VsxProtocolDriver

3.3.2+g6a18e23

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

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 propably 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 whithin 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.

V3.3.1

- 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:

7 Hierarchical Index

7.1 Class Hierarchy

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

8 Class Index

8.1 Class List

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

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-Scan3dPrincipalPointU)∗Scan3dBaseline/D(u, v) Y(u, v) = (v-Scan3dPrincipalPointV)∗Scan3dBaseline/D(u, v) Z(u, v) = Scan3dFocalLength∗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 {
  VSX_DRIVER_NO_ERROR = 0x0 , VSX_DRIVER_INIT_ERROR = -0x1 , VSX_DRIVER_TIMEOUT_ERROR = -0x2 , VSX_DRIVER_SAVE_FILE_ERROR = -0x3 ,
  VSX_DRIVER_DATA_ERROR = -0x4 , VSX_DRIVER_CONNECTION_ERROR = -0x5 , VSX_DRIVER_INVALID_DATA_ERRO = -0x6 , VSX_DRIVER_DEVICE_ERROR = -0x7 ,
  VSX_DRIVER_LOAD_FILE_ERROR = -0x8 , VSX_SESSION_ERROR = -0x9 , VSX_STRING_ERROR = -0x0A , VSX_VERSION_ERROR = -0x0B ,
  VSX_DRIVER_GENERAL_ERROR = -0x1000 }

  The id of the error.
- enum DisconnectEvent { RemoteHostConnectionClosed , DisconnectCalled , ConnectionError }

  The disconnect reason.
- enum Strategy { DROP_OLDEST , DROP_WRITE }

  The strategy which containers are removed when max number of items is reached.
- enum SerialConnectionType {
  USB_SSI , PROFIBUS , PROFINET , ETHERNET_IP ,
  RS485 , CANOPEN }

  The serial connection type.
- enum DeviceStatusScope { FULL , MULTI }

  Flag about the verbosity of the status message.

- enum Content : int {
  C = 0 , X , Y , Z ,
  Q , I , R , LINE_ID ,
  SEGMENT_ID , R0 , R1 , R2 ,
  R3 , R4 , R5 , R6 ,
  LENGTH }
- enum VsxType {
  VsxCaptureInformation , VsxDisparityDescriptor , VsxFile , VsxImage ,
  VsxLineData , VsxLogData , VsxResultData , VsxSessionData ,
  VsxTransformation , VsxDynamicContainer , VsxOlr2ModbusData , VsxOlr2CaptureInformation ,
  VsxApplicationResultData }

  The device data type.
- enum ImageFormat {
  SW8bit = 1 , SW16bit = 2 , SW32bit = 3 , Bayer8bit = 4 ,
  RGB24bit = 5 , Unknown = 255 , StandardFormat = 256 }
- enum ImageData2Format {
  Mono8 = 0x01080001 , Mono12 = 0x01100005 , Mono16 = 0x01100007 , Coord3D_A16 = 0x011000B6 ,
  Coord3D_B16 = 0x011000B7 , Coord3D_C16 = 0x011000B8 , Confidence8 = 0x010800C6 , Data8 = 0x01080116 ,
  Coord3D_A32f = 0x012000BD , Coord3D_B32f = 0x012000BE , Coord3D_C32f = 0x012000BF }

9.3.1 Enumeration Type Documentation

ErrorId

`enum PF.VsxProtocolDriver.Types.ErrorId`

The id of the error.

Enumerator

VSX_DRIVER_NO_ERROR	No error.
VSX_DRIVER_INIT_ERROR	Init error.
VSX_DRIVER_TIMEOUT_ERROR	Timeout error.
VSX_DRIVER_SAVE_FILE_ERROR	Save file error.
VSX_DRIVER_DATA_ERROR	Data error.
VSX_DRIVER_CONNECTION_ERROR	Connection error.
VSX_DRIVER_INVALID_DATA_ERROR	Invalid data error.
VSX_DRIVER_DEVICE_ERROR	Device error.
VSX_DRIVER_LOAD_FILE_ERROR	Load file error.
VSX_SESSION_ERROR	Device session error.
VSX_STRING_ERROR	String error.
VSX_VERSION_ERROR	Version error.
VSX_DRIVER_GENERAL_ERROR	General error.

DisconnectEvent

`enum PF.VsxProtocolDriver.Types.DisconnectEvent`

The disconnect reason.

Enumerator

RemoteHostConnectionClosed	
DisconnectCalled	
ConnectionError	

Strategy

`enum PF.VsxProtocolDriver.Types.Strategy`

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

Enumerator

DROP_OLDEST	Discards the oldest saved item if max number of items is reached.
DROP_WRITE	Discards the current item if max number of items is reached.

SerialConnectionType

`enum PF.VsxProtocolDriver.Types.SerialConnectionType`

The serial connection type.

Enumerator

USB_SSI	</summary>
PROFIBUS	</summary>
PROFINET	</summary>
ETHERNET_IP	</summary>
RS485	</summary>
CANOPEN	</summary>

DeviceStatusScope

`enum PF.VsxProtocolDriver.Types.DeviceStatusScope`

Flag about the verbosity of the status message.

Enumerator

FULL	All status items are included.
MULTI	Single or multiple status items are included.

Content

```
enum PF.VsxProtocolDriver.Types.Content :  int
```

Enumerator

C	</summary>
X	</summary>
Y	</summary>
Z	</summary>
Q	</summary>
I	</summary>
R	</summary>
LINE_ID	</summary>
SEGMENT_ID	</summary>
R0	</summary>
R1	</summary>
R2	</summary>
R3	</summary>
R4	</summary>
R5	</summary>

Enumerator

R6	</summary>
LENGTH	</summary>

VsxType

enum PF.VsxProtocolDriver.Types.VsxType

The device data type.

Enumerator

VsxCaptureInformation	
VsxDisparityDescriptor	
VsxFile	
VsxImage	
VsxLineData	
VsxLogData	
VsxResultData	
VsxSessionData	
VsxTransformation	
VsxDynamicContainer	
VsxOlr2ModbusData	
VsxOlr2CaptureInformation	
VsxApplicationResultData	

ImageFormat

enum PF.VsxProtocolDriver.Types.ImageFormat

Enumerator

SW8bit	
SW16bit	
SW32bit	
Bayer8bit	
RGB24bit	
Unknown	
StandardFormat	

ImageData2Format

enum PF.VsxProtocolDriver.Types.ImageData2Format

Enumerator

Mono8	
Mono12	
Mono16	
Coord3D_A16	
Coord3D_B16	
Coord3D_C16	
Confidence8	
Data8	
Coord3D_A32f	
Coord3D_B32f	
Coord3D_C32f	

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 coodinate.
- 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

ImageCoordinate

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

Gets or sets the image coordinate.

The image coordinate.

WorldCoordinate

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

Gets or sets the world coordinate.

The world coordinate.

Intensity

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

Intensity for this coodinate.

Quality

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

Quality for this coordinate.

Valid

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

Is coordinate valid?

LineId

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

The Id of the line the coordinate corresponds to.

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

Id

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

The id of the error.

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

Status

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

A status tag.

Msg

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

A status message.

Error

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

A status error tag.

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

Major

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

Major version.

Minor

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

Minor version.

Build

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

Build version.

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

Id

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

The Id of the item tuple.

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

Lines

```
List<ILineSingle> PF.VsxProtocolDriver.Types.ILineMulti.Lines  [get]
```

The list of the lines.

MinX

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

The lines min X value.

MaxX

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

The lines max X value.

MinZ

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

The lines min Z value.

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

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 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 dependend of the parameters ValueType.
- object Max `[get]`

  The maximum value of the parameter, type dependend of the parameters ValueType.
- object Step `[get]`

  The step size of the parameter, type dependend of the parameters ValueType.
- object Value `[get, set]`

  The current value of the parameter, type dependend of 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 dependend of the parameters ValueType.
- string Unit `[get]`

  The unit of the parameter.
- List< IItemTuple > 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

Clone()

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

Copy constructor.

