



VsxProtocolDriver Python-Wrapper

3.4.3+g6e2c12f

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

1 Introduction	1
1.1 Supported devices	1
1.2 Requirements	1
2 Usage with Python interface	2
2.1 Requirements	2
2.2 Installation	2
2.3 Usage	2
3 Examples	2
4 Device parameter	3
5 Changelog	4
6 Namespace Index	6
6.1 Namespace List	6
7 Hierarchical Index	6
7.1 Class Hierarchy	6
8 Class Index	8
8.1 Class List	8
9 Namespace Documentation	9
9.1 VsxProtocolDriver Namespace Reference	9
9.2 VsxProtocolDriver.DataContainer Namespace Reference	10
9.3 VsxProtocolDriver.Definitions Namespace Reference	10
9.4 VsxProtocolDriver.Interface Namespace Reference	10
9.5 VsxProtocolDriver.Sensor Namespace Reference	11
10 Class Documentation	11
10.1 VsxProtocolDriver.DataContainer.DataContainer Class Reference	11
10.1.1 Constructor & Destructor Documentation	12
10.1.2 Member Function Documentation	12
10.1.3 Member Data Documentation	17
10.2 VsxProtocolDriver.Definitions.DeviceAuthenticationMode Class Reference	17
10.2.1 Member Data Documentation	18
10.3 VsxProtocolDriver.Definitions.DeviceStatusScope Class Reference	18
10.3.1 Member Data Documentation	18
10.4 VsxProtocolDriver.Definitions.DisconnectEvent Class Reference	19
10.4.1 Member Data Documentation	19
10.5 VsxProtocolDriver.DataContainer.ImageData2Format Class Reference	19
10.5.1 Member Data Documentation	20
10.6 VsxProtocolDriver.Interface.Interface Class Reference	21
10.6.1 Member Function Documentation	23

10.6.2 Member Data Documentation	23
10.7 VsxProtocolDriver.Definitions.Parameter Class Reference	33
10.7.1 Constructor & Destructor Documentation	34
10.7.2 Member Data Documentation	34
10.8 VsxProtocolDriver.Sensor.Sensor Class Reference	35
10.8.1 Constructor & Destructor Documentation	38
10.8.2 Member Function Documentation	38
10.8.3 your implementation (not working thread, so watch thread safety!)	46
10.8.4 your implementation (not working thread, so watch thread safety!)	47
10.8.5 your implementation (not working thread, so watch thread safety!)	58
10.9 VsxProtocolDriver.Definitions.SerialConnectionType Class Reference	59
10.9.1 Member Data Documentation	59
10.10 VsxProtocolDriver.Definitions.SessionTypes Class Reference	60
10.10.1 Member Data Documentation	60
10.11 VsxProtocolDriver.Definitions.StatusCode Class Reference	62
10.11.1 Member Data Documentation	64
10.12 VsxProtocolDriver.Definitions.StatusItem Class Reference	77
10.12.1 Constructor & Destructor Documentation	78
10.12.2 Member Data Documentation	78
10.13 VsxProtocolDriver.Definitions.Strategy Class Reference	79
10.13.1 Member Data Documentation	79
10.14 VsxProtocolDriver.Definitions.ValueType Class Reference	80
10.14.1 Member Data Documentation	80
10.15 VsxProtocolDriver.DataContainer.VsxCaptureInformation Class Reference	82
10.15.1 Detailed Description	82
10.15.2 Constructor & Destructor Documentation	82
10.16 VsxProtocolDriver.Interface.VsxCaptureInformationStructure Class Reference	82
10.17 VsxProtocolDriver.Interface.VsxDataContainerHandle Class Reference	82
10.17.1 Member Function Documentation	83
10.18 VsxProtocolDriver.Interface.VsxDevice Class Reference	83
10.19 VsxProtocolDriver.Interface.VsxDeviceList Class Reference	83
10.20 VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2 Class Reference	83
10.20.1 Constructor & Destructor Documentation	83
10.21 VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure Class Reference	84
10.22 VsxProtocolDriver.Interface.VsxImage Class Reference	84
10.23 VsxProtocolDriver.DataContainer.VsxLineCoordinate Class Reference	84
10.23.1 Constructor & Destructor Documentation	84
10.24 VsxProtocolDriver.Interface.VsxLineCoordinateStructure Class Reference	84
10.25 VsxProtocolDriver.Interface.VsxLineData Class Reference	85
10.26 VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation Class Reference	85
10.26.1 Constructor & Destructor Documentation	85
10.27 VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure Class Reference	85

10.28 VsxProtocolDriver.DataContainer.VsxOlr2ModbusData Class Reference	85
10.28.1 Constructor & Destructor Documentation	85
10.29 VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure Class Reference	86
10.30 VsxProtocolDriver.Interface.VsxParameter Class Reference	86
10.31 VsxProtocolDriver.Interface.VsxParameterEnumItem Class Reference	86
10.32 VsxProtocolDriver.Interface.VsxParameterList Class Reference	86
10.32.1 Constructor & Destructor Documentation	87
10.32.2 Member Data Documentation	87
10.33 VsxProtocolDriver.Interface.VsxStatusItem Class Reference	87
10.34 VsxProtocolDriver.Interface.VsxStatusItemList Class Reference	87
10.35 VsxProtocolDriver.Interface.VsxSystemHandle Class Reference	88
10.35.1 Member Function Documentation	88
10.36 VsxProtocolDriver.Interface.VsxTagList Class Reference	88
10.37 VsxProtocolDriver.DataContainer.VsxTransformation Class Reference	88
10.37.1 Detailed Description	88
10.37.2 Constructor & Destructor Documentation	89
10.38 VsxProtocolDriver.Interface.VsxTransformationStructure Class Reference	89
Index	91

1 Introduction

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

1.1 Supported devices

The official supported devices are the following:

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

1.2 Requirements

The driver is available for multiple architecture

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

The main driver is based on the C# (.NET). There are wrapper for C and Python programming language available. For usage the Microsoft .NET Runtime 6.0.x framework or higher must be installed (See <https://dotnet.microsoft.com/en-us/download/dotnet>).

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

2 Usage with Python interface

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

The main driver is implemented in C# and requires .NET 6.0 or higher. Python accesses the functionality via the C-wrapper interface.

The functions of the Python-wrapper can only be used synchronously.

2.1 Requirements

The driver is available as Python library and header for multiple architecture

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

Python should be at least 3.9.

The driver is based on the `VsxProtocolDriver`, which is based on C#. So for usage the Microsoft .NET Runtime 6.0.x framework or higher must be installed (See <https://dotnet.microsoft.com/en-us/download/dotnet>). There is also still support for .Net 5.0, but this will probably be dropped in the next version.

2.2 Installation

In order to use the SDK, the file are located inside the folder `Python\package` as an zipped package file.

Install with pip:

```
pip install VsxProtocolDriver-<x.x.x>.tar.gz
```

2.3 Usage

The main entry is the class `VsxProtocolDriver.Sensor`, where the functions to detect sensors on the network and the initialization of the sensor (`InitTcpSensor` and `InitSerialSensor`) are located.

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 CMake project in the `Python\example\` subfolder. It support the detection of different sensors and show the parametrization and the grabbing of data from the sensor.

```
import VsxProtocolDriver
ret, devices = VsxProtocolDriver.Sensor.GetUdpDeviceList()
if len(devices) > 0:
    sensor = VsxProtocolDriver.Sensor.InitTcpSensor(devices[0]["ipAddress"], "")
else:
    sensor = VsxProtocolDriver.Sensor.InitTcpSensor("192.168.2.4", "")

ret = sensor.Connect()
# Optional login
ret = sensor.Login("<user>", "<password>")
ret, exposure_time = sensor.GetSingleParameterValue(1, "Base", 1, "ExposureTime")
ret = sensor.Disconnect()
```

4 Device parameter

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

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

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

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

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

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

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

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

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

Additional notes:

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

5 Changelog

This is the changelog for the Python implementation of the [VsxProtocolDriver](#). It is based on the .NET implementation (C#) and the C wrapper of the [VsxProtocolDriver](#). Please use also the .NET and C documentation for additional information about the release.

V3.4.3

- based on [VsxProtocolDriver](#) 3.4.3

V3.4.2

- based on [VsxProtocolDriver](#) 3.4.2

V3.4.1

- Add support for new "InitializeDevice" function
- based on [VsxProtocolDriver](#) 3.4.1

V3.4.0

- Minimum requirement .net 6.0
- based on [VsxProtocolDriver](#) 3.4.0

V3.3.2

- Updated to [VsxProtocolDriver](#) 3.3.2

V3.3.1

- Updated to [VsxProtocolDriver](#) 3.3.1
- Fix "SetSingleParameterValue" type checking introduced in V3.3.0

V3.3.0

- Updated to [VsxProtocolDriver](#) 3.3.0
- Fix incorrect return values in:
 - LogMessageQueueSize, MissingLogMessagesCounter, MissingContainerFramesCounter
 - DynamicContainerQueueSize, NumberOfCachedContainers
- Add more code documentation

V3.2.1

- Updated to [VsxProtocolDriver](#) 3.2.1

- Add support for "GetSingleParameterValueInt32", "GetSingleParameterValueDouble"
- Add support for "SetSingleParameterValue" for int, float & str values

V3.1.5

- Updated to [VsxProtocolDriver](#) 3.1.5
 - Modified "Olr2CaptureInformation" data structure (incompatible with V3.1.0 and following, only for Olr2!)

V3.1.4

- Update new wrapper 3.1.4 (based on [VsxProtocolDriver](#) 3.1.3)
 - Fix memory leak inside line data allocation

V3.1.3

- Updated to [VsxProtocolDriver](#) 3.1.3
 - use given ip address instead udp response to connect

V3.1.2

- Updated to [VsxProtocolDriver](#) 3.1.2
- Support for "ApplicationResultData"

V3.1.0

- Updated to [VsxProtocolDriver](#) 3.0.6
- Add function "SendSessionKeepAlive" (reply to timeout announcement message)

V3.0.7

- Added support for "Olr2ModbusData" & "Olr2CaptureInformation"
- Fix "_OnDisconnect" function (incorrect converting of type)

V3.0.6

- Updated to [VsxProtocolDriver](#) 3.0.6

V3.0.5

- Updated to [VsxProtocolDriver](#) 3.0.5
- Adapted "SetSingleParameter" internally for C-API changes.

V3.0.4

- Updated to [VsxProtocolDriver](#) 3.0.4
- Remove "InitTcpSensorEx" & "ReConnectTcpDeviceEx" function from 3.0.2/3.0.3 again (not needed, if direct UDP from sensor supported)

V3.0.3

- Updated to [VsxProtocolDriver](#) 3.0.3
- Fixes "ReConnectTcpDevice" from V3.0.2 (incorrect function declaration)
- Add function "ReConnectTcpDeviceEx" with port number support

V3.0.2

- Updated to [VsxProtocolDriver](#) 3.0.2
- Added "InitTcpSensorEx" function with "port" parameter
- Fixes installation script (for newer setuptools)

V3.0.0

- Updated to [VsxProtocolDriver](#) 3.0.0

6 Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

VsxProtocolDriver	9
VsxProtocolDriver.DataContainer	10
VsxProtocolDriver.Definitions	10
VsxProtocolDriver.Interface	10
VsxProtocolDriver.Sensor	11

7 Hierarchical Index

7.1 Class Hierarchy

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

VsxProtocolDriver.DataContainer.DataContainer object	11
VsxProtocolDriver.Interface.Interface	21
VsxProtocolDriver.Sensor.Sensor	35
VsxProtocolDriver.Definitions.Parameter	33
VsxProtocolDriver.Definitions.StatusItem Structure	77
VsxProtocolDriver.Interface.VsxCaptureInformationStructure	82
VsxProtocolDriver.Interface.VsxDataContainerHandle	82
VsxProtocolDriver.Interface.VsxDevice	83
VsxProtocolDriver.Interface.VsxDeviceList	83
VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure	84
VsxProtocolDriver.Interface.VsxImage	84
VsxProtocolDriver.Interface.VsxLineCoordinateStructure	84
VsxProtocolDriver.Interface.VsxLineData	85
VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure	85
VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure	86
VsxProtocolDriver.Interface.VsxParameter	86
VsxProtocolDriver.Interface.VsxParameterEnumItem	86
VsxProtocolDriver.Interface.VsxParameterList	86
VsxProtocolDriver.Interface.VsxStatusItem	87
VsxProtocolDriver.Interface.VsxStatusItemList	87
VsxProtocolDriver.Interface.VsxSystemHandle	88
VsxProtocolDriver.Interface.VsxTagList	88
VsxProtocolDriver.Interface.VsxTransformationStructure	89
VsxProtocolDriver.DataContainer.VsxCaptureInformation	82
VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2	83
VsxProtocolDriver.DataContainer.VsxLineCoordinate	84
VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation	85
VsxProtocolDriver.DataContainer.VsxOlr2ModbusData	85
VsxProtocolDriver.DataContainer.VsxTransformation IntEnum	88
VsxProtocolDriver.DataContainer.ImageData2Format	19

VsxProtocolDriver.Definitions.DeviceAuthenticationMode	17
VsxProtocolDriver.Definitions.DeviceStatusScope	18
VsxProtocolDriver.Definitions.DisconnectEvent	19
VsxProtocolDriver.Definitions.SerialConnectionType	59
VsxProtocolDriver.Definitions.SessionTypes	60
VsxProtocolDriver.Definitions.StatusCode	62
VsxProtocolDriver.Definitions.Strategy	79
VsxProtocolDriver.Definitions.ValueType	80

8 Class Index

8.1 Class List

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

VsxProtocolDriver.DataContainer.DataContainer	11
VsxProtocolDriver.Definitions.DeviceAuthenticationMode	17
VsxProtocolDriver.Definitions.DeviceStatusScope	18
VsxProtocolDriver.Definitions.DisconnectEvent	19
VsxProtocolDriver.DataContainer.ImageData2Format	19
VsxProtocolDriver.Interface.Interface	21
VsxProtocolDriver.Definitions.Parameter	33
VsxProtocolDriver.Sensor.Sensor	35
VsxProtocolDriver.Definitions.SerialConnectionType	59
VsxProtocolDriver.Definitions.SessionTypes	60
VsxProtocolDriver.Definitions.StatusCode	62
VsxProtocolDriver.Definitions.StatusItem	77
VsxProtocolDriver.Definitions.Strategy	79
VsxProtocolDriver.Definitions.ValueType	80
VsxProtocolDriver.DataContainer.VsxCaptureInformation	
CaptureInformation contains information identifying and describing the captured image	82
VsxProtocolDriver.Interface.VsxCaptureInformationStructure	82
VsxProtocolDriver.Interface.VsxDataContainerHandle	82
VsxProtocolDriver.Interface.VsxDevice	83

