



VsxProtocolDriver

3.4.3+ged90229

Driver package (.NET) to communicate with P+F SmartRunner devices via VSX protocol

1 Introduction	1
1.1 Supported devices	1
1.2 Requirements	1
2 Usage with dotnet (C#)	1
3 Examples	2
4 Device parameter	2
5 Changelog	3
6 Namespace Index	5
6.1 Namespace List	5
7 Hierarchical Index	5
7.1 Class Hierarchy	5
8 Class Index	6
8.1 Class List	6
9 Namespace Documentation	8
9.1 PF Namespace Reference	8
9.2 PF.VsxProtocolDriver Namespace Reference	8
9.3 PF.VsxProtocolDriver.Types Namespace Reference	9
9.3.1 Enumeration Type Documentation	11
10 Class Documentation	12
10.1 PF.VsxProtocolDriver.Types.ICoordinate Interface Reference	12
10.1.1 Detailed Description	12
10.1.2 Property Documentation	12
10.2 PF.VsxProtocolDriver.Types.IError Interface Reference	13
10.2.1 Detailed Description	14
10.2.2 Property Documentation	14
10.3 PF.VsxProtocolDriver.Types.IFirmwareState Interface Reference	14
10.3.1 Detailed Description	14
10.3.2 Property Documentation	14
10.4 PF.VsxProtocolDriver.Types.IFirmwareVersion Interface Reference	15
10.4.1 Detailed Description	15
10.4.2 Property Documentation	15
10.5 PF.VsxProtocolDriver.Types.IItemTuple Interface Reference	16
10.5.1 Detailed Description	16
10.5.2 Property Documentation	16
10.6 PF.VsxProtocolDriver.Types.ILineMulti Interface Reference	17
10.6.1 Detailed Description	17
10.6.2 Property Documentation	17

10.7 PF.VsxProtocolDriver.Types.ILineSingle Interface Reference	18
10.7.1 Detailed Description	18
10.7.2 Property Documentation	18
10.8 PF.VsxProtocolDriver.Types.IParameter Interface Reference	18
10.8.1 Detailed Description	19
10.8.2 Member Function Documentation	19
10.8.3 Property Documentation	20
10.9 PF.VsxProtocolDriver.Types.IPoint2D Interface Reference	22
10.9.1 Detailed Description	22
10.9.2 Property Documentation	22
10.10 PF.VsxProtocolDriver.Types.IPoint3D Interface Reference	23
10.10.1 Detailed Description	23
10.10.2 Property Documentation	23
10.11 PF.VsxProtocolDriver.Types.IStatusItem Interface Reference	23
10.11.1 Detailed Description	24
10.11.2 Property Documentation	24
10.12 PF.VsxProtocolDriver.Types.IVsxApplicationResultData Interface Reference	25
10.12.1 Detailed Description	26
10.12.2 Property Documentation	26
10.13 PF.VsxProtocolDriver.Types.IVsxCaptureInformation Interface Reference	26
10.13.1 Detailed Description	27
10.13.2 Property Documentation	27
10.14 PF.VsxProtocolDriver.Types.IVsxData Interface Reference	28
10.14.1 Detailed Description	29
10.14.2 Property Documentation	29
10.15 PF.VsxProtocolDriver.Types.IVsxDeviceInformation Interface Reference	29
10.15.1 Detailed Description	30
10.15.2 Property Documentation	30
10.16 PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor Interface Reference	32
10.16.1 Detailed Description	32
10.16.2 Property Documentation	32
10.17 PF.VsxProtocolDriver.Types.IVsxDynamicContainer Interface Reference	33
10.17.1 Detailed Description	34
10.17.2 Member Function Documentation	34
10.17.3 Property Documentation	34
10.18 PF.VsxProtocolDriver.Types.IVsxFile Interface Reference	34
10.18.1 Detailed Description	35
10.18.2 Property Documentation	35
10.19 PF.VsxProtocolDriver.Types.IVsxImage Interface Reference	35
10.19.1 Detailed Description	36
10.19.2 Property Documentation	36
10.20 PF.VsxProtocolDriver.Types.IVsxLineData Interface Reference	38

10.20.1 Detailed Description	39
10.20.2 Property Documentation	39
10.21 PF.VsxProtocolDriver.Types.IVsxLogData Interface Reference	40
10.21.1 Detailed Description	40
10.21.2 Property Documentation	41
10.22 PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation Interface Reference	41
10.22.1 Detailed Description	42
10.22.2 Property Documentation	42
10.23 PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData Interface Reference	45
10.23.1 Detailed Description	45
10.23.2 Property Documentation	45
10.24 PF.VsxProtocolDriver.IVsxProtocolDriver Interface Reference	46
10.24.1 Detailed Description	49
10.24.2 Member Function Documentation	49
10.24.3 Property Documentation	62
10.24.4 Event Documentation	63
10.25 PF.VsxProtocolDriver.IVsxProtocolDriverSync Interface Reference	64
10.25.1 Detailed Description	67
10.25.2 Member Function Documentation	67
10.25.3 Member Data Documentation	72
10.25.4 Property Documentation	80
10.25.5 Event Documentation	82
10.26 PF.VsxProtocolDriver.Types.IVsxResultData Interface Reference	83
10.26.1 Detailed Description	83
10.26.2 Property Documentation	83
10.27 PF.VsxProtocolDriver.Types.IVsxSessionData Interface Reference	83
10.27.1 Detailed Description	84
10.27.2 Property Documentation	84
10.28 PF.VsxProtocolDriver.Types.IVsxTransformation Interface Reference	84
10.28.1 Detailed Description	85
10.28.2 Property Documentation	85
10.29 PF.VsxProtocolDriver.Types.Vsx Class Reference	86
10.29.1 Member Enumeration Documentation	86
10.29.2 Member Function Documentation	89
10.29.3 Member Data Documentation	89
10.30 PF.VsxProtocolDriver.VsxProtocolDriver Class Reference	90
10.30.1 Detailed Description	91
10.30.2 Member Function Documentation	91
10.30.3 Member Data Documentation	93
10.31 PF.VsxProtocolDriver.VsxProtocolDriverSync Class Reference	94
10.31.1 Detailed Description	95
10.31.2 Member Function Documentation	95

10.31.3 Member Data Documentation	97
Index	99

1 Introduction

The driver `VsxProtocolDriver` (`VsxSdk`) provides full access to the input and output data of the sensor. The driver connects to the sensor and handles communication in accordance with the communication protocol. The user can access functions for setting parameters on the sensor, retrieving parameter values from the sensor, and saving and loading entire parameter sets both locally and on the sensor. The user can also receive sensor data like images, 3D-data or lines. Each function also contains an error object from which information can be obtained in the event of an error in the function.

1.1 Supported devices

The official supported devices are the following:

- SmartRunner 3D (Stereo + ToF)
- SmartRunner 2D

1.2 Requirements

The driver is available for multiple architecture

- Windows 64 bit / 32 bit
- Linux AMD64, ARM64, ARM32

The main driver is based on the C# (.NET). There are wrapper for C and Python programming language available.

For usage the Microsoft .NET Runtime 6.0.x framework or higher must be installed (See <https://dotnet.microsoft.com/en-us/download/dotnet>).

Important note: There is also still support for .Net 5.0, but this will probably be dropped in the next version, as this release has reached end of life support by Microsoft.

2 Usage with dotnet (C#)

The driver `VsxProtocolDriver` (`VsxSdk`) facilitates integration in a C#- based programming environment.

The driver is implemented in C# and requires .NET 6.0 or higher.

The functions of the driver can be used synchronously or asynchronously. For this, the required instance must be created using the `Init` function of the respective classes.

- The `VsxProtocolDriver` class provides the asynchronous driver.
- The `VsxProtocolDriverSync` class provides the synchronous interface.

In order to use the SDK, the NuGet must be integrated. This can be done in Visual Studio using the NuGet package manager, for example. The SDK can be found on the product page of the corresponding sensor from Pepperl+Fuchs in the software folder. The NuGet is located in the ZIP file stored in the folder `NET\package\`.

Important note: The following chapters describe the functions and structures of the `VsxProtocolDriver`. When implementing and using it in your own code, only the elements described here from the namespaces mentioned should be used.

3 Examples

In the following the usage of the `VsxProtocolDriver` is shown with a short code example.

The complete examples can be found as a Visual Studio project in the `NET\example\` subfolder. It support the detection of different sensors and show the parametrization and the grabbing of data from the sensor.

```
class Program
{
    // The instance to the async VsxProtocolDriver
    public static VsxProtocolDriver _vsxProtocolDriver;
    static async Task Main()
    {
        //First discover devices via UDP and use the ip of the first device found
        var sensors = await VsxProtocolDriver.UdpDeviceList();
        if (sensors.Succ && sensors.DeviceList.Count > 0)
        {
            // create a new VsxProtocolDriver instance
            var dev = sensors.DeviceList[0];
            Console.WriteLine($"Device found:  {dev.SensorType}({dev.IpAddress})");
            _vsxProtocolDriver = VsxProtocolDriver.InitTcpDevice(sensors.DeviceList[0].IpAddress);
        }
        else //use fix ip address
        {
            // create a new VsxProtocolDriver instance with fix ip address
            _vsxProtocolDriver = VsxProtocolDriver.InitTcpDevice("192.168.2.4");
        }
        // Connect with device
        var connRes = await _vsxProtocolDriver.Connect();
        if (!connRes.Succ)
        {
            Console.WriteLine($"Unable to connect with device:  {connRes.ErrorDesc.Message}");
            return;
        }
        // Get the current device information
        var deviceInformationRes = await _vsxProtocolDriver.GetDeviceInformation();
        if (deviceInformationRes.Succ)
        {
            Console.WriteLine($"Connected with device {deviceInformationRes.CurrentDevice.SensorType}, IP: {deviceInformationRes.CurrentDevice.IpAddress}");
            Console.WriteLine();
        }
        else
        {
            Console.WriteLine($"Unable to receive device information:  {connRes.ErrorDesc.Message}");
            return;
        }
    }
}
```

4 Device parameter

In this chapter some information about the structure of the device parameters shall be given.

The device parameters are organized in two levels. The first level includes one or more configuration groups. Each of these configuration groups in turn contains one or more parameters. To uniquely identify parameters, each configuration has a unique Id. Each parameter also contains an Id that is unique within its configuration.

In order to keep different firmware versions compatible with each other, an additional versioning exists. This comprises on the one hand a settings version, which determines, which configurations are present up-to-date, and a configuration version, which determines which parameters are present at the moment in this configuration. If changes are made to configurations or parameters, the respective version number is increased.

Four arguments are hence required to uniquely define a device parameter:

- *settingsVersion*: Version number, which tells the device which configurations are available
- *configurationId*: Id of the current configuration group

- *configurationVersion*: Version number, which tells the device which parameters are available within the current configuration group and how they are handled
- *parameterId*: Id of the current parameter

In order to know the individual parameters with their ids and versions, files for all supported sensor types and their various firmware versions are stored in a source file in the example subfolder named with `<sensor_name>ParameterIdentifier`. The required informations can be taken from these files.

Example: The value of the following parameter for the Smartrunner 3-D device:

- *settingsVersion*: 2
- *configurationId*: "Base"
- *configurationVersion*: 2
- *parameterId*: "ExposureTime"

can be received using the driver via the function `GetSingleParameterValue(settingsVersion:2, configId:"Base", configVersion:2, parameterId:"ExposureTime")`.

Additional notes:

- if a configuration or parameter does not contain a version attribute, use the default value of "1".
- in addition to the information on version and Id, the xml files also contain further information on the parameters such as name, value range, etc.
- to trigger event parameters these must be set to a value of "1".
- for the Smartrunner 2-D, only a part of the parameters is listed in the corresponding xml file. Only these parameters should be used for parameterization of the device.

5 Changelog

This is the changelog for the .NET implementation (C#) of the VsxProtocolDriver. The C and Python wrapper can have additional informations in their documentation.

V3.4.3

- Fix wrong dependencies for Linux OS

V3.4.2

- Fix wrong dependencies for "unsupported" .net5.0 target framework

V3.4.1

- Nuget support for .net6.0 & .net8.0
- Add "InitializeDevice" helper function (set authentication mode of brand-new devices)
- Minor fixes

V3.4.0

- Minimum requirement .net6.0
- Optimize firmware update for ethernet based sensors
- Speed optimization for ethernet communication
- Minor fixes

V3.3.1

- Enable keepalive for ethernet connections to recognize connection failure of sensor behind network switch

V3.3.0

- Better performance for line data transfer
- Optimized performance for receiving data over ethernet
- Fix possible endless loop (internal), when disconnecting sensor

V3.2.0

- Use VsxProtocolDriver as base for Vision Configurator
 - Integrate last changes from libraries
- Change namespace for "PF.Types" to "PF.VsxProtocolDriver.Internal.Types" to avoid collision with old "Vsx↔Driver" driver.

V3.1.5

- Changed Olr2CaptureInformation to match requirements

V3.1.3

- Allow usage of vsx sensor (UDP response) behind router

V3.1.2

- Add "ApplicationResultData" message

V3.1.0

- Add "SendKeepAlive" function (reply to timeout announcement message)
- Add "IVsxOlr2CaptureInformation" & "IVsxOlr2ModbusData" type support

V3.0.5

- Added possibility to save IVsxDynamicContainer
- Fixed bug that the directory to save to must already exist
- Added Appendix with parameter explanation and parameter files for each supported device

V3.0.0

- Initial release

6 Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

PF	8
PF.VsxProtocolDriver	8
PF.VsxProtocolDriver.Types	9

7 Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

PF.VsxProtocolDriver.Types.ICoordinate	12
IDisposable	
PF.VsxProtocolDriver.IVsxProtocolDriver	46
PF.VsxProtocolDriver.IVsxProtocolDriverSync	64
PF.VsxProtocolDriver.Types.IError	13
PF.VsxProtocolDriver.Types.IFirmwareState	14
PF.VsxProtocolDriver.Types.IFirmwareVersion	15
PF.VsxProtocolDriver.Types.IItemTuple	16
PF.VsxProtocolDriver.Types.ILineMulti	17
PF.VsxProtocolDriver.Types.ILineSingle	18
PF.VsxProtocolDriver.Types.IParameter	18
PF.VsxProtocolDriver.Types.IPoint2D	22
PF.VsxProtocolDriver.Types.IPoint3D	23
PF.VsxProtocolDriver.Types.IStatusItem	23
PF.VsxProtocolDriver.Types.IVsxData	28
PF.VsxProtocolDriver.Types.IVsxApplicationResultData	25
PF.VsxProtocolDriver.Types.IVsxCaptureInformation	26
PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor	32
PF.VsxProtocolDriver.Types.IVsxDynamicContainer	33
PF.VsxProtocolDriver.Types.IVsxFile	34

PF.VsxProtocolDriver.Types.IVsxImage	35
PF.VsxProtocolDriver.Types.IVsxLineData	38
PF.VsxProtocolDriver.Types.IVsxLogData	40
PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation	41
PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData	45
PF.VsxProtocolDriver.Types.IVsxResultData	83
PF.VsxProtocolDriver.Types.IVsxSessionData	83
PF.VsxProtocolDriver.Types.IVsxTransformation	84
PF.VsxProtocolDriver.Types.IVsxDeviceInformation	29
PF.VsxProtocolDriver.Types.Vsx	86
PF.VsxProtocolDriver.VsxProtocolDriver	90
PF.VsxProtocolDriver.VsxProtocolDriverSync	94

8 Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

PF.VsxProtocolDriver.Types.ICoordinate A point of a line with coordinates and some attributes	12
PF.VsxProtocolDriver.Types.IError Error object with detailed description when an error occurs.	13
PF.VsxProtocolDriver.Types.IFirmwareState Detailed status description sent while making a firmware update.	14
PF.VsxProtocolDriver.Types.IFirmwareVersion The current device firmware version.	15
PF.VsxProtocolDriver.Types.IItemTuple An item tuple of id and value.	16
PF.VsxProtocolDriver.Types.ILineMulti A collection of lines received from the device.	17
PF.VsxProtocolDriver.Types.ILineSingle One single line received from the device.	18
PF.VsxProtocolDriver.Types.IParameter Represents a single parameter of the device. Some properties are optional depending on the Type of the parameter	18
PF.VsxProtocolDriver.Types.IPoint2D A 2D point of a line.	22

PF.VsxProtocolDriver.Types.IPoint3D	
A 3D point of a line.	23
PF.VsxProtocolDriver.Types.IStatusItem	
A status item received from the device.	23
PF.VsxProtocolDriver.Types.IVsxApplicationResultData	
Application result data received from device.	25
PF.VsxProtocolDriver.Types.IVsxCaptureInformation	
CaptureInformation contains information identifying and describing the captured image. The timestamp reflects a monotonic clock, without further synchronization timestamps are only meaningful relative to one another. The frame counter counts the number of actually captured images, while the trigger counter also includes missed trigger events (if they could be detected at all). The JobId and ParameterId jointly specify the parameters that were used for evaluating the image (including disparity computation). They do not necessarily reflect any settings that affect the actual image capturing. These are given separately instead. The reason for that behavior is that capture settings updates are hard to do atomically if you do not want to disable triggers (which would affect latency and in the worst case miss triggers).	26
PF.VsxProtocolDriver.Types.IVsxData	
Device data header. Specifies the data type of the received data.	28
PF.VsxProtocolDriver.Types.IVsxDeviceInformation	
Contains the information of a device.	29
PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor	
These parameters of the calibrated stereo camera system permit the 3D reconstruction in the Cartesian camera coordinate system from the disparity $D(u,v)$ at pixel column u and row v : $X(u, v) = (u - \text{Scan3dPrincipalPointU}) * \text{Scan3dBaseline} / D(u, v)$ $Y(u, v) = (v - \text{Scan3dPrincipalPointV}) * \text{Scan3dBaseline} / D(u, v)$ $Z(u, v) = \text{Scan3dFocalLength} * \text{Scan3dBaseline} / D(u, v)$	32
PF.VsxProtocolDriver.Types.IVsxDynamicContainer	
This container contains a collection of IVsxData messages. Depending on the connected device, certain messages are always or optionally available.	33
PF.VsxProtocolDriver.Types.IVsxFile	
This data represents a file, that could be transferred from the device to the user.	34
PF.VsxProtocolDriver.Types.IVsxImage	
A device image. Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.	35
PF.VsxProtocolDriver.Types.IVsxLineData	
A device line.	38
PF.VsxProtocolDriver.Types.IVsxLogData	
A log message received from device.	40
PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation	
Contains information identifying and describing the captured line data of the SpinTop G2 Sensor.	41
PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData	
Contains data which was set via modbus protocol.	45

PF.VsxProtocolDriver.IVsxProtocolDriver

Driver to communicate with a device/sensor by using Asynchronous programming.

Use `IVsxProtocolDriver` driver = `VsxProtocolDriver.InitTcpDevice(...)`

or

`IVsxProtocolDriver` driver = `VsxProtocolDriver.InitSerialDevice(...)`

to get an instance and use the driver.

46

PF.VsxProtocolDriver.IVsxProtocolDriverSync

Driver to communicate with a device/sensor by using synchronous programming.

Use `IVsxProtocolDriverSync` driver = `VsxProtocolDriver.InitTcpDevice(...)`

or

`IVsxProtocolDriverSync` driver = `VsxProtocolDriver.InitSerialDevice(...)`

to get an instance and use the driver.

64

PF.VsxProtocolDriver.Types.IVsxResultData

Result data received from device.

83

PF.VsxProtocolDriver.Types.IVsxSessionData

A session message. These messages deal with logging in and out, and setting passwords.

83

PF.VsxProtocolDriver.Types.IVsxTransformation

Used to transform raw point cloud data from device.

84

PF.VsxProtocolDriver.Types.Vsx

86

PF.VsxProtocolDriver.VsxProtocolDriver

Driver to communicate with a device/sensor by using Asynchronous programming.

Use `IVsxProtocolDriver` driver = `VsxProtocolDriver.InitTcpDevice(...)`

or

`IVsxProtocolDriver` driver = `VsxProtocolDriver.InitSerialDevice(...)`

to get an instance and use the driver.

90

PF.VsxProtocolDriver.VsxProtocolDriverSync

Driver to communicate with a device/sensor by using synchronous programming.

Use `IVsxProtocolDriverSync` driver = `VsxProtocolDriver.InitTcpDevice(...)`

or

`IVsxProtocolDriverSync` driver = `VsxProtocolDriver.InitSerialDevice(...)`

to get an instance and use the driver.

94

9 Namespace Documentation

9.1 PF Namespace Reference

Namespaces

- namespace [VsxProtocolDriver](#)

9.2 PF.VsxProtocolDriver Namespace Reference

Namespaces

- namespace [Types](#)

Classes

- interface [IVsxProtocolDriver](#)
Driver to communicate with a device/sensor by using Asynchronous programming.
 Use [IVsxProtocolDriver](#) `driver = VsxProtocolDriver.InitTcpDevice(...)` or
[IVsxProtocolDriver](#) `driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.
- interface [IVsxProtocolDriverSync](#)
Driver to communicate with a device/sensor by using synchronous programming.
 Use [IVsxProtocolDriverSync](#) `driver = VsxProtocolDriver.InitTcpDevice(...)` or
[IVsxProtocolDriverSync](#) `driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.
- class [VsxProtocolDriver](#)
Driver to communicate with a device/sensor by using Asynchronous programming.
 Use [IVsxProtocolDriver](#) `driver = VsxProtocolDriver.InitTcpDevice(...)` or
[IVsxProtocolDriver](#) `driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.
- class [VsxProtocolDriverSync](#)
Driver to communicate with a device/sensor by using synchronous programming.
 Use [IVsxProtocolDriverSync](#) `driver = VsxProtocolDriver.InitTcpDevice(...)` or
[IVsxProtocolDriverSync](#) `driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

9.3 PF.VsxProtocolDriver.Types Namespace Reference

Classes

- interface [ICoordinate](#)
A point of a line with coordinates and some attributes
- interface [IError](#)
Error object with detailed description when an error occurs.
- interface [IFirmwareState](#)
Detailed status description sent while making a firmware update.
- interface [IFirmwareVersion](#)
The current device firmware version.
- interface [IItemTuple](#)
An item tuple of id and value.
- interface [ILineMulti](#)
A collection of lines received from the device.
- interface [ILineSingle](#)
One single line received from the device.
- interface [IParameter](#)
Represents a single parameter of the device. Some properties are optional depending on the [Type](#) of the parameter
- interface [IPoint2D](#)
A 2D point of a line.
- interface [IPoint3D](#)
A 3D point of a line.
- interface [IStatusItem](#)
A status item received from the device.
- interface [IVsxApplicationResultData](#)
Application result data received from device.
- interface [IVsxCaptureInformation](#)

CaptureInformation contains information identifying and describing the captured image. The timestamp reflects a monotonic clock, without further synchronization timestamps are only meaningful relative to one another. The frame counter counts the number of actually captured images, while the trigger counter also includes missed trigger events (if they could be detected at all). The *JobId* and *ParameterId* jointly specify the parameters that were used for evaluating the image (including disparity computation). They do not necessarily reflect any settings that affect the actual image capturing. These are given separately instead. The reason for that behavior is that capture settings updates are hard to do atomically if you do not want to disable triggers (which would affect latency and in the worst case miss triggers).