Returns

10.8.3 Property Documentation

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).

ConfigVersion

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

The config version of the configuration node the parameter belongs to.

ConfigId

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

The configuration id of the parameter. The cref=ConfigId together with the ParameterId is unique.

ParameterId

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

The parameter id of the parameter. The cref=ConfigId together with the ParameterId is unique.

Name

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

The display name of the parameter.

Type

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

The type of the parameter which determines the control type.

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.

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.

Visible

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

Determines if the parameter is visible or not.

Min

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

The minimum value of the parameter, type dependend of the parameters ValueType.

Max

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

The maximum value of the parameter, type dependend of the parameters ValueType.

Step

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

The step size of the parameter, type dependend of the parameters ValueType.

Value

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

The current value of the parameter, type dependend of the parameters ValueType. Note: The value is either bool, a string, a 64 bit integer or a 64 bit floating point double value.

DefaultValue

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

The default value of the parameter, type dependend of the parameters ValueType.

Unit

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

The unit of the parameter.

Items

```
List<IItemTuple> 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

X

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

The x coordinate.

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

X

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

The x coordinate.

Y

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

The y coordinate.

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< IItemTuple > 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

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).

ConfigVersion

```
ushort PF.VsxProtocolDriver.Types.IStatusItem.ConfigVersion [get]
```

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

StatusItemId

```
string PF.VsxProtocolDriver.Types.IStatusItem.StatusItemId  [get]
```

The id of the status item.

Name

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

The display name of the status item.

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.

Value

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

The current value of the status item, type dependend of the status items ValueType.

Items

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

List of possible values.

Time

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

Last changed timestamp of the status signal.

SensorTime

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

Current timestamp of the sensor clock.

ConfigurationClass

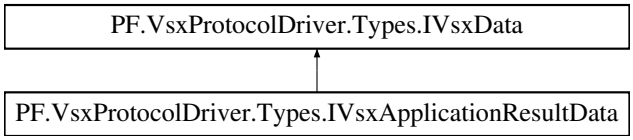
```
string PF.VsxProtocolDriver.Types.IStatusItem.ConfigurationClass  [get]
```

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]`

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType `[get]`

10.12.1 Detailed Description

Application result data received from device.

10.12.2 Property Documentation

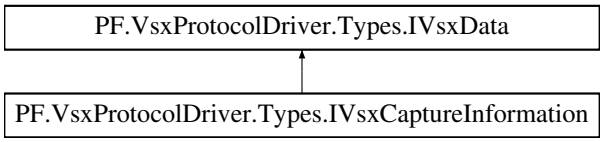
ApplicationResultString

```
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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [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

TriggerCtr

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

ParameterId

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

JobId

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

RotaryEncoder

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

FrameCounter

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

Timestamp

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

ExposureTime

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

Gain

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

Illumination

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

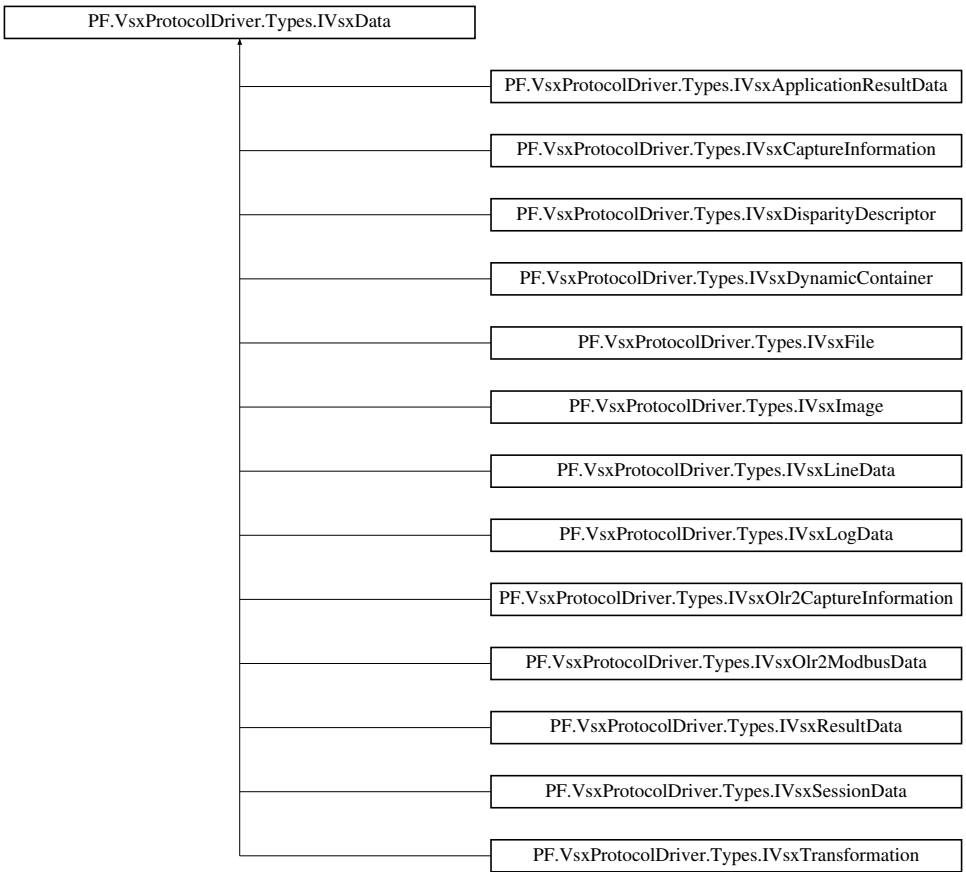
TriggerSource

```
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

VsxDataType

```
VsxType 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.

10.15.1 Detailed Description

Contains the information of a device.

10.15.2 Property Documentation

IpAddress

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

NetworkMask

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

The network mask of the device.

Gateway

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

The standard gateway of the device.

MacAddress

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

The MAC address of the device.

FirmwareVersion

```
string PF.VsxProtocolDriver.Types.IVsxDeviceInformation.FirmwareVersion  [get]
```

The firmware version string of the device.

SensorType

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

The type of the device.

SensorName

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

The name of the device.

Busy

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

Indicates if device is already connected or not.

DeviceVsxVersionMajor

```
int PF.VsxProtocolDriver.Types.IVsxDeviceInformation.DeviceVsxVersionMajor  [get]
```

The device major VSX protocol version.

DeviceVsxVersionMinor

```
int PF.VsxProtocolDriver.Types.IVsxDeviceInformation.DeviceVsxVersionMinor  [get]
```

The device minor VSX protocol version.

ComPort

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

The comport if device is a serial one.

Baudrate

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

The baudrate if device is a serial one.

Headaddress

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

The headaddress if device is a serial one.

IsLoginNeeded

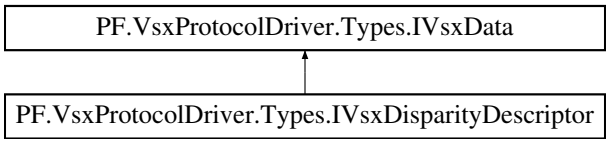
```
bool PF.VsxProtocolDriver.Types.IVsxDeviceInformation.IsLoginNeeded  [get]
```

Gets a value indicating whether the device needs login.

