIP	System.String	The static ip address to be used in case of failure of DHCP. E.g. 192.168.0.250
subnetMask	System.String	The subnet mask of the network. E.g. 255.255.255.0
gateway	System.String	The gateway address of the network. E.g. 192.168.0.1

ConfigureNetwork(configuration) `method`

Summary

Configures the network settings of an Ox sensor. ConfigureNetwork is an another varient of this API.
Use GetNetworkConfiguration API to read the current network configuration from the sensor.

Parameters

Name	Type	Description
configuration	Baumer.OXApi.Types.NetworkConfiguration	A NetworkConfiguration object containing DHCP state, static ip address, subnet mask, and gateway address.

ConfigureProcessInterfaces(enableModbus,enableOpcUa,enableUdpStreaming,udpDestinationIp,udpDestinationPort,realtimeProtocolId,ioLinkProcessDataLayout) `method`

Summary

Configures the sensors process interfaces.
Current state of the process interfaces can be obtained using GetProcessInterfaces API.

Parameters

Name	Type	Description
enableModbus	System.Boolean	Enables or disables Modbus TCP server.
enableOpcUa	System.Boolean	Enables or disables OPC UA server.
enableUdpStreaming	System.Boolean	Enables or disables UDP streaming.
udpDestinationIp	System.String	Destination ip address for UDP streaming.
udpDestinationPort	System.UInt32	Destination port for UDP streaming.

Name	Type	Description
realtimeProtocolId	System.UInt32	Id of the realtime protocol. IDs can be obtained using GetProcessInterfacesInfo API.
ioLinkProcessDataLayout	System.UInt32	Id of the IO-Link process data layout. IDs can be obtained using GetProcessInterfacesInfo API.

ConfigureProfileAlgorithm(algorithmID) `method`

Summary

Configures the algorithm used for profile calculation. List of all avaliable algorithms can be obtained from GetProfileAlgorithms API.

Parameters

Name	Type	Description
algorithmID	System.UInt32	The id of the algorithm to be used for profile calculation.

ConfigureProfileAlgorithmParameters(algorithmId,minPeakHeight,thresholdValue,thresholdType,minPeakWidth) `method`

Summary

Configures the parameters for a specific profile computation algorithm.
The limits and units of the parameters can be obtained from GetProfileAlgorithmParamsLimits and GetProfileAlgorithmParamsInfo APIs respectively.
Get parameters of an algorithm using GetProfileAlgorithmParameters API.

```
Usage:
OX.ConfigureProfileAlgorithmParameters(0, 20, 12, 1, 5);
```

Parameters

Name	Type	Description
algorithmId	System.UInt32	The id of the algorithm to configure.
minPeakHeight	System.UInt32	Minimum peak height.
thresholdValue	System.UInt32	Threshold value.
thresholdType	System.UInt32	Type of the threshold.
minPeakWidth	System.UInt32	Minimum peak width.

ConfigureProfileFilter(movingAverageEnabled,movingAverageLength) `method`

Summary

Enables/Disables and sets the length of the moving average filter which is used to filter the profile points.
The length should be between the min-max limits returned by GetProfileFilterLimits API.
You can get the current filter status from GetProfileFilter API.

```
Usage:
OX.ConfigureProfileFilter(true, 5); //Enables the filter and sets the filter
length to 5.
```

Parameters

Name	Type	Description
movingAverageEnabled	System.Boolean	Enables or disables the profile filter.
movingAverageLength	System.UInt32	Length of the moving average filter.

ConfigureProfileFilter(profileFilter) `method`

Summary

Enables/Disables and sets the length of the moving average filter which is used to filter the profile points.
Another varient of ConfigureProfileFilter API.

Parameters

Name	Type	Description
profileFilter	Baumer.OXApi.Types.ProfileFilter	An object containing the profile filter settings.