- interface [IVsxData](#)
Device data header. Specifies the data type of the received data.
- interface [IVsxDeviceInformation](#)
Contains the information of a device.
- interface [IVsxDisparityDescriptor](#)
These parameters of the calibrated stereo camera system permit the 3D reconstruction in the Cartesian camera coordinate system from the disparity $D(u,v)$ at pixel column u and row v : $X(u, v) = (u - \text{Scan3dPrincipalPointU}) * \text{Scan3dBaseline} / D(u, v)$ $Y(u, v) = (v - \text{Scan3dPrincipalPointV}) * \text{Scan3dBaseline} / D(u, v)$ $Z(u, v) = \text{Scan3dFocalLength} * \text{Scan3dBaseline} / D(u, v)$
- interface [IVsxDynamicContainer](#)
This container contains a collection of [IVsxData](#) messages. Depending on the connected device, certain messages are always or optionally available.
- interface [IVsxFile](#)
This data represents a file, that could be transferred from the device to the user.
- interface [IVsxImage](#)
A device image. Depending on the *ImageData2Format*, the image data is either saved in *ImageData* or in *ImageDataFloats*.
- interface [IVsxLineData](#)
A device line.
- interface [IVsxLogData](#)
A log message received from device.
- interface [IVsxOlr2CaptureInformation](#)
Contains information identifying and describing the captured line data of the SpinTop G2 Sensor.
- interface [IVsxOlr2ModbusData](#)
Contains data which was set via modbus protocol.
- interface [IVsxResultData](#)
Result data received from device.
- interface [IVsxSessionData](#)
A session message. These messages deal with logging in and out, and setting passwords.
- interface [IVsxTransformation](#)
Used to transform raw point cloud data from device.
- class [Vsx](#)

Enumerations

- enum [ErrorId](#)
The id of the error.
- enum [DisconnectEvent](#)
The disconnect reason.
- enum [Strategy](#)
The strategy which containers are removed when max number of items is reached.
- enum [SerialConnectionType](#)
The serial connection type.
- enum [DeviceStatusScope](#)
Flag about the verbosity of the status message.

- enum [Content](#)
- enum [VsxType](#)
 - The device data type.*
- enum [ImageFormat](#)
- enum [ImageData2Format](#)

9.3.1 Enumeration Type Documentation

9.3.1.1 ErrorId enum [PF.VsxProtocolDriver.Types.ErrorId](#)

The id of the error.

9.3.1.2 DisconnectEvent enum [PF.VsxProtocolDriver.Types.DisconnectEvent](#)

The disconnect reason.

9.3.1.3 Strategy enum [PF.VsxProtocolDriver.Types.Strategy](#)

The strategy which containers are removed when max number of items is reached.

9.3.1.4 SerialConnectionType enum [PF.VsxProtocolDriver.Types.SerialConnectionType](#)

The serial connection type.

9.3.1.5 DeviceStatusScope enum [PF.VsxProtocolDriver.Types.DeviceStatusScope](#)

Flag about the verbosity of the status message.

9.3.1.6 Content enum [PF.VsxProtocolDriver.Types.Content](#)

9.3.1.7 VsxType enum [PF.VsxProtocolDriver.Types.VsxType](#)

The device data type.

9.3.1.8 ImageFormat enum `PF.VsxProtocolDriver.Types.ImageFormat`

9.3.1.9 ImageData2Format enum `PF.VsxProtocolDriver.Types.ImageData2Format`

10 Class Documentation

10.1 PF.VsxProtocolDriver.Types.ICoordinate Interface Reference

A point of a line with coordinates and some attributes

Properties

- `IPoint2D ImageCoordinate` [get]
Gets or sets the image coordinate.
- `IPoint3D WorldCoordinate` [get]
Gets or sets the world coordinate.
- float `Intensity` [get]
Intensity for this coordinate
- float `Quality` [get]
Quality for this coordinate
- bool `Valid` [get]
Is coordinate valid?
- int `LineId` [get]
The id of the line the coordinate corresponds to
- int `SegmentId` [get]
The id of the line segment the coordinate corresponds to

10.1.1 Detailed Description

A point of a line with coordinates and some attributes

10.1.2 Property Documentation

10.1.2.1 ImageCoordinate `IPoint2D` `PF.VsxProtocolDriver.Types.ICoordinate.ImageCoordinate` [get]

Gets or sets the image coordinate.

The image coordinate.

10.1.2.2 WorldCoordinate `IPoint3D PF.VsxProtocolDriver.Types.ICoordinate.WorldCoordinate [get]`

Gets or sets the world coordinate.

The world coordinate.

10.1.2.3 Intensity `float PF.VsxProtocolDriver.Types.ICoordinate.Intensity [get]`

Intensity for this coordinate

10.1.2.4 Quality `float PF.VsxProtocolDriver.Types.ICoordinate.Quality [get]`

Quality for this coordinate

10.1.2.5 Valid `bool PF.VsxProtocolDriver.Types.ICoordinate.Valid [get]`

Is coordinate valid?

10.1.2.6 LineId `int PF.VsxProtocolDriver.Types.ICoordinate.LineId [get]`

The id of the line the coordinate corresponds to

10.1.2.7 SegmentId `int PF.VsxProtocolDriver.Types.ICoordinate.SegmentId [get]`

The id of the line segment the coordinate corresponds to

10.2 PF.VsxProtocolDriver.Types.IError Interface Reference

Error object with detailed description when an error occurs.

Properties

- `ErrorId Id [get]`
The id of the error.
- `string Message [get]`
Detailed error description.

10.2.1 Detailed Description

Error object with detailed description when an error occurs.

10.2.2 Property Documentation

10.2.2.1 Id `ErrorId PF.VsxProtocolDriver.Types.IError.Id [get]`

The id of the error.

10.2.2.2 Message `string PF.VsxProtocolDriver.Types.IError.Message [get]`

Detailed error description.

10.3 PF.VsxProtocolDriver.Types.IFirmwareState Interface Reference

Detailed status description sent while making a firmware update.

Properties

- int `Status` [get]
A status tag.
- string `Msg` [get]
A status message.
- int `Error` [get]
A status error tag.
- int `LastResult` [get]
The last result tag.

10.3.1 Detailed Description

Detailed status description sent while making a firmware update.

10.3.2 Property Documentation

10.3.2.1 Status `int PF.VsxProtocolDriver.Types.IFirmwareState.Status [get]`

A status tag.

10.3.2.2 Msg `string PF.VsxProtocolDriver.Types.IFirmwareState.Msg [get]`

A status message.

10.3.2.3 Error `int PF.VsxProtocolDriver.Types.IFirmwareState.Error [get]`

A status error tag.

10.3.2.4 LastResult `int PF.VsxProtocolDriver.Types.IFirmwareState.LastResult [get]`

The last result tag.

10.4 PF.VsxProtocolDriver.Types.IFirmwareVersion Interface Reference

The current device firmware version.

Properties

- int **Major** [get]
Major version
- int **Minor** [get]
Minor version
- int **Build** [get]
Build version
- int **Revision** [get]
Revision version

10.4.1 Detailed Description

The current device firmware version.

10.4.2 Property Documentation

10.4.2.1 Major `int PF.VsxProtocolDriver.Types.IFirmwareVersion.Major [get]`

Major version

10.4.2.2 Minor `int PF.VsxProtocolDriver.Types.IFirmwareVersion.Minor [get]`

Minor version

10.4.2.3 Build `int PF.VsxProtocolDriver.Types.IFirmwareVersion.Build [get]`

Build version

10.4.2.4 Revision `int PF.VsxProtocolDriver.Types.IFirmwareVersion.Revision [get]`

Revision version

10.5 PF.VsxProtocolDriver.Types.IItemTuple Interface Reference

An item tuple of id and value.

Properties

- string `Id` [get]
The id of the item tuple
- string `Name` [get]
The Name of the item tuple

10.5.1 Detailed Description

An item tuple of id and value.

10.5.2 Property Documentation

10.5.2.1 Id `string PF.VsxProtocolDriver.Types.IItemTuple.Id [get]`

The id of the item tuple

10.5.2.2 Name `string PF.VsxProtocolDriver.Types.IItemTuple.Name [get]`

The Name of the item tuple

10.6 PF.VsxProtocolDriver.Types.ILineMulti Interface Reference

A collection of lines received from the device.

Properties

- List< [ILineSingle](#) > [Lines](#) [get]
The list of the lines.
- double [MinX](#) [get]
The lines min X value
- double [MaxX](#) [get]
The lines max X value
- double [MinZ](#) [get]
The lines min Z value
- double [MaxZ](#) [get]
The lines max Z value

10.6.1 Detailed Description

A collection of lines received from the device.

10.6.2 Property Documentation

10.6.2.1 Lines List<[ILineSingle](#)> PF.VsxProtocolDriver.Types.ILineMulti.Lines [get]

The list of the lines.

10.6.2.2 MinX double PF.VsxProtocolDriver.Types.ILineMulti.MinX [get]

The lines min X value

10.6.2.3 MaxX double PF.VsxProtocolDriver.Types.ILineMulti.MaxX [get]

The lines max X value

10.6.2.4 MinZ double PF.VsxProtocolDriver.Types.ILineMulti.MinZ [get]

The lines min Z value

10.6.2.5 MaxZ `double PF.VsxProtocolDriver.Types.ILineMulti.MaxZ [get]`

The lines max Z value

10.7 PF.VsxProtocolDriver.Types.ILineSingle Interface Reference

One single line received from the device.

Properties

- `List< ICoordinate > Points [get]`
A list of all line points.

10.7.1 Detailed Description

One single line received from the device.

10.7.2 Property Documentation

10.7.2.1 Points `List<ICoordinate> PF.VsxProtocolDriver.Types.ILineSingle.Points [get]`

A list of all line points.

10.8 PF.VsxProtocolDriver.Types.IParameter Interface Reference

Represents a single parameter of the device. Some properties are optional depending on the [Type](#) of the parameter

Public Member Functions

- `IParameter Clone ()`
Copy constructor

Properties

- ushort [SettingsVersion](#) [get]
The settings version of the settings node the parameter belongs to (identical for all parameters of a device).
- ushort [ConfigVersion](#) [get]
The config version of the configuration node the parameter belongs to
- string [ConfigId](#) [get]
The configuration id of the parameter. The cref=ConfigId together with the [ParameterId](#) is unique.
- string [ConfigType](#) [get]
The type of the configuration the parameter belongs to.
- string [ParameterId](#) [get]
The parameter id of the parameter. The cref=ConfigId together with the [ParameterId](#) is unique.
- string [Name](#) [get]
The display name of the parameter.
- [Vsx.ParameterTypes](#) Type [get]
The type of the parameter which determines the control type.
- [Vsx.ValueTypes](#) ValueType [get]
The value type of the parameter which determines the data type of the parameters cref="Value". Note: The value is either bool, a 64 bit integer or a 64 bit floating point double value.
- bool [Enable](#) [get]
Determines if the parameter is enabled and allowed to edit (depending on the user level) or not.
- bool [Visible](#) [get]
Determines if the parameter is visible or not.
- object? [Min](#) [get]
The minimum value of the parameter, type depends on the parameters [ValueType](#).
- object? [Max](#) [get]
The maximum value of the parameter, type depends on the parameters [ValueType](#).
- object? [Step](#) [get]
The step size of the parameter, type depends on the parameters [ValueType](#).
- object? [Value](#) [get, set]
The current value of the parameter, type depends on the parameters [ValueType](#). Note: The value is either bool, a string, a 64 bit integer or a 64 bit floating point double value.
- object? [DefaultValue](#) [get]
The default value of the parameter, type depends on the parameters [ValueType](#).
- string [Unit](#) [get]
The unit of the parameter.
- List< [ItemTuple](#) > [Items](#) [get]
Value entries if [Type](#) is Combobox

10.8.1 Detailed Description

Represents a single parameter of the device. Some properties are optional depending on the [Type](#) of the parameter

10.8.2 Member Function Documentation

10.8.2.1 Clone() `IPParameter PF.VsxProtocolDriver.Types.IParameter.Clone ()`

Copy constructor

Returns

10.8.3 Property Documentation

10.8.3.1 SettingsVersion `ushort PF.VsxProtocolDriver.Types.IParameter.SettingsVersion [get]`

The settings version of the settings node the parameter belongs to (identical for all parameters of a device).

10.8.3.2 ConfigVersion `ushort PF.VsxProtocolDriver.Types.IParameter.ConfigVersion [get]`

The config version of the configuration node the parameter belongs to

10.8.3.3 ConfigId `string PF.VsxProtocolDriver.Types.IParameter.ConfigId [get]`

The configuration id of the parameter. The cref=ConfigId together with the [ParameterId](#) is unique.

10.8.3.4 ConfigType `string PF.VsxProtocolDriver.Types.IParameter.ConfigType [get]`

The type of the configuration the parameter belongs to.

10.8.3.5 ParameterId `string PF.VsxProtocolDriver.Types.IParameter.ParameterId [get]`

The parameter id of the parameter. The cref=ConfigId together with the [ParameterId](#) is unique.

10.8.3.6 Name `string PF.VsxProtocolDriver.Types.IParameter.Name [get]`

The display name of the parameter.

10.8.3.7 Type `Vsx.ParameterTypes PF.VsxProtocolDriver.Types.IParameter.Type [get]`

The type of the parameter which determines the control type.

10.8.3.8 ValueType `Vsx.ValueTypes PF.VsxProtocolDriver.Types.IParameter.ValueType [get]`

The value type of the parameter which determines the data type of the parameters cref="Value". Note: The value is either bool, a 64 bit integer or a 64 bit floating point double value.

10.8.3.9 Enable `bool PF.VsxProtocolDriver.Types.IParameter.Enable [get]`

Determines if the parameter is enabled and allowed to edit (depending on the user level) or not.

10.8.3.10 Visible `bool PF.VsxProtocolDriver.Types.IParameter.Visible [get]`

Determines if the parameter is visible or not.

10.8.3.11 Min `object? PF.VsxProtocolDriver.Types.IParameter.Min [get]`

The minimum value of the parameter, type depends on the parameters [ValueType](#).

10.8.3.12 Max `object? PF.VsxProtocolDriver.Types.IParameter.Max [get]`

The maximum value of the parameter, type depends on the parameters [ValueType](#).

10.8.3.13 Step `object? PF.VsxProtocolDriver.Types.IParameter.Step [get]`

The step size of the parameter, type depends on the parameters [ValueType](#).

10.8.3.14 Value `object? PF.VsxProtocolDriver.Types.IParameter.Value [get], [set]`

The current value of the parameter, type depends on the parameters [ValueType](#). Note: The value is either bool, a string, a 64 bit integer or a 64 bit floating point double value.

10.8.3.15 DefaultValue `object? PF.VsxProtocolDriver.Types.IParameter.DefaultValue [get]`

The default value of the parameter, type depends on the parameters [ValueType](#).

10.8.3.16 Unit `string PF.VsxProtocolDriver.Types.IParameter.Unit [get]`

The unit of the parameter.

10.8.3.17 Items `List<ITuple> PF.VsxProtocolDriver.Types.IParameter.Items [get]`

Value entries if [Type](#) is Combobox

10.9 PF.VsxProtocolDriver.Types.IPoint2D Interface Reference

A 2D point of a line.

Properties

- double [X](#) [get]
The x coordinate.
- double [Y](#) [get]
The y coordinate.

10.9.1 Detailed Description

A 2D point of a line.

10.9.2 Property Documentation

10.9.2.1 X `double PF.VsxProtocolDriver.Types.IPoint2D.X [get]`

The x coordinate.

10.9.2.2 Y `double PF.VsxProtocolDriver.Types.IPoint2D.Y [get]`

The y coordinate.

10.10 PF.VsxProtocolDriver.Types.IPoint3D Interface Reference

A 3D point of a line.

Properties

- double **X** [get]
The x coordinate.
- double **Y** [get]
The y coordinate.
- double **Z** [get]
The z coordinate.

10.10.1 Detailed Description

A 3D point of a line.

10.10.2 Property Documentation

10.10.2.1 X double PF.VsxProtocolDriver.Types.IPoint3D.X [get]

The x coordinate.

10.10.2.2 Y double PF.VsxProtocolDriver.Types.IPoint3D.Y [get]

The y coordinate.

10.10.2.3 Z double PF.VsxProtocolDriver.Types.IPoint3D.Z [get]

The z coordinate.

10.11 PF.VsxProtocolDriver.Types.IStatusItem Interface Reference

A status item received from the device.

Properties

- ushort **SettingsVersion** [get]
The settings version of the settings node the status item belongs to (identical for all status items of a device).
- ushort **ConfigVersion** [get]
The config version of the configuration node the status item belongs to
- string **StatusItemId** [get]
The id of the status item.
- string **Name** [get]
The display name of the status item.
- **Vsx.ValueTypes.ValueType** [get]
*The value type of the status item which determines the data type of the status items **Value**.*
- object? **Value** [get]
*The current value of the status item, type dependend of the status items **ValueType**.*
- List< **ItemTuple** >? **Items** [get]
List of possible values.
- ulong **Time** [get]
Last changed timestamp of the status signal.
- ulong **SensorTime** [get]
Current timestamp of the sensor clock.
- string **ConfigurationClass** [get]
The class the status item belongs to.

10.11.1 Detailed Description

A status item received from the device.

10.11.2 Property Documentation

10.11.2.1 **SettingsVersion** ushort PF.VsxProtocolDriver.Types.IStatusItem.SettingsVersion [get]

The settings version of the settings node the status item belongs to (identical for all status items of a device).

10.11.2.2 **ConfigVersion** ushort PF.VsxProtocolDriver.Types.IStatusItem.ConfigVersion [get]

The config version of the configuration node the status item belongs to

10.11.2.3 **StatusItemId** string PF.VsxProtocolDriver.Types.IStatusItem.StatusItemId [get]

The id of the status item.

10.11.2.4 Name `string PF.VsxProtocolDriver.Types.IStatusItem.Name [get]`

The display name of the status item.

10.11.2.5 ValueType `Vsx.ValueTypes PF.VsxProtocolDriver.Types.IStatusItem.ValueType [get]`

The value type of the status item which determines the data type of the status items [Value](#).

10.11.2.6 Value `object? PF.VsxProtocolDriver.Types.IStatusItem.Value [get]`

The current value of the status item, type dependend of the status items [ValueType](#).

10.11.2.7 Items `List<IItemTuple>? PF.VsxProtocolDriver.Types.IStatusItem.Items [get]`

List of possible values.

10.11.2.8 Time `ulong PF.VsxProtocolDriver.Types.IStatusItem.Time [get]`

Last changed timestamp of the status signal.

10.11.2.9 SensorTime `ulong PF.VsxProtocolDriver.Types.IStatusItem.SensorTime [get]`

Current timestamp of the sensor clock.

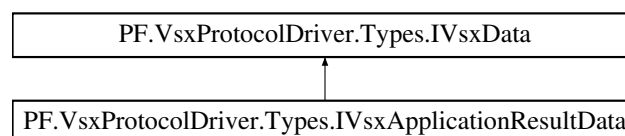
10.11.2.10 ConfigurationClass `string PF.VsxProtocolDriver.Types.IStatusItem.ConfigurationClass [get]`

The class the status item belongs to.

10.12 PF.VsxProtocolDriver.Types.IVsxApplicationResultData Interface Reference

Application result data received from device.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxApplicationResultData:



Properties

- string [ApplicationResultString](#) [get]

10.12.1 Detailed Description

Application result data received from device.

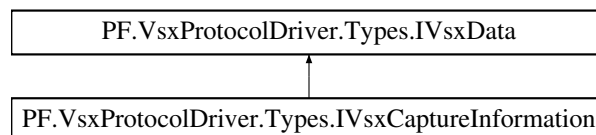
10.12.2 Property Documentation

10.12.2.1 ApplicationResultString string PF.VsxProtocolDriver.Types.IVsxApplicationResultData.
ApplicationResultString [get]

10.13 PF.VsxProtocolDriver.Types.IVsxCaptureInformation Interface Reference

CaptureInformation contains information identifying and describing the captured image. The timestamp reflects a monotonic clock, without further synchronization timestamps are only meaningful relative to one another. The frame counter counts the number of actually captured images, while the trigger counter also includes missed trigger events (if they could be detected at all). The JobId and ParameterId jointly specify the parameters that were used for evaluating the image (including disparity computation). They do not necessarily reflect any settings that affect the actual image capturing. These are given separately instead. The reason for that behavior is that capture settings updates are hard to do atomically if you do not want to disable triggers (which would affect latency and in the worst case miss triggers).

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxCaptureInformation:



Properties

- ulong [TriggerCtr](#) [get]
- ulong [ParameterId](#) [get]
- ulong [JobId](#) [get]
- long [RotaryEncoder](#) [get]
- ulong [FrameCounter](#) [get]
- ulong [Timestamp](#) [get]
- uint [ExposureTime](#) [get]
- uint [Gain](#) [get]
- byte [Illumination](#) [get]
- byte [TriggerSource](#) [get]

10.13.1 Detailed Description

CaptureInformation contains information identifying and describing the captured image. The timestamp reflects a monotonic clock, without further synchronization timestamps are only meaningful relative to one another. The frame counter counts the number of actually captured images, while the trigger counter also includes missed trigger events (if they could be detected at all). The JobId and ParameterId jointly specify the parameters that were used for evaluating the image (including disparity computation). They do not necessarily reflect any settings that affect the actual image capturing. These are given separately instead. The reason for that behavior is that capture settings updates are hard to do atomically if you do not want to disable triggers (which would affect latency and in the worst case miss triggers).