VsxProtocolDriver.Interface.VsxDeviceList	83
VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2	83
VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure	84
VsxProtocolDriver.Interface.VsxImage	84
VsxProtocolDriver.DataContainer.VsxLineCoordinate	84
VsxProtocolDriver.Interface.VsxLineCoordinateStructure	84
VsxProtocolDriver.Interface.VsxLineData	85
VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation	85
VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure	85
VsxProtocolDriver.DataContainer.VsxOlr2ModbusData	85
VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure	86
VsxProtocolDriver.Interface.VsxParameter	86
VsxProtocolDriver.Interface.VsxParameterEnumItem	86
VsxProtocolDriver.Interface.VsxParameterList	86
VsxProtocolDriver.Interface.VsxStatusItem	87
VsxProtocolDriver.Interface.VsxStatusItemList	87
VsxProtocolDriver.Interface.VsxSystemHandle	88
VsxProtocolDriver.Interface.VsxTagList	88
VsxProtocolDriver.DataContainer.VsxTransformation	88
Used to transform raw point cloud data from device	
VsxProtocolDriver.Interface.VsxTransformationStructure	89

9 Namespace Documentation

9.1 VsxProtocolDriver Namespace Reference

Namespaces

- namespace [DataContainer](#)
- namespace [Definitions](#)
- namespace [Interface](#)
- namespace [Sensor](#)

9.2 VsxProtocolDriver.DataContainer Namespace Reference

Classes

- class [DataContainer](#)
- class [ImageData2Format](#)
- class [VsxCaptureInformation](#)
CaptureInformation contains information identifying and describing the captured image.
- class [VsxDisparityDescriptor2](#)
- class [VsxLineCoordinate](#)
- class [VsxOlr2CaptureInformation](#)
- class [VsxOlr2ModbusData](#)
- class [VsxTransformation](#)
Used to transform raw point cloud data from device.

9.3 VsxProtocolDriver.Definitions Namespace Reference

Classes

- class [DeviceAuthenticationMode](#)
- class [DeviceStatusScope](#)
- class [DisconnectEvent](#)
- class [Parameter](#)
- class [SerialConnectionType](#)
- class [SessionTypes](#)
- class [StatusCode](#)
- class [StatusItem](#)
- class [Strategy](#)
- class [ValueType](#)

9.4 VsxProtocolDriver.Interface Namespace Reference

Classes

- class [Interface](#)
- class [VsxCaptureInformationStructure](#)
- class [VsxDataContainerHandle](#)
- class [VsxDevice](#)
- class [VsxDeviceList](#)
- class [VsxDisparityDescriptor2Structure](#)
- class [VsxImage](#)
- class [VsxLineCoordinateStructure](#)
- class [VsxLineData](#)
- class [VsxOlr2CaptureInformationStructure](#)
- class [VsxOlr2ModbusDataStructure](#)
- class [VsxParameter](#)
- class [VsxParameterEnumItem](#)
- class [VsxParameterList](#)
- class [VsxStatusItem](#)
- class [VsxStatusItemList](#)
- class [VsxSystemHandle](#)
- class [VsxTagList](#)
- class [VsxTransformationStructure](#)

9.5 VsxProtocolDriver.Sensor Namespace Reference

Classes

- class [Sensor](#)

10 Class Documentation

10.1 VsxProtocolDriver.DataContainer.DataContainer Class Reference

Public Member Functions

- def [__init__](#) (self, *_is_direct=True)
- def [__del__](#) (self)
- int [SaveData](#) (self, str tag, str file_name)
Saves a VsxMessage to the given filename.
- int [Save3DPointCloudData](#) (self, str tag_x, str tag_y, str tag_z, str file_name)
Saves a 3D point cloud as pcd to the given filename.
- def [GetTagList](#) (self)
Returns all available tags from a dynamic container.
- Tuple[int, Optional[VsxCaptureInformation]] [GetCaptureInformation](#) (self, str tag)
Get capture information from a dynamic container.
- Tuple[int, Optional[VsxTransformation]] [GetTransformation](#) (self, str tag)
Get transformation from a dynamic container.
- Tuple[int, Optional[VsxDisparityDescriptor2]] [GetDisparityDescriptor2](#) (self, str tag)
Get disparity descriptor from a dynamic container.
- Tuple[int, Optional[VsxOlr2CaptureInformation]] [GetOlr2CaptureInformation](#) (self, str tag)
Get olr2 capture information from a dynamic container.
- Tuple[int, Optional[VsxOlr2ModbusData]] [GetOlr2ModbusData](#) (self, str tag)
Get modbus data for olr2 sensor from a dynamic container.
- Tuple[int, Optional[np.ndarray], Optional[dict]] [GetImage](#) (self, str image_tag)
Get image from a dynamic container, access via numpy array.
- Tuple[int, Optional[List[List[VsxLineCoordinate]]], Optional[dict]] [GetLine](#) (self, str image_tag)
Get line data from a dynamic container.
- Tuple[int, Optional[str]] [GetResultXml](#) (self, str tag_result)
Returns the complete xml response from an result inside data container.
- Tuple[int, Optional[str]] [GetResultElementString](#) (self, str tag_result, str xpath)
Return certain value from a result inside data container.
- Tuple[int, Optional[int]] [GetResultElementInt32](#) (self, str tag_result, str xpath)
Return certain value from a result inside data container.
- Tuple[int, Optional[int]] [GetResultElementInt64](#) (self, str tag_result, str xpath)
Return certain value from a result inside data container.
- Tuple[int, Optional[float]] [GetResultElementDouble](#) (self, str tag_result, str xpath)
Return certain value from a result inside data container.

Public Attributes

- [data_container_handle](#)

10.1.1 Constructor & Destructor Documentation

10.1.1.1 `__init__()` `def VsxProtocolDriver.DataContainer.DataContainer.__init__ (`
`self,`
`* _is_direct = True)`

10.1.1.2 `__del__()` `def VsxProtocolDriver.DataContainer.DataContainer.__del__ (`
`self)`

10.1.2 Member Function Documentation

10.1.2.1 `SaveData()` `int VsxProtocolDriver.DataContainer.DataContainer.SaveData (`
`self,`
`str tag,`
`str file_name)`

Saves a VsxMessage to the given filename.

Parameters

<i>tag</i>	Specify which tag from container should be saved ("*" save complete container)
<i>file_name</i>	Path and filename where to save the message

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.1.2.2 `Save3DPointCloudData()` `int VsxProtocolDriver.DataContainer.DataContainer.Save3D↵`
`DPointCloudData (`
`self,`
`str tag_x,`
`str tag_y,`
`str tag_z,`
`str file_name)`

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

Parameters

<i>tag_x</i>	The x image tag name
<i>tag_y</i>	The y image tag name
<i>tag_z</i>	The z image tag name
<i>file_name</i>	Path and filename where to save the data

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.1.2.3 GetTagList() `def VsxProtocolDriver.DataContainer.DataContainer.GetTagList (
 self)`

Returns all available tags from a dynamic container.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and tag list object

10.1.2.4 GetCaptureInformation() `Tuple[int, Optional[VsxCaptureInformation]] VsxProtocolDriver.DataContainer.DataContainer.GetCaptureInformation (
 self,
 str tag)`

Get capture information from a dynamic container.

Parameters

<i>tag</i>	Tag name of data
------------	------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, [VsxCaptureInformation](#) object

10.1.2.5 GetTransformation() `Tuple[int, Optional[VsxTransformation]] VsxProtocolDriver.DataContainer.DataContainer.GetTransformation (
 self,
 str tag)`

Get transformation from a dynamic container.

Parameters

<i>tag</i>	Tag name of data
------------	------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, transformation object

10.1.2.6 GetDisparityDescriptor2() Tuple[int, Optional[VsxDisparityDescriptor2]] VsxProtocol↔
Driver.DataContainer.DataContainer.GetDisparityDescriptor2 (
 self,
 str *tag*)

Get disparity descriptor from a dynamic container.

Parameters

<i>tag</i>	Tag name of data
------------	------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, [VsxDisparityDescriptor2](#) object

10.1.2.7 GetOlr2CaptureInformation() Tuple[int, Optional[VsxOlr2CaptureInformation]] VsxProtocol↔
Driver.DataContainer.DataContainer.GetOlr2CaptureInformation (
 self,
 str *tag*)

Get olr2 capture information from a dynamic container.

Parameters

<i>tag</i>	Tag name of data
------------	------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, [VsxOlr2CaptureInformation](#) object

10.1.2.8 GetOlr2ModbusData() Tuple[int, Optional[VsxOlr2ModbusData]] VsxProtocolDriver.Data↔
Container.DataContainer.GetOlr2ModbusData (
 self,
 str *tag*)

Get modbus data for olr2 sensor from a dynamic container.

Parameters

<i>tag</i>	Tag name of data
------------	------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, [VsxOlr2ModbusData](#) object

10.1.2.9 GetImage() Tuple[int, Optional[np.ndarray], Optional[dict]] VsxProtocolDriver.DataContainer.DataContainer.GetImage (

```

        self,
        str image_tag )

```

Get image from a dynamic container, access via numpy array.

Parameters

<i>image_tag</i>	Tag name of image data
------------------	------------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, numpy array, dictionary of attributes

10.1.2.10 GetLine() Tuple[int, Optional[List[List[VsxLineCoordinate]]], Optional[dict]] VsxProtocolDriver.DataContainer.DataContainer.GetLine (

```

        self,
        str image_tag )

```

Get line data from a dynamic container.

Parameters

<i>image_tag</i>	Tag name of line data
------------------	-----------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, list of coordinates, dictionary of attributes

10.1.2.11 GetResultXml() Tuple[int, Optional[str]] VsxProtocolDriver.DataContainer.DataContainer.GetResultXml (

```

        self,
        str tag_result )

```

Returns the complete xml response from an result inside data container.

Parameters

<i>tag_result</i>	Name of result
-------------------	----------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, complete result xml as string

10.1.2.12 GetResultElementString() Tuple[int, Optional[str]] VsxProtocolDriver.DataContainer.↔
 DataContainer.GetResultElementString (
 self,
 str *tag_result*,
 str *xpath*)

Return certain value from a result inside data container.

Parameters

<i>tag_result</i>	Name of result
<i>xpath</i>	xPath definition

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, result as string

10.1.2.13 GetResultElementInt32() Tuple[int, Optional[int]] VsxProtocolDriver.DataContainer.↔
 DataContainer.GetResultElementInt32 (
 self,
 str *tag_result*,
 str *xpath*)

Return certain value from a result inside data container.

Parameters

<i>tag_result</i>	Name of result
<i>xpath</i>	xPath definition

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, result as integer number

10.1.2.14 GetResultElementInt64() Tuple[int, Optional[int]] VsxProtocolDriver.DataContainer.↔
 DataContainer.GetResultElementInt64 (
 self,
 str *tag_result*,
 str *xpath*)

Return certain value from a result inside data container.

Parameters

<i>tag_result</i>	Name of result
<i>xpath</i>	xPath definition

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, result as integer number

10.1.2.15 GetResultElementDouble() Tuple[int, Optional[float]] VsxProtocolDriver.DataContainer.↔
 DataContainer.GetResultElementDouble (
 self,
 str *tag_result*,
 str *xpath*)

Return certain value from a result inside data container.

Parameters

<i>tag_result</i>	Name of result
<i>xpath</i>	xPath definition

Returns

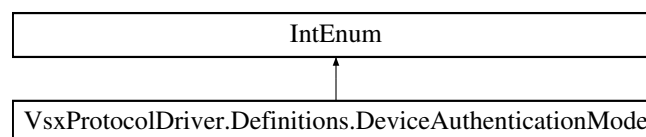
: Returns VSX_STATUS_SUCCESS(0) on success, result as float number

10.1.3 Member Data Documentation

10.1.3.1 data_container_handle VsxProtocolDriver.DataContainer.DataContainer.data_container_↔
 handle

10.2 VsxProtocolDriver.Definitions.DeviceAuthenticationMode Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.DeviceAuthenticationMode:

**Static Public Attributes**

- int [NONE](#) = 0
- int [CONFIGURATION_ONLY](#) = 1
- int [FULL](#) = 2

10.2.1 Member Data Documentation

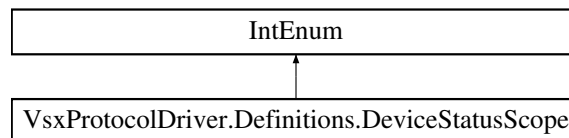
10.2.1.1 NONE `int VsxProtocolDriver.Definitions.DeviceAuthenticationMode.NONE = 0 [static]`

10.2.1.2 CONFIGURATION_ONLY `int VsxProtocolDriver.Definitions.DeviceAuthenticationMode.
CONFIGURATION_ONLY = 1 [static]`

10.2.1.3 FULL `int VsxProtocolDriver.Definitions.DeviceAuthenticationMode.FULL = 2 [static]`

10.3 VsxProtocolDriver.Definitions.DeviceStatusScope Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.DeviceStatusScope:



Static Public Attributes

- `int FULL = 0`
- `int MULTI = 1`

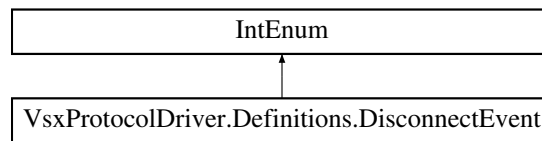
10.3.1 Member Data Documentation

10.3.1.1 FULL `int VsxProtocolDriver.Definitions.DeviceStatusScope.FULL = 0 [static]`

10.3.1.2 MULTI `int VsxProtocolDriver.Definitions.DeviceStatusScope.MULTI = 1 [static]`

10.4 VsxProtocolDriver.Definitions.DisconnectEvent Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.DisconnectEvent:



Static Public Attributes

- int [REMOTE_HOST_CONNECTION_CLOSED](#) = 0
- int [DISCONNECT_CALLED](#) = 1
- int [CONNECTION_ERROR](#) = 2

10.4.1 Member Data Documentation

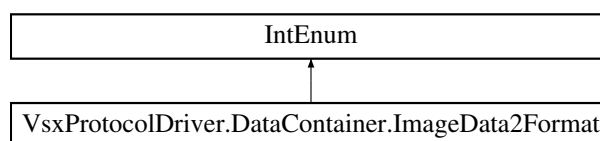
10.4.1.1 REMOTE_HOST_CONNECTION_CLOSED int VsxProtocolDriver.Definitions.DisconnectEvent.REMOTE_HOST_CONNECTION_CLOSED = 0 [static]