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-Scan3dPrincipalPointU)∗↩ Scan3dBaseline/D(u, v) Y(u, v) = (v-Scan3dPrincipalPointV)∗Scan3dBaseline/D(u, v) Z(u, v) = Scan3dFocal↩ Length∗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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [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-Scan3dPrincipalPointU)∗↩ Scan3dBaseline/D(u, v) Y(u, v) = (v-Scan3dPrincipalPointV)∗Scan3dBaseline/D(u, v) Z(u, v) = Scan3dFocal↩ Length∗Scan3dBaseline/D(u, v)

10.16.2 Property Documentation

FocalLength

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

PrincipalPointU

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

PrincipalPointV

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

Baseline

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

OffsetLeftRectifiedToDisparityU

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

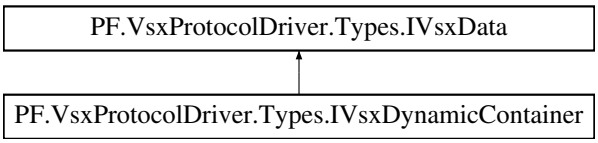
OffsetLeftRectifiedToDisparityV

```
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]`

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType `[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

ContainsMessage()

```
bool PF.VsxProtocolDriver.Types.IVsxDynamicContainer.ContainsMessage (
            string tag )
```

GetMessage()

```
IVsxData PF.VsxProtocolDriver.Types.IVsxDynamicContainer.GetMessage (
            string tag )
```

Dispose()

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

Clone()

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

10.17.3 Property Documentation

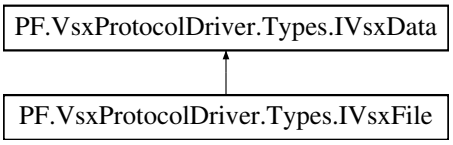
TagListOfContainedMessages

```
List<string> PF.VsxProtocolDriver.Types.IVsxDynamicContainer.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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [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

FileStatus

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

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

Path

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

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

MimeType

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

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

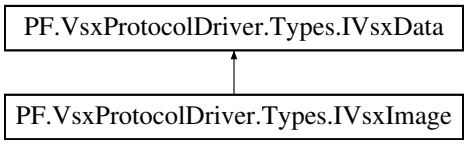
Contents

```
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.

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

10.19.1 Detailed Description

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

10.19.2 Property Documentation

ImageType

```
ImageFormat PF.VsxProtocolDriver.Types.IVsxImage.ImageType [get]
```

TransformationMatrix

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

Image

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

ImageData

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

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

Format

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

Width

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

Height

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

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.

FrameCounter

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

64 bit counter with cyclic overflow

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.

CoordinateOffset

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

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

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).

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).

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)

ImageDataFloats

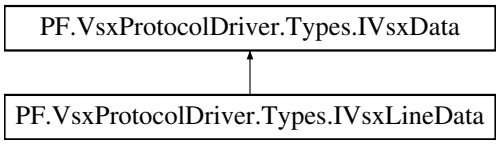
```
float [] PF.VsxProtocolDriver.Types.IVsxImage.ImageDataFloats  [get]
```

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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [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] = Data value C / ScaleC
Position image line: R [px] = Counter + 0.5
Position world: X [mm] = Data value X / ScaleXYZ
Position world: Y [mm] = Data value Y / ScaleXYZ
Position world: Z [mm] = Data value Z / ScaleXYZ
Goodness: Q [%] = Data value Q / (2^15-1) ∗ 100
Brightness: I [%] = 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

CountLines

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

Format

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

Scale

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

ScaleXYZ

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

MinX

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

MaxX

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

MinZ

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

MaxZ

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

FrameCounter

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

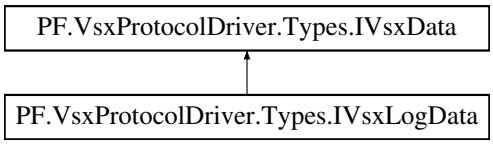
Lines

```
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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

10.21.1 Detailed Description

A log message received from device.

10.21.2 Property Documentation

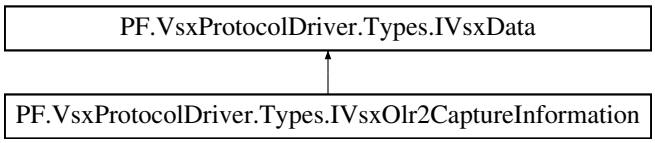
Log

```
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.

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

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

FrameCounter

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

Number of captured lines.

TriggerCounter

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

Number of module triggers (includes missed trigger events).

CurrentPosition

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

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

IoState

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

Current state of the sensor GPIs.

Timestamp

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

Monotonic clock counter, unit=µs.

LmaExposureTime1

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

Main exposure time of laser module A, unit=µs.

LmaExposureTime2

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

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

LmbExposureTime1

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

Main exposure time of laser module B, unit=µs.

LmbExposureTime2

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

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

LmaRoiOffsetX

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

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

LmaRoiLengthX

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

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

LmaRoiOffsetZ

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

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

LmaRoiLengthZ

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

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

LmbRoiOffsetX

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

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

LmbRoiLengthX

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

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

LmbRoiOffsetZ

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

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

LmbRoiLengthZ

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

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

AutoTriggerFrameRate

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

Auto trigger frame rate, unit=Hz.

TriggerSource

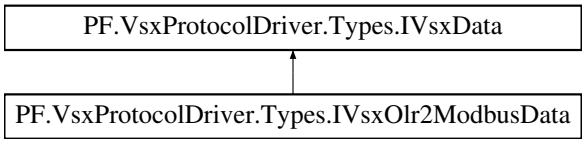
```
byte PF.VsxProtocolDriver.Types.IVsxOlr2CaptureInformation.TriggerSource [get]
```

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.

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType `[get]`

10.23.1 Detailed Description

Contains data which was set via modbus protocol.

10.23.2 Property Documentation

ActivationTimer

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

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

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.

TargetPosition

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

Parameterized rotational offset between rotating and fixed sensor side.

RobotData

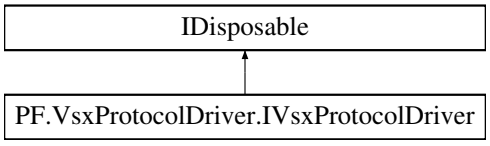
```
ushort [] PF.VsxProtocolDriver.Types.IVsxOlr2ModbusData.RobotData  [get]
```

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.↩ Vsxport)

  Disconnects the device and reconnects with new connection settings.
- Task<(bool Succ, IError ErrorDesc)> 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.
- 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 settings↩ Version, 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 imcomplete parameter file to the device. From Vsx-Version 4.
- Task<(bool Succ, IError ErrorDesc)> SendFirmware (string filename)

  Sends a firmware update file to the device. NOTE: not completely implemented yet, the file is send 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 authorization↩ Password, string username, string newPassword)

  Sets a new password for a device account.

- 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 VsxProtocolDriver.DefaultEthernet↩ TimeoutMs.

- 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 SubscribeToDeviceStatus↩ Data).
- 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 = VsxProtocolDriver.InitTcpDevice(...)` or `IVsxProtocolDriver driver = VsxProtocolDriver.InitSerialDevice(...)` to get an instance and use the driver.

10.24.2 Member Function Documentation

Connect()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.Connect (
            int timeout = VsxProtocolDriver.DefaultEthernetTimeoutMs )
```

Connect with the device.

Parameters

timeout	The timeout for a connection attempt, default is VsxProtocolDriver.DefaultEthernetTimeoutMs.

Returns

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

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

username	The login username.
password	The login password.
timeout	The timeout for a connection attempt.

Returns

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

Disconnect()

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

Disconnect with the device.

Returns

Always true and empty error.

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

ipAddress	The new IPAddress.
port	The new port.

Returns

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

ReConnectAndLoginTcpDevice()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.ReConnectAnd↩
LoginTcpDevice (
            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

ipAddress	The new IPAddress.
username	The login username.
password	The login password.
port	The new port.

Returns

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

ReConnectSerialDevice()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.ReConnectSerial↩
Device (
            string serialPort,
            int baudrate,
            SerialConnectionType connectionType )
```

Disconnects the device and reconnects with new connection settings.

Parameters

serialPort	The new serial port.
baudrate	The new baudrate.
connectionType	The new connection type.

Returns

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

GetDeviceInformation()

```
Task<(bool Succ, IVsxDeviceInformation CurrentDevice, IError ErrorDesc)> PF.VsxProtocol↩
Driver.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.