10.13.2 Property Documentation

10.13.2.1 TriggerCtr `ulong PF.VsxProtocolDriver.Types.IVsxCaptureInformation.TriggerCtr` [get]

10.13.2.2 ParameterId `ulong PF.VsxProtocolDriver.Types.IVsxCaptureInformation.ParameterId`
[get]

10.13.2.3 JobId `ulong PF.VsxProtocolDriver.Types.IVsxCaptureInformation.JobId` [get]

10.13.2.4 RotaryEncoder `long PF.VsxProtocolDriver.Types.IVsxCaptureInformation.RotaryEncoder`
[get]

10.13.2.5 FrameCounter `ulong PF.VsxProtocolDriver.Types.IVsxCaptureInformation.FrameCounter`
[get]

10.13.2.6 Timestamp `ulong PF.VsxProtocolDriver.Types.IVsxCaptureInformation.Timestamp` [get]

10.13.2.7 ExposureTime `uint PF.VsxProtocolDriver.Types.IVsxCaptureInformation.ExposureTime`
[get]

10.13.2.8 Gain `uint PF.VsxProtocolDriver.Types.IVsxCaptureInformation.Gain [get]`

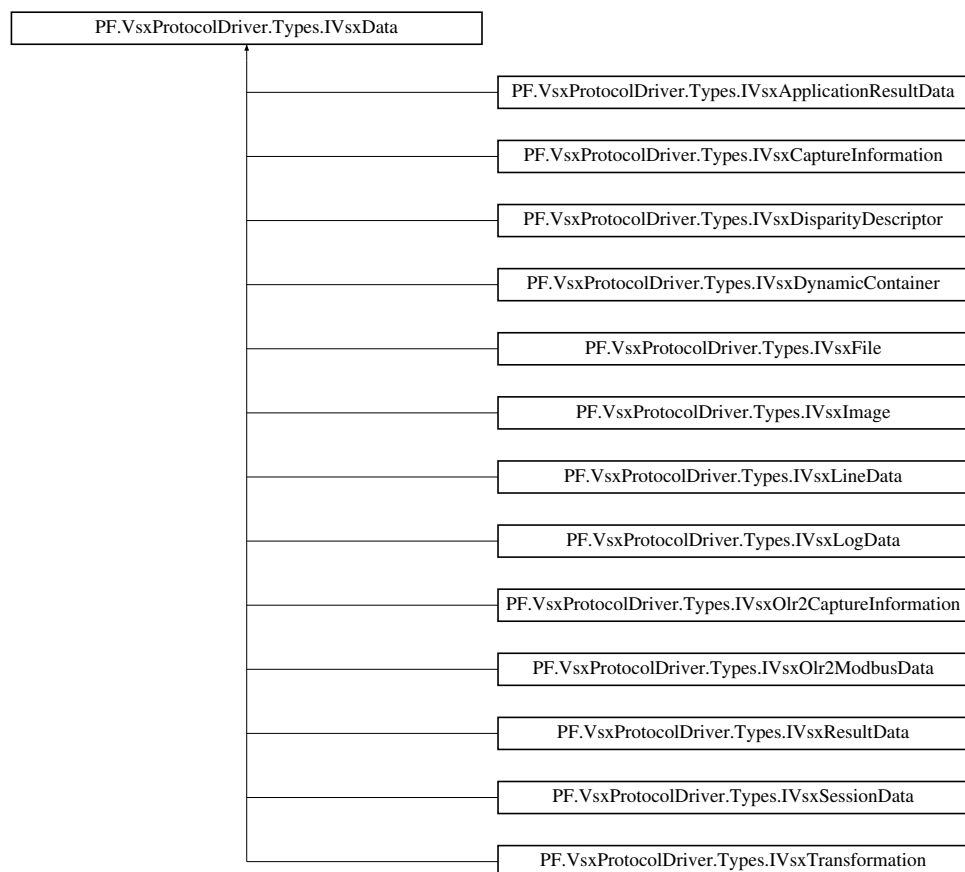
10.13.2.9 Illumination `byte PF.VsxProtocolDriver.Types.IVsxCaptureInformation.Illumination [get]`

10.13.2.10 TriggerSource `byte PF.VsxProtocolDriver.Types.IVsxCaptureInformation.TriggerSource [get]`

10.14 PF.VsxProtocolDriver.Types.IVsxData Interface Reference

Device data header. Specifies the data type of the received data.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxData:



Properties

- [VsxType VsxDatatype](#) [get]

10.14.1 Detailed Description

Device data header. Specifies the data type of the received data.

10.14.2 Property Documentation

10.14.2.1 VsxDatatype [VsxDatatype](#) PF.VsxProtocolDriver.Types.IVsxData.VsxDatatype [get]

10.15 PF.VsxProtocolDriver.Types.IVsxDeviceInformation Interface Reference

Contains the information of a device.

Properties

- string [IpAddress](#) [get]
- string [NetworkMask](#) [get]
The network mask of the device.
- string [Gateway](#) [get]
The standard gateway of the device.
- string [MacAddress](#) [get]
The MAC address of the device.
- string [FirmwareVersion](#) [get]
The firmware version string of the device.
- string [SensorType](#) [get]
The type of the device.
- string [SensorName](#) [get]
The name of the device.
- bool [Busy](#) [get]
Indicates if device is already connected or not.
- int [DeviceVsxVersionMajor](#) [get]
The device major VSX protocol version.
- int [DeviceVsxVersionMinor](#) [get]
The device minor VSX protocol version.
- string [ComPort](#) [get]
The comport if device is a serial one.
- int [Baudrate](#) [get]
The baudrate if device is a serial one.
- string [Headaddress](#) [get]
The headaddress if device is a serial one.
- bool [IsLoginNeeded](#) [get]
Gets a value indicating whether the device needs login.
- [VsxDatatype](#) [DeviceStatus](#) [get]
Gets the device status.
- string [ActiveApplicationIdentifier](#) [get]
Gets the identifier of currently running application.

10.15.1 Detailed Description

Contains the information of a device.

10.15.2 Property Documentation

10.15.2.1 IpAddress `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.IpAddress [get]`

10.15.2.2 NetworkMask `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.NetworkMask [get]`

The network mask of the device.

10.15.2.3 Gateway `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Gateway [get]`

The standard gateway of the device.

10.15.2.4 MacAddress `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.MacAddress [get]`

The MAC address of the device.

10.15.2.5 FirmwareVersion `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Firmware↔Version [get]`

The firmware version string of the device.

10.15.2.6 SensorType `string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.SensorType [get]`

The type of the device.

10.15.2.7 SensorName string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.SensorName
[get]

The name of the device.

10.15.2.8 Busy bool PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Busy [get]

Indicates if device is already connected or not.

10.15.2.9 DeviceVsxVersionMajor int PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Device↔
VsxVersionMajor [get]

The device major VSX protocol version.

10.15.2.10 DeviceVsxVersionMinor int PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Device↔
VsxVersionMinor [get]

The device minor VSX protocol version.

10.15.2.11 ComPort string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.ComPort [get]

The comport if device is a serial one.

10.15.2.12 Baudrate int PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Baudrate [get]

The baudrate if device is a serial one.

10.15.2.13 Headaddress string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.Headaddress
[get]

The headaddress if device is a serial one.

10.15.2.14 IsLoginNeeded bool PF.VsxProtocolDriver.Types.IVsxDeviceInformation.IsLoginNeeded
[get]

Gets a value indicating whether the device needs login.

10.15.2.15 DeviceStatus `Vsx.DeviceStatus` `PF.VsxProtocolDriver.Types.IVsxDeviceInformation.`
`DeviceStatus` [get]

Gets the device status.

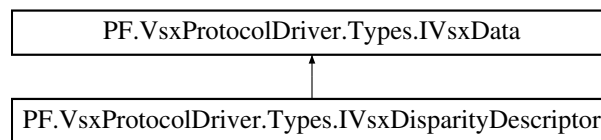
10.15.2.16 ActiveApplicationIdentifier `string` `PF.VsxProtocolDriver.Types.IVsxDeviceInformation.`
`ActiveApplicationIdentifier` [get]

Gets the identifier of currently running application.

10.16 PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor Interface Reference

These parameters of the calibrated stereo camera system permit the 3D reconstruction in the Cartesian camera coordinate system from the disparity $D(u,v)$ at pixel column u and row v : $X(u, v) = (u - \text{Scan3dPrincipalPointU}) * \text{Scan3dBaseline} / D(u, v)$ $Y(u, v) = (v - \text{Scan3dPrincipalPointV}) * \text{Scan3dBaseline} / D(u, v)$ $Z(u, v) = \text{Scan3dFocalLength} * \text{Scan3dBaseline} / D(u, v)$

Inheritance diagram for `PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor`:



Properties

- `double` `FocalLength` [get]
- `double` `PrincipalPointU` [get]
- `double` `PrincipalPointV` [get]
- `double` `Baseline` [get]
- `double` `OffsetLeftRectifiedToDisparityU` [get]
- `double` `OffsetLeftRectifiedToDisparityV` [get]

10.16.1 Detailed Description

These parameters of the calibrated stereo camera system permit the 3D reconstruction in the Cartesian camera coordinate system from the disparity $D(u,v)$ at pixel column u and row v : $X(u, v) = (u - \text{Scan3dPrincipalPointU}) * \text{Scan3dBaseline} / D(u, v)$ $Y(u, v) = (v - \text{Scan3dPrincipalPointV}) * \text{Scan3dBaseline} / D(u, v)$ $Z(u, v) = \text{Scan3dFocalLength} * \text{Scan3dBaseline} / D(u, v)$

10.16.2 Property Documentation

10.16.2.1 FocalLength double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.FocalLength [get]

10.16.2.2 PrincipalPointU double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.PrincipalPointU [get]

10.16.2.3 PrincipalPointV double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.PrincipalPointV [get]

10.16.2.4 Baseline double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.Baseline [get]

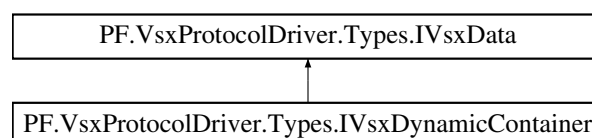
10.16.2.5 OffsetLeftRectifiedToDisparityU double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.OffsetLeftRectifiedToDisparityU [get]

10.16.2.6 OffsetLeftRectifiedToDisparityV double PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor.OffsetLeftRectifiedToDisparityV [get]

10.17 PF.VsxProtocolDriver.Types.IVsxDynamicContainer Interface Reference

This container contains a collection of [IVsxData](#) messages. Depending on the connected device, certain messages are always or optionally available.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxDynamicContainer:



Public Member Functions

- bool [ContainsMessage](#) (string tag)
- [IVsxData GetMessage](#) (string tag)
- void [Dispose](#) ()
- object [Clone](#) ()

Properties

- List< string > [TagListOfContainedMessages](#) [get]

10.17.1 Detailed Description

This container contains a collection of [IVsxData](#) messages. Depending on the connected device, certain messages are always or optionally available.

10.17.2 Member Function Documentation

10.17.2.1 ContainsMessage() bool PF.VsxProtocolDriver.Types.IVsxDynamicContainer.Contains↔
Message (
 string tag)

10.17.2.2 GetMessage() [IVsxData](#) PF.VsxProtocolDriver.Types.IVsxDynamicContainer.GetMessage (
 string tag)

10.17.2.3 Dispose() void PF.VsxProtocolDriver.Types.IVsxDynamicContainer.Dispose ()

10.17.2.4 Clone() object PF.VsxProtocolDriver.Types.IVsxDynamicContainer.Clone ()

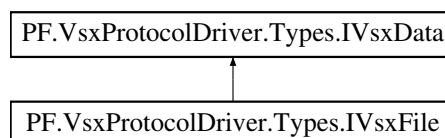
10.17.3 Property Documentation

10.17.3.1 TagListOfContainedMessages List<string> PF.VsxProtocolDriver.Types.IVsxDynamic↔
Container.TagListOfContainedMessages [get]

10.18 PF.VsxProtocolDriver.Types.IVsxFile Interface Reference

This data represents a file, that could be transferred from the device to the user.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxFile:



Properties

- uint **FileStatus** [get]
0: Ok 1: Error opening file 2: Error reading data 3: Error data exceed 32bit size
- string **Path** [get]
Name of file transferred, e.g. "boot.gz"
- string **MimeType** [get]
Type declaration of file, e.g. "application/gzip"
- string **Contents** [get]

10.18.1 Detailed Description

This data represents a file, that could be transferred from the device to the user.

10.18.2 Property Documentation

10.18.2.1 FileStatus uint PF.VsxProtocolDriver.Types.IVsxFile.FileStatus [get]

0: Ok 1: Error opening file 2: Error reading data 3: Error data exceed 32bit size

10.18.2.2 Path string PF.VsxProtocolDriver.Types.IVsxFile.Path [get]

Name of file transferred, e.g. "boot.gz"

10.18.2.3 MimeType string PF.VsxProtocolDriver.Types.IVsxFile.MimeType [get]

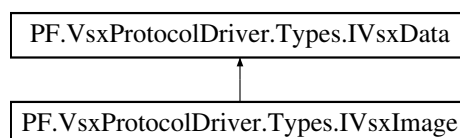
Type declaration of file, e.g. "application/gzip"

10.18.2.4 Contents string PF.VsxProtocolDriver.Types.IVsxFile.Contents [get]

10.19 PF.VsxProtocolDriver.Types.IVsxImage Interface Reference

A device image. Depending on the ImageData2Format, the image data is either saved in ImageData or in Image↔DataFloats.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxImage:



Properties

- [ImageFormat ImageType](#) [get]
- Matrix [TransformationMatrix](#) [get]
- Bitmap [Image](#) [get]
- byte[] [ImageData](#) [get]

Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.

- [ImageData2Format Format](#) [get]
- int [Width](#) [get]
- int [Height](#) [get]
- int [LinePitch](#) [get]

Total number of bytes between the starts of 2 consecutive lines. This feature is used to facilitate alignment of image data. This might be useful if the system has specific limitations, such as having the lines aligned on 32-bit boundaries.

- long [FrameCounter](#) [get]

64 bit counter with cyclic overflow

- double [CoordinateScale](#) [get]

Scale factor when transforming a pixel from relative coordinates to world coordinates. A negative scale mirrors the axis. For rectified image axes it is the distance between samples in the rectified image along this axis.

- double [CoordinateOffset](#) [get]

Offset when transforming a pixel from relative coordinates to world coordinates.

- double [AxisMin](#) [get]

Minimum valid transmitted coordinate value of the selected Axis. The values are for information purposes to e.g. facilitate scaling of display resolution. The value references the raw data value (not the scaled / offset transformed value).

- double [AxisMax](#) [get]

Maximum valid transmitted coordinate value of the selected Axis. The values are for information purposes to e.g. facilitate scaling of display resolution. The value references the raw data value (not the scaled / offset transformed value).

- double [InvalidDataValue](#) [get]

Value which identifies an invalid pixel. Typically, the invalid data is flagged in one coordinate (Z/Rho) only, but it can be applied to each coordinate. If the pixel format is integer the value must be mapped to (rounded to) an integer register in the device. Using a floating point NaN during pixel data processing might incur performance penalties, it might be desirable to avoid such values within pixel data whenever possible. When set to -INF, this value is not used (default)

- float[] [ImageDataFloats](#) [get]

Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.

10.19.1 Detailed Description

A device image. Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.

10.19.2 Property Documentation

10.19.2.1 ImageType [ImageFormat](#) `PF.VsxProtocolDriver.Types.IVsxImage.ImageType` [get]

10.19.2.2 TransformationMatrix Matrix PF.VsxProtocolDriver.Types.IVsxImage.TransformationMatrix [get]

10.19.2.3 Image Bitmap PF.VsxProtocolDriver.Types.IVsxImage.Image [get]

10.19.2.4 ImageData byte [] PF.VsxProtocolDriver.Types.IVsxImage.ImageData [get]

Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.

10.19.2.5 Format ImageData2Format PF.VsxProtocolDriver.Types.IVsxImage.Format [get]

10.19.2.6 Width int PF.VsxProtocolDriver.Types.IVsxImage.Width [get]

10.19.2.7 Height int PF.VsxProtocolDriver.Types.IVsxImage.Height [get]

10.19.2.8 LinePitch int PF.VsxProtocolDriver.Types.IVsxImage.LinePitch [get]

Total number of bytes between the starts of 2 consecutive lines. This feature is used to facilitate alignment of image data. This might be useful if the system has specific limitations, such as having the lines aligned on 32-bit boundaries.

10.19.2.9 FrameCounter long PF.VsxProtocolDriver.Types.IVsxImage.FrameCounter [get]

64 bit counter with cyclic overflow

10.19.2.10 CoordinateScale double PF.VsxProtocolDriver.Types.IVsxImage.CoordinateScale [get]

Scale factor when transforming a pixel from relative coordinates to world coordinates. A negative scale mirrors the axis. For rectified image axes it is the distance between samples in the rectified image along this axis.

10.19.2.11 CoordinateOffset double PF.VsxProtocolDriver.Types.IVsxImage.CoordinateOffset [get]

Offset when transforming a pixel from relative coordinates to world coordinates.

10.19.2.12 AxisMin double PF.VsxProtocolDriver.Types.IVsxImage.AxisMin [get]

Minimum valid transmitted coordinate value of the selected Axis. The values are for information purposes to e.g. facilitate scaling of display resolution. The value references the raw data value (not the scaled / offset transformed value).

10.19.2.13 AxisMax double PF.VsxProtocolDriver.Types.IVsxImage.AxisMax [get]

Maximum valid transmitted coordinate value of the selected Axis. The values are for information purposes to e.g. facilitate scaling of display resolution. The value references the raw data value (not the scaled / offset transformed value).

10.19.2.14 InvalidDataValue double PF.VsxProtocolDriver.Types.IVsxImage.InvalidDataValue [get]

Value which identifies an invalid pixel. Typically, the invalid data is flagged in one coordinate (Z/Rho) only, but it can be applied to each coordinate. If the pixel format is integer the value must be mapped to (rounded to) an integer register in the device. Using a floating point NaN during pixel data processing might incur performance penalties, it might be desirable to avoid such values within pixel data whenever possible. When set to -INF, this value is not used (default)

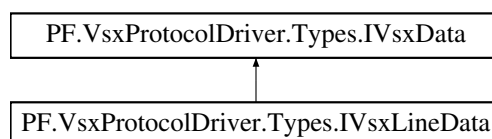
10.19.2.15 ImageDataFloats float [] PF.VsxProtocolDriver.Types.IVsxImage.ImageDataFloats [get]

Depending on the ImageData2Format, the image data is either saved in ImageData or in ImageDataFloats.

10.20 PF.VsxProtocolDriver.Types.IVsxLineData Interface Reference

A device line.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxLineData:



Properties

- ushort [CountLines](#) [get]
- ushort [Format](#) [get]
- ushort [Scale](#) [get]
- ushort [ScaleXYZ](#) [get]
- short [MinX](#) [get]
- short [MaxX](#) [get]
- short [MinZ](#) [get]
- short [MaxZ](#) [get]
- ushort [FrameCounter](#) [get]
- [ILineMulti Lines](#) [get]

10.20.1 Detailed Description

A device line.

In 32 bit mode the single data values are displayed as 32 bit, otherwise in 16 bit. A line consists of "width" data packages. Each data package consists of max 6 int16 or int32 values. The exact number of values is determined by the format bits. A line data message can contain several such lines.

So the values come e.g. as follows:

CXYZQICXYZQICXYZQI... if the format is 0x003F.

The values must be converted as follows to get floating point numbers again:

Position image column: $C [px] = \text{Data value } C / \text{ScaleC}$

Position image line: $R [px] = \text{Counter} + 0.5$

Position world: $X [mm] = \text{Data value } X / \text{ScaleXYZ}$

Position world: $Y [mm] = \text{Data value } Y / \text{ScaleXYZ}$

Position world: $Z [mm] = \text{Data value } Z / \text{ScaleXYZ}$

Goodness: $Q [\%] = \text{Data value } Q / (2^{15}-1) * 100$

Brightness: $I [\%] = \text{Data value } I / (2^{15}-1) * 100$

The same as for the world coordinates applies to the min/max values.

10.20.2 Property Documentation

10.20.2.1 CountLines ushort PF.VsxProtocolDriver.Types.IVsxLineData.CountLines [get]

10.20.2.2 Format ushort PF.VsxProtocolDriver.Types.IVsxLineData.Format [get]

10.20.2.3 Scale ushort PF.VsxProtocolDriver.Types.IVsxLineData.Scale [get]

10.20.2.4 ScaleXYZ `ushort PF.VsxProtocolDriver.Types.IVsxLineData.ScaleXYZ [get]`

10.20.2.5 MinX `short PF.VsxProtocolDriver.Types.IVsxLineData.MinX [get]`

10.20.2.6 MaxX `short PF.VsxProtocolDriver.Types.IVsxLineData.MaxX [get]`

10.20.2.7 MinZ `short PF.VsxProtocolDriver.Types.IVsxLineData.MinZ [get]`

10.20.2.8 MaxZ `short PF.VsxProtocolDriver.Types.IVsxLineData.MaxZ [get]`

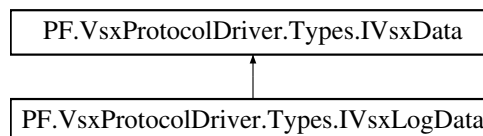
10.20.2.9 FrameCounter `ushort PF.VsxProtocolDriver.Types.IVsxLineData.FrameCounter [get]`

10.20.2.10 Lines `ILineMulti PF.VsxProtocolDriver.Types.IVsxLineData.Lines [get]`

10.21 PF.VsxProtocolDriver.Types.IVsxLogData Interface Reference

A log message received from device.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxLogData:



Properties

- string `Log` [get]

10.21.1 Detailed Description

A log message received from device.

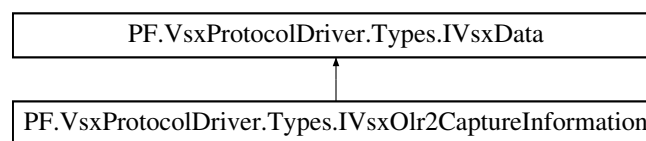
10.21.2 Property Documentation

10.21.2.1 Log `string PF.VsxProtocolDriver.Types.IVsxLogData.Log [get]`

10.22 PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation Interface Reference

Contains information identifying and describing the captured line data of the SpinTop G2 Sensor.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation:



Properties

- `ulong FrameCounter [get]`
Number of captured lines.
- `ulong TriggerCounter [get]`
Number of module triggers (includes missed trigger events).
- `double CurrentPosition [get]`
Current rotational offset between rotating and fixed sensor side, unit=degree.
- `ulong IoState [get]`
Current state of the sensor GPIs.
- `ulong Timestamp [get]`
Monotonic clock counter, unit=μs.
- `uint LmaExposureTime1 [get]`
Main exposure time of laser module A, unit=μs.
- `uint LmaExposureTime2 [get]`
Second exposure time used for HDR of laser module A, unit=μs.
- `uint LmbExposureTime1 [get]`
Main exposure time of laser module B, unit=μs.
- `uint LmbExposureTime2 [get]`
Second exposure time used for HDR of laser module B, unit=μs.
- `ushort LmaRoiOffsetX [get]`
X offset for region of interest of laser module A, unit=px.
- `ushort LmaRoiLengthX [get]`
X length for region of interest of laser module A, unit=px.
- `ushort LmaRoiOffsetZ [get]`
Z offset for region of interest of laser module A, unit=px.
- `ushort LmaRoiLengthZ [get]`
Z length for region of interest of laser module A, unit=px.
- `ushort LmbRoiOffsetX [get]`
X offset for region of interest of laser module B, unit=px.

- ushort **LmbRoiLengthX** [get]
X length for region of interest of laser module B, unit=px.
- ushort **LmbRoiOffsetZ** [get]
Z offset for region of interest of laser module B, unit=px.
- ushort **LmbRoiLengthZ** [get]
Z length for region of interest of laser module B, unit=px.
- ushort **AutoTriggerFrameRate** [get]
Auto trigger frame rate, unit=Hz.
- byte **TriggerSource** [get]
0x0: SoftwareTrigger, 0x2: AutoTrigger.

10.22.1 Detailed Description

Contains information identifying and describing the captured line data of the SpinTop G2 Sensor.