ConfigureResampling(enabled,gridValue) `method`

Summary

Enables/Disables rasterization of the profile and sets the horizontal distance between the profile points to the given value.
The resampling grid value should be between allowed min - max values retuned by the GetResamplingInfo API.
The resampling active status can be queried using IsResamplingEnabled API and current grid value can be obtained using GetResamplingGridValue API.

Parameters

Name	Type	Description
enabled	System.Boolean	Enables or disabled the profile resampling.
gridValue	System.Double	The grid value used for the profile resampling.

ConfigureResolution(xResolution,zResolution) `method`

Summary

Configures the x and z resolution.
The current resolution and available resolution values can be obtained from GetResolution and GetResolutionInfo APIs respectively.

```
Usage:
OX.ConfigureResolution(2, 4); //Sets the x resolution to 1/2 and  z resolution to
1/4
```

Parameters

Name	Type	Description
xResolution	System.UInt32	The x resolution value to be set.
zResolution	System.UInt32	The z resolution value to be set.

ConfigureStartupSetup(storageNumber) `method`

Summary

Sets the startup parameter setup to given preset storage number.
Get the current startup setup number using GetStartupSetup API.

Parameters

Name	Type	Description
storageNumber	System.UInt32	The storage number.

ConfigureTimeServer(useNTP,timeServers) `method`

Summary

Enables or disables the time synchronization. Sets the IP adresses of the NTP servers.
Current configurations can be read from the sensor using GetTimeServerConfiguration API.

```
Usage:
string [] TimeServers ={"192.168.10.1","192.168.10.2"};
OX.ConfigureTimeServer(true, TimeServers); //Enables the time servers and sets
the server ip addresses
```

Parameters

Name	Type	Description
useNTP	System.Boolean	Enable or disable time synchronization.
timeServers	System.String[]	Array of strings containing IP adresses of the NTP servers.

ConfigureTrigger(triggerMode,modeOption,triggerTime,encoderSteps) `method`

Summary

Configures the trigger to a given trigger mode and option. The names and ids of the trigger modes supported by the sensor can be obtained from GetTriggerInfo API.
The min-max limits for the trigger time and encoder steps can be obtained from GetTriggerLimits API.
Use GetTrigger API to obtain current trigger settings.

```
            Usage:
            OX.ConfigureTrigger(3, 0, 0, 0); //Software trigger mode id: 3
            OX.Trigger(numberOfTriggers); //numberOfTriggers: Number of times the
sensor triggers in a free run mode.
```

Parameters

Name	Type	Description
triggerMode	System.UInt32	Id of the trigger mode to be set.
modeOption	System.UInt32	Id of the trigger mode option to be set.
triggerTime	System.UInt32	Trigger time to be used in fixedTime trigger mode.
encoderSteps	System.UInt32	Encoder steps to be used in encoder trigger mode.

ConfigureUdpStreams(streamIds) `method`

Summary

This API is used to enable/disable the UDP streams.
Everytime the function is called, only the UDP streams that are passed to the function are enabled and all other streams are disabled.
Send an empty array to disable all UDP streams. You can get the current status of the UDP streams using GetActiveUdpStreams API.

```
            Usage:
            uint[] ids = { 0, 1 };
            OX.ConfigureUdpStreams(ids); //Stream with Id 0 and 1 will be enabled
and other streams will be diabled.
```

Parameters

Name	Type	Description
streamIds	System.UInt32[]	An array containing all UDP stream ids to be enabled.

ConfigureZAxis(zAxisId) `method`

Summary

Configures the z-axis(e.g. distance or height). Use GetAxesInfo to obtain all supported z-axes.

Parameters

Name	Type	Description
zAxisId	System.UInt32	Id of the z-axis to be set.

Connect() `method`

Summary

Establishes a connection to the Ox sensor.
Sensor can be configured or measurement data can be read only after establishing a connection to the sensor.

Create(ipAddress,streamingPort) `method`

Summary

This API creates an object of the Ox sensor and returns it.
The object created using this API can be used to connect and interact with the sensor.
No connection will be established by this method. Use Connect() API on the Ox sensor object to connect.