10.4.1.2 DISCONNECT_CALLED int VsxProtocolDriver.Definitions.DisconnectEvent.DISCONNECT_CALLED = 1 [static]

10.4.1.3 CONNECTION_ERROR int VsxProtocolDriver.Definitions.DisconnectEvent.CONNECTION_ERROR = 2 [static]

10.5 VsxProtocolDriver.DataContainer.ImageData2Format Class Reference

Inheritance diagram for VsxProtocolDriver.DataContainer.ImageData2Format:



Static Public Attributes

- `int VSX_IMAGE_DATA2_FORMAT_MONO8 = 17301505`
- `int VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8 = 17301702`
- `int VSX_IMAGE_DATA2_FORMAT_MONO12 = 17825797`
- `int VSX_IMAGE_DATA2_FORMAT_MONO16 = 17825799`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_A16 = 17825974`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_B16 = 17825975`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_C16 = 17825976`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f = 18874557`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f = 18874558`
- `int VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f = 18874559`

10.5.1 Member Data Documentation

10.5.1.1 VSX_IMAGE_DATA2_FORMAT_MONO8 `int VsxProtocolDriver.DataContainer.ImageData2↔Format.VSX_IMAGE_DATA2_FORMAT_MONO8 = 17301505 [static]`

10.5.1.2 VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8 `int VsxProtocolDriver.DataContainer.Image↔Data2Format.VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8 = 17301702 [static]`

10.5.1.3 VSX_IMAGE_DATA2_FORMAT_MONO12 `int VsxProtocolDriver.DataContainer.ImageData2↔Format.VSX_IMAGE_DATA2_FORMAT_MONO12 = 17825797 [static]`

10.5.1.4 VSX_IMAGE_DATA2_FORMAT_MONO16 `int VsxProtocolDriver.DataContainer.ImageData2↔Format.VSX_IMAGE_DATA2_FORMAT_MONO16 = 17825799 [static]`

10.5.1.5 VSX_IMAGE_DATA2_FORMAT_COORD3D_A16 `int VsxProtocolDriver.DataContainer.Image↔Data2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_A16 = 17825974 [static]`

10.5.1.6 VSX_IMAGE_DATA2_FORMAT_COORD3D_B16 `int VsxProtocolDriver.DataContainer.Image↔Data2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_B16 = 17825975 [static]`

10.5.1.7 VSX_IMAGE_DATA2_FORMAT_COORD3D_C16 `int VsxProtocolDriver.DataContainer.Image↔Data2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_C16 = 17825976 [static]`

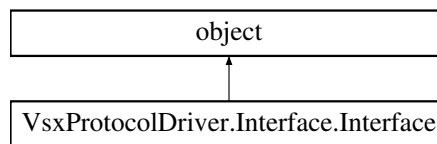
10.5.1.8 VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f `int VsxProtocolDriver.DataContainer.↔ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f = 18874557 [static]`

10.5.1.9 VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f `int VsxProtocolDriver.DataContainer.↔ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f = 18874558 [static]`

10.5.1.10 VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f `int VsxProtocolDriver.DataContainer.↔ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f = 18874559 [static]`

10.6 VsxProtocolDriver.Interface.Interface Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.Interface:



Public Member Functions

- def `init_driver` (cls)

Static Public Attributes

- `callback_on_disconnect` = WINFUNCTYPE(None, c_int32, c_char_p, c_int32, c_char_p)
- `callback_on_session_message_received` = WINFUNCTYPE(None, c_int32, c_int32, c_int32)
- `callback_on_device_status_received` = WINFUNCTYPE(None, c_int32, c_int32, POINTER(VsxStatusItemList))
- `release_string` = None
- `get_library_version` = None
- `get_error_text` = None
- `init_tcp_sensor` = None
- `init_serial_sensor` = None
- `release_sensor` = None
- `reconnect_tcp_device` = None
- `reconnect_and_login_tcp_device` = None
- `reconnect_serial_device` = None
- `connect` = None
- `connect_ex` = None
- `connect_and_login` = None

- [connect_ex_and_login](#) = None
- [login](#) = None
- [logout](#) = None
- [set_password](#) = None
- [initialize_device](#) = None
- [get_connected](#) = None
- [disconnect](#) = None
- [register_on_disconnect](#) = None
- [deregister_on_disconnect](#) = None
- [register_on_session_message_received](#) = None
- [deregister_on_session_message_received](#) = None
- [send_session_keep_alive](#) = None
- [test_system](#) = None
- [test_system_ex](#) = None
- [get_wait_timeout](#) = None
- [set_wait_timeout](#) = None
- [upload_data](#) = None
- [send_firmware](#) = None
- [send_xml_data_message](#) = None
- [set_network_settings](#) = None
- [set_network_settings_via_udp](#) = None
- [reset_dynamic_container_grabber](#) = None
- [get_data_container](#) = None
- [get_cached_container](#) = None
- [release_data_container](#) = None
- [save_data](#) = None
- [save_3d_point_cloud_data](#) = None
- [get_capture_information](#) = None
- [release_capture_information](#) = None
- [get_transformation](#) = None
- [release_transformation](#) = None
- [get_disparity_descriptor2](#) = None
- [release_disparity_descriptor2](#) = None
- [get_olr2_capture_information](#) = None
- [release_olr2_capture_information](#) = None
- [get_olr2_modbus_data](#) = None
- [release_olr2_modbus_data](#) = None
- [get_image](#) = None
- [release_image](#) = None
- [get_line](#) = None
- [release_line](#) = None
- [get_tag_list](#) = None
- [release_tag_list](#) = None
- [get_missing_container_frames_counter](#) = None
- [get_dynamic_container_queue_size](#) = None
- [get_number_of_cached_containers](#) = None
- [get_device_information](#) = None
- [release_device](#) = None
- [get_udp_device_list](#) = None
- [release_device_list](#) = None
- [reset_log_message_grabber](#) = None
- [get_log_message](#) = None
- [get_log_message_queue_size](#) = None
- [get_missing_log_messages_counter](#) = None
- [set_single_parameter_value](#) = None

- [set_single_parameter_value_int32](#) = None
- [set_single_parameter_value_double](#) = None
- [get_single_parameter_value](#) = None
- [get_single_parameter_value_int32](#) = None
- [get_single_parameter_value_double](#) = None
- [load_default_parameter_set_on_device](#) = None
- [load_parameter_set_on_device](#) = None
- [save_parameter_set_on_device](#) = None
- [upload_parameter_set](#) = None
- [download_parameter_set](#) = None
- [get_parameter_list](#) = None
- [upload_parameter_list](#) = None
- [set_single_parameter_double](#) = None
- [set_single_parameter_int32](#) = None
- [set_single_parameter_string](#) = None
- [get_single_parameter](#) = None
- [release_parameter](#) = None
- [release_parameter_list](#) = None
- [get_result_xml](#) = None
- [get_result_element_string](#) = None
- [get_result_element_int32](#) = None
- [get_result_element_int64](#) = None
- [get_result_element_double](#) = None
- [get_all_device_status_data](#) = None
- [release_status_item_list](#) = None
- [register_on_device_status_received](#) = None
- [deregister_on_device_status_received](#) = None
- [subscribe_to_device_status_data](#) = None
- [unsubscribe_to_device_status_data](#) = None

10.6.1 Member Function Documentation

10.6.1.1 `init_driver()` `def VsxProtocolDriver.Interface.Interface.init_driver (`
 cls)

10.6.2 Member Data Documentation

10.6.2.1 `callback_on_disconnect` `VsxProtocolDriver.Interface.Interface.callback_on_disconnect =`
`WINFUNCTYPE(None, c_int32, c_char_p, c_int32, c_char_p) [static]`

10.6.2.2 `callback_on_session_message_received` `VsxProtocolDriver.Interface.Interface.callback↵`
`_on_session_message_received = WINFUNCTYPE(None, c_int32, c_int32, c_int32) [static]`

10.6.2.3 callback_on_device_status_received `VsxProtocolDriver.Interface.Interface.callback_on_device_status_received = WINFUNCTYPE(None, c_int32, c_int32, POINTER(VsxStatusItemList))`
[static]

10.6.2.4 release_string `VsxProtocolDriver.Interface.Interface.release_string = None` [static]

10.6.2.5 get_library_version `VsxProtocolDriver.Interface.Interface.get_library_version = None`
[static]

10.6.2.6 get_error_text `VsxProtocolDriver.Interface.Interface.get_error_text = None` [static]

10.6.2.7 init_tcp_sensor `VsxProtocolDriver.Interface.Interface.init_tcp_sensor = None` [static]

10.6.2.8 init_serial_sensor `VsxProtocolDriver.Interface.Interface.init_serial_sensor = None`
[static]

10.6.2.9 release_sensor `VsxProtocolDriver.Interface.Interface.release_sensor = None` [static]

10.6.2.10 reconnect_tcp_device `VsxProtocolDriver.Interface.Interface.reconnect_tcp_device = None` [static]

10.6.2.11 reconnect_and_login_tcp_device `VsxProtocolDriver.Interface.Interface.reconnect_and_login_tcp_device = None` [static]

10.6.2.12 reconnect_serial_device `VsxProtocolDriver.Interface.Interface.reconnect_serial_device = None` [static]

10.6.2.13 connect VsxProtocolDriver.Interface.Interface.connect = None [static]

10.6.2.14 connect_ex VsxProtocolDriver.Interface.Interface.connect_ex = None [static]

10.6.2.15 connect_and_login VsxProtocolDriver.Interface.Interface.connect_and_login = None [static]

10.6.2.16 connect_ex_and_login VsxProtocolDriver.Interface.Interface.connect_ex_and_login = None [static]

10.6.2.17 login VsxProtocolDriver.Interface.Interface.login = None [static]

10.6.2.18 logout VsxProtocolDriver.Interface.Interface.logout = None [static]

10.6.2.19 set_password VsxProtocolDriver.Interface.Interface.set_password = None [static]

10.6.2.20 initialize_device VsxProtocolDriver.Interface.Interface.initialize_device = None [static]

10.6.2.21 get_connected VsxProtocolDriver.Interface.Interface.get_connected = None [static]

10.6.2.22 disconnect VsxProtocolDriver.Interface.Interface.disconnect = None [static]

10.6.2.23 register_on_disconnect VsxProtocolDriver.Interface.Interface.register_on_disconnect = None [static]

10.6.2.24 deregister_on_disconnect `VsxProtocolDriver.Interface.Interface.deregister_on_disconnect = None [static]`

10.6.2.25 register_on_session_message_received `VsxProtocolDriver.Interface.Interface.register↔_on_session_message_received = None [static]`

10.6.2.26 deregister_on_session_message_received `VsxProtocolDriver.Interface.Interface.↔deregister_on_session_message_received = None [static]`

10.6.2.27 send_session_keep_alive `VsxProtocolDriver.Interface.Interface.send_session_keep_↔alive = None [static]`

10.6.2.28 test_system `VsxProtocolDriver.Interface.Interface.test_system = None [static]`

10.6.2.29 test_system_ex `VsxProtocolDriver.Interface.Interface.test_system_ex = None [static]`

10.6.2.30 get_wait_timeout `VsxProtocolDriver.Interface.Interface.get_wait_timeout = None [static]`

10.6.2.31 set_wait_timeout `VsxProtocolDriver.Interface.Interface.set_wait_timeout = None [static]`

10.6.2.32 upload_data `VsxProtocolDriver.Interface.Interface.upload_data = None [static]`

10.6.2.33 send_firmware `VsxProtocolDriver.Interface.Interface.send_firmware = None [static]`

10.6.2.34 send_xml_data_message VsxProtocolDriver.Interface.Interface.send_xml_data_message = None [static]

10.6.2.35 set_network_settings VsxProtocolDriver.Interface.Interface.set_network_settings = None [static]

10.6.2.36 set_network_settings_via_udp VsxProtocolDriver.Interface.Interface.set_network↔ settings_via_udp = None [static]

10.6.2.37 reset_dynamic_container_grabber VsxProtocolDriver.Interface.Interface.reset_dynamic↔ _container_grabber = None [static]

10.6.2.38 get_data_container VsxProtocolDriver.Interface.Interface.get_data_container = None [static]

10.6.2.39 get_cached_container VsxProtocolDriver.Interface.Interface.get_cached_container = None [static]

10.6.2.40 release_data_container VsxProtocolDriver.Interface.Interface.release_data_container = None [static]

10.6.2.41 save_data VsxProtocolDriver.Interface.Interface.save_data = None [static]

10.6.2.42 save_3d_point_cloud_data VsxProtocolDriver.Interface.Interface.save_3d_point_cloud↔ _data = None [static]

10.6.2.43 get_capture_information VsxProtocolDriver.Interface.Interface.get_capture_information = None [static]

10.6.2.44 release_capture_information VsxProtocolDriver.Interface.Interface.release_capture_↔
information = None [static]

10.6.2.45 get_transformation VsxProtocolDriver.Interface.Interface.get_transformation = None
[static]

10.6.2.46 release_transformation VsxProtocolDriver.Interface.Interface.release_transformation =
None [static]

10.6.2.47 get_disparity_descriptor2 VsxProtocolDriver.Interface.Interface.get_disparity_descriptor2
= None [static]

10.6.2.48 release_disparity_descriptor2 VsxProtocolDriver.Interface.Interface.release_disparity_↔
_descriptor2 = None [static]

10.6.2.49 get_olr2_capture_information VsxProtocolDriver.Interface.Interface.get_olr2_capture_↔
information = None [static]

10.6.2.50 release_olr2_capture_information VsxProtocolDriver.Interface.Interface.release_olr2_↔
capture_information = None [static]

10.6.2.51 get_olr2_modbus_data VsxProtocolDriver.Interface.Interface.get_olr2_modbus_data =
None [static]

10.6.2.52 release_olr2_modbus_data VsxProtocolDriver.Interface.Interface.release_olr2_modbus_↔
_data = None [static]