Used abbreviations:

Lma = Laser Module A

Lmb = Laser Module B

10.22.2 Property Documentation

10.22.2.1 FrameCounter `ulong PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.FrameCounter [get]`

Number of captured lines.

10.22.2.2 TriggerCounter `ulong PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.TriggerCounter [get]`

Number of module triggers (includes missed trigger events).

10.22.2.3 CurrentPosition `double PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.CurrentPosition [get]`

Current rotational offset between rotating and fixed sensor side, unit=degree.

10.22.2.4 IoState `ulong PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.IoState [get]`

Current state of the sensor GPIs.

10.22.2.5 Timestamp `ulong PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.Timestamp [get]`

Monotonic clock counter, unit= μ s.

10.22.2.6 LmaExposureTime1 `uint PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaExposureTime1 [get]`

Main exposure time of laser module A, unit= μ s.

10.22.2.7 LmaExposureTime2 `uint PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaExposureTime2 [get]`

Second exposure time used for HDR of laser module A, unit= μ s.

10.22.2.8 LmbExposureTime1 `uint PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbExposureTime1 [get]`

Main exposure time of laser module B, unit= μ s.

10.22.2.9 LmbExposureTime2 `uint PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbExposureTime2 [get]`

Second exposure time used for HDR of laser module B, unit= μ s.

10.22.2.10 LmaRoiOffsetX `ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaRoiOffsetX [get]`

X offset for region of interest of laser module A, unit=px.

10.22.2.11 LmaRoiLengthX `ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaRoiLengthX [get]`

X length for region of interest of laser module A, unit=px.

10.22.2.12 LmaRoiOffsetZ ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaRoiOffsetZ [get]

Z offset for region of interest of laser module A, unit=px.

10.22.2.13 LmaRoiLengthZ ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmaRoiLengthZ [get]

Z length for region of interest of laser module A, unit=px.

10.22.2.14 LmbRoiOffsetX ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbRoiOffsetX [get]

X offset for region of interest of laser module B, unit=px.

10.22.2.15 LmbRoiLengthX ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbRoiLengthX [get]

X length for region of interest of laser module B, unit=px.

10.22.2.16 LmbRoiOffsetZ ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbRoiOffsetZ [get]

Z offset for region of interest of laser module B, unit=px.

10.22.2.17 LmbRoiLengthZ ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.LmbRoiLengthZ [get]

Z length for region of interest of laser module B, unit=px.

10.22.2.18 AutoTriggerFrameRate ushort PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.AutoTriggerFrameRate [get]

Auto trigger frame rate, unit=Hz.

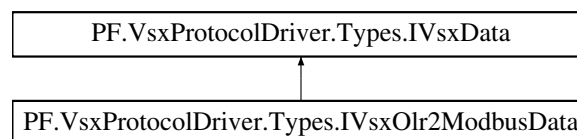
10.22.2.19 TriggerSource `byte PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.Trigger↔Source [get]`

0x0: SoftwareTrigger, 0x2: AutoTrigger.

10.23 PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData Interface Reference

Contains data which was set via modbus protocol.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData:



Properties

- ushort **ActivationTimer** [get]
Number of milliseconds since the last valid set of modbus data was written.
- ushort **CompareBuffer** [get]
CRC16 calculated over all RobotData registers. The Polynom is $0x8005 (x^{16} + x^{15} + x^2 + 1)$. Init value is 0 with no XOR at output.
- ushort **TargetPosition** [get]
Parameterized rotational offset between rotating and fixed sensor side.
- ushort[] **RobotData** [get]
Array with robot data containing the following values:
 RobotData[0]: Robot X Position: lower word.
 RobotData[1]: Robot X Position: upper word.
 RobotData[2]: Robot Y Position: lower word.
 RobotData[3]: Robot Y Position: upper word.
 RobotData[4]: Robot Z Position: lower word.
 RobotData[5]: Robot Z Position: upper word.
 RobotData[6]: Robot rotation around X axis: lower word.
 RobotData[7]: Robot rotation around X axis: upper word.
 RobotData[8]: Robot rotation around Y axis: lower word.
 RobotData[9]: Robot rotation around Y axis: upper word.
 RobotData[10]: Robot rotation around Z axis: lower word.
 RobotData[11]: Robot rotation around Z axis: upper word.
 RobotData[12]: Not specified. For general purpose usage.

10.23.1 Detailed Description

Contains data which was set via modbus protocol.

10.23.2 Property Documentation

10.23.2.1 ActivationTimer `ushort PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData.ActivationTimer [get]`

Number of milliseconds since the last valid set of modbus data was written.

10.23.2.2 CompareBuffer `ushort PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData.CompareBuffer [get]`

CRC16 calculated over all RobotData registers. The Polynom is $0x8005 (x^{16} + x^{15} + x^2 + 1)$. Init value is 0 with no XOR at output.

10.23.2.3 TargetPosition `ushort PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData.TargetPosition [get]`

Parameterized rotational offset between rotating and fixed sensor side.

10.23.2.4 RobotData `ushort [] PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData.RobotData [get]`

Array with robot data containing the following values:

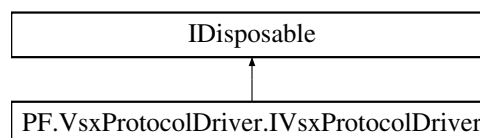
RobotData[0]: Robot X Position: lower word.
 RobotData[1]: Robot X Position: upper word.
 RobotData[2]: Robot Y Position: lower word.
 RobotData[3]: Robot Y Position: upper word.
 RobotData[4]: Robot Z Position: lower word.
 RobotData[5]: Robot Z Position: upper word.
 RobotData[6]: Robot rotation around X axis: lower word.
 RobotData[7]: Robot rotation around X axis: upper word.
 RobotData[8]: Robot rotation around Y axis: lower word.
 RobotData[9]: Robot rotation around Y axis: upper word.
 RobotData[10]: Robot rotation around Z axis: lower word.
 RobotData[11]: Robot rotation around Z axis: upper word.
 RobotData[12]: Not specified. For general purpose usage.

10.24 PF.VsxProtocolDriver.IVsxProtocolDriver Interface Reference

Driver to communicate with a device/sensor by using Asynchronous programming.

Use `IVsxProtocolDriver driver = VsxProtocolDriver.InitTcpDevice(...)` or `IVsxProtocolDriver driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

Inheritance diagram for `PF.VsxProtocolDriver.IVsxProtocolDriver`:



Public Member Functions

- Task<(bool Succ, [IError](#) ErrorDesc)> [Connect](#) (int timeout=VsxProtocolDriver.DefaultEthernetTimeoutMs)
Connect with the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [ConnectAndLogin](#) (string username, string password, int timeout=VsxProtocolDriver.ConnectionTimeoutMs)
Connect with the device and attempt a login on any open connection.
- Task<(bool Succ, [IError](#) ErrorDesc)> [Disconnect](#) ()
Disconnect with the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [ReConnectTcpDevice](#) (string ipAddress, int port=VsxProtocolDriver.Vslexport)
Disconnects the device and reconnects with new connection settings.
- Task<(bool Succ, [IError](#) ErrorDesc)> [ReConnectAndLoginTcpDevice](#) (string ipAddress, string username, string password, int port=VsxProtocolDriver.Vslexport)
Disconnects the device and reconnects with new connection settings. Attempts a login on any reopened connection.
- Task<(bool Succ, [IError](#) ErrorDesc)> [ReConnectSerialDevice](#) (string serialPort, int baudrate, [SerialConnectionType](#) connectionType)
Disconnects the device and reconnects with new connection settings.
- Task<(bool Succ, [IVsxDeviceInformation](#) CurrentDevice, [IError](#) ErrorDesc)> [GetDeviceInformation](#) ()
Returns a list of the complete parameter set of the device including current values.
- Task<(bool Succ, float XmlVersion, Hashtable FeatureList, [IError](#) ErrorDesc)> [GetFeatureList](#) ()
Returns a hashtable with the features from the device and the xml version of the device.
- Task<(bool Succ, List< [IParameter](#) > ParameterList, [IError](#) ErrorDesc)> [GetParameterList](#) ()
Returns a list of the complete parameter set of the device including current values.
- Task<(bool Succ, object? ParameterValue, [IError](#) ErrorDesc)> [GetSingleParameterValue](#) ([IParameter](#) parameter)
Returns the current value of the given parameter from device.
- Task<(bool Succ, object? ParameterValue, [IError](#) ErrorDesc)> [GetSingleParameterValue](#) (ushort settingsVersion, ushort configVersion, string configId, string parameterId)
Returns the current value of the given parameter from device.
- Task<(bool Succ, List< [IParameter](#) >? DependendParameters, [IError](#) ErrorDesc)> [SetSingleParameterValue](#) ([IParameter](#)? parameter, object? value)
Sets the parameter to a value on the device.
- Task<(bool Succ, List< [IParameter](#) >? DependendParameters, [IError](#) ErrorDesc)> [SetSingleParameterValue](#) (ushort settingsVersion, ushort configVersion, string configId, string parameterId, object? value)
Sets the parameter to a value on the device.
- Task<(bool Succ, List< [IParameter](#) >? DependendParameters, [IError](#) ErrorDesc)> [SetMultipleParameterValues](#) (string sourceFileName)
Uploads an incomplete parameter file to the device. From Vsx-Version 4.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SendFirmware](#) (string filename, bool quitAfterFileSent=false)
Sends a firmware update file to the device. NOTE: not completely implemented yet, the file is sent only to the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SetNetworkSettings](#) (string ipAddress, string networkMask, string gateway)
Sends new network settings to the device. After device has accepted, connection to the device will be closed.
- Task<(bool Succ, string Command, string OutputValue, string Status, [IError](#) ErrorDesc)> [SendTestSystemCommand](#) (string command, string inputValue, int timeout=VsxProtocolDriver.DefaultEthernetTimeoutMs)
Sends a test system command to the device (only for internal usage).
- Task<(bool Succ, [IError](#) ErrorDesc)> [DownloadParameterSet](#) (string destinationFileName)
Save the current parameter set to a file.
- Task<(bool Succ, [IError](#) ErrorDesc)> [UploadParameterSet](#) (string sourceFileName)
Uploads a parameter file to the device.
- Task<(bool Succ, List< [IParameter](#) >? DependendParameters, [IError](#) ErrorDesc)> [UploadParameterSet](#) (List< [IParameter](#) >? parameterSet)

Uploads a list of parameters and their values to the device. From Vsx-Version 4.

- Task<(bool Succ, [IError](#) ErrorDesc)> [SaveParameterSetOnDevice](#) ()
Saves the current parameter set on device. Parameter values will be loaded when device starts.
- Task<(bool Succ, List< [IParameter](#) > ParameterList, [IError](#) ErrorDesc)> [LoadParameterSetOnDevice](#) ()
Loads the parameter set saved on device and returns a list of the complete parameter set of the device including current values.
- Task<(bool Succ, List< [IParameter](#) > ParameterList, [IError](#) ErrorDesc)> [LoadDefaultParameterSetOnDevice](#) ()
Resets the devices parameters to factory settings and returns a list of the complete parameter set of the device including current values.
- void [ResetDynamicContainerGrabber](#) (int bufferSize, [Strategy](#) strategy=Strategy.DROP_OLDEST)
Restarts the internal dynamic container grabber. Saving the items will be new initialized.
- Task<(bool Succ, [IVsxDynamicContainer?](#) Container, int NumberOfDiscardedItems, [IError](#) ErrorDesc)> [GetDynamicContainer](#) (int timeoutMs=Timeout.Infinite, CancellationTokenSource? cts=null)
Gets the oldest saved item and removes it internally.
- [IVsxDynamicContainer?](#) [GetCachedContainer](#) (int pos)
Gets a cached dynamic container.
- void [ResetLogMessageGrabber](#) (int bufferSize, int typeMask, [Strategy](#) strategy=Strategy.DROP_OLDEST)
Starts the internal log message grabber. Saving the items will be new initialized.
- Task<(bool Succ, [IVsxLogData?](#) LogMessage, int NumberOfDiscardedItems, [IError](#) ErrorDesc)> [GetLogMessage](#) (int timeoutMs=Timeout.Infinite, CancellationTokenSource? cts=null)
Gets the oldest saved item and removes it internally.
- Task<(bool Succ, [IError](#) ErrorDesc)> [UploadData](#) (string filename)
Sends a data file (either image data or dynamic container data) to the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SendXmlDataMessage](#) (string xml)
Sends a string to the device. NOTE: function does not wait for any device reply.
- Task<(bool Succ, List< [IStatusItem](#) > StatusItems, [IError](#) ErrorDesc)> [GetAllDeviceStatusData](#) ()
Get the full status data set from device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SubscribeToDeviceStatusData](#) ()
Starts cyclic sending of status data from device. Cyclic status data can be received by the [OnDeviceStatusReceived](#) event.
- Task<(bool Succ, [IError](#) ErrorDesc)> [UnsubscribeToDeviceStatusData](#) ()
Stops cyclic sending of status data from device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [Login](#) (string username, string password)
Sends login data to the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SetPassword](#) (string authorizationUsername, string authorizationPassword, string username, string newPassword)
Sets a new password for a device account.
- Task<(bool Succ, [IError](#) ErrorDesc)> [InitializeDevice](#) (string password, [Vsx.DeviceAuthenticationMode](#) authenticationMode)
Initializes a brand-new (factory reset) device supporting multiple authentication modes. This must be done first before device can be used.
- Task<(bool Succ, [IError](#) ErrorDesc)> [Logout](#) ()
Sends a logout request for the current user to the device.
- Task<(bool Succ, [IError](#) ErrorDesc)> [SendSessionKeepAlive](#) ()
Sends a keep alive command to the device. Is needed to reset the timer if the session on the device would otherwise expire. This is reported via a session message with [TimeoutAnnouncement](#).

Properties

- ChannelReader< [IFirmwareState](#) > [FirmwareStateChannelReader](#) [get]
Channel reader for incoming state messages while firmware update is in progress.
- int [WaitTimeout](#) [get, set]
Gets or sets the time in ms, the driver waits for response from device. Default is [VsxDriver.DefaultEthernetTimeoutMs](#).
- int [MissingContainerFramesCounter](#) [get]
Gets the missing frame counter for image grabbing
- int [MissingLogMessagesCounter](#) [get]
Gets the missing log messages counter for log message grabbing
- int [DynamicContainerQueueSize](#) [get]
Gets the current size of the dynamic container message queue
- int [NumberOfCachedContainers](#) [get]
Gets the current number of cached container messages.
- int [LogMessageQueueSize](#) [get]
Gets the current size of the log message queue
- [Strategy ContainerGrabberStrategy](#) [get]
Gets the current strategy of the dynamic container message queue
- [Strategy LogGrabberStrategy](#) [get]
Gets the current strategy of the log message queue
- bool [Connected](#) [get]
Indicates current connection state with the device.

Events

- Action< string, [DisconnectEvent](#), string > [OnDisconnect](#)
Event called when a TPC/IP device is disconnected.
- Action< [DeviceStatusScope](#), List< [IStatusItem](#) > > [OnDeviceStatusReceived](#)
Event called when status data is received from device after cyclic sending is started (see [SubscribeToDeviceStatusData](#)).
- Action< [IVsxSessionData](#) > [OnSessionMessageReceived](#)
Event called when session data is received from device which means that something changed with the login status.

10.24.1 Detailed Description

Driver to communicate with a device/sensor by using Asynchronous programming.

Use [IVsxProtocolDriver](#) driver = VsxDriver.InitTcpDevice(...) or
[IVsxProtocolDriver](#) driver = VsxDriver.InitSerialDevice(...) to get an instance and use the driver.

10.24.2 Member Function Documentation

10.24.2.1 Connect() Task<(bool Succ, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.Connect (
int timeout = VsxDriver.DefaultEthernetTimeoutMs)

Connect with the device.

Parameters

<i>timeout</i>	The timeout for a connection attempt, default is VsxProtocolDriver.DefaultEthernetTimeoutMs.
----------------	--

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.2 ConnectAndLogin() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.ConnectAndLogin (
 string username,
 string password,
 int timeout = VsxProtocolDriver.ConnectionTimeoutMs)

Connect with the device and attempt a login on any open connection.

Parameters

<i>username</i>	The login username.
<i>password</i>	The login password.
<i>timeout</i>	The timeout for a connection attempt.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.3 Disconnect() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.Disconnect ()

Disconnect with the device.

Returns

Always true and empty error.

10.24.2.4 ReConnectTcpDevice() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.ReConnectTcpDevice (
 string ipAddress,
 int port = VsxProtocolDriver.Vsxport)

Disconnects the device and reconnects with new connection settings.

Parameters

<i>ipAddress</i>	The new IPAddress.
<i>port</i>	The new port.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.5 ReConnectAndLoginTcpDevice() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocol↔
Driver.IVsxProtocolDriver.ReConnectAndLoginTcpDevice (

```

    string ipAddress,
    string username,
    string password,
    int port = VsxProtocolDriver.Vsxport )

```

Disconnects the device and reconnects with new connection settings. Attempts a login on any reopened connection.

Parameters

<i>ipAddress</i>	The new IPAddress.
<i>username</i>	The login username.
<i>password</i>	The login password.
<i>port</i>	The new port.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.6 ReConnectSerialDevice() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.↔
IVsxProtocolDriver.ReConnectSerialDevice (

```

    string serialPort,
    int baudrate,
    SerialConnectionType connectionType )

```

Disconnects the device and reconnects with new connection settings.

Parameters

<i>serialPort</i>	The new serial port.
<i>baudrate</i>	The new baudrate.
<i>connectionType</i>	The new connection type.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.7 GetDeviceInformation() Task<(bool Succ, IVsxDeviceInformation CurrentDevice, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetDeviceInformation ()

Returns a list of the complete parameter set of the device including current values.

Returns

(bool Succ, IVsxDeviceInformation CurrentDevice, IError ErrorDesc) true/device/empty error or false/empty device/error description.

10.24.2.8 GetFeatureList() Task<(bool Succ, float XmlVersion, Hashtable FeatureList, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetFeatureList ()

Returns a hashtable with the features from the device and the xml version of the device.

Returns

(bool Succ, float XmlVersion, Hashtable FeatureList, IError ErrorDesc) true/xml version/feature list/empty error or false/0/empty hashtable/error description.

10.24.2.9 GetParameterList() Task<(bool Succ, List< IParameter > ParameterList, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetParameterList ()

Returns a list of the complete parameter set of the device including current values.

Returns

(bool Succ, List(Parameter) ParameterList, IError ErrorDesc) true/parameter list/empty error or false/empty list/error description.

10.24.2.10 GetSingleParameterValue() [1/2] Task<(bool Succ, object? ParameterValue, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetSingleParameterValue (
 IParameter parameter)

Returns the current value of the given parameter from device.

Parameters

<i>parameter</i>	The parameter its value is asked for.
------------------	---------------------------------------

Returns

(bool Succ, object ParameterValue, IError ErrorDesc) true/the value with type parameters value type/empty error or false/0/error description.

10.24.2.11 GetSingleParameterValue() [2/2] Task<(bool Succ, object? ParameterValue, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetSingleParameterValue (
 ushort *settingsVersion*,
 ushort *configVersion*,
 string *configId*,
 string *parameterId*)

Returns the current value of the given parameter from device.

Parameters

<i>settingsVersion</i>	The settings version of the parameter its value is asked for.
<i>configVersion</i>	The config version of the parameter its value is asked for.
<i>configId</i>	The config id of the parameter its value is asked for.
<i>parameterId</i>	The id of the parameter its value is asked for.

Returns

(bool Succ, object ParameterValue, IError ErrorDesc) true/the value with type parameters value type/empty error or false/0/error description.

10.24.2.12 SetSingleParameterValue() [1/2] Task<(bool Succ, List< IParameter >? DependendParameters, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SetSingleParameterValue (
 IParameter? *parameter*,
 object? *value*)

Sets the parameter to a value on the device.

Parameters

<i>parameter</i>	The parameter the value should be set from.
<i>value</i>	The new value.

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

10.24.2.13 SetSingleParameterValue() [2/2] Task<(bool Succ, List< IPParameter >? DependendParameters, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SetSingleParameterValue (

```

    ushort settingsVersion,
    ushort configVersion,
    string configId,
    string parameterId,
    object? value )

```

Sets the parameter to a value on the device.

Parameters

<i>settingsVersion</i>	The settings version of the parameter which should be set.
<i>configVersion</i>	The config version of the parameter which should be set.
<i>configId</i>	The config id of the parameter which should be set.
<i>parameterId</i>	The id of the parameter which should be set.
<i>value</i>	The new value.

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

10.24.2.14 SetMultipleParameterValues() Task<(bool Succ, List< IPParameter >? DependendParameters, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SetMultipleParameterValues (

```

    string sourceFileName )

```

Uploads an incomplete parameter file to the device. From Vsx-Version 4.

Parameters

<i>sourceFileName</i>	Path and filename to upload.
-----------------------	------------------------------

///

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

10.24.2.15 SendFirmware() Task<(bool Succ, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SendFirmware (

```

    string filename,
    bool quitAfterFileSent = false )

```

Sends a firmware update file to the device. NOTE: not completely implemented yet, the file is sent only to the device.

Parameters

<i>filename</i>	The path and filename of the firmware file.
<i>quitAfterFileSent</i>	If set to true, method quits after file is sent to device and update runs automatically on device.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.16 SetNetworkSettings() Task<(bool Succ, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SetNetworkSettings (

```

    string ipAddress,
    string networkMask,
    string gateway )

```

Sends new network settings to the device. After device has accepted, connection to the device will be closed.

Parameters

<i>ipAddress</i>	The new IP Address.
<i>networkMask</i>	The new network mask.
<i>gateway</i>	The new gateway.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.17 SendTestSystemCommand() Task<(bool Succ, string Command, string OutputValue, string Status, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SendTestSystemCommand (

```

    string command,
    string inputValue,
    int timeout = VsxProtocolDriver.DefaultEthernetTimeoutMs )

```

Sends a test system command to the device (only for internal usage).

Parameters

<i>command</i>	The test system command.
<i>inputValue</i>	
<i>timeout</i>	Wait time for device reply.

Returns

10.24.2.18 DownloadParameterSet() Task<(bool Succ, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.
IVsxProtocolDriver.DownloadParameterSet (
 string destinationFileName)

Save the current parameter set to a file.

Parameters

<i>destinationFileName</i>	Path and file name to save to.
----------------------------	--------------------------------

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.19 UploadParameterSet() [1/2] Task<(bool Succ, [IError](#) ErrorDesc)> PF.VsxProtocol
Driver.IVsxProtocolDriver.UploadParameterSet (
 string sourceFileName)

Uploads a parameter file to the device.

Parameters

<i>sourceFileName</i>	Path and filename to upload.
-----------------------	------------------------------

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.20 UploadParameterSet() [2/2] Task<(bool Succ, List< [IParameter](#) >? DependendParameters, [IError](#) ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.UploadParameterSet (
 List< [IParameter](#) >? parameterSet)

Uploads a list of parameters and their values to the device. From Vsx-Version 4.