Parameters

Name	Type	Description
ipAddress	System.String	The ip address of the sensor to connect, e.g. 192.168.0.250
streamingPort	System.UInt32	UDP port used for streaming. Default port number is 1234

CreateStream() `method`

Summary

Returns a singleton instance of the streaming client to access profiles and measurements provided by UDP steaming service.

Disconnect() `method`

Summary

Closes the connection to the sensor.

EnableLongSession() `method`

Summary

Enables long session timeouts.

GetActiveSetup() `method`

Summary

Returns the active setup number and its saved state.

GetActiveUdpStreams() `method`

Summary

Returns the IDs of all activated UDP streams.
The streams associated with the Ids can be obtained using GetUdpStreamingInfo API.
UDP streams can be enabled/disabled using ConfigureActiveUdpStreams API.

GetAxesInfo() `method`

Summary

Returns the ids and names of all supported z-axes (e.g., Distance, Height).
To change to different Z axis mode, use ConfigureZAxis API.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Z Axis Id	Z Axis Name
1	height
0	distance

GetExposureTime() `method`

Summary

Returns the current exposure time of the sensor(typically in µs).
One can get the exposure time units from GetExposureTimeResolution API.
Use ConfigureExposureTime to configure the exposure time.

GetExposureTimeLimits() `method`

Summary

Returns the minimum and maximum exposure time values(typically in µs) of the OX sensor.
Use ConfigureExposureTime to configure the exposure time.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Max Exposure Time	Min Exposure Time
3000	100

GetExposureTimeResolution() `method`

Summary

Returns the resolution of the exposure time used in ConfigureExposureTime, GetExposureTime and GetExposureTimeLimits APIs.

```
          Usage:
          String TimeResolutionUnit = OX.GetExposureTimeResolution(); //Returns
µs for OX200 sensor.
```

GetFieldOfView() `method`

Summary

Returns the actual field of view settings(z-axis measured in height) from the sensor containing left limit, right limit, height and offset.
The field of view can be configured using ConfigureFieldOfView API.

GetFieldOfViewDistance() `method`

Summary

Returns the actual field of view settings(z-axis measured in distance from the sensor) from the sensor.
It contains left limit, right limit, near distance and far distance.
One can configure field of view in distance mode using ConfigureFieldOfViewDistance API.

GetFieldOfViewInfo() `method`

Summary

Returns the x and z axes units and x and z axes precision for the field of view.
One can configure the Field of View using ConfigureFieldOfView API.

GetFieldOfViewLimits() `method`

Summary

Returns the actual field of view limits from the sensor. The returned object contains left limit, right limit, min width, min height and max height of the field of view.
Use GetFieldOfViewInfo to obtain precison as well as units of x and z axes limits.
One can configure the Field of View using ConfigureFieldOfView API.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

FOV Max Height(Z)	50
FOV Min Height(Z)	5
FOV Z Precision	10
FOV Z Unit	mm
FOV Max X	36
FOV Min X	-36
FOV Min Width(Delta X)	2
FOV X Precision	10
FOV X Unit	mm

GetImage() `method`

Summary

Returns the last measured raw image from the sensor containing a pixel array (integers 0-255), ROI height, ROI width, Row and Column binning and offset.

```
Usage:
var image = OX.GetImage();
var pixels = image.Pixels; //Array of grayscale pixels
var roiHeight = image.RoiHeight;
var roiWidth = image.RoiWidth;
var rowBinning = image.RowBinning;
var rowOffset = image.RowOffset;
```

GetImageInfo() `method`

Summary

Returns information about the image sensor height, width and max ROI pixel count.