GetFeatureList()

```
Task<(bool Succ, float XmlVersion, Hashtable FeatureList, IError ErrorDesc)> PF.VsxProtocol↩
Driver.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.

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.

GetSingleParameterValue() [1/2]

```
Task<(bool Succ, object ParameterValue, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocol↩
Driver.GetSingleParameterValue (
            IParameter parameter )
```

Returns the current value of the given parameter from device.

Parameters

parameter	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.

GetSingleParameterValue() [2/2]

```
Task<(bool Succ, object ParameterValue, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocol↩
Driver.GetSingleParameterValue (
            ushort settingsVersion,
            ushort configVersion,
            string configId,
            string parameterId )
```

Returns the current value of the given parameter from device.

Parameters

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

SetSingleParameterValue() [1/2]

```
Task<(bool Succ, List< IParameter > DependendParameters, IError ErrorDesc)> PF.VsxProtocol↩
Driver.IVsxProtocolDriver.SetSingleParameterValue (
            IParameter parameter,
            object value )
```

Sets the parameter to a value on the device.

Parameters

parameter	The parameter the value should be set from.
value	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.

SetSingleParameterValue() [2/2]

```
Task<(bool Succ, List< IParameter > DependendParameters, IError ErrorDesc)> PF.VsxProtocol↩
Driver.IVsxProtocolDriver.SetSingleParameterValue (
            ushort settingsVersion,
            ushort configVersion,
```

```
            string configId,
            string parameterId,
            object value )
```

Sets the parameter to a value on the device.

Parameters

settingsVersion	The settings version of the parameter which should be set.
configVersion	The config version of the parameter which should be set.
configId	The config id of the parameter which should be set.
parameterId	The id of the parameter which should be set.
value	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.

SetMultipleParameterValues()

```
Task<(bool Succ, List< IParameter > DependendParameters, IError ErrorDesc)> PF.VsxProtocol↩
Driver.IVsxProtocolDriver.SetMultipleParameterValues (
            string sourceFileName )
```

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

Parameters

sourceFileName	Path and filename to upload.

///

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.

SendFirmware()

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

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

Parameters

filename	The path and filename of the firmware file.

Returns

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

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

ipAddress	The new IP Address.
networkMask	The new network mask.
gateway	The new gateway.

Returns

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

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

command	The test system command.
inputValue	
timeout	Wait time for device reply.

Returns

DownloadParameterSet()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.DownloadParameter↩
Set (
            string destinationFileName )
```

Save the current parameter set to a file.

Parameters

destinationFileName	Path and file name to save to.

Returns

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

UploadParameterSet() [1/2]

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.UploadParameterSet
(
            string sourceFileName )
```

Uploads a parameter file to the device.

Parameters

sourceFileName	Path and filename to upload.

Returns

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

UploadParameterSet() [2/2]

```
Task<(bool Succ, List< IParameter > DependendParameters, IError ErrorDesc)> PF.VsxProtocol↩
Driver.IVsxProtocolDriver.UploadParameterSet (
            List< IParameter > parameterSet )
```

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

Parameters

parameterSet	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.

SaveParameterSetOnDevice()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SaveParameterSet↩
OnDevice ( )
```

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.

LoadParameterSetOnDevice()

```
Task<(bool Succ, List< IParameter > 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.

LoadDefaultParameterSetOnDevice()

```
Task<(bool Succ, List< IParameter > 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.

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

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

GetDynamicContainer()

```
Task<(bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, IError Error↩
Desc)> PF.VsxProtocolDriver.IVsxProtocolDriver.GetDynamicContainer (
            int timeoutMs = Timeout.Infinite,
            CancellationTokenSource cts = null )
```

Gets the oldest saved item and removes it internally.

Parameters

timeoutMs	The maximum time in ms to try reading an item.
cts	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.

GetCachedContainer()

```
IVsxDynamicContainer PF.VsxProtocolDriver.IVsxProtocolDriver.GetCachedContainer (
            int pos )
```

Gets a cached dynamic container.

Parameters

pos	Position of the container in cache.

Returns

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

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

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

timeoutMs	The maximum time in ms to try reading an item.
cts	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.

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

filename	The path and filename of the data file.

Returns

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

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

xml	The message.

Returns

Always true and empty error.

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.

SubscribeToDeviceStatusData()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SubscribeTo↩
DeviceStatusData ( )
```

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.

UnsubscribeToDeviceStatusData()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.UnsubscribeTo↩
DeviceStatusData ( )
```

Stops cyclic sending of status data from device.

Returns

true/empty error or false/error description.

Login()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.Login (
            string username,
            string password )
```

Sends login data to the device.

Parameters

username	The username.
password	The password.

Returns

true/empty error or false/error description.

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

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

Returns

true/empty error or false/error description.

Logout()

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

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

Returns

true/empty error or false/error description.

SendSessionKeepAlive()

```
Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.IVsxProtocolDriver.SendSessionKeep↩
Alive ( )
```

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

FirmwareStateChannelReader

```
ChannelReader<IFirmwareState> PF.VsxProtocolDriver.IVsxProtocolDriver.FirmwareStateChannel↩
Reader  [get]
```

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

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.Default↩
EthernetTimeoutMs.

MissingContainerFramesCounter

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

Gets the missing frame counter for image grabbing.

MissingLogMessagesCounter

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

Gets the missing log messages counter for log message grabbing.

DynamicContainerQueueSize

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

Gets the current size of the dynamic container message queue.

NumberOfCachedContainers

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

Gets the current number of cached container messages.

LogMessageQueueSize

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

Gets the current size of the log message queue.

ContainerGrabberStrategy

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

Gets the current strategy of the dynamic container message queue.

LogGrabberStrategy

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

Gets the current strategy of the log message queue.

Connected

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

Indicates current connection state with the device.

10.24.4 Event Documentation

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	Errror message as string

Returns

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

OnDeviceStatusReceived

```
Action<DeviceStatusScope, List<IStatusItem> > PF.VsxProtocolDriver.IVsxProtocolDriver.On↩
DeviceStatusReceived
```

Event called when status data is received from device after cyclic sending is started (see SubscribeToDevice↩ StatusData).

Returns

Device status scope wheter 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

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.

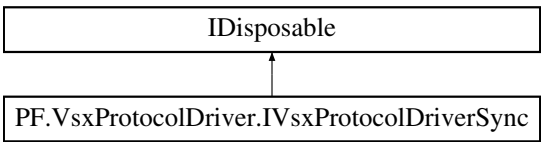
Returns

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 config↩ Version, string configId, string parameterId, object value)
- bool List< IParameter > IError ErrorDesc SetMultipleParameterValues (string sourceFileName)
- bool IError ErrorDesc SendFirmware (string filename)
- 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, CancellationTokenSource 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, CancellationToken↩ Source 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 username, string newPassword)
- 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.DefaultEthernet↩ TimeoutMs.
- 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 SubscribeToDeviceStatus↩ Data).
- 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

Connect()

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

ConnectAndLogin()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.ConnectAndLogin (
            string username,
            string password,
            int timeout = VsxProtocolDriver.ConnectionTimeoutMs )
```

Disconnect()

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

ReConnectTcpDevice()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.ReConnectTcpDevice (
            string ipAddress,
            int port = VsxProtocolDriver.Vsxport )
```

ReConnectAndLoginTcpDevice()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.ReConnectAndLoginTcpDevice (
            string ipAddress,
            string username,
            string password,
            int port = VsxProtocolDriver.Vsxport )
```

ReConnectSerialDevice()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.ReConnectSerialDevice (
            string serialPort,
            int baudrate,
            SerialConnectionType connectionType )
```

GetDeviceInformation()

```
bool IVsxDeviceInformation IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Get↩
DeviceInformation ( )
```

GetFeatureList()

```
bool float Hashtable IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetFeature↩
List ( )
```

GetParameterList()

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Get↩
ParameterList ( )
```

GetSingleParameterValue() [1/2]

```
bool object IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetSingleParameter↩
Value (
            IParameter parameter )
```

GetSingleParameterValue() [2/2]

```
bool object IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetSingleParameter↩
Value (
            ushort settingsVersion,
            ushort configVersion,
            string configId,
            string parameterId )
```