Parameters

<i>parameterSet</i>	A list of parameter objects.
---------------------	------------------------------

///

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

10.24.2.21 SaveParameterSetOnDevice() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SaveParameterSetOnDevice ()

Saves the current parameter set on device. Parameter values will be loaded when device starts.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.24.2.22 LoadParameterSetOnDevice() Task<(bool Succ, List< IPParameter > ParameterList, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.LoadParameterSetOnDevice ()

Loads the parameter set saved on device and returns a list of the complete parameter set of the device including current values.

Returns

(bool Succ, List(Parameter) ParameterList, IError ErrorDesc) true/parameter list/empty error or false/empty list/error description.

10.24.2.23 LoadDefaultParameterSetOnDevice() Task<(bool Succ, List< IPParameter > ParameterList, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.LoadDefaultParameterSetOnDevice ()

Resets the devices parameters to factory settings and returns a list of the complete parameter set of the device including current values.

Returns

(bool Succ, List(Parameter) ParameterList, IError ErrorDesc) true/parameter list/empty error or false/empty list/error description.

10.24.2.24 ResetDynamicContainerGrabber() void PF.VsxProtocolDriver.IVsxProtocolDriver.ResetDynamicContainerGrabber (
 int bufferSize,
 Strategy strategy = Strategy.DROP_OLDEST)

Restarts the internal dynamic container grabber. Saving the items will be new initialized.

Parameters

<i>bufferSize</i>	The maximum number of items which will be internally saved, if less than 0, number is infinity.
<i>strategy</i>	The strategy, which items will be discarded if maximum number of items is reached.

10.24.2.25 GetDynamicContainer() Task<(bool Succ, IVsxDynamicContainer? Container, int NumberOfDiscardedItems, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetDynamicContainer (

```

    int timeoutMs = Timeout.Infinite,
    CancellationTokenSource? cts = null )

```

Gets the oldest saved item and removes it internally.

Parameters

<i>timeoutMs</i>	The maximum time in ms to try reading an item.
<i>cts</i>	A cancellation token source to cancel operation earlier than timeoutMs.

Returns

(bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, IError ErrorDesc) true/item/no of discarded items until now/empty error or false/empty container/no of discarded items until now/error description.

10.24.2.26 GetCachedContainer() IVsxDynamicContainer? PF.VsxProtocolDriver.IVsxProtocolDriver.GetCachedContainer (

```

    int pos )

```

Gets a cached dynamic container.

Parameters

<i>pos</i>	Position of the container in cache.
------------	-------------------------------------

Returns

10.24.2.27 ResetLogMessageGrabber() void PF.VsxProtocolDriver.IVsxProtocolDriver.ResetLogMessageGrabber (

```

    int bufferSize,
    int typeMask,
    Strategy strategy = Strategy.DROP_OLDEST )

```

Starts the internal log message grabber. Saving the items will be new initialized.

Parameters

<i>bufferSize</i>	The maximum number of items which will be internally saved, if less than 0, number is infinity.
<i>typeMask</i>	Mask which log message types will be sent by device.
<i>strategy</i>	The strategy, which items will be discarded if maximum number of items is reached.

10.24.2.28 GetLogMessage() Task<(bool Succ, IVsxLogData? LogMessage, int NumberOfDiscardedItems, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetLogMessage (
 int timeoutMs = Timeout.Infinite,
 CancellationTokenSource? cts = null)

Gets the oldest saved item and removes it internally.

Parameters

<i>timeoutMs</i>	The maximum time in ms to try reading an item.
<i>cts</i>	A cancellation token source to cancel operation earlier than timeoutMs.

Returns

(bool Succ, IVsxLogData LogMessage, int NumberOfDiscardedItems, IError ErrorDesc) true/item/no of discarded items until now/empty error or false/empty log message/no of discarded items until now/error description.

10.24.2.29 UploadData() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.UploadData (
 string filename)

Sends a data file (either image data or dynamic container data) to the device.

Parameters

<i>filename</i>	The path and filename of the data file.
-----------------	---

Returns

(bool Succ, IError ErrorDesc) true/empty error if upload was successful or false/error description.

10.24.2.30 SendXmlDataMessage() Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SendXmlDataMessage (
 string xml)

Sends a string to the device. NOTE: function does not wait for any device reply.

Parameters

<i>xml</i>	The message.
------------	--------------

Returns

Always true and empty error.

10.24.2.31 GetAllDeviceStatusData() `Task<(bool Succ, List< IStatusItem > StatusItems, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetAllDeviceStatusData ()`

Get the full status data set from device.

Returns

true/status items list/empty error or false/empty list/error description.

10.24.2.32 SubscribeToDeviceStatusData() `Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocol↔Driver.IVsxProtocolDriver.SubscribeToDeviceStatusData ()`

Starts cyclic sending of status data from device. Cyclic status data can be received by the [OnDeviceStatusReceived](#) event.

Returns

true/status items list/empty error or false/empty list/error description.

10.24.2.33 UnsubscribeToDeviceStatusData() `Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocol↔Driver.IVsxProtocolDriver.UnsubscribeToDeviceStatusData ()`

Stops cyclic sending of status data from device.

Returns

true/empty error or false/error description.

10.24.2.34 Login() `Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocol↔Driver.Login (`
 `string username,`
 `string password)`

Sends login data to the device.

Parameters

<i>username</i>	The username.
<i>password</i>	The password.

Returns

true/empty error or false/error description.

10.24.2.35 SetPassword()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SetPassword (
    string authorizationUsername,
    string authorizationPassword,
    string username,
    string newPassword )
```

Sets a new password for a device account.

Parameters

<i>authorizationUsername</i>	The username who wants to set the password.
<i>authorizationPassword</i>	The password of the user who wants to set the password.
<i>username</i>	The username of the account for which the new password should be set.
<i>newPassword</i>	The new password.

Returns

true/empty error or false/error description.

10.24.2.36 InitializeDevice()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.InitializeDevice (
    string password,
    Vsx.DeviceAuthenticationMode authenticationMode )
```

Initializes a brand-new (factory reset) device supporting multiple authentication modes. This must be done first before device can be used.

Parameters

<i>password</i>	A new admin password.
<i>authenticationMode</i>	The desired authentication mode on device.

Returns

true/empty error or false/error description.

10.24.2.37 Logout() `Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.
Driver.Logout ()`

Sends a logout request for the current user to the device.

Returns

true/empty error or false/error description.

10.24.2.38 SendSessionKeepAlive() `Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.
SendSessionKeepAlive ()`

Sends a keep alive command to the device. Is needed to reset the timer if the session on the device would otherwise expire. This is reported via a session message with TimeoutAnnouncement.

Returns

true/empty error or false/error description.

10.24.3 Property Documentation

10.24.3.1 FirmwareStateChannelReader `ChannelReader<IFirmwareState> PF.VsxProtocolDriver.IVsxProtocolDriver.
FirmwareStateChannelReader [get]`

Channel reader for incoming state messages while firmware update is in progress.

10.24.3.2 WaitTimeout `int PF.VsxProtocolDriver.IVsxProtocolDriver.WaitTimeout [get], [set]`

Gets or sets the time in ms, the driver waits for response from device. Default is VsxProtocolDriver.DefaultEthernetTimeoutMs.

10.24.3.3 MissingContainerFramesCounter `int PF.VsxProtocolDriver.IVsxProtocolDriver.MissingContainerFramesCounter [get]`

Gets the missing frame counter for image grabbing

10.24.3.4 MissingLogMessagesCounter `int PF.VsxProtocolDriver.IVsxProtocolDriver.MissingLogMessagesCounter [get]`

Gets the missing log messages counter for log message grabbing

10.24.3.5 DynamicContainerQueueSize `int PF.VsxProtocolDriver.IVsxProtocolDriver.DynamicContainerQueueSize [get]`

Gets the current size of the dynamic container message queue

10.24.3.6 NumberOfCachedContainers `int PF.VsxProtocolDriver.IVsxProtocolDriver.NumberOfCachedContainers [get]`

Gets the current number of cached container messages.

10.24.3.7 LogMessageQueueSize `int PF.VsxProtocolDriver.IVsxProtocolDriver.LogMessageQueueSize [get]`

Gets the current size of the log message queue

10.24.3.8 ContainerGrabberStrategy `Strategy PF.VsxProtocolDriver.IVsxProtocolDriver.ContainerGrabberStrategy [get]`

Gets the current strategy of the dynamic container message queue

10.24.3.9 LogGrabberStrategy `Strategy PF.VsxProtocolDriver.IVsxProtocolDriver.LogGrabberStrategy [get]`

Gets the current strategy of the log message queue

10.24.3.10 Connected `bool PF.VsxProtocolDriver.IVsxProtocolDriver.Connected [get]`

Indicates current connection state with the device.

10.24.4 Event Documentation

10.24.4.1 OnDisconnect `Action<string, DisconnectEvent, string> PF.VsxProtocolDriver.IVsxProtocolDriver.OnDisconnect`

Event called when a TPC/IP device is disconnected.

CurrentIpAddress	Actual ip address.
DisconnectEvent	Enumeration of disconnect reason
ErrorMessage	Error message as string

Returns

string with ip of the disconnected device and a descriptions why device was disconnected.

10.24.4.2 OnDeviceStatusReceived `Action<DeviceStatusScope, List<IStatusItem> > PF.VsxProtocolDriver.IVsxProtocolDriver.OnDeviceStatusReceived`

Event called when status data is received from device after cyclic sending is started (see [SubscribeToDeviceStatusData](#)).

Returns

Device status scope whether device sends full or multi status data and the status data list.

DeviceStatusScope	The library or executable built from a compilation.
List<IStatusItem>	List of status items returned

10.24.4.3 OnSessionMessageReceived `Action<IVsxSessionData> PF.VsxProtocolDriver.IVsxProtocolDriver.OnSessionMessageReceived`

Event called when session data is received from device which means that something changed with the login status.

Returns

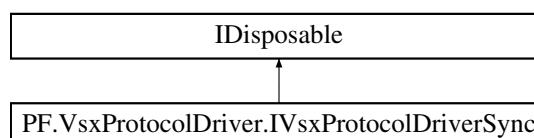
A session status message.

10.25 PF.VsxProtocolDriver.IVsxProtocolDriverSync Interface Reference

Driver to communicate with a device/sensor by using synchronous programming.

Use `IVsxProtocolDriverSync driver = VsxProtocolDriver.InitTcpDevice(...)` or `IVsxProtocolDriverSync driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

Inheritance diagram for `PF.VsxProtocolDriver.IVsxProtocolDriverSync`:



Public Member Functions

- bool [IError](#) ErrorDesc [Connect](#) (int timeout=VsxProtocolDriver.DefaultEthernetTimeoutMs)
- bool [IError](#) ErrorDesc [ConnectAndLogin](#) (string username, string password, int timeout=VsxProtocolDriver.↵ ConnectionTimeoutMs)
- bool [IError](#) ErrorDesc [Disconnect](#) ()
- bool [IError](#) ErrorDesc [ReConnectTcpDevice](#) (string ipAddress, int port=VsxProtocolDriver.Vsxport)
- bool [IError](#) ErrorDesc [ReConnectAndLoginTcpDevice](#) (string ipAddress, string username, string password, int port=VsxProtocolDriver.Vsxport)
- bool [IError](#) ErrorDesc [ReConnectSerialDevice](#) (string serialPort, int baudrate, [SerialConnectionType](#) connectionType)
- bool [IVsxDeviceInformation](#) [IError](#) ErrorDesc [GetDeviceInformation](#) ()
- bool float Hashtable [IError](#) ErrorDesc [GetFeatureList](#) ()
- bool List< [IParameter](#) > [IError](#) ErrorDesc [GetParameterList](#) ()
- bool? object [IError](#) ErrorDesc [GetSingleParameterValue](#) ([IParameter](#) parameter)
- bool? object [IError](#) ErrorDesc [GetSingleParameterValue](#) (ushort settingsVersion, ushort configVersion, string configId, string parameterId)
- bool? List< [IParameter](#) > [IError](#) ErrorDesc [SetSingleParameterValue](#) ([IParameter](#)? parameter, object? value)
- bool? List< [IParameter](#) > [IError](#) ErrorDesc [SetSingleParameterValue](#) (ushort settingsVersion, ushort configVersion, string configId, string parameterId, object? value)
- bool? List< [IParameter](#) > [IError](#) ErrorDesc [SetMultipleParameterValues](#) (string sourceFileName)
- bool [IError](#) ErrorDesc [SendFirmware](#) (string filename, bool quitAfterFileSent=false)
- bool [IError](#) ErrorDesc [SetNetworkSettings](#) (string ipAddress, string networkMask, string gateway)
- bool string string string [IError](#) ErrorDesc [SendTestSystemCommand](#) (string command, string inputValue, int timeout=VsxProtocolDriver.DefaultEthernetTimeoutMs)
- bool [IError](#) ErrorDesc [DownloadParameterSet](#) (string destinationFileName)
- bool [IError](#) ErrorDesc [UploadParameterSet](#) (string sourceFileName)
- bool? List< [IParameter](#) > [IError](#) ErrorDesc [UploadParameterSet](#) (List< [IParameter](#) >? parameterSet)
- bool [IError](#) ErrorDesc [SaveParameterSetOnDevice](#) ()
- bool List< [IParameter](#) > [IError](#) ErrorDesc [LoadParameterSetOnDevice](#) ()
- bool List< [IParameter](#) > [IError](#) ErrorDesc [LoadDefaultParameterSetOnDevice](#) ()
- void [ResetDynamicContainerGrabber](#) (int bufferSize, [Strategy](#) strategy=Strategy.DROP_OLDEST)
Restarts the internal dynamic container grabber. Saving the items will be new initialized.
- bool? [IVsxDynamicContainer](#) int [IError](#) ErrorDesc [GetDynamicContainer](#) (int timeoutMs=Timeout.Infinite, CancellationTokentSource? cts=null)
- [IVsxDynamicContainer](#)? [GetCachedContainer](#) (int pos)
Gets a cached dynamic container.
- void [ResetLogMessageGrabber](#) (int bufferSize, int typeMask, [Strategy](#) strategy=Strategy.DROP_OLDEST)
Starts the internal log message grabber. Saving the items will be new initialized.
- bool? [IVsxLogData](#) int [IError](#) ErrorDesc [GetLogMessage](#) (int timeoutMs=Timeout.Infinite, Cancellation↵ TokentSource? cts=null)
- bool [IError](#) ErrorDesc [UploadData](#) (string filename)
- bool [IError](#) ErrorDesc [SendXmlDataMessage](#) (string xml)
- bool List< [IStatusItem](#) > [IError](#) ErrorDesc [GetAllDeviceStatusData](#) ()
- bool [IError](#) ErrorDesc [SubscribeToDeviceStatusData](#) ()
- bool [IError](#) ErrorDesc [UnsubscribeToDeviceStatusData](#) ()
- bool [IError](#) ErrorDesc [Login](#) (string username, string password)
- bool [IError](#) ErrorDesc [SetPassword](#) (string authorizationUsername, string authorizationPassword, string user-name, string newPassword)
- bool [IError](#) ErrorDesc [InitializeDevice](#) (string password, [Vsx.DeviceAuthenticationMode](#) authenticationMode)
- bool [IError](#) ErrorDesc [Logout](#) ()
- bool [IError](#) ErrorDesc [SendSessionKeepAlive](#) ()

Public Attributes

- bool [Succ](#)
Connect with the device.
- bool [IVsxDeviceInformation CurrentDevice](#)
- bool float [XmlVersion](#)
- bool float Hashtable [FeatureList](#)
- bool List< [IParameter](#) > [ParameterList](#)
- bool? object [ParameterValue](#)
- bool? List< [IParameter](#) > [DependendParameters](#)
- bool string [Command](#)
- bool string string [OutputValue](#)
- bool string string string [Status](#)
- bool? [IVsxDynamicContainer Container](#)
- bool? [IVsxDynamicContainer](#) int [NumberOfDiscardedItems](#)
- bool? [IVsxLogData LogMessage](#)
- bool? [IVsxLogData](#) int [NumberOfDiscardedItems](#)
- bool List< [IStatusItem](#) > [StatusItems](#)

Properties

- ChannelReader< [IFirmwareState](#) > [FirmwareStateChannelReader](#) [get]
Channel reader for incoming state messages while firmware update is in progress.
- int [WaitTimeout](#) [get, set]
Gets or sets the time in ms, the driver waits for response from device. Default is VsxProtocolDriver.DefaultEthernetTimeoutMs.
- int [MissingContainerFramesCounter](#) [get]
Gets the missing frame counter for image grabbing
- int [MissingLogMessagesCounter](#) [get]
Gets the missing log messages counter for log message grabbing
- int [DynamicContainerQueueSize](#) [get]
Gets the current size of the dynamic container message queue
- int [NumberOfCachedContainers](#) [get]
Gets the current number of cached container messages.
- int [LogMessageQueueSize](#) [get]
Gets the current size of the log message queue
- [Strategy ContainerGrabberStrategy](#) [get]
Gets the current strategy of the dynamic container message queue
- [Strategy LogGrabberStrategy](#) [get]
Gets the current strategy of the log message queue
- bool [Connected](#) [get]
Indicates current connection state with the device.

Events

- Action< string, [DisconnectEvent](#), string > [OnDisconnect](#)
Event called when a TPC/IP device is disconnected.
- Action< [DeviceStatusScope](#), List< [IStatusItem](#) > > [OnDeviceStatusReceived](#)
Event called when status data is received from device after cyclic sending is started (see [SubscribeToDeviceStatusData](#)).
- Action< [IVsxSessionData](#) > [OnSessionMessageReceived](#)
Event called when session data is received from device which means that something changed with the login status.

10.25.1 Detailed Description

Driver to communicate with a device/sensor by using synchronous programming.

Use `IVsxProtocolDriverSync driver = VsxProtocolDriver.InitTcpDevice(...)` or `IVsxProtocolDriverSync driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

10.25.2 Member Function Documentation

10.25.2.1 Connect() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Connect (`
`int timeout = VsxProtocolDriver.DefaultEthernetTimeoutMs)`

10.25.2.2 ConnectAndLogin() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriver↔`
`Sync.ConnectAndLogin (`
`string username,`
`string password,`
`int timeout = VsxProtocolDriver.ConnectionTimeoutMs)`

10.25.2.3 Disconnect() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.↔`
`Disconnect ()`

10.25.2.4 ReConnectTcpDevice() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriver↔`
`Sync.ReConnectTcpDevice (`
`string ipAddress,`
`int port = VsxProtocolDriver.Vsxport)`

10.25.2.5 ReConnectAndLoginTcpDevice() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsx↔`
`ProtocolDriverSync.ReConnectAndLoginTcpDevice (`
`string ipAddress,`
`string username,`
`string password,`
`int port = VsxProtocolDriver.Vsxport)`

10.25.2.6 ReConnectSerialDevice() bool **IErr**or Desc PF.VsxProtocolDriver.IVsxProtocol↔
DriverSync.ReConnectSerialDevice (
 string serialPort,
 int baudrate,
 SerialConnectionType connectionType)

10.25.2.7 GetDeviceInformation() bool **IVsxDeviceInformation** **IErr**or Desc PF.VsxProtocol↔
Driver.IVsxProtocolDriverSync.GetDeviceInformation ()

10.25.2.8 GetFeatureList() bool float Hashtable **IErr**or Desc PF.VsxProtocolDriver.IVsx↔
ProtocolDriverSync.GetFeatureList ()

10.25.2.9 GetParameterList() bool List< **IP**arameter > **IErr**or Desc PF.VsxProtocolDriver.↔
IVsxProtocolDriverSync.GetParameterList ()

10.25.2.10 GetSingleParameterValue() [1/2] bool? object **IErr**or Desc PF.VsxProtocol↔
Driver.IVsxProtocolDriverSync.GetSingleParameterValue (
 IParameter parameter)

10.25.2.11 GetSingleParameterValue() [2/2] bool? object **IErr**or Desc PF.VsxProtocol↔
Driver.IVsxProtocolDriverSync.GetSingleParameterValue (
 ushort settingsVersion,
 ushort configVersion,
 string configId,
 string parameterId)

10.25.2.12 SetSingleParameterValue() [1/2] bool? List< **IP**arameter > **IErr**or Desc PF.Vsx↔
ProtocolDriver.IVsxProtocolDriverSync.SetSingleParameterValue (
 IParameter? parameter,
 object? value)

10.25.2.13 SetSingleParameterValue() [2/2] bool? List< **IP**arameter > **IErr**or Desc PF.Vsx↔
ProtocolDriver.IVsxProtocolDriverSync.SetSingleParameterValue (
 ushort settingsVersion,
 ushort configVersion,
 string configId,
 string parameterId,
 object? value)

10.25.2.14 SetMultipleParameterValues() bool? List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SetMultipleParameterValues (

```

    string sourceFileName )

```

10.25.2.15 SendFirmware() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SendFirmware (

```

    string filename,
    bool quitAfterFileSent = false )

```

10.25.2.16 SetNetworkSettings() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SetNetworkSettings (

```

    string ipAddress,
    string networkMask,
    string gateway )

```

10.25.2.17 SendTestSystemCommand() bool string string string IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SendTestSystemCommand (

```

    string command,
    string inputValue,
    int timeout = VsxProtocolDriver.DefaultEthernetTimeoutMs )

```

10.25.2.18 DownloadParameterSet() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.DownloadParameterSet (

```

    string destinationFileName )

```

10.25.2.19 UploadParameterSet() [1/2] bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UploadParameterSet (

```

    string sourceFileName )

```

10.25.2.20 UploadParameterSet() [2/2] bool? List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UploadParameterSet (

```

    List< IParameter >? parameterSet )

```

10.25.2.21 SaveParameterSetOnDevice() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SaveParameterSetOnDevice ()

10.25.2.22 LoadParameterSetOnDevice() `bool List< IParameter > IError ErrorDesc PF.VsxProtocol↔
Driver.IVsxProtocolDriverSync.LoadParameterSetOnDevice ()`

10.25.2.23 LoadDefaultParameterSetOnDevice() `bool List< IParameter > IError ErrorDesc PF.Vsx↔
ProtocolDriver.IVsxProtocolDriverSync.LoadDefaultParameterSetOnDevice ()`

10.25.2.24 ResetDynamicContainerGrabber() `void PF.VsxProtocolDriver.IVsxProtocolDriverSync.↔
ResetDynamicContainerGrabber (`
 `int bufferSize,`
 `Strategy strategy = Strategy.DROP_OLDEST)`

Restarts the internal dynamic container grabber. Saving the items will be new initialized.

Parameters

<i>bufferSize</i>	The maximum number of items which will be internally saved, if less than 0, number is infinity.
<i>strategy</i>	The strategy, which items will be discarded if maximum number of items is reached.

10.25.2.25 GetDynamicContainer() `bool? IVsxDynamicContainer int IError ErrorDesc PF.Vsx↔
ProtocolDriver.IVsxProtocolDriverSync.GetDynamicContainer (`
 `int timeoutMs = Timeout.Infinite,`
 `CancellationTokenSource? cts = null)`

10.25.2.26 GetCachedContainer() `IVsxDynamicContainer? PF.VsxProtocolDriver.IVsxProtocol↔
DriverSync.GetCachedContainer (`
 `int pos)`

Gets a cached dynamic container.