GetIntensityProfile() `method`

Summary

Returns the last measured intensity profile from the sensor.
This API delivers intensity information of the profile points in addition to the information returned by GetProfile API.

```
Usage:
var profile = OX.GetIntensityProfile();
var xdata = ( profile.X[i] + profile.XStart ) / profile.Precision;
var zdata = profile.Z[i] / profile.Precision;
var intensity = profile.I[i];
var profile_info = OX.GetProfileInfo();
var xunits = profile_info.XUnits;
var zunits = profile_info.ZUnits;
```

GetLaserPower() `method`

Summary

Returns the current laser power value.
Use ConfigureLaserPower to configure laser power.

GetLaserPowerInfo() `method`

Summary

Returns laser power factor precision and unit.

GetLaserPowerLimits() `method`

Summary

Returns all predefined laser power factors along with the min and max power factor values.
Use ConfigureLaserPower to change the laser power factor.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Max Power	Min Power	Predefined Factors
3	0,5	3, 2, 1, 0,5,

GetMeasurement() `method`

Summary

Returns an array of latest measurement values, digital output values, encoder value, alarm status, measurement rate, quality id and timestamp.
Use GetMeasurementInfo API to know more about QualityId and units of TimeStamp.
Use GetMeasurementValuesInfo API to know more about tool name, id, mode and other metadata associated with the measurement
results returned by this API.

```
Usage:
var MeasValues = OX.GetMeasurement();
var MeasResults = MeasValues.Values //Double array containing values of enabled
tools
var DigitalOuts = MeasValues.DigitalOuts; // Boolian array containing digital
outputs
var Alarm = MeasValues.Alarm; //Alarm status
var TimeStamp = MeasValues.TimeStamp;
var QualityId = MeasValues.QualityId;
var MeasurementRate = MeasValues.MeasurementRate;
var ConfigMode = MeasValues.ConfigMode;
var EncoderValue = MeasValues.EncoderValue;
```

GetMeasurementInfo() `method`

Summary

Returns the information about measurements such as time stamp units, measurement rate units, measurement rate precision and all quality value names and ids.

```
Usage:
var MeasInfo = OX.GetMeasurementInfo();
var QualityValues = MeasInfo.QualityValues; //Array of Id and Name pair.
var TimeStampUnits = MeasInfo.TimeStampUnits; //Array of time stamp units.
var MeasRateUnit = MeasInfo.MeasurementRateUnit;
var MeasRatePrecision = MeasInfo.MeasurementRatePrecision;
```

GetMeasurementValuesInfo() `method`

Summary

Returns the information about the measurement values such as Tool ID, Tool Mode, Tool Name, Unit, Precision and Limits.

```
Usage:
var MeasValuesInfo = OX.GetMeasurementValuesInfo();
var MeasTypes = MeasValuesInfo.MeasurementTypes; //Contains information about all
active measurements.
//Use foreach loop to iterate over all active measurements.
var AnActiveMeas = MeasTypes[0];
var MeasName = AnActiveMeas.Name;
var MeasMode = AnActiveMeas.Mode;
var MeasTool = AnActiveMeas.Tool;
var MeasToolID = AnActiveMeas.ToolId;
var MeasUnit = AnActiveMeas.Unit;
var MeasPrecision = AnActiveMeas.Precision;
var MeasMinValue = AnActiveMeas.Minimum;
var MeasMaxValue = AnActiveMeas.Maximum;
```

GetNetworkConfiguration() `method`

Summary

Returns the current network configuration of the Ox sensor.
The returned object contains DHCP state, static ip address, subnet mask, gateway address and mac address.
You can set the network configuration using ConfigureNetwork API.

GetNumberOfSetups() `method`

Summary

Returns the total number of sensor configuration preset storages available in the sensor.

GetNumberOfTimeServers() `method`

Summary

Returns the maximum number of time servers supported by the sensor for time synchronization.

GetParameterSetup(storageNumber) `method`

Summary

Returns the sensor configuration data saved in a given preset number as a json string.

Parameters

Name	Type	Description
storageNumber	System.UInt32	The storage number.