10.6.2.53 get_image VsxProtocolDriver.Interface.Interface.get_image = None [static]

10.6.2.54 release_image VsxProtocolDriver.Interface.Interface.release_image = None [static]

10.6.2.55 get_line VsxProtocolDriver.Interface.Interface.get_line = None [static]

10.6.2.56 release_line VsxProtocolDriver.Interface.Interface.release_line = None [static]

10.6.2.57 get_tag_list VsxProtocolDriver.Interface.Interface.get_tag_list = None [static]

10.6.2.58 release_tag_list VsxProtocolDriver.Interface.Interface.release_tag_list = None [static]

10.6.2.59 get_missing_container_frames_counter VsxProtocolDriver.Interface.Interface.get_↵
missing_container_frames_counter = None [static]

10.6.2.60 get_dynamic_container_queue_size VsxProtocolDriver.Interface.Interface.get_dynamic_↵
_container_queue_size = None [static]

10.6.2.61 get_number_of_cached_containers VsxProtocolDriver.Interface.Interface.get_number_↵
of_cached_containers = None [static]

10.6.2.62 get_device_information VsxProtocolDriver.Interface.Interface.get_device_information =
None [static]

10.6.2.63 release_device VsxProtocolDriver.Interface.Interface.release_device = None [static]

10.6.2.64 get_udp_device_list VsxProtocolDriver.Interface.Interface.get_udp_device_list = None
[static]

10.6.2.65 release_device_list VsxProtocolDriver.Interface.Interface.release_device_list = None
[static]

10.6.2.66 reset_log_message_grabber VsxProtocolDriver.Interface.Interface.reset_log_message_↔
grabber = None [static]

10.6.2.67 get_log_message VsxProtocolDriver.Interface.Interface.get_log_message = None [static]

10.6.2.68 get_log_message_queue_size VsxProtocolDriver.Interface.Interface.get_log_message_↔
queue_size = None [static]

10.6.2.69 get_missing_log_messages_counter VsxProtocolDriver.Interface.Interface.get_missing_↔
_log_messages_counter = None [static]

10.6.2.70 set_single_parameter_value VsxProtocolDriver.Interface.Interface.set_single_parameter_↔
_value = None [static]

10.6.2.71 set_single_parameter_value_int32 VsxProtocolDriver.Interface.Interface.set_single_↔
parameter_value_int32 = None [static]

10.6.2.72 set_single_parameter_value_double VsxProtocolDriver.Interface.Interface.set_single_↔
parameter_value_double = None [static]

10.6.2.73 get_single_parameter_value VsxProtocolDriver.Interface.Interface.get_single_parameter_↔
_value = None [static]

10.6.2.74 get_single_parameter_value_int32 VsxProtocolDriver.Interface.Interface.get_single↔
parameter_value_int32 = None [static]

10.6.2.75 get_single_parameter_value_double VsxProtocolDriver.Interface.Interface.get_single↔
parameter_value_double = None [static]

10.6.2.76 load_default_parameter_set_on_device VsxProtocolDriver.Interface.Interface.load↔
default_parameter_set_on_device = None [static]

10.6.2.77 load_parameter_set_on_device VsxProtocolDriver.Interface.Interface.load_parameter↔
set_on_device = None [static]

10.6.2.78 save_parameter_set_on_device VsxProtocolDriver.Interface.Interface.save_parameter↔
set_on_device = None [static]

10.6.2.79 upload_parameter_set VsxProtocolDriver.Interface.Interface.upload_parameter_set =
None [static]

10.6.2.80 download_parameter_set VsxProtocolDriver.Interface.Interface.download_parameter_set
= None [static]

10.6.2.81 get_parameter_list VsxProtocolDriver.Interface.Interface.get_parameter_list = None
[static]

10.6.2.82 upload_parameter_list VsxProtocolDriver.Interface.Interface.upload_parameter_list =
None [static]

10.6.2.83 set_single_parameter_double VsxProtocolDriver.Interface.Interface.set_single_parameter↔
_double = None [static]

10.6.2.84 set_single_parameter_int32 VsxProtocolDriver.Interface.Interface.set_single_parameter↔
_int32 = None [static]

10.6.2.85 set_single_parameter_string VsxProtocolDriver.Interface.Interface.set_single_parameter↔
_string = None [static]

10.6.2.86 get_single_parameter VsxProtocolDriver.Interface.Interface.get_single_parameter =
None [static]

10.6.2.87 release_parameter VsxProtocolDriver.Interface.Interface.release_parameter = None
[static]

10.6.2.88 release_parameter_list VsxProtocolDriver.Interface.Interface.release_parameter_list =
None [static]

10.6.2.89 get_result_xml VsxProtocolDriver.Interface.Interface.get_result_xml = None [static]

10.6.2.90 get_result_element_string VsxProtocolDriver.Interface.Interface.get_result_element↔
string = None [static]

10.6.2.91 get_result_element_int32 VsxProtocolDriver.Interface.Interface.get_result_element↔
int32 = None [static]

10.6.2.92 get_result_element_int64 VsxProtocolDriver.Interface.Interface.get_result_element↔
int64 = None [static]

10.6.2.93 get_result_element_double VsxProtocolDriver.Interface.Interface.get_result_element↔
double = None [static]

10.6.2.94 get_all_device_status_data VsxProtocolDriver.Interface.Interface.get_all_device_↔
status_data = None [static]

10.6.2.95 release_status_item_list VsxProtocolDriver.Interface.Interface.release_status_item_↔
list = None [static]

10.6.2.96 register_on_device_status_received VsxProtocolDriver.Interface.Interface.register_on_↔
_device_status_received = None [static]

10.6.2.97 deregister_on_device_status_received VsxProtocolDriver.Interface.Interface.deregister_↔
_on_device_status_received = None [static]

10.6.2.98 subscribe_to_device_status_data VsxProtocolDriver.Interface.Interface.subscribe_to_↔
device_status_data = None [static]

10.6.2.99 unsubscribe_to_device_status_data VsxProtocolDriver.Interface.Interface.unsubscribe_↔
_to_device_status_data = None [static]

10.7 VsxProtocolDriver.Definitions.Parameter Class Reference

Public Member Functions

- def `__init__` (self, int settings_version, str config_id, int config_version, str parameter_id, str [name](#), object [value](#), [ValueType](#) value_type, dict enum_items)

Public Attributes

- [settingsVersion](#)
- [configId](#)
- [configVersion](#)
- [parameterId](#)
- [name](#)
- [value](#)
- [valueType](#)
- [enumItems](#)

10.7.1 Constructor & Destructor Documentation

10.7.1.1 `__init__()` `def VsxProtocolDriver.Definitions.Parameter.__init__ (`
 `self,`
 `int settings_version,`
 `str config_id,`
 `int config_version,`
 `str parameter_id,`
 `str name,`
 `object value,`
 `ValueType value_type,`
 `dict enum_items)`

10.7.2 Member Data Documentation

10.7.2.1 `settingsVersion` `VsxProtocolDriver.Definitions.Parameter.settingsVersion`

10.7.2.2 `configId` `VsxProtocolDriver.Definitions.Parameter.configId`

10.7.2.3 `configVersion` `VsxProtocolDriver.Definitions.Parameter.configVersion`

10.7.2.4 `parameterId` `VsxProtocolDriver.Definitions.Parameter.parameterId`

10.7.2.5 `name` `VsxProtocolDriver.Definitions.Parameter.name`

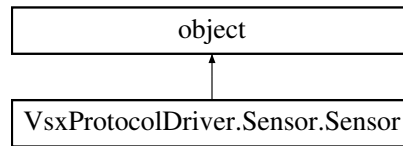
10.7.2.6 `value` `VsxProtocolDriver.Definitions.Parameter.value`

10.7.2.7 `valueType` `VsxProtocolDriver.Definitions.Parameter.valueType`

10.7.2.8 enumItems VsxProtocolDriver.Definitions.Parameter.enumItems

10.8 VsxProtocolDriver.Sensor.Sensor Class Reference

Inheritance diagram for VsxProtocolDriver.Sensor.Sensor:



Public Member Functions

- def `__init__` (self, *_is_direct=True)
- def `__del__` (self)
- Optional[bool] `Handle` (self)
Get the actual used handle from the actual sensor instance.
- def `InitTcpSensor` (cls, str ip_address, str plugin_name="")
Initialize a new tcp based sensor.
- def `InitSerialSensor` (cls, str serial_port, int baud_rate, str sensor_type, [SerialConnectionType](#) connection_type, str plugin_name="")
Init an instance to communicate with a Vsx-Device via serial protocol.
- Tuple[int, Optional[str], int] `TestSystem` (self, str command, str input_value)
Sends a test system command to the device.
- Tuple[int, Optional[str], int] `TestSystemEx` (self, str command, str input_value, int timeout_ms)
Sends a test system command to the device.
- int `SendFirmware` (self, str file_name)
Sends a firmware update file to the device.
- int `UploadData` (self, str file_name)
Sends a data file (either image data or dynamic container data) to the device.
- int `UploadParameterSet` (self, str file_name)
Uploads a parameter file to the device.
- int `DownloadParameterSet` (self, str file_name)
Save the current parameter set to a file.
- int `SendXmlDataMessage` (self, str xml_command)
Sends a string to the device.
- int `LoadDefaultParameterSetOnDevice` (self)
Resets the devices parameters to factory settings and returns a list of the complete parameter set of the device including current values.
- int `LoadParameterSetOnDevice` (self)
Loads the parameter set saved on device and returns a list of the complete parameter set of the device including current values.
- int `SaveParameterSetOnDevice` (self)
Saves the current parameter set on device.
- int `Connect` (self)
Connect with the device.
- int `ConnectEx` (self, int timeout_ms)
Connect with the device.
- int `ConnectAndLogin` (self, str username, str password)

- Connect with the device.*

 - int [ConnectExAndLogin](#) (self, str username, str password, int timeout_ms)
- Connect with the device.*

 - int [Login](#) (self, str username, str password)
- Login to the device.*

 - int [Logout](#) (self)
- Logout from device.*

 - int [InitializeDevice](#) (self, str password, [DeviceAuthenticationMode](#) authentication_mode)
- Initializes a brand-new (factory reset) device supporting multiple authentication modes.*

 - int [SetPassword](#) (self, str authorization_username, str authorization_password, str username, str password)
- Set new password on the device.*

 - Optional[bool] [Connected](#) (self)
- Indicates current connection state with the device.*

 - int [Disconnect](#) (self)
- Disconnect with the device.*

 - int [RegisterOnDisconnect](#) (self, Callable[[int, str, [DisconnectEvent](#), str], None] func)
- Register a callback function for a disconnect event.*

 - int [DeregisterOnDisconnect](#) (self)
- Function to deregister already existing callback function.*

 - int [RegisterOnSessionMessageReceived](#) (self, Callable[[int, [SessionTypes](#), int], None] func)
- Register a callback function for a session message event.*

 - int [DeregisterOnSessionMessageReceived](#) (self)
- Function to deregister already existing callback function.*

 - int [SendSessionKeepAlive](#) (self)
- Send session keep alive to sensor.*

 - int [ReConnectTcpSensor](#) (self, str ip_address)
 - int [ReConnectTcpDevice](#) (self, str ip_address)
- Disconnects the device and reconnects with new connection settings.*

 - int [ReConnectAndLoginTcpDevice](#) (self, str ip_address, str username, str password)
- Disconnects the device and reconnects with new connection settings.*

 - int [ReConnectSerialSensor](#) (self, str serial_port, int baud_rate, str _, [SerialConnectionType](#) connection_type)
 - int [ReConnectSerialDevice](#) (self, str serial_port, int baud_rate, [SerialConnectionType](#) connection_type)
- Disconnects the device and reconnects with new connection settings.*

 - int [ReleaseSensor](#) (self)
- Frees the given sensor :return: Returns VSX_STATUS_SUCCESS(0) on success.*

 - int [SetWaitTimeout](#) (self, timeout_ms)
- Sets the time in ms, the driver waits for response from device.*

 - Tuple[int, int] [GetWaitTimeout](#) (self)
- Gets the time in ms, the driver waits for response from device.*

 - def [SetNetworkParameter](#) (self, str ip_address, str network_mask, str gateway)
 - def [SetNetworkSettings](#) (self, str ip_address, str network_mask, str gateway)
- Sets the network settings of the device.*

 - int [SetSingleDataValue](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id, str value)
 - int [SetSingleParameterValue](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id, Union[str, int, float] value)
- Sets the parameter to a value on the device.*

 - Tuple[int, str] [GetSingleDataValue](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id)
 - Tuple[int, Optional[str]] [GetSingleParameterValue](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id)

- Returns the current value of the given parameter from device.*
- Tuple[int, Optional[int]] [GetSingleParameterValueInt32](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id)
- Returns the current value of the given parameter from device.*
- Tuple[int, Optional[float]] [GetSingleParameterValueDouble](#) (self, int settings_version, str configuration_id, int configuration_version, str parameter_id)
- Returns the current value of the given parameter from device.*
- Tuple[int, Optional[List[Parameter]]] [GetParameterList](#) (self)
- Returns a list of the complete parameter set of the device including their current values.*
- int [UploadParameterList](#) (self, List[Parameter] parameter_list_data)
- Uploads a parameter list to the device.*
- int [SetSingleParameter](#) (self, Parameter parameter)
- Sets the parameter to a value on the device.*
- Tuple[int, Optional[Parameter]] [GetSingleParameter](#) (self, Parameter parameter)
- Returns the current value of the given parameter from device.*
- int [ResetLogMessageGrabber](#) (self, int buffer_size, int type_mask, Strategy strategy)
- Starts the internal log message grabber.*
- Tuple[int, Optional[str]] [GetLogMessage](#) (self, int timeout_ms)
- Gets the oldest saved item and removes it internally.*
- Optional[int] [LogMessageQueueSize](#) (self)
- Gets the current size of the log message queue.*
- Optional[int] [MissingLogMessagesCounter](#) (self)
- Gets the missing log messages counter for log message grabbing.*
- int [ResetDynamicContainerGrabber](#) (self, int buffer_size, int __, Strategy strategy)
- int [ResetDynamicContainerGrabberEx](#) (self, int buffer_size, Strategy strategy)
- Restarts the internal dynamic container grabber.*
- Tuple[int, Optional[DataContainer]] [GetDataContainer](#) (self, int timeout_ms)
- Gets the oldest saved item and removes it internally.*
- Tuple[int, Optional[DataContainer]] [GetCachedContainer](#) (self, int position)
- Gets a cached dynamic container.*
- Optional[int] [MissingContainerFramesCounter](#) (self)
- Gets the missing frame counter from image grabbing.*
- Optional[int] [DynamicContainerQueueSize](#) (self)
- Gets the current size of the dynamic container message queue.*
- Optional[int] [NumberOfCachedContainers](#) (self)
- Gets the current number of cached container messages.*
- Tuple[int, Optional[Dict[str, str]]] [GetCurrentDeviceInformation](#) (self)
- Tuple[int, Optional[Dict[str, str]]] [GetDeviceInformation](#) (self)
- Returns a device object with network information about the current device.*
- Tuple[int, Optional[List[StatusItem]]] [GetAllDeviceStatusData](#) (self)
- Get the full status data set from device.*
- int [RegisterOnDeviceStatusReceived](#) (self, Callable[[int, DeviceStatusScope, List[StatusItem]], None] func)
- Register a callback function for a disconnect event.*
- int [DeregisterOnDeviceStatusReceived](#) (self)
- Function to deregister already existing callback function.*
- int [SubscribeToDeviceStatusData](#) (self)
- Subscribe status data from sensor to the client.*
- int [UnsubscribeToDeviceStatusData](#) (self)
- Unsubscribe status data from sensor.*

Static Public Member Functions

- Optional[str] [GetLibraryVersion](#) ()
Returns the actual library version.
- Optional[str] [GetErrorText](#) (int error_code)
Return the error text to a given error code.
- def [SetNetworkSettingsViaUdp](#) (str mac_address, str ip_address, str network_mask, str gateway)
Sets the network settings of the device identified by the macAddress via UDP.
- Tuple[int, Optional[List[Dict[str, str]]]] [GetUdpDeviceList](#) ()
Searches for all devices in a subnet via udp and returns a list with all devices found.