Parameters

<i>pos</i>	Position of the container in cache.
------------	-------------------------------------

Returns

10.25.2.27 ResetLogMessageGrabber() void PF.VsxProtocolDriver.IVsxProtocolDriverSync.ResetLogMessageGrabber (
 int bufferSize,
 int typeMask,
 Strategy strategy = Strategy.DROP_OLDEST)

Starts the internal log message grabber. Saving the items will be new initialized.

Parameters

<i>bufferSize</i>	The maximum number of items which will be internally saved, if less than 0, number is infinity.
<i>typeMask</i>	Mask which log message types will be sent by device.
<i>strategy</i>	The strategy, which items will be discarded if maximum number of items is reached.

10.25.2.28 GetLogMessage() bool? IVsxLogData int IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetLogMessage (
 int timeoutMs = Timeout.Infinite,
 CancellationTokenSource? cts = null)

10.25.2.29 UploadData() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UploadData (
 string filename)

10.25.2.30 SendXmlDataMessage() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SendXmlDataMessage (
 string xml)

10.25.2.31 GetAllDeviceStatusData() bool List< IStatusItem > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetAllDeviceStatusData ()

10.25.2.32 SubscribeToDeviceStatusData() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SubscribeToDeviceStatusData ()

10.25.2.33 UnsubscribeToDeviceStatusData() bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UnsubscribeToDeviceStatusData ()

10.25.2.34 Login() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Login (`
 `string username,`
 `string password)`

10.25.2.35 SetPassword() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.↵`
`SetPassword (`
 `string authorizationUsername,`
 `string authorizationPassword,`
 `string username,`
 `string newPassword)`

10.25.2.36 InitializeDevice() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriver↵`
`Sync.InitializeDevice (`
 `string password,`
 `Vsx.DeviceAuthenticationMode authenticationMode)`

10.25.2.37 Logout() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Logout (`
 `)`

10.25.2.38 SendSessionKeepAlive() `bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocol↵`
`DriverSync.SendSessionKeepAlive ()`

10.25.3 Member Data Documentation

10.25.3.1 Succ `bool PF.VsxProtocolDriver.IVsxProtocolDriverSync.Succ`

Connect with the device.

Sends a keep alive command to the device. Is needed to reset the timer if the session on the device would otherwise expire. This is reported via a session message with TimeoutAnnouncement.

Sends a logout request for the current user to the device.

Initializes a brand-new (factory reset) device supporting multiple authentication modes. This must be done first before device can be used.

Sets a new password for a device account.

Sends login data to the device.

Stops cyclic sending of status data from device.

Starts cyclic sending of status data from device. Cyclic status data can be received by the [OnDeviceStatusReceived](#) event.

Get the full status data set from device.

Sends a string to the device. NOTE: function does not wait for any device reply.

Sends a data file (either image data or dynamic container data) to the device.

Gets the oldest saved item and removes it internally.

Resets the devices parameters to factory settings and returns a list of the complete parameter set of the device including current values.

Loads the parameter set saved on device and returns a list of the complete parameter set of the device including current values.

Saves the current parameter set on device. Parameter values will be loaded when device starts.

Uploads a list of parameters and their values to the device. From Vsx-Version 4.

Uploads a parameter file to the device.

Save the current parameter set to a file.

Sends a test system command to the device (only for internal usage).

Sends new network settings to the device. After device has accepted, connection to the device will be closed.

Sends a firmware update file to the device. NOTE: not completely implemented yet, the file is sent only to the device.

Uploads an incomplete parameter file to the device. From Vsx-Version 4.

Sets the parameter to a value on the device.

Returns the current value of the given parameter from device.

Returns a hashtable with the features from the device and the xml version of the device.

Returns a list of the complete parameter set of the device including current values.

Disconnects the device and reconnects with new connection settings. Attempts a login on any reopened connection.

Disconnects the device and reconnects with new connection settings.

Disconnect with the device.

Connect with the device and attempt a login on any open connection.

Parameters

<i>timeout</i>	The timeout for a connection attempt, default is VsxProtocolDriver.DefaultEthernetTimeoutMs.
----------------	--

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>username</i>	The login username.
<i>password</i>	The login password.
<i>timeout</i>	The timeout for a connection attempt.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Returns

Always true and empty error.

Parameters

<i>ipAddress</i>	The new IPAddress.
<i>port</i>	The new port.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>ipAddress</i>	The new IPAddress.
<i>username</i>	The login username.
<i>password</i>	The login password.
<i>port</i>	The new port.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>serialPort</i>	The new serial port.
<i>baudrate</i>	The new baudrate.
<i>connectionType</i>	The new connection type.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Returns

(bool Succ, IVsxDeviceInformation CurrentDevice, IError ErrorDesc) true/device/empty error or false/empty device/error description.

Returns

(bool Succ, float XmlVersion, Hashtable FeatureList, IError ErrorDesc) true/xml version/feature list/empty error or false/0/empty hashtable/error description.

Returns

(bool Succ, List(Parameter) ParameterList, IError ErrorDesc) true/parameter list/empty error or false/empty list/error description.

Parameters

<i>parameter</i>	The parameter its value is asked for.
------------------	---------------------------------------

Returns

(bool Succ, object ParameterValue, IError ErrorDesc) true/the value with type parameters value type/empty error or false/0/error description.

Parameters

<i>settingsVersion</i>	The settings version of the parameter its value is asked for.
<i>configVersion</i>	The config version of the parameter its value is asked for.
<i>configId</i>	The config id of the parameter its value is asked for.
<i>parameterId</i>	The id of the parameter its value is asked for.

Returns

(bool Succ, object ParameterValue, IError ErrorDesc) true/the value with type parameters value type/empty error or false/0/error description.

Parameters

<i>parameter</i>	The parameter the value should be set from.
<i>value</i>	The new value.

Returns

(bool Succ, List(IParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

Parameters

<i>settingsVersion</i>	The settings version of the parameter which should be set.
------------------------	--

Parameters

<i>configVersion</i>	The config version of the parameter which should be set.
<i>configId</i>	The config id of the parameter which should be set.
<i>parameterId</i>	The id of the parameter which should be set.
<i>value</i>	The new value.

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

Parameters

<i>sourceFileName</i>	Path and filename to upload.
-----------------------	------------------------------

///

Returns

(bool Succ, List(IPParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

Parameters

<i>filename</i>	The path and filename of the firmware file.
<i>quitAfterFileSent</i>	If set to true, method quits after file is sent to device and update runs automatically on device.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>ipAddress</i>	The new IP Address.
<i>networkMask</i>	The new network mask.
<i>gateway</i>	The new gateway.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>command</i>	The test system command.
<i>inputValue</i>	
<i>timeout</i>	Wait time for device reply.

Returns**Parameters**

<i>destinationFileName</i>	Path and file name to save to.
----------------------------	--------------------------------

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>sourceFileName</i>	Path and filename to upload.
-----------------------	------------------------------

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>parameterSet</i>	A list of parameter objects.
---------------------	------------------------------

///

Returns

(bool Succ, List(IParameter) DependendParameters, IError ErrorDesc) true/empty error or false/error description. If other parameters have automatically changed as a result of this parameter change, they will be returned with their new values as a list.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>timeoutMs</i>	The maximum time in ms to try reading an item.
<i>cts</i>	A cancellation token source to cancel operation earlier than timeoutMs.

Returns

(bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, IError ErrorDesc) true/item/no of discarded items until now/empty error or false/empty container/no of discarded items until now/error description.

Parameters

<i>timeoutMs</i>	The maximum time in ms to try reading an item.
<i>cts</i>	A cancellation token source to cancel operation earlier than timeoutMs.

Returns

(bool Succ, IVsxLogData LogMessage, int NumberOfDiscardedItems, IError ErrorDesc) true/item/no of discarded items until now/empty error or false/empty log message/no of discarded items until now/error description.

Parameters

<i>filename</i>	The path and filename of the data file.
-----------------	---

Returns

(bool Succ, IError ErrorDesc) true/empty error if upload was successful or false/error description.

Parameters

<i>xml</i>	The message.
------------	--------------

Returns

Always true and empty error.

Returns

true/status items list/empty error or false/empty list/error description.

Returns

true/empty error or false/error description.

Parameters

<i>username</i>	The username.
<i>password</i>	The password.

Returns

true/empty error or false/error description.

Parameters

<i>authorizationUsername</i>	The username who wants to set the password.
<i>authorizationPassword</i>	The password of the user who wants to set the password.

Parameters

<i>username</i>	The username of the account for which the new password should be set.
<i>newPassword</i>	The new password.

Returns

true/empty error or false/error description.

Parameters

<i>password</i>	A new admin password.
<i>authenticationMode</i>	The desired authentication mode on device.

Returns

true/empty error or false/error description.

10.25.3.2 CurrentDevice bool [IVsxDeviceInformation](#) PF.VsxProtocolDriver.IVsxProtocolDriver↔
Sync.CurrentDevice

10.25.3.3 XmlVersion bool float PF.VsxProtocolDriver.IVsxProtocolDriverSync.XmlVersion

10.25.3.4 FeatureList bool float Hashtable PF.VsxProtocolDriver.IVsxProtocolDriverSync.Feature↔
List

10.25.3.5 ParameterList bool List< [IParameter](#) > PF.VsxProtocolDriver.IVsxProtocolDriverSync.↔
ParameterList

10.25.3.6 ParameterValue bool object PF.VsxProtocolDriver.IVsxProtocolDriverSync.Parameter↔
Value

10.25.3.7 DependendParameters bool List< [IParameter](#) > PF.VsxProtocolDriver.IVsxProtocol↔
DriverSync.DependendParameters

10.25.3.8 Command bool string PF.VsxProtocolDriver.IVsxProtocolDriverSync.Command

10.25.3.9 OutputValue bool string string PF.VsxProtocolDriver.IVsxProtocolDriverSync.Output↔
Value

10.25.3.10 Status bool string string string PF.VsxProtocolDriver.IVsxProtocolDriverSync.Status

10.25.3.11 Container bool? [IVsxDynamicContainer](#) PF.VsxProtocolDriver.IVsxProtocolDriver↔
Sync.Container

10.25.3.12 NumberOfDiscardedItems [1/2] bool? [IVsxDynamicContainer](#) int PF.VsxProtocol↔
Driver.IVsxProtocolDriverSync.NumberOfDiscardedItems

10.25.3.13 LogMessage bool? [IVsxLogData](#) PF.VsxProtocolDriver.IVsxProtocolDriverSync.Log↔
Message

10.25.3.14 NumberOfDiscardedItems [2/2] bool? [IVsxLogData](#) int PF.VsxProtocolDriver.IVsx↔
ProtocolDriverSync.NumberOfDiscardedItems

10.25.3.15 StatusItems bool List<[IStatusItem](#)> PF.VsxProtocolDriver.IVsxProtocolDriverSync.↔
StatusItems

10.25.4 Property Documentation

10.25.4.1 FirmwareStateChannelReader ChannelReader<[IFirmwareState](#)> PF.VsxProtocolDriver.↔
IVsxProtocolDriverSync.FirmwareStateChannelReader [get]

Channel reader for incoming state messages while firmware update is in progress.

10.25.4.2 WaitTimeout `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.WaitTimeout [get], [set]`

Gets or sets the time in ms, the driver waits for response from device. Default is VsxProtocolDriver.DefaultEthernetTimeoutMs.

10.25.4.3 MissingContainerFramesCounter `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.MissingContainerFramesCounter [get]`

Gets the missing frame counter for image grabbing

10.25.4.4 MissingLogMessagesCounter `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.MissingLogMessagesCounter [get]`

Gets the missing log messages counter for log message grabbing

10.25.4.5 DynamicContainerQueueSize `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.DynamicContainerQueueSize [get]`

Gets the current size of the dynamic container message queue

10.25.4.6 NumberOfCachedContainers `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.NumberOfCachedContainers [get]`

Gets the current number of cached container messages.

10.25.4.7 LogMessageQueueSize `int PF.VsxProtocolDriver.IVsxProtocolDriverSync.LogMessageQueueSize [get]`

Gets the current size of the log message queue

10.25.4.8 ContainerGrabberStrategy `Strategy PF.VsxProtocolDriver.IVsxProtocolDriverSync.ContainerGrabberStrategy [get]`

Gets the current strategy of the dynamic container message queue

10.25.4.9 LogGrabberStrategy `Strategy PF.VsxProtocolDriver.IVsxProtocolDriverSync.LogGrabber↔Strategy [get]`

Gets the current strategy of the log message queue

10.25.4.10 Connected `bool PF.VsxProtocolDriver.IVsxProtocolDriverSync.Connected [get]`

Indicates current connection state with the device.

10.25.5 Event Documentation

10.25.5.1 OnDisconnect `Action<string, DisconnectEvent, string> PF.VsxProtocolDriver.IVsx↔ProtocolDriverSync.OnDisconnect`

Event called when a TPC/IP device is disconnected.

CurrentIpAddress	Actual ip address.
DisconnectEvent	Enumeration of disconnect reason
ErrorMessage	Error message as string

Returns

string with ip of the disconnected device and a descriptions why device was disconnected.

10.25.5.2 OnDeviceStatusReceived `Action<DeviceStatusScope, List<IStatusItem> > PF.Vsx↔ProtocolDriver.IVsxProtocolDriverSync.OnDeviceStatusReceived`

Event called when status data is received from device after cyclic sending is started (see [SubscribeToDeviceStatusData](#)).

Returns

Device status scope whether device sends full or multi status data and the status data list.

DeviceStatusScope	The library or executable built from a compilation.
List<IStatusItem>	List of status items returned

10.25.5.3 OnSessionMessageReceived `Action<IVsxSessionData> PF.VsxProtocolDriver.IVsx↔ProtocolDriverSync.OnSessionMessageReceived`

Event called when session data is received from device which means that something changed with the login status.

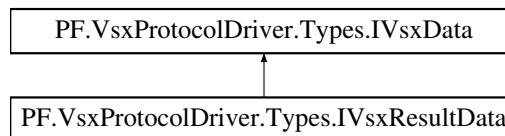
Returns

A session status message.

10.26 PF.VsxProtocolDriver.Types.IVsxResultData Interface Reference

Result data received from device.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxResultData:

**Properties**

- string [ResultString](#) [get]

10.26.1 Detailed Description

Result data received from device.

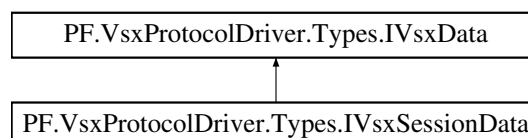
10.26.2 Property Documentation

10.26.2.1 ResultString `string PF.VsxProtocolDriver.Types.IVsxResultData.ResultString [get]`

10.27 PF.VsxProtocolDriver.Types.IVsxSessionData Interface Reference

A session message. These messages deal with logging in and out, and setting passwords.

Inheritance diagram for PF.VsxProtocolDriver.Types.IVsxSessionData:



Properties

- `Vsx.SessionTypes.SessionType` [get]
The session type of the message.
- string `Message` [get]
The message string.
- `TimeSpan Timeout` [get]
The timeout if `SessionType` is `Vsx.SessionTypes.TimeoutAnnouncement`.

10.27.1 Detailed Description

A session message. These messages deal with logging in and out, and setting passwords.

10.27.2 Property Documentation

10.27.2.1 SessionType `Vsx.SessionTypes` `PF.VsxProtocolDriver.Types.IVsxSessionData.SessionType` [get]

The session type of the message.

10.27.2.2 Message string `PF.VsxProtocolDriver.Types.IVsxSessionData.Message` [get]

The message string.

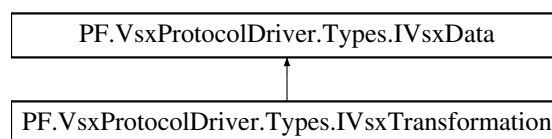
10.27.2.3 Timeout `TimeSpan` `PF.VsxProtocolDriver.Types.IVsxSessionData.Timeout` [get]

The timeout if `SessionType` is `Vsx.SessionTypes.TimeoutAnnouncement`.

10.28 PF.VsxProtocolDriver.Types.IVsxTransformation Interface Reference

Used to transform raw point cloud data from device.

Inheritance diagram for `PF.VsxProtocolDriver.Types.IVsxTransformation`:



Properties

- double [TranslationTX](#) [get]
- double [TranslationTY](#) [get]
- double [TranslationTZ](#) [get]
- double [QuaternionQ0](#) [get]
- double [QuaternionQ1](#) [get]
- double [QuaternionQ2](#) [get]
- double [QuaternionQ3](#) [get]

10.28.1 Detailed Description

Used to transform raw point cloud data from device.

10.28.2 Property Documentation

10.28.2.1 TranslationTX double PF.VsxProtocolDriver.Types.IVsxTransformation.TranslationTX
[get]

10.28.2.2 TranslationTY double PF.VsxProtocolDriver.Types.IVsxTransformation.TranslationTY
[get]

10.28.2.3 TranslationTZ double PF.VsxProtocolDriver.Types.IVsxTransformation.TranslationTZ
[get]

10.28.2.4 QuaternionQ0 double PF.VsxProtocolDriver.Types.IVsxTransformation.QuaternionQ0
[get]

10.28.2.5 QuaternionQ1 double PF.VsxProtocolDriver.Types.IVsxTransformation.QuaternionQ1
[get]

10.28.2.6 QuaternionQ2 double PF.VsxProtocolDriver.Types.IVsxTransformation.QuaternionQ2
[get]

10.28.2.7 QuaternionQ3 `double PF.VsxProtocolDriver.Types.IVsxTransformation.QuaternionQ3`
[get]

10.29 PF.VsxProtocolDriver.Types.Vsx Class Reference

Public Types

- enum [Modifiers](#)
- enum [FunctionAttributes](#)
- enum [Elements](#)
- enum [Attributes](#)
- enum [ConstantValues](#)
- enum [ParameterTypes](#)
- enum [ValueTypes](#)
- enum [LogMessageTypes](#)
- enum [Features](#)
- enum [Colors](#)
- enum [Sizes](#)
- enum [SensorTypes](#)
- enum [SensorErrors](#)
- enum [PgvMeasurement](#)
- enum [PcvMeasurement](#)
- enum [PxxSilMeasurement](#)
- enum [SrMeasurement](#)
- enum [SessionTypes](#)
- enum [SessionStatus](#)
- enum [DeviceStatus](#)
- enum [Sharpness](#)
- enum [Results](#)
- enum [DeviceAuthenticationMode](#)

Static Public Member Functions

- static int [GetValueFromString< T >](#) (string enumString)
- static ? string [GetStringFromValue< T >](#) (int value)
- static T [GetEnumFromString< T >](#) (string value)
- static bool T string errorDesc [GetEnumFromStringWithEM< T >](#) (string value)
- static T [GetEnumFromStringWithException< T >](#) (string? value)

Static Public Attributes

- static readonly float [VcVsxVersion](#) = 2
- static readonly float [VcXmlVersion](#) = 2
- static float [SensorVsxVersion](#) = 1
- static float [SensorXmlVersion](#) = 1
- static bool [Succ](#)
- static bool T [EnumEntry](#)

10.29.1 Member Enumeration Documentation

10.29.1.1 Modifiers enum [PF.VsxProtocolDriver.Types.Vsx.Modifiers](#)

10.29.1.2 FunctionAttributes enum [PF.VsxProtocolDriver.Types.Vsx.FunctionAttributes](#)

10.29.1.3 Elements enum [PF.VsxProtocolDriver.Types.Vsx.Elements](#)

10.29.1.4 Attributes enum [PF.VsxProtocolDriver.Types.Vsx.Attributes](#)

10.29.1.5 ConstantValues enum [PF.VsxProtocolDriver.Types.Vsx.ConstantValues](#)

10.29.1.6 ParameterTypes enum [PF.VsxProtocolDriver.Types.Vsx.ParameterTypes](#)

10.29.1.7 ValueTypes enum [PF.VsxProtocolDriver.Types.Vsx.ValueTypes](#)

10.29.1.8 LogMessageTypes enum [PF.VsxProtocolDriver.Types.Vsx.LogMessageTypes](#)

10.29.1.9 Features enum [PF.VsxProtocolDriver.Types.Vsx.Features](#)

10.29.1.10 Colors enum [PF.VsxProtocolDriver.Types.Vsx.Colors](#)

10.29.1.11 Sizes enum [PF.VsxProtocolDriver.Types.Vsx.Sizes](#)

10.29.1.12 SensorTypes enum `PF.VsxProtocolDriver.Types.Vsx.SensorTypes`

10.29.1.13 SensorErrors enum `PF.VsxProtocolDriver.Types.Vsx.SensorErrors`

10.29.1.14 PgvMeasurement enum `PF.VsxProtocolDriver.Types.Vsx.PgvMeasurement`

10.29.1.15 PcvMeasurement enum `PF.VsxProtocolDriver.Types.Vsx.PcvMeasurement`

10.29.1.16 PxxSilMeasurement enum `PF.VsxProtocolDriver.Types.Vsx.PxxSilMeasurement`

10.29.1.17 SrMeasurement enum `PF.VsxProtocolDriver.Types.Vsx.SrMeasurement`

10.29.1.18 SessionTypes enum `PF.VsxProtocolDriver.Types.Vsx.SessionTypes`

10.29.1.19 SessionStatus enum `PF.VsxProtocolDriver.Types.Vsx.SessionStatus`

10.29.1.20 DeviceStatus enum `PF.VsxProtocolDriver.Types.Vsx.DeviceStatus`

10.29.1.21 Sharpness enum `PF.VsxProtocolDriver.Types.Vsx.Sharpness`

10.29.1.22 Results enum `PF.VsxProtocolDriver.Types.Vsx.Results`

10.29.1.23 DeviceAuthenticationMode enum `PF.VsxProtocolDriver.Types.Vsx.DeviceAuthenticationMode`

10.29.2 Member Function Documentation

10.29.2.1 GetValueFromString< T >() static int PF.VsxProtocolDriver.Types.Vsx.GetValueFrom↵
String< T > (
 string *enumString*) [inline], [static]

10.29.2.2 GetStringFromValue< T >() static ? string PF.VsxProtocolDriver.Types.Vsx.GetString↵
FromValue< T > (
 int *value*) [inline], [static]

10.29.2.3 GetEnumFromString< T >() static T PF.VsxProtocolDriver.Types.Vsx.GetEnumFromString<↵
T > (
 string *value*) [inline], [static]

10.29.2.4 GetEnumFromStringWithEM< T >() static bool T string errorDesc PF.VsxProtocol↵
Driver.Types.Vsx.GetEnumFromStringWithEM< T > (
 string *value*) [inline], [static]

10.29.2.5 GetEnumFromStringWithException< T >() static T PF.VsxProtocolDriver.Types.Vsx.Get↵
EnumFromStringWithException< T > (
 string? *value*) [inline], [static]

10.29.3 Member Data Documentation

10.29.3.1 VcVsxVersion readonly float PF.VsxProtocolDriver.Types.Vsx.VcVsxVersion = 2 [static]

10.29.3.2 VcXmlVersion readonly float PF.VsxProtocolDriver.Types.Vsx.VcXmlVersion = 2 [static]

10.29.3.3 SensorVsxVersion float PF.VsxProtocolDriver.Types.Vsx.SensorVsxVersion = 1 [static]

10.29.3.4 SensorXmlVersion float PF.VsxProtocolDriver.Types.Vsx.SensorXmlVersion = 1 [static]

10.29.3.5 Succ bool PF.VsxProtocolDriver.Types.Vsx.Succ [static]

10.29.3.6 EnumEntry bool T PF.VsxProtocolDriver.Types.Vsx.EnumEntry [static]

10.30 PF.VsxProtocolDriver.VsxProtocolDriver Class Reference

Driver to communicate with a device/sensor by using Asynchronous programming.