GetProcessInterfaces() `method`

Summary

Returns the current state of all sensor process interfaces like modbus and OPCUA state,
UDP streaming status, destination UDP port and ip address. You can configure process interfaces using ConfigureProcessInterfaces

GetProcessInterfacesInfo() `method`

Summary

Returns the ids and names of the available real time protocols(including the id to disable real time communication).

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Process Id	Process Name
0	Disabled
1	Profinet
2	EtherNetIP

GetProfile() `method`

Summary

Returns the last measured profile from the sensor.
In addition to X and Z coordinate values of the profile points, the returned object also contains profile length, quality id, xstart, precision, and timestamp.

```
            Usage:
            var profile = OX.GetProfile();
            var xdata = ( profile.X[i] + profile.XStart ) / profile.Precision;
            var zdata = profile.Z[i] / profile.Precision;
            var profile_info = OX.GetProfileInfo();
            var xunits = profile_info.XUnits;
            var zunits = profile_info.ZUnits;
```

GetProfileAlgorithm() `method`

Summary

Returns an id of the algorithm used for profile calculation.
Use ConfigureProfileAlgorithm to change the profile algorithm.

GetProfileAlgorithmParameters(algorithmId) `method`

Summary

Returns the current configuration parameter values of a specific profile computation algorithm.
Configure profile algorithm parameters using ConfigureProfileAlgorithmParameters API.

Parameters

Name	Type	Description
algorithmId	System.UInt32	The id of the algorithm.

GetProfileAlgorithmParamsInfo() `method`

Summary

Returns the units of all profile algorithm parameters.

GetProfileAlgorithmParamsLimits(algorithmId) `method`

Summary

Returns all parameters' min-max limits for a specific profile computation algorithm.
Configure Profile Algorithm Params using ConfigureProfileAlgorithmParameters API.

```
Usage:
var ParamLimits = OX.GetProfileAlgorithmParamsLimits(algorithmId);
var Limits = ParamLimits.Limits;
var MinPeakHeightParamMin = Limits.MinPeakHeight.Minimum;
var MinPeakHeightParamMax = Limits.MinPeakHeight.Maximum;
var MinPeakWidthParamMin = Limits.MinPeakWidth.Minimum;
var MinPeakWidthParamMax = Limits.MinPeakWidth.Maximum;
var ThresholdValueMin = Limits.ThresholdValue.Minimum;
var ThresholdValueMax = Limits.ThresholdValue.Maximum;
var ThresholdTypes = Limits.ThresholdTypes; //Returns array containing names and
ids of supported types.
```

Parameters

Name	Type	Description
algorithmId	System.UInt32	The profile computation algorithm id.

GetProfileAlgorithms() `method`

Summary

Returns an object containing a list of names and ids of the profile algorithms supported by the sensor.
Use ConfigureProfileAlgorithm to use a different profile algorithm.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Algorithm Id	Algorithm Name
0	max
1	upper
2	lower

GetProfileFilter() `method`

Summary

Returns a profile filter object containing the active status and length of the moving average filter.
The filter can be configured using ConfigureProfileFilter API.

GetProfileFilterLimits() `method`

Summary

Returns an object containing minimum and maximum allowed profle filter length.
Use ConfigureProfileFilter to configure the profile filter.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Max Filter Length	Min Filter Length
15	3

GetProfileInfo() `method`

Summary

Returns information such as max length of the profile(in number of points), x and z axes units of the profile.

```
Usage:
var ProfileInformation = OX.GetProfileInfo();
uint MaxLength = ProfileInformation.MaxLength;
String XUnit = ProfileInformation.XUnit;
String ZUnit = ProfileInformation.ZUnit;
```

GetResamplingGridValue() `method`

Summary

Returns the current resampling grid value and active status of the profile resampling.
Use ConfigureResampling to enable/disable resampling and change the resampling grid value.
The unit and precision of the returned grid value can be obtained using GetResamplingInfo API.