10.8.1 Constructor & Destructor Documentation

10.8.1.1 `__init__()` `def VsxProtocolDriver.Sensor.Sensor.__init__ (`
 `self,`
 `* _is_direct = True)`

10.8.1.2 `__del__()` `def VsxProtocolDriver.Sensor.Sensor.__del__ (`
 `self)`

10.8.2 Member Function Documentation

10.8.2.1 `GetLibraryVersion()` `Optional[str] VsxProtocolDriver.Sensor.Sensor.GetLibraryVersion ()`
[static]

Returns the actual library version.

10.8.2.2 `GetErrorText()` `Optional[str] VsxProtocolDriver.Sensor.Sensor.GetErrorText (`
 `int error_code) [static]`

Return the error text to a given error code.

It also appends additional text from last error given.

Parameters

<code>error_code</code>	Input error code
-------------------------	------------------

Returns

: error text

10.8.2.3 Handle() `Optional[bool] VsxProtocolDriver.Sensor.Sensor.Handle (self)`

Get the actual used handle from the actual sensor instance.

10.8.2.4 InitTcpSensor() `def VsxProtocolDriver.Sensor.Sensor.InitTcpSensor (cls, str ip_address, str plugin_name = '')`

Initialize a new tcp based sensor.

Parameters

<i>ip_address</i>	e.g. "192.168.2.4"
<i>plugin_name</i>	Additional functionality for special sensors

Returns

: New instance of [Sensor](#) class

10.8.2.5 InitSerialSensor() `def VsxProtocolDriver.Sensor.Sensor.InitSerialSensor (cls, str serial_port, int baud_rate, str sensor_type, SerialConnectionType connection_type, str plugin_name = '')`

Initns an instance to communicate with a Vsx-Device via serial protocol.

Parameters

<i>serial_port</i>	The comport of the device
<i>baud_rate</i>	The baudrate of the device
<i>sensor_type</i>	The sensor type of the device
<i>connection_type</i>	The connection type of the device
<i>plugin_name</i>	Additional functionality for special sensors

Returns

:

10.8.2.6 TestSystem() Tuple[int, Optional[str], int] VsxProtocolDriver.Sensor.Sensor.Test↔
System (

```
    self,  
    str command,  
    str input_value )
```

Sends a test system command to the device.

Parameters

<i>command</i>	The test system command
<i>input_value</i>	Optional input value

Returns

: error code, output string of function call, status (1 on success and 0 on failure)

10.8.2.7 TestSystemEx() Tuple[int, Optional[str], int] VsxProtocolDriver.Sensor.Sensor.Test↔
SystemEx (

```
    self,  
    str command,  
    str input_value,  
    int timeout_ms )
```

Sends a test system command to the device.

Parameters

<i>command</i>	The test system command
<i>input_value</i>	Optional input value
<i>timeout_ms</i>	Timeout for function to execute command (in ms)

Returns

: error code, output string of function call, status (1 on success and 0 on failure)

10.8.2.8 SendFirmware() int VsxProtocolDriver.Sensor.Sensor.SendFirmware (

```
    self,  
    str file_name )
```

Sends a firmware update file to the device.

Parameters

<i>file_name</i>	The path and filename of the firmware file
------------------	--

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.9 UploadData() `int VsxProtocolDriver.Sensor.Sensor.UploadData (`
 `self,`
 `str file_name)`

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

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.10 UploadParameterSet() `int VsxProtocolDriver.Sensor.Sensor.UploadParameterSet (`
 `self,`
 `str file_name)`

Uploads a parameter file to the device.

Parameters

<i>file_name</i>	Path and filename to upload
------------------	-----------------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.11 DownloadParameterSet() `int VsxProtocolDriver.Sensor.Sensor.DownloadParameterSet (`
 `self,`
 `str file_name)`

Save the current parameter set to a file.

Parameters

<i>file_name</i>	Path and file name to save to
------------------	-------------------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.12 SendXmlDataMessage() `int VsxProtocolDriver.Sensor.Sensor.SendXmlDataMessage (`
 `self,`
 `str xml_command)`

Sends a string to the device.

NOTE: function does not wait for any device reply.

Parameters

<i>xml_command</i>	Command to send
--------------------	-----------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.13 LoadDefaultParameterSetOnDevice() `int VsxProtocolDriver.Sensor.Sensor.LoadDefault↔`
`ParameterSetOnDevice (`
 `self)`

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.14 LoadParameterSetOnDevice() `int VsxProtocolDriver.Sensor.Sensor.LoadParameterSetOn↔`
`Device (`
 `self)`

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.15 SaveParameterSetOnDevice() `int VsxProtocolDriver.Sensor.Sensor.SaveParameterSetOnDevice (
 self)`

Saves the current parameter set on device.

Parameter values will be loaded when device starts.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.16 Connect() `int VsxProtocolDriver.Sensor.Sensor.Connect (
 self)`

Connect with the device.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.17 ConnectEx() `int VsxProtocolDriver.Sensor.Sensor.ConnectEx (
 self,
 int timeout_ms)`

Connect with the device.

Parameters

<i>timeout_ms</i>	The timeout for a connection attempt (in ms)
-------------------	--

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.18 ConnectAndLogin() `int VsxProtocolDriver.Sensor.Sensor.ConnectAndLogin (
 self,
 str username,
 str password)`

Connect with the device.

Parameters

<i>username</i>	username for login
<i>password</i>	password for login

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.19 ConnectExAndLogin() `int VsxProtocolDriver.Sensor.Sensor.ConnectExAndLogin (`
 `self,`
 `str username,`
 `str password,`
 `int timeout_ms)`

Connect with the device.

Parameters

<i>username</i>	username for login
<i>password</i>	password for login
<i>timeout_ms</i>	The timeout for a connection attempt (in ms)

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.20 Login() `int VsxProtocolDriver.Sensor.Sensor.Login (`
 `self,`
 `str username,`
 `str password)`

Login to the device.

Parameters

<i>username</i>	username for login
<i>password</i>	password for login

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.21 Logout() `int VsxProtocolDriver.Sensor.Sensor.Logout (`
 `self)`

Logout from device.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.22 InitializeDevice() `int VsxProtocolDriver.Sensor.Sensor.InitializeDevice (`
 `self,`
 `str password,`
 `DeviceAuthenticationMode authentication_mode)`

Initializes a brand-new (factory reset) device supporting multiple authentication modes.

This must be done first before device can be used.

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.23 SetPassword() `int VsxProtocolDriver.Sensor.Sensor.SetPassword (`
 `self,`
 `str authorization_username,`
 `str authorization_password,`
 `str username,`
 `str password)`

Set new password on the device.

Parameters

<i>authorization_username</i>	username for authorization account
<i>authorization_password</i>	password for authorization account
<i>username</i>	username for account to set new password
<i>password</i>	password for account

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.24 Connected() `Optional[bool] VsxProtocolDriver.Sensor.Sensor.Connected (`
 `self)`

Indicates current connection state with the device.

Returns

: Returns True or False

10.8.2.25 Disconnect() `int VsxProtocolDriver.Sensor.Sensor.Disconnect (`
`self)`

Disconnect with the device.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.2.26 RegisterOnDisconnect() `int VsxProtocolDriver.Sensor.Sensor.RegisterOnDisconnect (`
`self,`
`Callable[[int, str, DisconnectEvent, str], None] func)`

Register a callback function for a disconnect event.

Watch out: The callback will be in another thread (this could be not debuggable).

The function should have the following definition:: `def MyOnDisconnect(handle: int, ip_address: str, disconnect↵
Event: int, description: str) -> None:`

10.8.3 your implementation (not working thread, so watch thread safety!)

Parameters

<i>func</i>	The function should have the following definition
-------------	---

Returns

: Return value

10.8.3.1 DeregisterOnDisconnect() `int VsxProtocolDriver.Sensor.Sensor.DeregisterOnDisconnect (`
`self)`

Function to deregister already existing callback function.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.3.2 RegisterOnSessionMessageReceived() `int VsxProtocolDriver.Sensor.Sensor.RegisterOn↵
SessionMessageReceived (`
`self,`
`Callable[[int, SessionTypes, int], None] func)`

Register a callback function for a session message event.

Watch out: The callback will be in another thread (this could be not debuggable).

The function should have the following definition:: `def MyOnSessionMessageReceived(handle: int, session_type:↵
int, timeout: int) -> None:`

10.8.4 your implementation (not working thread, so watch thread safety!)**Parameters**

<i>func</i>	The function should have the following definition
-------------	---

Returns

: Return value

10.8.4.1 DeregisterOnSessionMessageReceived() `int VsxProtocolDriver.Sensor.Sensor.DeregisterOnSessionMessageReceived (
 self)`

Function to deregister already existing callback function.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.2 SendSessionKeepAlive() `int VsxProtocolDriver.Sensor.Sensor.SendSessionKeepAlive (
 self)`

Send session keep alive to sensor.

Should be the reply from a timeout announcement message

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.3 ReConnectTcpSensor() `int VsxProtocolDriver.Sensor.Sensor.ReConnectTcpSensor (
 self,
 str ip_address)`

10.8.4.4 ReConnectTcpDevice() `int VsxProtocolDriver.Sensor.Sensor.ReConnectTcpDevice (
 self,
 str ip_address)`

Disconnects the device and reconnects with new connection settings.

Parameters

<i>ip_address</i>	The new IPAddress
-------------------	-------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.5 ReConnectAndLoginTcpDevice() `int VsxProtocolDriver.Sensor.Sensor.ReConnectAndLoginTcpDevice (`

```
    self,  
    str ip_address,  
    str username,  
    str password )
```

Disconnects the device and reconnects with new connection settings.

Parameters

<i>ip_address</i>	The new IPAddress
<i>username</i>	username for login
<i>password</i>	password for login

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.6 ReConnectSerialSensor() `int VsxProtocolDriver.Sensor.Sensor.ReConnectSerialSensor (`

```
    self,  
    str serial_port,  
    int baud_rate,  
    str _,  
    SerialConnectionType connection_type )
```

10.8.4.7 ReConnectSerialDevice() `int VsxProtocolDriver.Sensor.Sensor.ReConnectSerialDevice (`

```
    self,  
    str serial_port,  
    int baud_rate,  
    SerialConnectionType connection_type )
```

Disconnects the device and reconnects with new connection settings.

Parameters

<i>serial_port</i>	The new serial port
<i>baud_rate</i>	The new baudrate
<i>connection_type</i>	The new connection type

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.8 ReleaseSensor() `int VsxProtocolDriver.Sensor.Sensor.ReleaseSensor (
 self)`

Frees the given sensor :return: Returns VSX_STATUS_SUCCESS(0) on success.

10.8.4.9 SetWaitTimeout() `int VsxProtocolDriver.Sensor.Sensor.SetWaitTimeout (
 self,
 timeout_ms)`

Sets the time in ms, the driver waits for response from device.

Parameters

<i>timeout_ms</i>	Time in ms
-------------------	------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.10 GetWaitTimeout() `Tuple[int, int] VsxProtocolDriver.Sensor.Sensor.GetWaitTimeout (
 self)`

Gets the time in ms, the driver waits for response from device.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success, Time in ms

10.8.4.11 SetNetworkParameter() `def VsxProtocolDriver.Sensor.Sensor.SetNetworkParameter (`
 `self,`
 `str ip_address,`
 `str network_mask,`
 `str gateway)`

10.8.4.12 SetNetworkSettings() `def VsxProtocolDriver.Sensor.Sensor.SetNetworkSettings (`
 `self,`
 `str ip_address,`
 `str network_mask,`
 `str gateway)`

Sets the network settings of the device.

Parameters

<i>ip_address</i>	The new IP Address
<i>network_mask</i>	The new network mask
<i>gateway</i>	The new gateway address

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.13 SetNetworkSettingsViaUdp() `def VsxProtocolDriver.Sensor.Sensor.SetNetworkSettings↵`
`ViaUdp (`
 `str mac_address,`
 `str ip_address,`
 `str network_mask,`
 `str gateway) [static]`

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

Parameters

<i>mac_address</i>	The mac address of the device to set
<i>ip_address</i>	The new IP Address
<i>network_mask</i>	The new network mask
<i>gateway</i>	The new gateway address

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.14 SetSingleDataValue() `int VsxProtocolDriver.Sensor.Sensor.SetSingleDataValue (`
 `self,`
 `int settings_version,`
 `str configuration_id,`
 `int configuration_version,`
 `str parameter_id,`
 `str value)`

10.8.4.15 SetSingleParameterValue() `int VsxProtocolDriver.Sensor.Sensor.SetSingleParameterValue`
`(`
 `self,`
 `int settings_version,`
 `str configuration_id,`
 `int configuration_version,`
 `str parameter_id,`
 `Union[str, int, float] value)`

Sets the parameter to a value on the device.