SetSingleParameterValue() [1/2]

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Set↩
SingleParameterValue (
            IParameter parameter,
            object value )
```

SetSingleParameterValue() [2/2]

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Set↩
SingleParameterValue (
            ushort settingsVersion,
            ushort configVersion,
            string configId,
            string parameterId,
            object value )
```

SetMultipleParameterValues()

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Set↩
MultipleParameterValues (
            string sourceFileName )
```

SendFirmware()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SendFirmware (
            string filename )
```

SetNetworkSettings()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SetNetworkSettings (
            string ipAddress,
            string networkMask,
            string gateway )
```

SendTestSystemCommand()

```
bool string string string IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Send↩
TestSystemCommand (
            string command,
            string inputValue,
            int timeout = VsxProtocolDriver.DefaultEthernetTimeoutMs )
```

DownloadParameterSet()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.DownloadParameterSet (
            string destinationFileName )
```

UploadParameterSet() [1/2]

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UploadParameterSet (
            string sourceFileName )
```

UploadParameterSet() [2/2]

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Upload↩
ParameterSet (
            List< IParameter > parameterSet )
```

SaveParameterSetOnDevice()

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

LoadParameterSetOnDevice()

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Load↩
ParameterSetOnDevice ( )
```

LoadDefaultParameterSetOnDevice()

```
bool List< IParameter > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.Load↩
DefaultParameterSetOnDevice ( )
```

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

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

GetDynamicContainer()

```
bool IVsxDynamicContainer int IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.↩
GetDynamicContainer (
            int timeoutMs = Timeout.Infinite,
            CancellationTokenSource cts = null )
```

GetCachedContainer()

```
IVsxDynamicContainer PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetCachedContainer (
            int pos )
```

Gets a cached dynamic container.

Parameters

pos	Position of the container in cache.

Returns

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

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

GetLogMessage()

```
bool IVsxLogData int IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetLog↩
Message (
```

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

UploadData()

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

SendXmlDataMessage()

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

GetAllDeviceStatusData()

```
bool List< IStatusItem > IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.GetAll↩
DeviceStatusData ( )
```

SubscribeToDeviceStatusData()

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

UnsubscribeToDeviceStatusData()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.UnsubscribeToDeviceStatus↩
Data ( )
```

Login()

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

SetPassword()

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

Logout()

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

SendSessionKeepAlive()

```
bool IError ErrorDesc PF.VsxProtocolDriver.IVsxProtocolDriverSync.SendSessionKeepAlive ( )
```

10.25.3 Member Data Documentation

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.

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 send only to the device.

Uploads an imcomplete 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

timeout	The timeout for a connection attempt, default is VsxProtocolDriver.DefaultEthernetTimeoutMs.

Returns

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

Parameters

username	The login username.
password	The login password.
timeout	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

ipAddress	The new IPAddress.
port	The new port.

Returns

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

Parameters

ipAddress	The new IPAddress.
username	The login username.
password	The login password.
port	The new port.

Returns

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

Parameters

serialPort	The new serial port.
baudrate	The new baudrate.
connectionType	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

parameter	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

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

parameter	The parameter the value should be set from.
value	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

settingsVersion	The settings version of the parameter which should be set.
configVersion	The config version of the parameter which should be set.
configId	The config id of the parameter which should be set.
parameterId	The id of the parameter which should be set.
value	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

sourceFileName	Path and filename to upload.

///

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

filename	The path and filename of the firmware file.

Returns

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

Parameters

ipAddress	The new IP Address.
networkMask	The new network mask.
gateway	The new gateway.

Returns

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

Parameters

command	The test system command.
inputValue	
timeout	Wait time for device reply.

Returns

Parameters

destinationFileName	Path and file name to save to.

Returns

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

Parameters

sourceFileName	Path and filename to upload.

Returns

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

Parameters

parameterSet	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

timeoutMs	The maximum time in ms to try reading an item.
cts	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

timeoutMs	The maximum time in ms to try reading an item.
cts	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

filename	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

xml	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

username	The username.
password	The password.

Returns

true/empty error or false/error description.

Parameters

authorizationUsername	The username who wants to set the password.
authorizationPassword	The password of the user who wants to set the password.

Parameters

username	The username of the account for which the new password should be set.
newPassword	The new password.

Returns

true/empty error or false/error description.

CurrentDevice

```
bool IVsxDeviceInformation PF.VsxProtocolDriver.IVsxProtocolDriverSync.CurrentDevice
```

XmlVersion

```
bool float PF.VsxProtocolDriver.IVsxProtocolDriverSync.XmlVersion
```

FeatureList

```
bool float Hashtable PF.VsxProtocolDriver.IVsxProtocolDriverSync.FeatureList
```

ParameterList

```
bool List< IParameter > PF.VsxProtocolDriver.IVsxProtocolDriverSync.ParameterList
```

ParameterValue

```
bool object PF.VsxProtocolDriver.IVsxProtocolDriverSync.ParameterValue
```

DependendParameters

```
bool List< IParameter > PF.VsxProtocolDriver.IVsxProtocolDriverSync.DependendParameters
```

Command

```
bool string PF.VsxProtocolDriver.IVsxProtocolDriverSync.Command
```

OutputValue

```
bool string string PF.VsxProtocolDriver.IVsxProtocolDriverSync.OutputValue
```

Status

```
bool string string string PF.VsxProtocolDriver.IVsxProtocolDriverSync.Status
```

Container

```
bool IVsxDynamicContainer PF.VsxProtocolDriver.IVsxProtocolDriverSync.Container
```

NumberOfDiscardedItems [1/2]

```
bool IVsxDynamicContainer int PF.VsxProtocolDriver.IVsxProtocolDriverSync.NumberOfDiscarded↩
Items
```

LogMessage

```
bool IVsxLogData PF.VsxProtocolDriver.IVsxProtocolDriverSync.LogMessage
```

NumberOfDiscardedItems [2/2]

```
bool IVsxLogData int PF.VsxProtocolDriver.IVsxProtocolDriverSync.NumberOfDiscardedItems
```

StatusItems

```
bool List<IStatusItem> PF.VsxProtocolDriver.IVsxProtocolDriverSync.StatusItems
```

10.25.4 Property Documentation

FirmwareStateChannelReader

```
ChannelReader<IFirmwareState> PF.VsxProtocolDriver.IVsxProtocolDriverSync.FirmwareState↩
ChannelReader [get]
```

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

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.Default↩
EthernetTimeoutMs.

MissingContainerFramesCounter

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

Gets the missing frame counter for image grabbing.

MissingLogMessagesCounter

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

Gets the missing log messages counter for log message grabbing.

DynamicContainerQueueSize

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

Gets the current size of the dynamic container message queue.

NumberOfCachedContainers

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

Gets the current number of cached container messages.

LogMessageQueueSize

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

Gets the current size of the log message queue.

ContainerGrabberStrategy

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

Gets the current strategy of the dynamic container message queue.

LogGrabberStrategy

```
Strategy PF.VsxProtocolDriver.IVsxProtocolDriverSync.LogGrabberStrategy  [get]
```

Gets the current strategy of the log message queue.

Connected

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

Indicates current connection state with the device.

10.25.5 Event Documentation

OnDisconnect

```
Action<string, DisconnectEvent, string> PF.VsxProtocolDriver.IVsxProtocolDriverSync.OnDisconnect
```

Event called when a TPC/IP device is disconnected.

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

Returns

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

OnDeviceStatusReceived

```
Action<DeviceStatusScope, List<IStatusItem> > PF.VsxProtocolDriver.IVsxProtocolDriverSync.↩
OnDeviceStatusReceived
```

Event called when status data is received from device after cyclic sending is started (see SubscribeToDevice↩ StatusData).