GetResamplingInfo() `method`

Summary

Returns the precision, unit, allowed max and min values for the resampling grid value.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Max Grid Value	2
Min Grid Value	0,2

Grid Precision	10
Grid Value Unit	mm

GetResolution() `method`

Summary

Returns the current x and z resolution values. If the returned value is n then it implies that the current resolution is 1/n times that of max resolution.
One can configure resolution using ConfigureResolution API.

```
Usage:
var resolution = OX.GetResolution();
uint XResolution = resolution.XResolution;
uint ZResolution = resolution.ZResolution;
```

GetResolutionInfo() `method`

Summary

Returns the list of all available x and z resolutions.
One can configure resolution using ConfigureResolution

E.g.: Information below is fetched from OX200 sensor with the help of this API.

X Resolution	Z Resolution
1, 2, 4,	1, 2, 4,

GetSecondaryData() `method`

Summary

Returns secondary data such as Temperature, Boot-Up counts, Up Time, Operating Voltage and Operation Time.

```
Usage:
var SecondaryData = OX.GetSecondaryData();
uint BootUpCounter = SecondaryData.BootUpCounter;
uint OperatingTime = SecondaryData.OperatingTime;
uint UpTime = SecondaryData.UpTime;
uint Temperature = SecondaryData.Temperature;
uint OperatingVoltage = SecondaryData.OperatingVoltage;
```

GetSensorInfo() `method`

Summary

Returns sensor information such as type of the sensor, serial number, vendor name and the current firmware versions.

```
Usage:
var SensorInfo = OX.GetSensorInfo();
String SensorType = SensorInfo.Type;
String SerialNumber = SensorInfo.SerialNumber;
String VendorName = SensorInfo.VendorName;
String AggregateVersion = SensorInfo.AggregateVersion;
String SoftwareVersion = SensorInfo.SoftwareVersion;
```

GetStartupSetup() `method`

Summary

Returns the parameter setup number which is loaded during startup of the sensor.
Change the startup setup number using ConfigureStartupSetup API.

GetTimeServerConfiguration() `method`

Summary

Returns the current NTP time server configuration containing NTP state and time server IP adresses.
The state or adresses of the time servers can be set using ConfigureTimeServer API.

GetTrigger() `method`

Summary

Returns the current trigger configuration containing encode steps, trigger mode, trigger mode option and trigger time.
Trigger mode can be configured using ConfigureTrigger API.

GetTriggerInfo() `method`

Summary

Returns the information about all possible trigger configurations (Names and Ids of all trigger modes and supported trigger options).
To configure the trigger, use ConfigureTrigger API.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Mode Id	Mode Name	Supported Options
0	freerun	1
1	extSingleShot	4
2	fixedTime	1
4	encoder	1
3	software	

Option Id	Option Name
0	ignoreSyncIn
1	runWhileSyncInLow
2	runWhileSyncInHigh
3	risingEdge
4	fallingEdge
5	bothEdges

GetTriggerLimits() `method`

Summary

Returns the min - max limits for trigger time and encoder steps (used in fixed time trigger and encoder trigger modes).

E.g.: Information below is fetched from OX200 sensor with the help of this API.

Property	Value
Max Encoder Steps	65536
Min Encoder Steps	3
Max Interval Time	4000000
Min Interval Time	500

GetUdpStreamingInfo() `method`

Summary

Returns all available UDP stream names and Ids supported by the sensor.
Use GetActiveUdpStreams API to obtain IDs of all active UDP streams.
Use ConfigureActiveUdpStreams to enable/disable UDP streams.

E.g.: Information below is fetched from OX200 sensor with the help of this API.

UDP Stream Id	UDP Stream Name
0	zProfile
1	intensityProfile
2	allMeasurementValues

GetZAxis() `method`

Summary

Returns an id of the current z-axis. Use GetAxesInfo to check the corresponding name.
To change to different z-axis mode, use ConfigureZAxis API.