Parameters

<i>settings_version</i>	The settings version of the parameter which should be set
<i>configuration_id</i>	The config id of the parameter which should be set
<i>configuration_version</i>	The config version of the parameter which should be se
<i>parameter_id</i>	The id of the parameter which should be set
<i>value</i>	Value as string, float or integer number

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.16 GetSingleDataValue() `Tuple[int, str] VsxProtocolDriver.Sensor.Sensor.GetSingleData↔`
`Value (`
 `self,`
 `int settings_version,`
 `str configuration_id,`
 `int configuration_version,`
 `str parameter_id)`

10.8.4.17 GetSingleParameterValue() `Tuple[int, Optional[str]] VsxProtocolDriver.Sensor.Sensor.↔`
`GetSingleParameterValue (`
 `self,`
 `int settings_version,`
 `str configuration_id,`
 `int configuration_version,`
 `str parameter_id)`

Returns the current value of the given parameter from device.

Parameters

<i>settings_version</i>	The settings version of the parameter its value is asked for
<i>configuration_id</i>	The config id of the parameter its value is asked for
<i>configuration_version</i>	The config version of the parameter its value is asked for
<i>parameter_id</i>	The id of the parameter its value is asked for

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and value in string representation

10.8.4.18 GetSingleParameterValueInt32() Tuple[int, Optional[int]] VsxProtocolDriver.Sensor.↔

```
Sensor.GetSingleParameterValueInt32 (
    self,
    int settings_version,
    str configuration_id,
    int configuration_version,
    str parameter_id )
```

Returns the current value of the given parameter from device.

Parameters

<i>settings_version</i>	The settings version of the parameter its value is asked for
<i>configuration_id</i>	The config id of the parameter its value is asked for
<i>configuration_version</i>	The config version of the parameter its value is asked for
<i>parameter_id</i>	The id of the parameter its value is asked for

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and value in integer representation

10.8.4.19 GetSingleParameterValueDouble() Tuple[int, Optional[float]] VsxProtocolDriver.↔

```
Sensor.Sensor.GetSingleParameterValueDouble (
    self,
    int settings_version,
    str configuration_id,
    int configuration_version,
    str parameter_id )
```

Returns the current value of the given parameter from device.

Parameters

<i>settings_version</i>	The settings version of the parameter its value is asked for
<i>configuration_id</i>	The config id of the parameter its value is asked for
<i>configuration_version</i>	The config version of the parameter its value is asked for
<i>parameter_id</i>	The id of the parameter its value is asked for

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and value in float representation

10.8.4.20 GetParameterList() Tuple[int, Optional[List[Parameter]]] VsxProtocolDriver.Sensor.Sensor.GetParameterList (
self)

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

The list shows the current state of the parameters.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and parameter list data

10.8.4.21 UploadParameterList() int VsxProtocolDriver.Sensor.Sensor.UploadParameterList (
self,
List[Parameter] parameter_list_data)

Uploads a parameter list to the device.

Parameters

<i>parameter_list_data</i>	
----------------------------	--

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.22 SetSingleParameter() int VsxProtocolDriver.Sensor.Sensor.SetSingleParameter (
self,
Parameter parameter)

Sets the parameter to a value on the device.

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.23 GetSingleParameter() Tuple[int, Optional[Parameter]] VsxProtocolDriver.Sensor.Sensor.
 Sensor.GetSingleParameter (
 self,
 Parameter *parameter*)

Returns the current value of the given parameter from device.

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and the new parameter

10.8.4.24 ResetLogMessageGrabber() int VsxProtocolDriver.Sensor.Sensor.ResetLogMessageGrabber
 (
 self,
 int *buffer_size*,
 int *type_mask*,
 Strategy *strategy*)

Starts the internal log message grabber.

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.25 GetLogMessage() Tuple[int, Optional[str]] VsxProtocolDriver.Sensor.Sensor.GetLogMessage
 (
 self,
 int *timeout_ms*)

Gets the oldest saved item and removes it internally.

Parameters

<i>timeout_ms</i>	The maximum time in ms to try reading an item
-------------------	---

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.26 LogMessageQueueSize() Optional[int] VsxProtocolDriver.Sensor.Sensor.LogMessage↵
QueueSize (
 self)

Gets the current size of the log message queue.

Returns

: Actual log message counter

10.8.4.27 MissingLogMessagesCounter() Optional[int] VsxProtocolDriver.Sensor.Sensor.Missing↵
LogMessagesCounter (
 self)

Gets the missing log messages counter for log message grabbing.

Returns

: Missing log messages counter

10.8.4.28 ResetDynamicContainerGrabber() int VsxProtocolDriver.Sensor.Sensor.ResetDynamic↵
ContainerGrabber (
 self,
int *buffer_size*,
int *_*,
Strategy *strategy*)

10.8.4.29 ResetDynamicContainerGrabberEx() int VsxProtocolDriver.Sensor.Sensor.ResetDynamic↵
ContainerGrabberEx (
 self,
int *buffer_size*,
Strategy *strategy*)

Restarts the internal dynamic container grabber.

Parameters

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

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.4.30 GetDataContainer() Tuple[int, Optional[DataContainer]] VsxProtocolDriver.Sensor.↔
Sensor.GetDataContainer (
 self,
 int *timeout_ms*)

Gets the oldest saved item and removes it internally.

Parameters

<i>timeout_ms</i>	The maximum time in ms to try reading an item.
-------------------	--

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and new dynamic container

10.8.4.31 GetCachedContainer() Tuple[int, Optional[DataContainer]] VsxProtocolDriver.Sensor.↔
Sensor.GetCachedContainer (
 self,
 int *position*)

Gets a cached dynamic container.

Parameters

<i>position</i>	Position of the container in cache
-----------------	------------------------------------

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and new dynamic container

10.8.4.32 MissingContainerFramesCounter() Optional[int] VsxProtocolDriver.Sensor.Sensor.↔
MissingContainerFramesCounter (
 self)

Gets the missing frame counter from image grabbing.

Returns

: Actual missing frame counter

10.8.4.33 DynamicContainerQueueSize() `Optional[int] VsxProtocolDriver.Sensor.Sensor.Dynamic↔
ContainerQueueSize (
 self)`

Gets the current size of the dynamic container message queue.

Returns

: Returns actual queue size

10.8.4.34 NumberOfCachedContainers() `Optional[int] VsxProtocolDriver.Sensor.Sensor.NumberOf↔
CachedContainers (
 self)`

Gets the current number of cached container messages.

Returns

: Returns actual cached containers

10.8.4.35 GetCurrentDeviceInformation() `Tuple[int, Optional[Dict[str, str]]] VsxProtocolDriver.↔
Sensor.Sensor.GetCurrentDeviceInformation (
 self)`

10.8.4.36 GetDeviceInformation() `Tuple[int, Optional[Dict[str, str]]] VsxProtocolDriver.↔
Sensor.Sensor.GetDeviceInformation (
 self)`

Returns a device object with network information about the current device.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and device data object

10.8.4.37 GetUdpDeviceList() `Tuple[int, Optional[List[Dict[str, str]]]] VsxProtocolDriver.↔
Sensor.Sensor.GetUdpDeviceList () [static]`

Searches for all devices in a subnet via udp and returns a list with all devices found.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success and list of device information

10.8.4.38 GetAllDeviceStatusData() `Tuple[int, Optional[List[StatusItem]]]` `VsxProtocolDriver.Sensor.Sensor.GetAllDeviceStatusData (self)`

Get the full status data set from device.

Returns

: Returns `VSX_STATUS_SUCCESS(0)` on success and list of status data objects

10.8.4.39 RegisterOnDeviceStatusReceived() `int` `VsxProtocolDriver.Sensor.Sensor.RegisterOnDeviceStatusReceived (self, Callable[[int, DeviceStatusScope, List[StatusItem]], None] func)`

Register a callback function for a disconnect event.

Watch out: The callback will be in another thread (this could be not debuggable).

The function should have the following definition:: `def MyOnDeviceStatusReceived(handle: int, device_status_scope: DeviceStatusScope, status_item_list: List[StatusItem]) -> None:`

10.8.5 your implementation (not working thread, so watch thread safety!)

Parameters

<i>func</i>	The function should have the following definition:
-------------	--

Returns

:

10.8.5.1 DeregisterOnDeviceStatusReceived() `int` `VsxProtocolDriver.Sensor.Sensor.DeregisterOnDeviceStatusReceived (self)`

Function to deregister already existing callback function.

Returns

: Returns `VSX_STATUS_SUCCESS(0)` on success

10.8.5.2 SubscribeToDeviceStatusData() `int VsxProtocolDriver.Sensor.Sensor.SubscribeToDeviceStatusData (self)`

Subscribe status data from sensor to the client.

This will send periodically or in case of a problem status data to the client. This need a registered callback for "vsx_OnDeviceStatusReceived".

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.8.5.3 UnsubscribeToDeviceStatusData() `int VsxProtocolDriver.Sensor.Sensor.UnsubscribeToDeviceStatusData (self)`

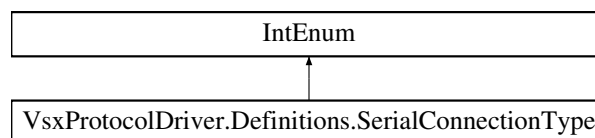
Unsubscribe status data from sensor.

Returns

: Returns VSX_STATUS_SUCCESS(0) on success

10.9 VsxProtocolDriver.Definitions.SerialConnectionType Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.SerialConnectionType:



Static Public Attributes

- int `USB_SSI` = 0
- int `PROFIBUS` = 1
- int `PROFINET` = 2
- int `ETHERNET_IP` = 3
- int `RS485` = 4
- int `CANOPEN` = 5

10.9.1 Member Data Documentation

10.9.1.1 USB_SSI `int VsxProtocolDriver.Definitions.SerialConnectionType.USB_SSI = 0 [static]`

10.9.1.2 PROFIBUS `int VsxProtocolDriver.Definitions.SerialConnectionType.PROFIBUS = 1 [static]`

10.9.1.3 PROFINET `int VsxProtocolDriver.Definitions.SerialConnectionType.PROFINET = 2 [static]`

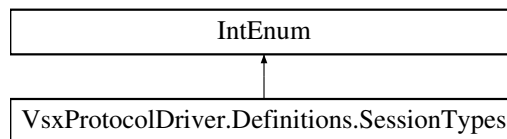
10.9.1.4 ETHERNET_IP `int VsxProtocolDriver.Definitions.SerialConnectionType.ETHERNET_IP = 3 [static]`

10.9.1.5 RS485 `int VsxProtocolDriver.Definitions.SerialConnectionType.RS485 = 4 [static]`

10.9.1.6 CANOPEN `int VsxProtocolDriver.Definitions.SerialConnectionType.CANOPEN = 5 [static]`

10.10 VsxProtocolDriver.Definitions.SessionTypes Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.SessionTypes:



Static Public Attributes

- `int LOGIN_REQUIRED = 0`
- `int INITIAL_PASSWORD_REQUIRED = 1`
- `int LOGIN = 2`
- `int LOGIN_REPLY = 3`
- `int SET_PASSWORD = 4`
- `int SET_PASSWORD_REPLY = 5`
- `int TIMEOUT_ANNOUNCEMENT = 6`
- `int TIMEOUT = 7`
- `int LOGOUT = 8`
- `int LOGOUT_REPLY = 9`
- `int UNKNOWN = 10`

10.10.1 Member Data Documentation

10.10.1.1 LOGIN_REQUIRED `int VsxProtocolDriver.Definitions.SessionTypes.LOGIN_REQUIRED = 0`
[static]

10.10.1.2 INITIAL_PASSWORD_REQUIRED `int VsxProtocolDriver.Definitions.SessionTypes.↔
INITIAL_PASSWORD_REQUIRED = 1` [static]

10.10.1.3 LOGIN `int VsxProtocolDriver.Definitions.SessionTypes.LOGIN = 2` [static]

10.10.1.4 LOGIN_REPLY `int VsxProtocolDriver.Definitions.SessionTypes.LOGIN_REPLY = 3` [static]

10.10.1.5 SET_PASSWORD `int VsxProtocolDriver.Definitions.SessionTypes.SET_PASSWORD = 4`
[static]

10.10.1.6 SET_PASSWORD_REPLY `int VsxProtocolDriver.Definitions.SessionTypes.SET_PASSWORD↔
_REPLY = 5` [static]

10.10.1.7 TIMEOUT_ANNOUNCEMENT `int VsxProtocolDriver.Definitions.SessionTypes.TIMEOUT_↔
ANNOUNCEMENT = 6` [static]

10.10.1.8 TIMEOUT `int VsxProtocolDriver.Definitions.SessionTypes.TIMEOUT = 7` [static]

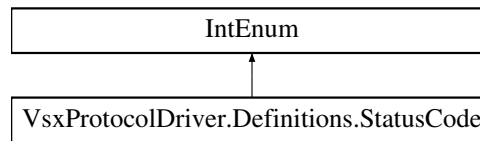
10.10.1.9 LOGOUT `int VsxProtocolDriver.Definitions.SessionTypes.LOGOUT = 8` [static]