Use [IVsxProtocolDriver](#) driver = VsxProtocolDriver.InitTcpDevice(...) or [IVsxProtocolDriver](#) driver = VsxProtocolDriver.InitSerialDevice(...) to get an instance and use the driver.

Static Public Member Functions

- static [IVsxProtocolDriver InitTcpDevice](#) (string ipAddress, int port=Vsxport, string pluginName="")
Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via TCP/IP protocol.
- static [IVsxProtocolDriver InitSerialDevice](#) (string serialPort, int baudrate, string sensorType, [SerialConnectionType](#) connectionType, string pluginName="")
Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via serial protocol.
- static async Task<(bool [Succ](#), List< [IVsxDeviceInformation](#) > DeviceList, [IError](#) ErrorDesc)> [UdpDeviceList](#) ()
Returns a list of the Vsx devices that are found on the network via a UDP broadcast.
- static async Task<(bool [Succ](#), List< [IVsxDeviceInformation](#) > DeviceList, [IError](#) ErrorDesc)> [GetUdpDeviceList](#) ()
Returns a list of the Vsx devices that are found on the network via a UDP broadcast.
- static async Task<(bool [Succ](#), [IVsxDeviceInformation](#) Device, [IError](#) ErrorDesc)> [GetSingleUdpDevice](#) (string ip)
Returns information retrieved from a udp discovery query to a specific device.
- static async Task<(bool [Succ](#), [IError](#) ErrorDesc)> [SetNetworkSettingsViaUdp](#) (string macAddress, string ipAddress, string networkMask, string gateway)
Sets the network settings of the device identified by the macAddress via UDP.
- static bool [IError](#) ErrorDesc [SaveData](#) (string filename, [IVsxData](#) message)
- static bool [IError](#) ErrorDesc [Save3DPointCloudData](#) (string filename, [IVsxImage](#) x, [IVsxImage](#) y, [IVsxImage](#) z)
- static bool? [IFirmwareVersion](#) [IError](#) ErrorDesc [GetFirmwareVersion](#) (string filename)

Static Public Attributes

- static bool [Succ](#)
Saves a VsxMessage to the given filename.
- static bool? [IFirmwareVersion](#) [FirmwareVersion](#)

10.30.1 Detailed Description

Driver to communicate with a device/sensor by using Asynchronous programming.

Use `IVsxProtocolDriver driver = VsxProtocolDriver.InitTcpDevice(...)` or `IVsxProtocolDriver driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

10.30.2 Member Function Documentation

10.30.2.1 InitTcpDevice() static `IVsxProtocolDriver` PF.VsxProtocolDriver.VsxProtocolDriver.
`InitTcpDevice (`
 `string ipAddress,`
 `int port = Vsxport,`
 `string pluginName = "") [inline], [static]`

Initiates an instance of the `VsxProtocolDriverSync` to communicate with a Vsx-Device via TCP/IP protocol.

Parameters

<i>ipAddress</i>	The ip address of the device.
<i>port</i>	The port of the device.
<i>pluginName</i>	The type of the device. (not used anymore)

Returns

The instance.

10.30.2.2 InitSerialDevice() static `IVsxProtocolDriver` PF.VsxProtocolDriver.VsxProtocolDriver.
`InitSerialDevice (`
 `string serialPort,`
 `int baudrate,`
 `string sensorType,`
 `SerialConnectionType connectionType,`
 `string pluginName = "") [inline], [static]`

Initiates an instance of the `VsxProtocolDriverSync` to communicate with a Vsx-Device via serial protocol.

Parameters

<i>serialPort</i>	The comport of the device.
<i>baudrate</i>	The baudrate of the device.
<i>sensorType</i>	The sensor type of the device.
<i>connectionType</i>	The connection type of the device.
<i>pluginName</i>	The type of the device. (not used anymore)

Returns

The instance.

10.30.2.3 UdpDeviceList() static async Task<(bool Succ, List< IVsxDeviceInformation > DeviceList, IError ErrorDesc)> PF.VsxProtocolDriver.VsxProtocolDriver.UdpDeviceList () [inline], [static]

10.30.2.4 GetUdpDeviceList() static async Task<(bool Succ, List< IVsxDeviceInformation > DeviceList, IError ErrorDesc)> PF.VsxProtocolDriver.VsxProtocolDriver.GetUdpDeviceList () [inline], [static]

Returns a list of the Vsx devices that are found on the network via a UDP broadcast.

Returns

(bool Succ, List<IVsxDeviceInformation> DeviceList, IError ErrorDesc) true/list with all devices/empty error or false/empty list/error description.

10.30.2.5 GetSingleUdpDevice() static async Task<(bool Succ, IVsxDeviceInformation Device, IError ErrorDesc)> PF.VsxProtocolDriver.VsxProtocolDriver.GetSingleUdpDevice (string ip) [inline], [static]

Returns information retrieved from a udp discovery query to a specific device.

Parameters

<i>ip</i>	The ip of the requested device.
-----------	---------------------------------

Returns

(bool Succ, IVsxDeviceInformation Device, IError ErrorDesc) true/device information/empty error or false/null/error description.

10.30.2.6 SetNetworkSettingsViaUdp() static async Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.VsxProtocolDriver.SetNetworkSettingsViaUdp (string macAddress, string ipAddress, string networkMask, string gateway) [inline], [static]

Sets the network settings of the device identified by the macAddress via UDP.

Parameters

<i>macAddress</i>	The macAddress of the device to set
<i>ipAddress</i>	The new IP Address
<i>networkMask</i>	The new network mask
<i>gateway</i>	The new gateway

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

10.30.2.7 SaveData() static bool IError ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriver.
SaveData (
 string filename,
 IVsxData message) [inline], [static]

10.30.2.8 Save3DPointCloudData() static bool IError ErrorDesc PF.VsxProtocolDriver.VsxProtocol
Driver.Save3DPointCloudData (
 string filename,
 IVsxImage x,
 IVsxImage y,
 IVsxImage z) [inline], [static]

10.30.2.9 GetFirmwareVersion() static bool? IFirmwareVersion IError ErrorDesc PF.VsxProtocol
Driver.VsxProtocolDriver.GetFirmwareVersion (
 string filename) [inline], [static]

10.30.3 Member Data Documentation

10.30.3.1 Succ static bool PF.VsxProtocolDriver.VsxProtocolDriver.Succ [static]

Saves a VsxMessage to the given filename.

Reads the firmware version from a given firmware filename.

Saves a 3D point cloud as pcd to the given filename.

Parameters

<i>filename</i>	Path and filename where to save the message.
<i>message</i>	The message to save.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description

Parameters

<i>filename</i>	Path and filename where to save the data.
<i>x</i>	The x image.
<i>y</i>	The y image.
<i>z</i>	The z image.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description

Parameters

<i>filename</i>	Path and filename of the firmware file.
-----------------	---

Returns

(bool Succ, IFirmwareVersion? FirmwareVersion, IError ErrorDesc) true/firmware version/empty error or false/null/error description

10.30.3.2 FirmwareVersion bool? [IFirmwareVersion](#) PF.VsxProtocolDriver.VsxProtocolDriver.↔
 FirmwareVersion [static]

10.31 PF.VsxProtocolDriver.VsxProtocolDriverSync Class Reference

Driver to communicate with a device/sensor by using synchronous programming.

Use [IVsxProtocolDriverSync](#) driver = VsxProtocolDriver.InitTcpDevice(...) or [IVsxProtocolDriverSync](#) driver = VsxProtocolDriver.InitSerialDevice(...) to get an instance and use the driver.

Static Public Member Functions

- static [IVsxProtocolDriverSync](#) [InitTcpDevice](#) (string ipAddress, int port=VsxProtocolDriver.Vsxport, string pluginName="")
Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via TCP/IP protocol.
- static [IVsxProtocolDriverSync](#) [InitSerialDevice](#) (string serialPort, int baudrate, string sensorType, [SerialConnectionType](#) connectionType, string pluginName="")
Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via serial protocol.
- static bool List< [IVsxDeviceInformation](#) > [IError](#) ErrorDesc [UdpDeviceList](#) ()
- static bool List< [IVsxDeviceInformation](#) > [IError](#) ErrorDesc [GetUdpDeviceList](#) ()
- static bool [IVsxDeviceInformation](#) [IError](#) ErrorDesc [GetSingleUdpDevice](#) (string ip)
- static bool [IError](#) ErrorDesc [SetNetworkSettingsViaUdp](#) (string macAddress, string ipAddress, string networkMask, string gateway)
- static bool [IError](#) ErrorDesc [SaveData](#) (string filename, [IVsxData](#) message)
- static bool [IError](#) ErrorDesc [Save3DPointCloudData](#) (string filename, [IVsxImage](#) x, [IVsxImage](#) y, [IVsxImage](#) z)
- static bool? [IFirmwareVersion](#) [IError](#) ErrorDesc [GetFirmwareVersion](#) (string filename)

Static Public Attributes

- static bool [Succ](#)
Returns a list of the Vsx devices that are found on the network via a UDP broadcast.
- static bool List< [IVsxDeviceInformation](#) > [DeviceList](#)
- static bool [IVsxDeviceInformation](#) [DeviceList](#)
- static bool? [IFirmwareVersion](#) [FirmwareVersion](#)

10.31.1 Detailed Description

Driver to communicate with a device/sensor by using synchronous programming.

Use [IVsxProtocolDriverSync](#) driver = [VsxProtocolDriver](#).InitTcpDevice(...) or [IVsxProtocolDriverSync](#) driver = [VsxProtocolDriver](#).InitSerialDevice(...) to get an instance and use the driver.

10.31.2 Member Function Documentation

10.31.2.1 InitTcpDevice() static [IVsxProtocolDriverSync](#) PF.VsxProtocolDriver.VsxProtocolDriver↔
Sync.InitTcpDevice (
 string *ipAddress*,
 int *port* = [VsxProtocolDriver](#).Vsxport,
 string *pluginName* = "") [inline], [static]

Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via TCP/IP protocol.

Parameters

<i>ipAddress</i>	The ip address of the device.
<i>port</i>	The port of the device.
<i>pluginName</i>	The type of the device. (not used anymore)

Returns

The instance.

10.31.2.2 InitSerialDevice() static [IVsxProtocolDriverSync](#) PF.VsxProtocolDriver.VsxProtocol↔
DriverSync.InitSerialDevice (
 string *serialPort*,
 int *baudrate*,
 string *sensorType*,
 [SerialConnectionType](#) *connectionType*,
 string *pluginName* = "") [inline], [static]

Initiates an instance of the [VsxProtocolDriverSync](#) to communicate with a Vsx-Device via serial protocol.

Parameters

<i>serialPort</i>	The comport of the device.
<i>baudrate</i>	The baudrate of the device.
<i>sensorType</i>	The sensor type of the device.
<i>connectionType</i>	The connection type of the device.
<i>pluginName</i>	The type of the device. (not used anymore)

Returns

The instance.

10.31.2.3 UdpDeviceList() static bool List< [IVsxDeviceInformation](#) > [IError](#) ErrorDesc PF.Vsx↔
ProtocolDriver.VsxProtocolDriverSync.UdpDeviceList () [inline], [static]

10.31.2.4 GetUdpDeviceList() static bool List< [IVsxDeviceInformation](#) > [IError](#) ErrorDesc PF.↔
VsxProtocolDriver.VsxProtocolDriverSync.GetUdpDeviceList () [inline], [static]

10.31.2.5 GetSingleUdpDevice() static bool [IVsxDeviceInformation](#) [IError](#) ErrorDesc PF.Vsx↔
ProtocolDriver.VsxProtocolDriverSync.GetSingleUdpDevice (
string ip) [static]

10.31.2.6 SetNetworkSettingsViaUdp() static bool [IError](#) ErrorDesc PF.VsxProtocolDriver.Vsx↔
ProtocolDriverSync.SetNetworkSettingsViaUdp (
string macAddress,
string ipAddress,
string networkMask,
string gateway) [static]

10.31.2.7 SaveData() static bool [IError](#) ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriver↔
Sync.SaveData (
string filename,
[IVsxData](#) message) [static]

10.31.2.8 Save3DPointCloudData() static bool [IError](#) ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriverSync.Save3DPointCloudData (
 string *filename*,
 [IVsxImage](#) *x*,
 [IVsxImage](#) *y*,
 [IVsxImage](#) *z*) [static]

10.31.2.9 GetFirmwareVersion() static bool? [IFirmwareVersion](#) [IError](#) ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriverSync.GetFirmwareVersion (
 string *filename*) [static]

10.31.3 Member Data Documentation

10.31.3.1 Succ static bool PF.VsxProtocolDriver.VsxProtocolDriverSync.Succ [static]

Returns a list of the Vsx devices that are found on the network via a UDP broadcast.

Reads the firmware version from a given firmware filename.

Saves a 3D point cloud as pcd to the given filename.

Saves a VsxMessage to the given filename.

Sets the network settings of the device identified by the macAddress via UDP.

Returns information retrieved from an udp discovery query to a specific device.

Returns

(bool Succ, List([IVsxDeviceInformation](#)) DeviceList, [IError](#) ErrorDesc) true/list with all devices/empty error or false/empty list/error description.

Parameters

<i>ip</i>	The ip of the requested device.
-----------	---------------------------------

Returns

(bool Succ, [IVsxDeviceInformation](#) Device, [IError](#) ErrorDesc) true/device information/empty error or false/null/error description.

Parameters

<i>macAddress</i>	The macAddress of the device to set
<i>ipAddress</i>	The new IP Address
<i>networkMask</i>	The new network mask
<i>gateway</i>	The new gateway

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description.

Parameters

<i>filename</i>	Path and filename where to save the message.
<i>message</i>	The message to save.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description

Parameters

<i>filename</i>	Path and filename where to save the data.
<i>x</i>	The x image.
<i>y</i>	The y image.
<i>z</i>	The z image.

Returns

(bool Succ, IError ErrorDesc) true/empty error or false/error description

Parameters

<i>filename</i>	Path and filename of the firmware file.
-----------------	---

Returns

(bool Succ, IFirmwareVersion? FirmwareVersion, IError ErrorDesc) true/firmware version/empty error or false/null/error description

10.31.3.2 DeviceList [1/2] static bool List< [IVsxDeviceInformation](#) > PF.VsxProtocolDriver.VsxProtocolDriverSync.DeviceList [static]

10.31.3.3 DeviceList [2/2] bool [IVsxDeviceInformation](#) PF.VsxProtocolDriver.VsxProtocolDriverSync.DeviceList [static]

10.31.3.4 FirmwareVersion bool? [IFirmwareVersion](#) PF.VsxProtocolDriver.VsxProtocolDriverSync.FirmwareVersion [static]

Index

- ActivationTimer
 - PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData, 45
- ActiveApplicationIdentifier
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 32
- ApplicationResultString
 - PF.VsxProtocolDriver.Types.IVsxApplicationResultData, 26
- Attributes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- AutoTriggerFrameRate
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 44
- AxisMax
 - PF.VsxProtocolDriver.Types.IVsxImage, 38
- AxisMin
 - PF.VsxProtocolDriver.Types.IVsxImage, 38
- Baseline
 - PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 33
- Baudrate
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- Build
 - PF.VsxProtocolDriver.Types.IFirmwareVersion, 16
- Busy
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- Clone
 - PF.VsxProtocolDriver.Types.IParameter, 19
 - PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 34
- Colors
 - PF.VsxProtocolDriver.Types.Vsx, 87
- Command
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- CompareBuffer
 - PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData, 46
- ComPort
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- ConfigId
 - PF.VsxProtocolDriver.Types.IParameter, 20
- ConfigType
 - PF.VsxProtocolDriver.Types.IParameter, 20
- ConfigurationClass
 - PF.VsxProtocolDriver.Types.IStatusItem, 25
- ConfigVersion
 - PF.VsxProtocolDriver.Types.IParameter, 20
 - PF.VsxProtocolDriver.Types.IStatusItem, 24
- Connect
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 49
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- ConnectAndLogin
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 50
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- Connected
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 63
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 82
- ConstantValues
 - PF.VsxProtocolDriver.Types.Vsx, 87
- Container
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- ContainerGrabberStrategy
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 63
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 81
- ContainsMessage
 - PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 34
- Content
 - PF.VsxProtocolDriver.Types, 11
- Contents
 - PF.VsxProtocolDriver.Types.IVsxFile, 35
- CoordinateOffset
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
- CoordinateScale
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
- CountLines
 - PF.VsxProtocolDriver.Types.IVsxLineData, 39
- CurrentDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- CurrentPosition
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 42
- DefaultValue
 - PF.VsxProtocolDriver.Types.IParameter, 21
- DependendParameters
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- DeviceAuthenticationMode
 - PF.VsxProtocolDriver.Types.Vsx, 88
- DeviceList
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 98
- DeviceStatus
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
 - PF.VsxProtocolDriver.Types.Vsx, 88
- DeviceStatusScope
 - PF.VsxProtocolDriver.Types, 11
- DeviceVsxVersionMajor
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- DeviceVsxVersionMinor
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- Disconnect
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 50

- PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- DisconnectEvent
 - PF.VsxProtocolDriver.Types, 11
- Dispose
 - PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 34
- DownloadParameterSet
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 56
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- DynamicContainerQueueSize
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 63
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 81
- Elements
 - PF.VsxProtocolDriver.Types.Vsx, 87
- Enable
 - PF.VsxProtocolDriver.Types.IParameter, 21
- EnumEntry
 - PF.VsxProtocolDriver.Types.Vsx, 90
- Error
 - PF.VsxProtocolDriver.Types.IFirmwareState, 15
- ErrorId
 - PF.VsxProtocolDriver.Types, 11
- ExposureTime
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
- FeatureList
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- Features
 - PF.VsxProtocolDriver.Types.Vsx, 87
- FileStatus
 - PF.VsxProtocolDriver.Types.IVsxFile, 35
- FirmwareStateChannelReader
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 62
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- FirmwareVersion
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
 - PF.VsxProtocolDriver.VsxProtocolDriver, 94
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 98
- FocalLength
 - PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 32
- Format
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
 - PF.VsxProtocolDriver.Types.IVsxLineData, 39
- FrameCounter
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
 - PF.VsxProtocolDriver.Types.IVsxLineData, 40
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 42
- FunctionAttributes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- Gain
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
- Gateway
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
- GetAllDeviceStatusData
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 60
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- GetCachedContainer
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 58
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 70
- GetDeviceInformation
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 52
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68
- GetDynamicContainer
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 58
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 70
- GetEnumFromString< T >
 - PF.VsxProtocolDriver.Types.Vsx, 89
- GetEnumFromStringWithEM< T >
 - PF.VsxProtocolDriver.Types.Vsx, 89
- GetEnumFromStringWithException< T >
 - PF.VsxProtocolDriver.Types.Vsx, 89
- GetFeatureList
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 52
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68
- GetFirmwareVersion
 - PF.VsxProtocolDriver.VsxProtocolDriver, 93
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 97
- GetLogMessage
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 59
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- GetMessage
 - PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 34
- GetParameterList
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 52
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68
- GetSingleParameterValue
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 52, 53
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68
- GetSingleUdpDevice
 - PF.VsxProtocolDriver.VsxProtocolDriver, 92
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- GetStringFromValue< T >
 - PF.VsxProtocolDriver.Types.Vsx, 89
- GetUdpDeviceList
 - PF.VsxProtocolDriver.VsxProtocolDriver, 92
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- GetValueFromString< T >
 - PF.VsxProtocolDriver.Types.Vsx, 89
- HeadAddress
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 31
- Height
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
- Id

- PF.VsxProtocolDriver.Types.IError, [14](#)
- PF.VsxProtocolDriver.Types.ItemTuple, [16](#)
- Illumination
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, [28](#)
- Image
 - PF.VsxProtocolDriver.Types.IVsxImage, [37](#)
- ImageCoordinate
 - PF.VsxProtocolDriver.Types.ICoordinate, [12](#)
- ImageData
 - PF.VsxProtocolDriver.Types.IVsxImage, [37](#)
- ImageData2Format
 - PF.VsxProtocolDriver.Types, [12](#)
- ImageDataFloats
 - PF.VsxProtocolDriver.Types.IVsxImage, [38](#)
- ImageFormat
 - PF.VsxProtocolDriver.Types, [11](#)
- ImageType
 - PF.VsxProtocolDriver.Types.IVsxImage, [36](#)
- InitializeDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [61](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [72](#)
- InitSerialDevice
 - PF.VsxProtocolDriver.VsxProtocolDriver, [91](#)
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, [95](#)
- InitTcpDevice
 - PF.VsxProtocolDriver.VsxProtocolDriver, [91](#)
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, [95](#)
- Intensity
 - PF.VsxProtocolDriver.Types.ICoordinate, [13](#)
- InvalidDataValue
 - PF.VsxProtocolDriver.Types.IVsxImage, [38](#)
- IoState
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [42](#)
- IpAddress
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, [30](#)
- IsLoginNeeded
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, [31](#)
- Items
 - PF.VsxProtocolDriver.Types.IParameter, [22](#)
 - PF.VsxProtocolDriver.Types.IStatusItem, [25](#)
- JobId
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, [27](#)
- LastResult
 - PF.VsxProtocolDriver.Types.IFirmwareState, [15](#)
- LineId
 - PF.VsxProtocolDriver.Types.ICoordinate, [13](#)
- LinePitch
 - PF.VsxProtocolDriver.Types.IVsxImage, [37](#)
- Lines
 - PF.VsxProtocolDriver.Types.ILineMulti, [17](#)
 - PF.VsxProtocolDriver.Types.IVsxLineData, [40](#)
- LmaExposureTime1
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmaExposureTime2
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmaRoiLengthX
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmaRoiLengthZ
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [44](#)
- LmaRoiOffsetX
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmaRoiOffsetZ
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmbExposureTime1
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmbExposureTime2
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [43](#)
- LmbRoiLengthX
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [44](#)
- LmbRoiLengthZ
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [44](#)
- LmbRoiOffsetX
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [44](#)
- LmbRoiOffsetZ
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, [44](#)
- LoadDefaultParameterSetOnDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [57](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [70](#)
- LoadParameterSetOnDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [57](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [69](#)
- Log
 - PF.VsxProtocolDriver.Types.IVsxLogData, [41](#)
- LogGrabberStrategy
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [63](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [81](#)
- Login
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [60](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [71](#)
- LogMessage
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [80](#)
- LogMessageQueueSize
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [63](#)
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, [81](#)
- LogMessageTypes
 - PF.VsxProtocolDriver.Types.Vsx, [87](#)
- Logout
 - PF.VsxProtocolDriver.IVsxProtocolDriver, [62](#)