IsProfileFilterEnabled() `method`

Summary

Returns active status of the profile filter.

```
Usage:
if(!OX.IsProfileFilterEnabled())
{
    OX.ConfigureProfileFilter(true, 5);
}
```

IsResamplingEnabled() `method`

Summary

Returns the current status of the profile resampling.

LoadParameterSetup(storageNumber) `method`

Summary

Loads the sensor configuration from a given parameter setup.

Parameters

Name	Type	Description
storageNumber	System.UInt32	The storage number.

Login(role,password) `method`

Summary

This API is used to enter into admin mode in order to change the sensor configuration.

Parameters

Name	Type	Description
role	System.String	The requested role. Default value is "admin".
password	System.String	The password for the requested role. Default value is "".

Logout() `method`

Summary

This API is used to leave admin mode.

ReadAllSettings() `method`

Summary

Reads all settings from the sensor.

Returns

An encoded string containing all settings.

ReadSetting(storageNumber) `method`

Summary

Reads a setting from the sensor.

Returns

ConfigureProfileAlgorithmParameters
An encoded string containing one setting.

Parameters

Name	Type	Description
storageNumber	System.UInt32	The storage number. "0" specifies device configuration parameters.

ResetAllSettings() `method`

Summary

Clears all saved parameter setups.

ResetSettings() `method`

Summary

Clears a saved parameter setup.

StoreParameterSetup(storageNumber) `method`

Summary

Stores the actual sensor configuration to the desired storage.

Parameters

Name	Type	Description
storageNumber	System.UInt32	The storage number.

Trigger(count) `method`

Summary

Generates a software trigger. The profiles are acquired in free running mode.
The trigger should be configured to software mode using ConfigureTrigger before using this API

Parameters

Name	Type	Description
count	System.UInt32	Number of trigger events.

WriteAllSettings(settings) `method`

Summary

Writes all settings to the sensor.

Parameters

Name	Type	Description
settings	System.String	The settings to write as encoded string (e.g. read by ReadAllSettings).

WriteSetting(setting,storageNumber) `method`

Summary

Writes a settings to the sensor.

Parameters

Name	Type	Description
setting	System.String	The setting to write as encoded string (e.g. read by ReadSetting).
storageNumber	System.UInt32	The storage number. "0" specifies device configuration parameters.

OxStream `type`

Namespace

Baumer.OXApi.UdpStreaming

Summary

This class is used to read data from active UPD streams using various class APIs.

ErrorOccured `property`

Summary

Returns true if at least one error occured.

FullQueueHandling `property`

Summary

Defines the behavior if the queue is full and new data arrives.
This enum can be set to 'DropOldest' or 'IgnoreNew' value.

MeasurementAvailable `property`

Summary

Returns true if at least one measurement is available.

MeasurementCount `property`

Summary

The number of queued measurements.

ProfileAvailable `property`

Summary

Returns true if at least one profile is available.

ProfileCount `property`

Summary

The number of queued profiles.

QueueSize `property`

Summary

The length of all queues. The default value is 10000.

ReceiveBufferSize `property`

Summary

The size of the receive buffer in bytes
If packets are lost increase this buffer size, the size depends
on the used system and its load

ClearMeasurementQueue() `method`

Summary

Clears the measurement queue.

ClearProfileQueue() `method`

Summary

Clears the profile queue.

Close() `method`

Summary

Closes the stream.

Dispose() `method`

Summary

Disposes the instance and releases all resources.

ReadError() `method`

Summary

Returns one error from the queue (the error will be removed from the queue).
Throws an exception if the queue is empty.

ReadMeasurement() `method`

Summary

Returns oldest measurement from the queue (the measurement will be removed from the queue).
Throws an exception if the queue is empty.

ReadProfile() `method`

Summary

Returns the oldest profile from the queue (the profile will be removed from the queue).
Throws an exception if the queue is empty.

Start() `method`

Summary

Starts reading data from the streaming sensor.

Stop() `method`

Summary

Stops reading data from the streaming sensor.