10.10.1.10 LOGOUT_REPLY `int VsxProtocolDriver.Definitions.SessionTypes.LOGOUT_REPLY = 9`
[static]

10.10.1.11 UNKNOWN `int VsxProtocolDriver.Definitions.SessionTypes.UNKNOWN = 10` [static]

10.11 VsxProtocolDriver.Definitions.StatusCode Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.StatusCode:



Static Public Attributes

- int [VSX_STATUS_SUCCESS](#) = 0
- int [VSX_STATUS_ERROR_DRIVER_INIT](#) = -0x1
- int [VSX_STATUS_ERROR_DRIVER_TIMEOUT](#) = -0x2
- int [VSX_STATUS_ERROR_DRIVER_SAVE_FILE](#) = -0x3
- int [VSX_STATUS_ERROR_DRIVER_DATA](#) = -0x4
- int [VSX_STATUS_ERROR_DRIVER_CONNECTION](#) = -0x5
- int [VSX_STATUS_ERROR_DRIVER_INVALID_DATA](#) = -0x6
- int [VSX_STATUS_ERROR_DRIVER_DEVICE](#) = -0x7
- int [VSX_STATUS_ERROR_DRIVER_LOAD_FILE](#) = -0x8
- int [VSX_STATUS_ERROR_SESSION](#) = -0x9
- int [VSX_STATUS_ERROR_STRING](#) = -0x0A
- int [VSX_STATUS_ERROR_VERSION](#) = -0x0B
- int [VSX_STATUS_ERROR_DRIVER_GENERAL](#) = -0x1000
- int [VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM](#) = -0x8001
- int [VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO](#) = -0x8002
- int [VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO](#) = -0x8003
- int [VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE](#) = -0x8004
- int [VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION](#) = -0x8005
- int [VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION](#) = -0x8006
- int [VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO](#) = -0x8007
- int [VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO](#) = -0x8008
- int [VSX_STATUS_ERROR_PARAMETER_ID_ZERO](#) = -0x8009
- int [VSX_STATUS_ERROR_VALUE_ZERO](#) = -0x800A
- int [VSX_STATUS_ERROR_COMMAND_ZERO](#) = -0x800B
- int [VSX_STATUS_ERROR_INPUT_VALUE_ZERO](#) = -0x800C
- int [VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO](#) = -0x800D
- int [VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO](#) = -0x800E
- int [VSX_STATUS_ERROR_VALUE_POINTER_ZERO](#) = -0x800F
- int [VSX_STATUS_ERROR_VALUE_NOT_ZERO](#) = -0x8010
- int [VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM](#) = -0x8011
- int [VSX_STATUS_ERROR_XML_COMMAND_ZERO](#) = -0x8012
- int [VSX_STATUS_ERROR_FILENAME_ZERO](#) = -0x8013
- int [VSX_STATUS_ERROR_STRING_POINTER_ZERO](#) = -0x8014
- int [VSX_STATUS_ERROR_STRING_ZERO](#) = -0x8015
- int [VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_POINTER_ZERO](#) = -0x8016
- int [VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER](#) = -0x8017
- int [VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_ZERO](#) = -0x8018
- int [VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_ZERO](#) = -0x8019
- int [VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_AVAILABLE](#) = -0x801A
- int [VSX_STATUS_ERROR_IMAGE_TAG_ZERO](#) = -0x801B
- int [VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER](#) = -0x801C

- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_CONTAINER = -0x801D`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_FORMAT = -0x801E`
- `int VSX_STATUS_ERROR_POINT_Z_ID_ZERO = -0x801F`
- `int VSX_STATUS_ERROR_POINT_Y_ID_ZERO = -0x8020`
- `int VSX_STATUS_ERROR_POINT_X_ID_ZERO = -0x8021`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_DATA_CONTAINER = -0x8022`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_DATA_CONTAINER = -0x8023`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_DATA_CONTAINER = -0x8024`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_DATA_FORMAT = -0x8025`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_DATA_FORMAT = -0x8026`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_DATA_FORMAT = -0x8027`
- `int VSX_STATUS_ERROR_LOG_POINTER_ZERO = -0x8028`
- `int VSX_STATUS_ERROR_LOG_NOT_ZERO = -0x8029`
- `int VSX_STATUS_ERROR_RESULT_NOT_ZERO = -0x802A`
- `int VSX_STATUS_ERROR_RESULT_POINTER_ZERO = -0x802B`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_DATA_CONTAINER = -0x802C`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_DATA_FORMAT = -0x802D`
- `int VSX_STATUS_ERROR_VERSION_POINTER_ZERO = -0x802E`
- `int VSX_STATUS_ERROR_VERSION_NOT_ZERO = -0x802F`
- `int VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO = -0x8030`
- `int VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO = -0x8031`
- `int VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE = -0x8032`
- `int VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VALUE = -0x8033`
- `int VSX_STATUS_ERROR_XPATH_ZERO = -0x8034`
- `int VSX_STATUS_ERROR_INVALID_DATA_FORMAT = -0x8035`
- `int VSX_STATUS_ERROR_NO_ELEMENT_FOUND = -0x8036`
- `int VSX_STATUS_ERROR_RESULT_TAG_ZERO = -0x8037`
- `int VSX_STATUS_ERROR_TAG_ZERO = -0x8038`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER = -0x8039`
- `int VSX_STATUS_ERROR_IP_ADDRESS_ZERO = -0x803A`
- `int VSX_STATUS_ERROR_NETWORK_MASK_ZERO = -0x803B`
- `int VSX_STATUS_ERROR_GATEWAY_ZERO = -0x803C`
- `int VSX_STATUS_ERROR_EXCEPTION_THROWN = -0x803D`
- `int VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO = -0x803E`
- `int VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO = -0x803F`
- `int VSX_STATUS_ERROR_VSX_IMAGE_ZERO = -0x8040`
- `int VSX_STATUS_ERROR_VSX_DEVICE_ZERO = -0x8041`
- `int VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO = -0x8042`
- `int VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO = -0x8043`
- `int VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO = -0x8044`
- `int VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO = -0x8045`
- `int VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO = -0x8046`
- `int VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO = -0x8047`
- `int VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO = -0x8048`
- `int VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO = -0x8049`
- `int VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO = -0x804A`
- `int VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO = -0x804B`
- `int VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO = -0x804C`
- `int VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO = -0x804D`
- `int VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO = -0x804E`
- `int VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO = -0x804F`
- `int VSX_STATUS_ERROR_MAC_ADDRESS_ZERO = -0x8050`
- `int VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND = -0x8051`
- `int VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO = -0x8052`
- `int VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO = -0x8053`

- `int VSX_STATUS_ERROR_VSX_PARAMETER_ZERO = -0x08054`
- `int VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO = -0x08055`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2_TAG = -0x08056`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2_ID_IN_DATA_CONTAINER = -0x08057`
- `int VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_ZERO = -0x08058`
- `int VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_ZERO = -0x08059`
- `int VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO = -0x0805A`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_ID_IN_DATA_CONTAINER = -0x0805B`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_TAG = -0x0805C`
- `int VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO = -0x0805D`
- `int VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO = -0x0805E`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_TAG = -0x0805F`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_ID_IN_DATA_CONTAINER = -0x08060`
- `int VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_POINTER_ZERO = -0x08061`
- `int VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_NOT_ZERO = -0x08062`
- `int VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_ZERO = -0x08063`
- `int VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO = -0x08064`
- `int VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO = -0x08065`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA_CONTAINER = -0x08066`
- `int VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_DATA_FORMAT = -0x08067`
- `int VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO = -0x08068`
- `int VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO = -0x08069`
- `int VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD = -0x0806A`
- `int VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME = -0x0806B`
- `int VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK_ZERO = -0x0806C`

10.11.1 Member Data Documentation

10.11.1.1 VSX_STATUS_SUCCESS `int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_SUCCESS = 0 [static]`

10.11.1.2 VSX_STATUS_ERROR_DRIVER_INIT `int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_INIT = -0x1 [static]`

10.11.1.3 VSX_STATUS_ERROR_DRIVER_TIMEOUT `int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_TIMEOUT = -0x2 [static]`

10.11.1.4 VSX_STATUS_ERROR_DRIVER_SAVE_FILE `int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_SAVE_FILE = -0x3 [static]`

10.11.1.5 VSX_STATUS_ERROR_DRIVER_DATA int VsxProtocolDriver.Definitions.StatusCode.↔
VSX_STATUS_ERROR_DRIVER_DATA = -0x4 [static]

10.11.1.6 VSX_STATUS_ERROR_DRIVER_CONNECTION int VsxProtocolDriver.Definitions.StatusCode.↔
Code.VSX_STATUS_ERROR_DRIVER_CONNECTION = -0x5 [static]

10.11.1.7 VSX_STATUS_ERROR_DRIVER_INVALID_DATA int VsxProtocolDriver.Definitions.StatusCode.↔
StatusCode.VSX_STATUS_ERROR_DRIVER_INVALID_DATA = -0x6 [static]

10.11.1.8 VSX_STATUS_ERROR_DRIVER_DEVICE int VsxProtocolDriver.Definitions.StatusCode.↔
VSX_STATUS_ERROR_DRIVER_DEVICE = -0x7 [static]

10.11.1.9 VSX_STATUS_ERROR_DRIVER_LOAD_FILE int VsxProtocolDriver.Definitions.StatusCode.↔
Code.VSX_STATUS_ERROR_DRIVER_LOAD_FILE = -0x8 [static]

10.11.1.10 VSX_STATUS_ERROR_SESSION int VsxProtocolDriver.Definitions.StatusCode.VSX_↔
STATUS_ERROR_SESSION = -0x9 [static]

10.11.1.11 VSX_STATUS_ERROR_STRING int VsxProtocolDriver.Definitions.StatusCode.VSX_↔
STATUS_ERROR_STRING = -0xA [static]

10.11.1.12 VSX_STATUS_ERROR_VERSION int VsxProtocolDriver.Definitions.StatusCode.VSX_↔
STATUS_ERROR_VERSION = -0xB [static]

10.11.1.13 VSX_STATUS_ERROR_DRIVER_GENERAL int VsxProtocolDriver.Definitions.StatusCode.↔
Code.VSX_STATUS_ERROR_DRIVER_GENERAL = -0x100 [static]

10.11.1.14 VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM int VsxProtocolDriver.Definitions.StatusCode.VSX_↔
Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM = -0x8001 [static]

10.11.1.15 VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO = -0x8002 [static]

10.11.1.16 VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO = -0x8003 [static]

10.11.1.17 VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE = -0x8004 [static]

10.11.1.18 VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION = -0x8005 [static]

10.11.1.19 VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION = -0x8006 [static]

10.11.1.20 VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO = -0x8007 [static]

10.11.1.21 VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO = -0x8008 [static]

10.11.1.22 VSX_STATUS_ERROR_PARAMETER_ID_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_PARAMETER_ID_ZERO = -0x8009 [static]

10.11.1.23 VSX_STATUS_ERROR_VALUE_ZERO int VsxProtocolDriver.Definitions.StatusCode.↔
VSX_STATUS_ERROR_VALUE_ZERO = -0x800A [static]

10.11.1.24 VSX_STATUS_ERROR_COMMAND_ZERO int VsxProtocolDriver.Definitions.StatusCode.↔
Code.VSX_STATUS_ERROR_COMMAND_ZERO = -0x800B [static]

10.11.1.25 VSX_STATUS_ERROR_INPUT_VALUE_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
Code.VSX_STATUS_ERROR_INPUT_VALUE_ZERO = -0x800C [static]

10.11.1.26 VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO = -0x800D [static]

10.11.1.27 VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO = -0x800E [static]

10.11.1.28 VSX_STATUS_ERROR_VALUE_POINTER_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_VALUE_POINTER_ZERO = -0x800F [static]

10.11.1.29 VSX_STATUS_ERROR_VALUE_NOT_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
Code.VSX_STATUS_ERROR_VALUE_NOT_ZERO = -0x8010 [static]

10.11.1.30 VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM = -0x8011 [static]

10.11.1.31 VSX_STATUS_ERROR_XML_COMMAND_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_XML_COMMAND_ZERO = -0x8012 [static]

10.11.1.32 VSX_STATUS_ERROR_FILENAME_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
Code.VSX_STATUS_ERROR_FILENAME_ZERO = -0x8013 [static]

10.11.1.33 VSX_STATUS_ERROR_STRING_POINTER_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
StatusCode.VSX_STATUS_ERROR_STRING_POINTER_ZERO = -0x8014 [static]

10.11.1.34 VSX_STATUS_ERROR_STRING_ZERO int VsxProtocolDriver.Definitions.StatusCode↔
VSX_STATUS_ERROR_STRING_ZERO = -0x8015 [static]

10.11.1.35 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_POINTER_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_POINTER_ZERO = -0x8016
[static]

10.11.1.36 VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER = -0x8017
[static]

10.11.1.37 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_ZERO = -0x8018
[static]

10.11.1.38 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_ZERO = -0x8019 [static]

10.11.1.39 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_AVAILABLE int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_AVAILABLE
= -0x801A [static]

10.11.1.40 VSX_STATUS_ERROR_IMAGE_TAG_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_IMAGE_TAG_ZERO = -0x801B [static]

10.11.1.41 VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER = -0x801C
[static]

10.11.1.42 VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_CONTAINER int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_↔
CONTAINER = -0x801D [static]

10.11.1.43 VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_FORMAT int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_FORMAT
= -0x801E [static]

10.11.1.44 VSX_STATUS_ERROR_POINT_Z_ID_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_POINT_Z_ID_ZERO = -0x801F [static]

10.11.1.45 VSX_STATUS_ERROR_POINT_Y_ID_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_POINT_Y_ID_ZERO = -0x8020 [static]

10.11.1.46 VSX_STATUS_ERROR_POINT_X_ID_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_POINT_X_ID_ZERO = -0x8021 [static]

10.11.1.47 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_DATA_CONTAINER int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_DATA_↔
CONTAINER = -0x8022 [static]

10.11.1.48 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_DATA_CONTAINER int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_DATA_↔
CONTAINER = -0x8023 [static]

10.11.1.49 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_DATA_CONTAINER int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_DATA_↔
CONTAINER = -0x8024 [static]

10.11.1.50 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_DATA_FORMAT int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_DATA_↔
FORMAT = -0x8025 [static]

10.11.1.51 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_DATA_FORMAT int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_DATA_↔
FORMAT = -0x8026 [static]

10.11.1.52 VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_DATA_FORMAT int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_DATA_↔
FORMAT = -0x8027 [static]

10.11.1.53 VSX_STATUS_ERROR_LOG_POINTER_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_LOG_POINTER_ZERO = -0x8028 [static]

10.11.1.54 VSX_STATUS_ERROR_LOG_NOT_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_LOG_NOT_ZERO = -0x8029 [static]

10.11.1.55 VSX_STATUS_ERROR_RESULT_NOT_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_RESULT_NOT_ZERO = -0x802A [static]

10.11.1.56 VSX_STATUS_ERROR_RESULT_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_RESULT_POINTER_ZERO = -0x802B [static]

10.11.1.57 VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_DATA_CONTAINER int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_DATA_↔
CONTAINER = -0x802C [static]

10.11.1.58 VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_DATA_FORMAT int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_DATA_FORMAT = -0x802D
[static]

10.11.1.59 VSX_STATUS_ERROR_VERSION_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VERSION_POINTER_ZERO = -0x802E [static]

10.11.1.60 VSX_STATUS_ERROR_VERSION_NOT_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_VERSION_NOT_ZERO = -0x802F [static]