Returns

Device status scope wheter 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

OnSessionMessageReceived

```
Action<IVsxSessionData> PF.VsxProtocolDriver.IVsxProtocolDriverSync.OnSessionMessageReceived
```

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

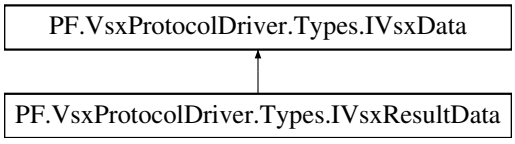
Returns

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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

10.26.1 Detailed Description

Result data received from device.

10.26.2 Property Documentation

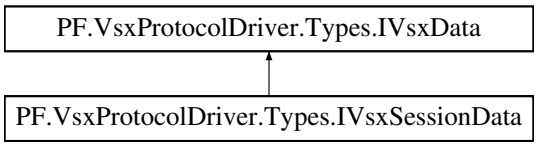
ResultString

```
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.

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

10.27.1 Detailed Description

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

10.27.2 Property Documentation

SessionType

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

The session type of the message.

Message

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

The message string.

Timeout

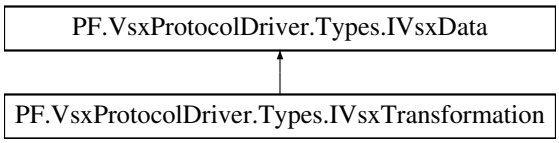
```
TimeSpan PF.VsxProtocolDriver.Types.IVsxSessionData.Timeout  [get]
```

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]

Properties inherited from PF.VsxProtocolDriver.Types.IVsxData

- VsxType VsxDataType [get]

10.28.1 Detailed Description

Used to transform raw point cloud data from device.

10.28.2 Property Documentation

TranslationTX

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

TranslationTY

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

TranslationTZ

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

QuaternionQ0

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

QuaternionQ1

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

QuaternionQ2

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

QuaternionQ3

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

10.29 PF.VsxProtocolDriver.Types.Vsx Class Reference

Public Types

- enum Modifiers {
  SetSingleData , GetSingleData , GetSpecificSingleData , SetConfigData ,
  GetConfigData , SetAllData , GetAllData , SaveData ,
  LoadData , LoadDefaultData , Echo , SetNetwork ,
  GetNetwork , SetLog , GetLog , Error ,
  SetUserData , GetUserData , Version , Disconnected ,
  AcceptData , TestSystem , Discovery , RestartNetwork ,
  GetAllSensorStatus , SubscribeSensorStatus , UnsubscribeSensorStatus , GetMultiSensorStatus ,
  SensorStatus , DiscoveryReply , SetMultipleData , Unknown }
- enum FunctionAttributes { Modifier , Reply , Check , Unknown }
- enum Elements {
  FUNCTION , SENSOR , SETTINGS , CONFIGURATION ,
  CATEGORY , PARAMETER , STRUCTURE , NETWORK ,
  LOG , ELEMENT , ITEM , VSX ,
  FEATURE , RESULT , ERROR , TEST ,
  IMAGETYPE , STATUS , MEASUREMENT , DATA ,
  DRAW , SESSION , PWLOGIN , PWSET ,
  TIMELEFT , INFORMATION , PORT , UNKNOWN }
- enum Attributes {
  Id , Name , Type , ValueType ,
  Default , Value , Help , Enable ,
  Visible , Location , UserLevel , Min ,
  Max , Step , Scale , Unit ,
  DecimalPlaces , MinLength , MaxLength , ReadOnly ,
  Multiple , MessageBoxType , IconType , Text ,
  FontSize , Color , IPAddress , Firmware ,
  MacAddress , TypeMask , Mask , Version ,
  StartupCommand , MessageIndex , Gateway , VCVersion ,
  FrameCounter , Command , InputValue , OutputValue ,
  Status , Identifier , Image , LinkSpeed ,
  Time , FillColor , Number , Position ,
  Orientation , IncompleteParameterset , Port , Baudrate ,
  HeadAddress , ConstrainGE , ConstrainLE , ConstrainGT ,
  ConstrainLT , ConstrainIN , ConstrainOUT , SubId ,
  State , Class , Protocol , Message ,
  Seconds , Username , Password , Busy ,
  TransportSecurity , Transport , Authentication , VsxVersion ,
  Unknown }
- enum ConstantValues {
  On , Off , lin , log ,
  Ok , OkCancel , YesNo , Information ,
  Error , Question , CONF_NAME , AlwaysVisible ,
  Update , Yes , No , Cancel ,
  Safe , Measurement , Unknown }
- enum ParameterTypes {
  Checkbox , Signal , Slider , Label ,
  Spinbox , Combobox , RadiobuttonGroup , Textbox ,
  TextboxButton , FunctionCall , MessageBox , Rectangle ,
  FillRectangle , Text , Output , Toolbar ,
  Image , Menu , Graph , Radiobuttons ,
  Bar , Data , Group , Select ,
  Table , Tab , Subtab , PolyLine ,
  Triangle , Result , RoiShape , Circle ,
  Clear , MarkCross , Position , Sharpness ,

Pixel , Line , Quad , PositionView ,
RoiDiagram , QuadDiagram , Diagram , Multiprofile ,
MultiprofileIndex , MultiprofileRow , Unknown }

- enum ValueTypes {
  BOOL , INT , LONG , UINT ,
  INT16 , FLOAT , DOUBLE , STRING ,
  HEXSTRING , BASE64 , ENUM , IP ,
  RECTANGLE , QUAD , POINT , CELLARRAY ,
  UNKNOWN }
- enum LogMessageTypes {
  DEBUG , INFO , RESULT_OK , RESULT_NOT_OK ,
  WARNING , ERROR , CRITICAL , ASSERT }
- enum Features {
  ImageView , ResultView , DiagramView , XmlView ,
  ParameterView , LogView , FirmwareUpdateDsp , FirmwareUpdateCommon ,
  FirmwareUpdate , NetworkConfiguration , SaveToPC , LoadFromPC ,
  LoadDefaultData , SaveToSensor , LoadFromSensor , ImageUpload ,
  ImageSave , Calibration , UserData , GetSpecificData ,
  PositionView , SharpnessView , ParameterCommands , Whitebalance ,
  DacValueDebug , DacValueDpmDebug , DeltaNearAdjust , ErrorLog ,
  Adjust , UseMultiprofile , Reboot , RescueSystem ,
  CodeStatistics , Unknown }
- enum Colors {
  cyan , black , blue , magenta ,
  gray , green , lime , maroon ,
  navy , olive , purple , red ,
  silver , teal , white , yellow }
- enum Sizes {
  xsmall = 4 , small = 6 , normal = 8 , medium = 9 ,
  large = 12 , xlarge = 16 , Unknown }
- enum SensorTypes { Vision , Triangular , Unknown }
- enum SensorErrors {
  Timelock , ChangeSensorState , Measurement , SensorDistanceAdjustNear ,
  SensorDistanceAdjustFar , Unknown }
- enum PgvMeasurement {
  XData , YData , YLeftData , YRightData ,
  Angle , AngleLeft , AngleRight , ControlCode1 ,
  SideOffset1 , Orientation1 , ControlCode2 , SideOffset2 ,
  Orientation2 , NoLane , LeftLane , RightLane ,
  XPosData , RelativePosition , LaneCount , Error ,
  NoPos , Warning , WarningDesc , ErrorDesc ,
  Cc , Tag , TagBit , AngleData ,
  AngleDataDegrees , EValid , XDataExtrapolated , YDataExtrapolated ,
  TagLength , TagData , Timestamp , Unknown }
- enum PcvMeasurement {
  NoOfTransmissions , FailedTransmissions , FailureValue , XPosition ,
  Speed , YPosition , DiagMessage , EventNo ,
  WarningNo , EventBit , NoPositionBit , ErrorBit ,
  WarningBit , Unknown }
- enum PxvSilMeasurement {
  HeadlineSil , NoOfTransmissionsSil , FailedTransmissionsSil , FailureValueSil ,
  XPositionSil , YPositionSil , PositionValid , Timestamp ,
  Valid , ReedSolomon , Color , ColorValid ,
  DataValid , Headline , EventBit , NoPositionBit ,
  ErrorBit , WarningBit , XPosition , Speed ,
  YPosition , ZData , Warning , Angle ,
  DiagnosisLow , DiagnosisHigh , XFinePosition , NumberOfCodes ,
  QualityValue , Unknown }

- enum SrMeasurement {
  Address , Result , Counter , EventBit ,
  NoMatch , ErrorBit , WarningBit , Checksum ,
  QualityGood , QualityVariation , QualityOutliers , XPosition ,
  ZPosition , ResultSignal , ObjectSignal , BackgroundSignal ,
  Result1 , Result2 , Result3 , Result4 ,
  Result5 , QualityGood1 , QualityGood2 , QualityGood3 ,
  QualityGood4 , QualityGood5 , EvaluationTime , Unknown }
- enum SessionTypes {