- PF.VsxProtocolDriver.IVsxProtocolDriverSync, 72
- MacAddress
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
- Major
 - PF.VsxProtocolDriver.Types.IFirmwareVersion, 15
- Max
 - PF.VsxProtocolDriver.Types.IParameter, 21
- MaxX
 - PF.VsxProtocolDriver.Types.ILineMulti, 17
 - PF.VsxProtocolDriver.Types.IVsxLineData, 40
- MaxZ
 - PF.VsxProtocolDriver.Types.ILineMulti, 17
 - PF.VsxProtocolDriver.Types.IVsxLineData, 40
- Message
 - PF.VsxProtocolDriver.Types.IError, 14
 - PF.VsxProtocolDriver.Types.IVsxSessionData, 84
- MimeType
 - PF.VsxProtocolDriver.Types.IVsxFile, 35
- Min
 - PF.VsxProtocolDriver.Types.IParameter, 21
- Minor
 - PF.VsxProtocolDriver.Types.IFirmwareVersion, 15
- MinX
 - PF.VsxProtocolDriver.Types.ILineMulti, 17
 - PF.VsxProtocolDriver.Types.IVsxLineData, 40
- MinZ
 - PF.VsxProtocolDriver.Types.ILineMulti, 17
 - PF.VsxProtocolDriver.Types.IVsxLineData, 40
- MissingContainerFramesCounter
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 62
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 81
- MissingLogMessagesCounter
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 62
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 81
- Modifiers
 - PF.VsxProtocolDriver.Types.Vsx, 86
- Msg
 - PF.VsxProtocolDriver.Types.IFirmwareState, 14
- Name
 - PF.VsxProtocolDriver.Types.IItemTuple, 16
 - PF.VsxProtocolDriver.Types.IParameter, 20
 - PF.VsxProtocolDriver.Types.IStatusItem, 24
- NetworkMask
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
- NumberOfCachedContainers
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 63
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 81
- NumberOfDiscardedItems
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- OffsetLeftRectifiedToDisparityU
 - PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 33
- OffsetLeftRectifiedToDisparityV
 - PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 33
- OnDeviceStatusReceived
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 64
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 82
- OnDisconnect
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 63
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 82
- OnSessionMessageReceived
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 64
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 82
- OutputValue
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- ParameterId
 - PF.VsxProtocolDriver.Types.IParameter, 20
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
- ParameterList
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- ParameterTypes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- ParameterValue
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79
- Path
 - PF.VsxProtocolDriver.Types.IVsxFile, 35
- PcvMeasurement
 - PF.VsxProtocolDriver.Types.Vsx, 88
- PF, 8
- PF.VsxProtocolDriver, 8
- PF.VsxProtocolDriver.IVsxProtocolDriver, 46
 - Connect, 49
 - ConnectAndLogin, 50
 - Connected, 63
 - ContainerGrabberStrategy, 63
 - Disconnect, 50
 - DownloadParameterSet, 56
 - DynamicContainerQueueSize, 63
 - FirmwareStateChannelReader, 62
 - GetAllDeviceStatusData, 60
 - GetCachedContainer, 58
 - GetDeviceInformation, 52
 - GetDynamicContainer, 58
 - GetFeatureList, 52
 - GetLogMessage, 59
 - GetParameterList, 52
 - GetSingleParameterValue, 52, 53
 - InitializeDevice, 61
 - LoadDefaultParameterSetOnDevice, 57
 - LoadParameterSetOnDevice, 57
 - LogGrabberStrategy, 63
 - Login, 60
 - LogMessageQueueSize, 63
 - Logout, 62
 - MissingContainerFramesCounter, 62
 - MissingLogMessagesCounter, 62
 - NumberOfCachedContainers, 63
 - OnDeviceStatusReceived, 64
 - OnDisconnect, 63

- OnSessionMessageReceived, [64](#)
- ReConnectAndLoginTcpDevice, [51](#)
- ReConnectSerialDevice, [51](#)
- ReConnectTcpDevice, [50](#)
- ResetDynamicContainerGrabber, [57](#)
- ResetLogMessageGrabber, [58](#)
- SaveParameterSetOnDevice, [57](#)
- SendFirmware, [55](#)
- SendSessionKeepAlive, [62](#)
- SendTestSystemCommand, [55](#)
- SendXmlDataMessage, [59](#)
- SetMultipleParameterValues, [54](#)
- SetNetworkSettings, [55](#)
- SetPassword, [61](#)
- SetSingleParameterValue, [53](#), [54](#)
- SubscribeToDeviceStatusData, [60](#)
- UnsubscribeToDeviceStatusData, [60](#)
- UploadData, [59](#)
- UploadParameterSet, [56](#)
- WaitTimeout, [62](#)
- PF.VsxProtocolDriver.IVsxProtocolDriverSync, [64](#)
 - Command, [79](#)
 - Connect, [67](#)
 - ConnectAndLogin, [67](#)
 - Connected, [82](#)
 - Container, [80](#)
 - ContainerGrabberStrategy, [81](#)
 - CurrentDevice, [79](#)
 - DependParameters, [79](#)
 - Disconnect, [67](#)
 - DownloadParameterSet, [69](#)
 - DynamicContainerQueueSize, [81](#)
 - FeatureList, [79](#)
 - FirmwareStateChannelReader, [80](#)
 - GetAllDeviceStatusData, [71](#)
 - GetCachedContainer, [70](#)
 - GetDeviceInformation, [68](#)
 - GetDynamicContainer, [70](#)
 - GetFeatureList, [68](#)
 - GetLogMessage, [71](#)
 - GetParameterList, [68](#)
 - GetSingleParameterValue, [68](#)
 - InitializeDevice, [72](#)
 - LoadDefaultParameterSetOnDevice, [70](#)
 - LoadParameterSetOnDevice, [69](#)
 - LogGrabberStrategy, [81](#)
 - Login, [71](#)
 - LogMessage, [80](#)
 - LogMessageQueueSize, [81](#)
 - Logout, [72](#)
 - MissingContainerFramesCounter, [81](#)
 - MissingLogMessagesCounter, [81](#)
 - NumberOfCachedContainers, [81](#)
 - NumberOfDiscardedItems, [80](#)
 - OnDeviceStatusReceived, [82](#)
 - OnDisconnect, [82](#)
 - OnSessionMessageReceived, [82](#)
 - OutputValue, [80](#)
 - ParameterList, [79](#)
 - ParameterValue, [79](#)
 - ReConnectAndLoginTcpDevice, [67](#)
 - ReConnectSerialDevice, [67](#)
 - ReConnectTcpDevice, [67](#)
 - ResetDynamicContainerGrabber, [70](#)
 - ResetLogMessageGrabber, [70](#)
 - SaveParameterSetOnDevice, [69](#)
 - SendFirmware, [69](#)
 - SendSessionKeepAlive, [72](#)
 - SendTestSystemCommand, [69](#)
 - SendXmlDataMessage, [71](#)
 - SetMultipleParameterValues, [68](#)
 - SetNetworkSettings, [69](#)
 - SetPassword, [72](#)
 - SetSingleParameterValue, [68](#)
 - Status, [80](#)
 - StatusItems, [80](#)
 - SubscribeToDeviceStatusData, [71](#)
 - Succ, [72](#)
 - UnsubscribeToDeviceStatusData, [71](#)
 - UploadData, [71](#)
 - UploadParameterSet, [69](#)
 - WaitTimeout, [80](#)
 - XmlVersion, [79](#)
- PF.VsxProtocolDriver.Types, [9](#)
 - Content, [11](#)
 - DeviceStatusScope, [11](#)
 - DisconnectEvent, [11](#)
 - ErrorId, [11](#)
 - ImageData2Format, [12](#)
 - ImageFormat, [11](#)
 - SerialConnectionType, [11](#)
 - Strategy, [11](#)
 - VsxType, [11](#)
- PF.VsxProtocolDriver.Types.ICoordinate, [12](#)
 - ImageCoordinate, [12](#)
 - Intensity, [13](#)
 - LineId, [13](#)
 - Quality, [13](#)
 - SegmentId, [13](#)
 - Valid, [13](#)
 - WorldCoordinate, [12](#)
- PF.VsxProtocolDriver.Types.IError, [13](#)
 - Id, [14](#)
 - Message, [14](#)
- PF.VsxProtocolDriver.Types.IFirmwareState, [14](#)
 - Error, [15](#)
 - LastResult, [15](#)
 - Msg, [14](#)
 - Status, [14](#)
- PF.VsxProtocolDriver.Types.IFirmwareVersion, [15](#)
 - Build, [16](#)
 - Major, [15](#)
 - Minor, [15](#)
 - Revision, [16](#)
- PF.VsxProtocolDriver.Types.ITuple, [16](#)
 - Id, [16](#)

- Name, 16
- PF.VsxProtocolDriver.Types.ILineMulti, 17
 - Lines, 17
 - MaxX, 17
 - MaxZ, 17
 - MinX, 17
 - MinZ, 17
- PF.VsxProtocolDriver.Types.ILineSingle, 18
 - Points, 18
- PF.VsxProtocolDriver.Types.IParameter, 18
 - Clone, 19
 - ConfigId, 20
 - ConfigType, 20
 - ConfigVersion, 20
 - DefaultValue, 21
 - Enable, 21
 - Items, 22
 - Max, 21
 - Min, 21
 - Name, 20
 - ParameterId, 20
 - SettingsVersion, 20
 - Step, 21
 - Type, 20
 - Unit, 22
 - Value, 21
 - ValueType, 21
 - Visible, 21
- PF.VsxProtocolDriver.Types.IPoint2D, 22
 - X, 22
 - Y, 22
- PF.VsxProtocolDriver.Types.IPoint3D, 23
 - X, 23
 - Y, 23
 - Z, 23
- PF.VsxProtocolDriver.Types.IStatusItem, 23
 - ConfigurationClass, 25
 - ConfigVersion, 24
 - Items, 25
 - Name, 24
 - SensorTime, 25
 - SettingsVersion, 24
 - StatusItemId, 24
 - Time, 25
 - Value, 25
 - ValueType, 25
- PF.VsxProtocolDriver.Types.IVsxApplicationResultData, 25
 - ApplicationResultString, 26
- PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 26
 - ExposureTime, 27
 - FrameCounter, 27
 - Gain, 27
 - Illumination, 28
 - JobId, 27
 - ParameterId, 27
 - RotaryEncoder, 27
 - Timestamp, 27
 - TriggerCtr, 27
 - TriggerSource, 28
- PF.VsxProtocolDriver.Types.IVsxData, 28
 - VsxDataType, 29
- PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 29
 - ActiveApplicationIdentifier, 32
 - Baudrate, 31
 - Busy, 31
 - ComPort, 31
 - DeviceStatus, 31
 - DeviceVsxVersionMajor, 31
 - DeviceVsxVersionMinor, 31
 - FirmwareVersion, 30
 - Gateway, 30
 - Headaddress, 31
 - IpAddress, 30
 - IsLoginNeeded, 31
 - MacAddress, 30
 - NetworkMask, 30
 - SensorName, 30
 - SensorType, 30
- PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 32
 - Baseline, 33
 - FocalLength, 32
 - OffsetLeftRectifiedToDisparityU, 33
 - OffsetLeftRectifiedToDisparityV, 33
 - PrincipalPointU, 33
 - PrincipalPointV, 33
- PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 33
 - Clone, 34
 - ContainsMessage, 34
 - Dispose, 34
 - GetMessage, 34
 - TagListOfContainedMessages, 34
- PF.VsxProtocolDriver.Types.IVsxFile, 34
 - Contents, 35
 - FileStatus, 35
 - MimeType, 35
 - Path, 35
- PF.VsxProtocolDriver.Types.IVsxImage, 35
 - AxisMax, 38
 - AxisMin, 38
 - CoordinateOffset, 37
 - CoordinateScale, 37
 - Format, 37
 - FrameCounter, 37
 - Height, 37
 - Image, 37
 - ImageData, 37
 - ImageDataFloats, 38
 - ImageType, 36
 - InvalidDataValue, 38
 - LinePitch, 37
 - TransformationMatrix, 36
 - Width, 37
- PF.VsxProtocolDriver.Types.IVsxLineData, 38
 - CountLines, 39
 - Format, 39

- FrameCounter, 40
- Lines, 40
- MaxX, 40
- MaxZ, 40
- MinX, 40
- MinZ, 40
- Scale, 39
- ScaleXYZ, 39
- PF.VsxProtocolDriver.Types.IVsxLogData, 40
 - Log, 41
- PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 41
 - AutoTriggerFrameRate, 44
 - CurrentPosition, 42
 - FrameCounter, 42
 - IoState, 42
 - LmaExposureTime1, 43
 - LmaExposureTime2, 43
 - LmaRoiLengthX, 43
 - LmaRoiLengthZ, 44
 - LmaRoiOffsetX, 43
 - LmaRoiOffsetZ, 43
 - LmbExposureTime1, 43
 - LmbExposureTime2, 43
 - LmbRoiLengthX, 44
 - LmbRoiLengthZ, 44
 - LmbRoiOffsetX, 44
 - LmbRoiOffsetZ, 44
 - Timestamp, 42
 - TriggerCounter, 42
 - TriggerSource, 44
- PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData, 45
 - ActivationTimer, 45
 - CompareBuffer, 46
 - RobotData, 46
 - TargetPosition, 46
- PF.VsxProtocolDriver.Types.IVsxResultData, 83
 - ResultString, 83
- PF.VsxProtocolDriver.Types.IVsxSessionData, 83
 - Message, 84
 - SessionType, 84
 - Timeout, 84
- PF.VsxProtocolDriver.Types.IVsxTransformation, 84
 - QuaternionQ0, 85
 - QuaternionQ1, 85
 - QuaternionQ2, 85
 - QuaternionQ3, 85
 - TranslationTX, 85
 - TranslationTY, 85
 - TranslationTZ, 85
- PF.VsxProtocolDriver.Types.Vsx, 86
 - Attributes, 87
 - Colors, 87
 - ConstantValues, 87
 - DeviceAuthenticationMode, 88
 - DeviceStatus, 88
 - Elements, 87
 - EnumEntry, 90
 - Features, 87
 - FunctionAttributes, 87
 - GetEnumFromString< T >, 89
 - GetEnumFromStringWithEM< T >, 89
 - GetEnumFromStringWithException< T >, 89
 - GetStringFromValue< T >, 89
 - GetValueFromString< T >, 89
 - LogMessageTypes, 87
 - Modifiers, 86
 - ParameterTypes, 87
 - PcvMeasurement, 88
 - PgvMeasurement, 88
 - PxvSilMeasurement, 88
 - Results, 88
 - SensorErrors, 88
 - SensorTypes, 87
 - SensorVsxVersion, 89
 - SensorXmlVersion, 89
 - SessionStatus, 88
 - SessionTypes, 88
 - Sharpness, 88
 - Sizes, 87
 - SrMeasurement, 88
 - Succ, 90
 - ValueTypes, 87
 - VcVsxVersion, 89
 - VcXmlVersion, 89
- PF.VsxProtocolDriver.VsxProtocolDriver, 90
 - FirmwareVersion, 94
 - GetFirmwareVersion, 93
 - GetSingleUdpDevice, 92
 - GetUdpDeviceList, 92
 - InitSerialDevice, 91
 - InitTcpDevice, 91
 - Save3DPointCloudData, 93
 - SaveData, 93
 - SetNetworkSettingsViaUdp, 92
 - Succ, 93
 - UdpDeviceList, 92
- PF.VsxProtocolDriver.VsxProtocolDriverSync, 94
 - DeviceList, 98
 - FirmwareVersion, 98
 - GetFirmwareVersion, 97
 - GetSingleUdpDevice, 96
 - GetUdpDeviceList, 96
 - InitSerialDevice, 95
 - InitTcpDevice, 95
 - Save3DPointCloudData, 96
 - SaveData, 96
 - SetNetworkSettingsViaUdp, 96
 - Succ, 97
 - UdpDeviceList, 96
- PgvMeasurement
 - PF.VsxProtocolDriver.Types.Vsx, 88
- Points
 - PF.VsxProtocolDriver.Types.ILineSingle, 18
- PrincipalPointU

- PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 33
- PrincipalPointV
 - PF.VsxProtocolDriver.Types.IVsxDisparityDescriptor, 33
- PxxSilMeasurement
 - PF.VsxProtocolDriver.Types.Vsx, 88
- Quality
 - PF.VsxProtocolDriver.Types.ICoordinate, 13
- QuaternionQ0
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- QuaternionQ1
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- QuaternionQ2
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- QuaternionQ3
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- ReConnectAndLoginTcpDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 51
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- ReConnectSerialDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 51
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- ReConnectTcpDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 50
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 67
- ResetDynamicContainerGrabber
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 57
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 70
- ResetLogMessageGrabber
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 58
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 70
- Results
 - PF.VsxProtocolDriver.Types.Vsx, 88
- ResultString
 - PF.VsxProtocolDriver.Types.IVsxResultData, 83
- Revision
 - PF.VsxProtocolDriver.Types.IFirmwareVersion, 16
- RobotData
 - PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData, 46
- RotaryEncoder
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
- Save3DPointCloudData
 - PF.VsxProtocolDriver.VsxProtocolDriver, 93
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- SaveData
 - PF.VsxProtocolDriver.VsxProtocolDriver, 93
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- SaveParameterSetOnDevice
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 57
- PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- Scale
 - PF.VsxProtocolDriver.Types.IVsxLineData, 39
- ScaleXYZ
 - PF.VsxProtocolDriver.Types.IVsxLineData, 39
- SegmentId
 - PF.VsxProtocolDriver.Types.ICoordinate, 13
- SendFirmware
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 55
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- SendSessionKeepAlive
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 62
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 72
- SendTestSystemCommand
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 55
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- SendXmlDataMessage
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 59
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- SensorErrors
 - PF.VsxProtocolDriver.Types.Vsx, 88
- SensorName
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
- SensorTime
 - PF.VsxProtocolDriver.Types.IStatusItem, 25
- SensorType
 - PF.VsxProtocolDriver.Types.IVsxDeviceInformation, 30
- SensorTypes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- SensorVsxVersion
 - PF.VsxProtocolDriver.Types.Vsx, 89
- SensorXmlVersion
 - PF.VsxProtocolDriver.Types.Vsx, 89
- SerialConnectionType
 - PF.VsxProtocolDriver.Types, 11
- SessionStatus
 - PF.VsxProtocolDriver.Types.Vsx, 88
- SessionType
 - PF.VsxProtocolDriver.Types.IVsxSessionData, 84
- SessionTypes
 - PF.VsxProtocolDriver.Types.Vsx, 88
- SetMultipleParameterValues
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 54
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68
- SetNetworkSettings
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 55
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- SetNetworkSettingsViaUdp
 - PF.VsxProtocolDriver.VsxProtocolDriver, 92
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- SetPassword
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 61
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 72
- SetSingleParameterValue
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 53, 54
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 68

- SettingsVersion
 - PF.VsxProtocolDriver.Types.IParameter, 20
 - PF.VsxProtocolDriver.Types.IStatusItem, 24
- Sharpness
 - PF.VsxProtocolDriver.Types.Vsx, 88
- Sizes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- SrMeasurement
 - PF.VsxProtocolDriver.Types.Vsx, 88
- Status
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
 - PF.VsxProtocolDriver.Types.IFirmwareState, 14
- StatusItemId
 - PF.VsxProtocolDriver.Types.IStatusItem, 24
- StatusItems
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- Step
 - PF.VsxProtocolDriver.Types.IParameter, 21
- Strategy
 - PF.VsxProtocolDriver.Types, 11
- SubscribeToDeviceStatusData
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 60
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- Succ
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 72
 - PF.VsxProtocolDriver.Types.Vsx, 90
 - PF.VsxProtocolDriver.VsxProtocolDriver, 93
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 97
- TagListOfContainedMessages
 - PF.VsxProtocolDriver.Types.IVsxDynamicContainer, 34
- TargetPosition
 - PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData, 46
- Time
 - PF.VsxProtocolDriver.Types.IStatusItem, 25
- Timeout
 - PF.VsxProtocolDriver.Types.IVsxSessionData, 84
- Timestamp
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 42
- TransformationMatrix
 - PF.VsxProtocolDriver.Types.IVsxImage, 36
- TranslationTX
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- TranslationTY
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- TranslationTZ
 - PF.VsxProtocolDriver.Types.IVsxTransformation, 85
- TriggerCounter
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 42
- TriggerCtr
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 27
- TriggerSource
 - PF.VsxProtocolDriver.Types.IVsxCaptureInformation, 28
 - PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation, 44
- Type
 - PF.VsxProtocolDriver.Types.IParameter, 20
- UdpDeviceList
 - PF.VsxProtocolDriver.VsxProtocolDriver, 92
 - PF.VsxProtocolDriver.VsxProtocolDriverSync, 96
- Unit
 - PF.VsxProtocolDriver.Types.IParameter, 22
- UnsubscribeToDeviceStatusData
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 60
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- UploadData
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 59
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 71
- UploadParameterSet
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 56
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 69
- Valid
 - PF.VsxProtocolDriver.Types.ICoordinate, 13
- Value
 - PF.VsxProtocolDriver.Types.IParameter, 21
 - PF.VsxProtocolDriver.Types.IStatusItem, 25
- ValueType
 - PF.VsxProtocolDriver.Types.IParameter, 21
 - PF.VsxProtocolDriver.Types.IStatusItem, 25
- ValueTypes
 - PF.VsxProtocolDriver.Types.Vsx, 87
- VcVsxVersion
 - PF.VsxProtocolDriver.Types.Vsx, 89
- VcXmlVersion
 - PF.VsxProtocolDriver.Types.Vsx, 89
- Visible
 - PF.VsxProtocolDriver.Types.IParameter, 21
- VsxDataType
 - PF.VsxProtocolDriver.Types.IVsxData, 29
- VsxType
 - PF.VsxProtocolDriver.Types, 11
- WaitTimeout
 - PF.VsxProtocolDriver.IVsxProtocolDriver, 62
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 80
- Width
 - PF.VsxProtocolDriver.Types.IVsxImage, 37
- WorldCoordinate
 - PF.VsxProtocolDriver.Types.ICoordinate, 12
- X
 - PF.VsxProtocolDriver.Types.IPoint2D, 22
 - PF.VsxProtocolDriver.Types.IPoint3D, 23
- XmlVersion
 - PF.VsxProtocolDriver.IVsxProtocolDriverSync, 79

Y

PF.VsxProtocolDriver.Types.IPoint2D, [22](#)

PF.VsxProtocolDriver.Types.IPoint3D, [23](#)

Z

PF.VsxProtocolDriver.Types.IPoint3D, [23](#)