  LoginRequired , InitialPasswordRequired , Login , LoginReply ,
  SetPassword , SetPasswordReply , TimeoutAnnouncement , Timeout ,
  Logout , LogoutReply , Unknown }
- enum SessionStatus { success , failure , Unknown }
- enum Sharpness {
  MaximumSharpness , ValueCenterRegion , Trend , MeanBrightness ,
  MeanDeviation , MeanValue , LowerMeanDeviation , UpperHalfMeanValue ,
  UpperMeanValue , TopLeftBrightness , LeftBrightness , BottomLeftBrightness ,
  TopBrightness , CenterBrightness , BottomBrightness , TopRightBrightness ,
  RightBrightness , BottomRightBrightness , CodeEdgeLength , DistanceInformation }
- enum Results { TriggerCounter }

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

Modifiers

enum PF.VsxProtocolDriver.Types.Vsx.Modifiers

Enumerator

SetSingleData	
GetSingleData	
GetSpecificSingleData	
SetConfigData	
GetConfigData	
SetAllData	

Enumerator

GetAllData	
SaveData	
LoadData	
LoadDefaultData	
Echo	
SetNetwork	
GetNetwork	
SetLog	
GetLog	
Error	
SetUserData	
GetUserData	
Version	
Disconnected	
AcceptData	
TestSystem	
Discovery	
RestartNetwork	
GetAllSensorStatus	
SubscribeSensorStatus	
UnsubscribeSensorStatus	
GetMultiSensorStatus	
SensorStatus	
DiscoveryReply	
SetMultipleData	
Unknown	

FunctionAttributes

`enum PF.VsxProtocolDriver.Types.Vsx.FunctionAttributes`

Enumerator

Modifier	
Reply	
Check	
Unknown	

Elements

`enum PF.VsxProtocolDriver.Types.Vsx.Elements`

Enumerator

FUNCTION	
SENSOR	

Enumerator

SETTINGS	
CONFIGURATION	
CATEGORY	
PARAMETER	
STRUCTURE	
NETWORK	
LOG	
ELEMENT	
ITEM	
VSX	
FEATURE	
RESULT	
ERROR	
TEST	
IMAGETYPE	
STATUS	
MEASUREMENT	
DATA	
DRAW	
SESSION	
PWLOGIN	
PWSET	
TIMELEFT	
INFORMATION	
PORT	
UNKNOWN	

Attributes

enum PF.VsxProtocolDriver.Types.Vsx.Attributes

Enumerator

Id	
Name	
Type	
ValueType	
Default	
Value	
Help	
Enable	
Visible	
Location	
UserLevel	
Min	
Max	
Step	
Scale	
Unit	

Enumerator

DecimalPlaces	
MinLength	
MaxLength	
ReadOnly	
Multiple	
MessageBoxType	
IconType	
Text	
FontSize	
Color	
IPAddress	
Firmware	
MacAddress	
TypeMask	
Mask	
Version	
StartupCommand	
MessageIndex	
Gateway	
VCVersion	
FrameCounter	
Command	
InputValue	
OutputValue	
Status	
Identifier	
Image	
LinkSpeed	
Time	
FillColor	
Number	
Position	
Orientation	
IncompleteParameterset	
Port	
Baudrate	
HeadAddress	
ConstrainGE	
ConstrainLE	
ConstrainGT	
ConstrainLT	
ConstrainIN	
ConstrainOUT	
SubId	
State	
Class	
Protocol	
Message	
Seconds	

Enumerator

Username	
Password	
Busy	
TransportSecurity	
Transport	
Authentication	
VsxVersion	
Unknown	

ConstantValues

enum PF.VsxProtocolDriver.Types.Vsx.ConstantValues

Enumerator

On	
Off	
lin	
log	
Ok	
OkCancel	
YesNo	
Information	
Error	
Question	
CONF_NAME	
AlwaysVisible	
Update	
Yes	
No	
Cancel	
Safe	
Measurement	
Unknown	

ParameterTypes

enum PF.VsxProtocolDriver.Types.Vsx.ParameterTypes

Enumerator

Checkbox	
Signal	
Slider	
Label	
Spinbox	

Enumerator

Combobox	
RadiobuttonGroup	
Textbox	
TextboxButton	
FunctionCall	
MessageBox	
Rectangle	
FillRectangle	
Text	
Output	
Toolbar	
Image	
Menu	
Graph	
Radiobuttons	
Bar	
Data	
Group	
Select	
Table	
Tab	
Subtab	
PolyLine	
Triangle	
Result	
RoiShape	
Circle	
Clear	
MarkCross	
Position	
Sharpness	
Pixel	
Line	
Quad	
PositionView	
RoiDiagram	
QuadDiagram	
Diagram	
Multiprofile	
MultiprofileIndex	
MultiprofileRow	
Unknown	

ValueTypes

enum PF.VsxProtocolDriver.Types.Vsx.ValueTypes

Enumerator

BOOL	
INT	
LONG	
UINT	
INT16	
FLOAT	
DOUBLE	
STRING	
HEXSTRING	
BASE64	
ENUM	
IP	
RECTANGLE	
QUAD	
POINT	
CELLARRAY	
UNKNOWN	

LogMessageTypes

enum PF.VsxProtocolDriver.Types.Vsx.LogMessageTypes

Enumerator

DEBUG	
INFO	
RESULT_OK	
RESULT_NOT_OK	
WARNING	
ERROR	
CRITICAL	
ASSERT	

Features

enum PF.VsxProtocolDriver.Types.Vsx.Features

Enumerator

ImageView	
ResultView	
DiagramView	
XmlView	
ParameterView	
LogView	
FirmwareUpdateDsp	
FirmwareUpdateCommon	

Enumerator

FirmwareUpdate	
NetworkConfiguration	
SaveToPC	
LoadFromPC	
LoadDefaultData	
SaveToSensor	
LoadFromSensor	
ImageUpload	
ImageSave	
Calibration	
UserData	
GetSpecificData	
PositionView	
SharpnessView	
ParameterCommands	
Whitebalance	
DacValueDebug	
DacValueDpmDebug	
DeltaNearAdjust	
ErrorLog	
Adjust	
UseMultiprofile	
Reboot	
RescueSystem	
CodeStatistics	
Unknown	

Colors

`enum PF.VsxProtocolDriver.Types.Vsx.Colors`

Enumerator

cyan	
black	
blue	
magenta	
gray	
green	
lime	
maroon	
navy	
olive	
purple	
red	
silver	
teal	
white	
yellow	

Sizes

enum PF.VsxProtocolDriver.Types.Vsx.Sizes

Enumerator

xsmall	
small	
normal	
medium	
large	
xlarge	
Unknown	

SensorTypes

enum PF.VsxProtocolDriver.Types.Vsx.SensorTypes

Enumerator

Vision	
Triangular	
Unknown	

SensorErrors

enum PF.VsxProtocolDriver.Types.Vsx.SensorErrors

Enumerator

Timelock	
ChangeSensorState	
Measurement	
SensorDistanceAdjustNear	
SensorDistanceAdjustFar	
Unknown	

PgvMeasurement

enum PF.VsxProtocolDriver.Types.Vsx.PgvMeasurement

Enumerator

XData	
YData	
YLeftData	

Enumerator

YRightData	
Angle	
AngleLeft	
AngleRight	
ControlCode1	
SideOffset1	
Orientation1	
ControlCode2	
SideOffset2	
Orientation2	
NoLane	
LeftLane	
RightLane	
XPosData	
RelativePosition	
LaneCount	
Error	
NoPos	
Warning	
WarningDesc	
ErrorDesc	
Cc	
Tag	
TagBit	
AngleData	
AngleDataDegrees	
EValid	
XDataExtrapolated	
YDataExtrapolated	
TagLength	
TagData	
Timestamp	
Unknown	

PcvMeasurement

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

Enumerator

NoOfTransmissions	
FailedTransmissions	
FailureValue	
XPosition	
Speed	
YPosition	
DiagMessage	
EventNo	

Enumerator

WarningNo	
EventBit	
NoPositionBit	
ErrorBit	
WarningBit	
Unknown	

PxvSilMeasurement

enum PF.VsxProtocolDriver.Types.Vsx.PxvSilMeasurement

Enumerator

HeadlineSil	
NoOfTransmissionsSil	
FailedTransmissionsSil	
FailureValueSil	
XPositionSil	
YPositionSil	
PositionValid	
Timestamp	
Valid	
ReedSolomon	
Color	
ColorValid	
DataValid	
Headline	
EventBit	
NoPositionBit	
ErrorBit	
WarningBit	
XPosition	
Speed	
YPosition	
ZData	
Warning	
Angle	
DiagnosisLow	
DiagnosisHigh	
XFinePosition	
NumberOfCodes	
QualityValue	
Unknown	

SrMeasurement

enum PF.VsxProtocolDriver.Types.Vsx.SrMeasurement

Enumerator

Address	
Result	
Counter	
EventBit	
NoMatch	
ErrorBit	
WarningBit	
Checksum	
QualityGood	
QualityVariation	
QualityOutliers	
XPosition	
ZPosition	
ResultSignal	
ObjectSignal	
BackgroundSignal	
Result1	
Result2	
Result3	
Result4	
Result5	
QualityGood1	
QualityGood2	
QualityGood3	
QualityGood4	
QualityGood5	
EvaluationTime	
Unknown	

SessionTypes

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

Enumerator

LoginRequired	
InitialPasswordRequired	
Login	
LoginReply	
SetPassword	
SetPasswordReply	
TimeoutAnnouncement	
Timeout	
Logout	
LogoutReply	
Unknown	

SessionStatus

enum PF.VsxProtocolDriver.Types.Vsx.SessionStatus

Enumerator

success	
failure	
Unknown	

Sharpness

enum PF.VsxProtocolDriver.Types.Vsx.Sharpness

Enumerator

MaximumSharpness	
ValueCenterRegion	
Trend	
MeanBrightness	
MeanDeviation	
MeanValue	
LowerMeanDeviation	
UpperHalfMeanValue	
UpperMeanValue	
TopLeftBrightness	
LeftBrightness	
BottomLeftBrightness	
TopBrightness	
CenterBrightness	
BottomBrightness	
TopRightBrightness	
RightBrightness	
BottomRightBrightness	
CodeEdgeLength	
DistanceInformation	

Results

enum PF.VsxProtocolDriver.Types.Vsx.Results

Enumerator

TriggerCounter	

10.29.2 Member Function Documentation

GetValueFromString< T >()

```
static int PF.VsxProtocolDriver.Types.Vsx.GetValueFromString< T > (
            string enumString )  [inline], [static]
```

GetStringFromValue< T >()

```
static string PF.VsxProtocolDriver.Types.Vsx.GetStringFromValue< T > (
            int value )  [inline], [static]
```

GetEnumFromString< T >()

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

GetEnumFromStringWithEM< T >()

```
static bool T string errorDesc PF.VsxProtocolDriver.Types.Vsx.GetEnumFromStringWithEM< T > (
            string value )  [inline], [static]
```

GetEnumFromStringWithException< T >()

```
static T PF.VsxProtocolDriver.Types.Vsx.GetEnumFromStringWithException< T > (
            string value )  [inline], [static]
```

10.29.3 Member Data Documentation

VcVsxVersion

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

VcXmlVersion

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

SensorVsxVersion

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

SensorXmlVersion

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

Succ

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

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="")

  Inits 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="")

  Inits 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, 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

InitTcpDevice()

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

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

Parameters

ipAddress	The ip address of the device.
port	The port of the device.
pluginName	The type of the device.

Returns

The instance.

InitSerialDevice()

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

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

Parameters

serialPort	The comport of the device.
baudrate	The baudrate of the device.
sensorType	The sensor type of the device.
connectionType	The connection type of the device.
pluginName	The type of the device.

Returns

The instance.

UdpDeviceList()

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

SetNetworkSettingsViaUdp()

```
static async Task<(bool Succ, IError ErrorDesc)> PF.VsxProtocolDriver.VsxProtocolDriver.Set↩
NetworkSettingsViaUdp (
            string macAddress,
            string ipAddress,
            string networkMask,
            string gateway )  [inline], [static]
```

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

Parameters

macAddress	The macAddress of the device to set
ipAddress	The new IP Address
networkMask	The new network mask
gateway	The new gateway

Returns

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

SaveData()

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

Save3DPointCloudData()

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

GetFirmwareVersion()

```
static bool IFirmwareVersion IError ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriver.Get↩
FirmwareVersion (
            string filename )  [inline], [static]
```

10.30.3 Member Data Documentation

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

filename	Path and filename where to save the message.
message	The message to save.

Returns

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

Parameters

filename	Path and filename where to save the data.
x	The x image.
y	The y image.
z	The z image.

Returns

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

Parameters

filename	Path and filename of the firmware file.

Returns

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

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="")

  Inits 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="")

  Inits an instance of the VsxProtocolDriverSync to communicate with a Vsx-Device via serial protocol.
- static bool List< IVsxDeviceInformation > IError ErrorDesc UdpDeviceList ()
- 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 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

InitTcpDevice()

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

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

Parameters

ipAddress	The ip address of the device.
port	The port of the device.
pluginName	The type of the device.

Returns

The instance.

InitSerialDevice()

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

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

Parameters

serialPort	The comport of the device.
baudrate	The baudrate of the device.
sensorType	The sensor type of the device.
connectionType	The connection type of the device.
pluginName	The type of the device.

Returns

The instance.

UdpDeviceList()

```
static bool List< IVsxDeviceInformation > IError ErrorDesc PF.VsxProtocolDriver.VsxProtocol↩
DriverSync.UdpDeviceList ( )  [inline], [static]
```

SetNetworkSettingsViaUdp()

```
static bool IError ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriverSync.SetNetworkSettings↩
ViaUdp (
            string macAddress,
            string ipAddress,
            string networkMask,
            string gateway )  [static]
```

SaveData()

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

Save3DPointCloudData()

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

GetFirmwareVersion()

```
static bool IFirmwareVersion IError ErrorDesc PF.VsxProtocolDriver.VsxProtocolDriverSync.Get↩
FirmwareVersion (
             string filename )  [static]
```

10.31.3 Member Data Documentation

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

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

Parameters

macAddress	The macAddress of the device to set
ipAddress	The new IP Address
networkMask	The new network mask
gateway	The new gateway

Returns

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

Parameters

filename	Path and filename where to save the message.
message	The message to save.

Returns

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

Parameters

filename	Path and filename where to save the data.
x	The x image.
y	The y image.
z	The z image.

Returns

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

Parameters

filename	Path and filename of the firmware file.

Returns

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

DeviceList

```
bool List<IVsxDeviceInformation> PF.VsxProtocolDriver.VsxProtocolDriverSync.DeviceList  [static]
```

FirmwareVersion

```
bool IFirmwareVersion PF.VsxProtocolDriver.VsxProtocolDriverSync.FirmwareVersion  [static]
```

Index