10.11.1.61 VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO = -0x8030 [static]

10.11.1.62 VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO = -0x8031 [static]

10.11.1.63 VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE = -0x8032 [static]

10.11.1.64 VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VALUE int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VALUE = -0x8033 [static]

10.11.1.65 VSX_STATUS_ERROR_XPATH_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_XPATH_ZERO = -0x8034 [static]

10.11.1.66 VSX_STATUS_ERROR_INVALID_DATA_FORMAT int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_INVALID_DATA_FORMAT = -0x8035 [static]

10.11.1.67 VSX_STATUS_ERROR_NO_ELEMENT_FOUND int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_NO_ELEMENT_FOUND = -0x8036 [static]

10.11.1.68 VSX_STATUS_ERROR_RESULT_TAG_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_RESULT_TAG_ZERO = -0x8037 [static]

10.11.1.69 VSX_STATUS_ERROR_TAG_ZERO int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_TAG_ZERO = -0x8038 [static]

10.11.1.70 VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER = -0x8039
[static]

10.11.1.71 VSX_STATUS_ERROR_IP_ADDRESS_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_IP_ADDRESS_ZERO = -0x803A [static]

10.11.1.72 VSX_STATUS_ERROR_NETWORK_MASK_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_NETWORK_MASK_ZERO = -0x803B [static]

10.11.1.73 VSX_STATUS_ERROR_GATEWAY_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_GATEWAY_ZERO = -0x803C [static]

10.11.1.74 VSX_STATUS_ERROR_EXCEPTION_THROWN int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_EXCEPTION_THROWN = -0x803D [static]

10.11.1.75 VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO = -0x803E [static]

10.11.1.76 VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO = -0x803F [static]

10.11.1.77 VSX_STATUS_ERROR_VSX_IMAGE_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_VSX_IMAGE_ZERO = -0x8040 [static]

10.11.1.78 VSX_STATUS_ERROR_VSX_DEVICE_ZERO int VsxProtocolDriver.Definitions.Status↔
Code.VSX_STATUS_ERROR_VSX_DEVICE_ZERO = -0x8041 [static]

10.11.1.79 VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO = -0x8042 [static]

10.11.1.80 VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO = -0x8043 [static]

10.11.1.81 VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO = -0x8044 [static]

10.11.1.82 VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO = -0x8045 [static]

10.11.1.83 VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO = -0x8046 [static]

10.11.1.84 VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO = -0x8047 [static]

10.11.1.85 VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO = -0x8048 [static]

10.11.1.86 VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO = -0x8049 [static]

10.11.1.87 VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO = -0x804A [static]

10.11.1.88 VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO = -0x804B [static]

10.11.1.89 VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO = -0x0804C [static]

10.11.1.90 VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO = -0x0804D [static]

10.11.1.91 VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO = -0x0804E [static]

10.11.1.92 VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO = -0x0804F [static]

10.11.1.93 VSX_STATUS_ERROR_MAC_ADDRESS_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_MAC_ADDRESS_ZERO = -0x08050 [static]

10.11.1.94 VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND = -0x08051 [static]

10.11.1.95 VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO = -0x08052 [static]

10.11.1.96 VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO = -0x08053 [static]

10.11.1.97 VSX_STATUS_ERROR_VSX_PARAMETER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_PARAMETER_ZERO = -0x08054 [static]

10.11.1.98 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO = -0x08055
[static]

10.11.1.99 VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2_TAG int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2↔
_TAG = -0x08056 [static]

10.11.1.100 VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2_ID_IN_DATA_CONTAINER
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_↔
DESCRIPTOR2_ID_IN_DATA_CONTAINER = -0x08057 [static]

10.11.1.101 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_ZERO = -0x08058
[static]

10.11.1.102 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_ZERO = -0x08059 [static]

10.11.1.103 VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO = -0x0805A
[static]

10.11.1.104 VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_ID_IN_DATA_CONTAINER
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_↔
ID_IN_DATA_CONTAINER = -0x0805B [static]

10.11.1.105 VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_TAG int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_TAG = -0x0805C
[static]

10.11.1.106 VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO = -0x0805D [static]

10.11.1.107 VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO = -0x0805E [static]

10.11.1.108 VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_TAG int Vsx↔
ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_TAG
= -0x0805F [static]

10.11.1.109 VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_ID_IN_DATA_CONTAINER
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION↔
_ID_IN_DATA_CONTAINER = -0x08060 [static]

10.11.1.110 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_POINTER_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_POINTER_ZERO = -0x08061
[static]

10.11.1.111 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_NOT_ZERO int VsxProtocol↔
Driver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_NOT_ZERO = -0x08062
[static]

10.11.1.112 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_ZERO int VsxProtocolDriver.↔
Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_ZERO = -0x08063 [static]

10.11.1.113 VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO = -0x08064 [static]

10.11.1.114 VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO int VsxProtocolDriver.Definitions.↔
StatusCode.VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO = -0x08065 [static]

10.11.1.115 VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA_CONTAINER int Vsx↔

ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA_CONTAINER
= -0x08066 [static]

10.11.1.116 VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_DATA_FORMAT int VsxProtocol↔

Driver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_DATA_FORMAT = -0x08067
[static]

10.11.1.117 VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO int VsxProtocolDriver.Definitions.↔

StatusCode.VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO = -0x08068 [static]

10.11.1.118 VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO int VsxProtocolDriver.Definitions.↔

StatusCode.VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO = -0x08069 [static]

10.11.1.119 VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD int VsxProtocolDriver.Definitions.↔

StatusCode.VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD = -0x0806A [static]

10.11.1.120 VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME int VsxProtocolDriver.Definitions.↔

StatusCode.VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME = -0x0806B [static]

10.11.1.121 VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK_ZERO int Vsx↔

ProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK_↔
ZERO = -0x0806C [static]

10.12 VsxProtocolDriver.Definitions.StatusItem Class Reference

Public Member Functions

- def `__init__` (self, int settings_version, str configuration_class, int config_version, str status_item_id, str [name](#), object [value](#), [ValueType](#) value_type, int [time](#), int sensor_time)

Public Attributes

- [settingsVersion](#)
- [configurationClass](#)
- [configVersion](#)
- [statusItemId](#)
- [name](#)
- [value](#)
- [valueType](#)
- [time](#)
- [sensorTime](#)

10.12.1 Constructor & Destructor Documentation

10.12.1.1 `__init__()` `def VsxProtocolDriver.Definitions.StatusItem.__init__ (`
 `self,`
 `int settings_version,`
 `str configuration_class,`
 `int config_version,`
 `str status_item_id,`
 `str name,`
 `object value,`
 `ValueType value_type,`
 `int time,`
 `int sensor_time)`

10.12.2 Member Data Documentation

10.12.2.1 `settingsVersion` `VsxProtocolDriver.Definitions.StatusItem.settingsVersion`

10.12.2.2 `configurationClass` `VsxProtocolDriver.Definitions.StatusItem.configurationClass`

10.12.2.3 `configVersion` `VsxProtocolDriver.Definitions.StatusItem.configVersion`

10.12.2.4 `statusItemId` `VsxProtocolDriver.Definitions.StatusItem.statusItemId`

10.12.2.5 name `VsxProtocolDriver.Definitions.StatusItem.name`

10.12.2.6 value `VsxProtocolDriver.Definitions.StatusItem.value`

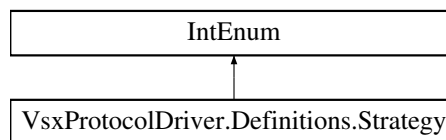
10.12.2.7 valueType `VsxProtocolDriver.Definitions.StatusItem.valueType`

10.12.2.8 time `VsxProtocolDriver.Definitions.StatusItem.time`

10.12.2.9 sensorTime `VsxProtocolDriver.Definitions.StatusItem.sensorTime`

10.13 VsxProtocolDriver.Definitions.Strategy Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.Strategy:



Static Public Attributes

- int `DROP_OLDEST` = 0
- int `DROP_WRITE` = 1

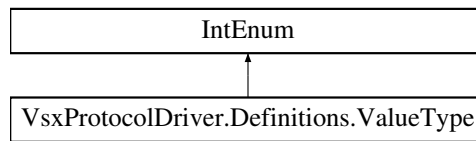
10.13.1 Member Data Documentation

10.13.1.1 DROP_OLDEST `int VsxProtocolDriver.Definitions.Strategy.DROP_OLDEST = 0 [static]`

10.13.1.2 DROP_WRITE `int VsxProtocolDriver.Definitions.Strategy.DROP_WRITE = 1 [static]`

10.14 VsxProtocolDriver.Definitions.ValueType Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.ValueType:



Static Public Attributes

- int `BOOL` = 0
- int `INT` = 1
- int `LONG` = 2
- int `UINT` = 3
- int `INT16` = 4
- int `FLOAT` = 5
- int `DOUBLE` = 6
- int `STRING` = 7
- int `HEXSTRING` = 8
- int `BASE64` = 9
- int `ENUM` = 10
- int `IP` = 11
- int `RECTANGLE` = 12
- int `QUAD` = 13
- int `POINT` = 14
- int `UNKNOWN` = 15

10.14.1 Member Data Documentation

10.14.1.1 `BOOL` `int VsxProtocolDriver.Definitions.ValueType.BOOL = 0` `[static]`

10.14.1.2 `INT` `int VsxProtocolDriver.Definitions.ValueType.INT = 1` `[static]`

10.14.1.3 `LONG` `int VsxProtocolDriver.Definitions.ValueType.LONG = 2` `[static]`

10.14.1.4 `UINT` `int VsxProtocolDriver.Definitions.ValueType.UINT = 3` `[static]`

10.14.1.5 INT16 `int VsxProtocolDriver.Definitions.ValueType.INT16 = 4 [static]`

10.14.1.6 FLOAT `int VsxProtocolDriver.Definitions.ValueType.FLOAT = 5 [static]`

10.14.1.7 DOUBLE `int VsxProtocolDriver.Definitions.ValueType.DOUBLE = 6 [static]`

10.14.1.8 STRING `int VsxProtocolDriver.Definitions.ValueType.STRING = 7 [static]`

10.14.1.9 HEXSTRING `int VsxProtocolDriver.Definitions.ValueType.HEXSTRING = 8 [static]`

10.14.1.10 BASE64 `int VsxProtocolDriver.Definitions.ValueType.BASE64 = 9 [static]`

10.14.1.11 ENUM `int VsxProtocolDriver.Definitions.ValueType.ENUM = 10 [static]`

10.14.1.12 IP `int VsxProtocolDriver.Definitions.ValueType.IP = 11 [static]`

10.14.1.13 RECTANGLE `int VsxProtocolDriver.Definitions.ValueType.RECTANGLE = 12 [static]`

10.14.1.14 QUAD `int VsxProtocolDriver.Definitions.ValueType.QUAD = 13 [static]`

10.14.1.15 POINT `int VsxProtocolDriver.Definitions.ValueType.POINT = 14 [static]`

10.14.1.16 UNKNOWN `int VsxProtocolDriver.Definitions.ValueType.UNKNOWN = 15 [static]`

10.15 VsxProtocolDriver.DataContainer.VsxCaptureInformation Class Reference

CaptureInformation contains information identifying and describing the captured image.

Public Member Functions

- `def __init__ (self)`

10.15.1 Detailed Description

CaptureInformation contains information identifying and describing the captured image.

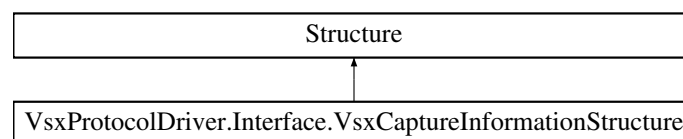
The timestamp reflects a monotonic clock, without further synchronization timestamps are only meaningful relative to one another. The frame counter counts the number of actually captured images, while the trigger counter also includes missed trigger events (if they could be detected at all). The JobId 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).

10.15.2 Constructor & Destructor Documentation

10.15.2.1 `__init__()` `def VsxProtocolDriver.DataContainer.VsxCaptureInformation.__init__ (self)`

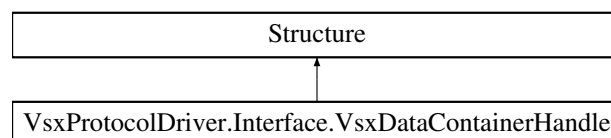
10.16 VsxProtocolDriver.Interface.VsxCaptureInformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxCaptureInformationStructure:



10.17 VsxProtocolDriver.Interface.VsxDataContainerHandle Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDataContainerHandle:



Public Member Functions

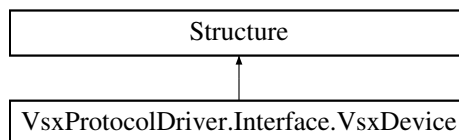
- def `__repr__` (self)

10.17.1 Member Function Documentation

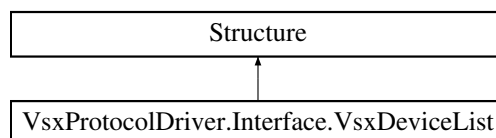
10.17.1.1 `__repr__()` `def VsxProtocolDriver.Interface.VsxDataContainerHandle.__repr__ (self)`

10.18 VsxProtocolDriver.Interface.VsxDevice Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDevice:

**10.19 VsxProtocolDriver.Interface.VsxDeviceList Class Reference**

Inheritance diagram for VsxProtocolDriver.Interface.VsxDeviceList:

**10.20 VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2 Class Reference****Public Member Functions**

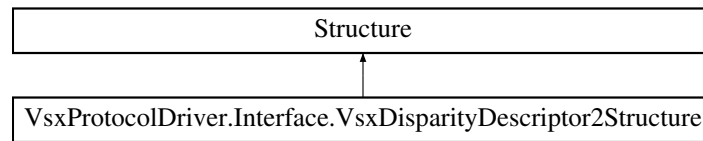
- def `__init__` (self)

10.20.1 Constructor & Destructor Documentation

10.20.1.1 `__init__()` `def VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2.__init__ (self)`

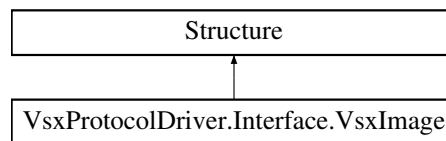
10.21 VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure:



10.22 VsxProtocolDriver.Interface.VsxImage Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxImage:



10.23 VsxProtocolDriver.DataContainer.VsxLineCoordinate Class Reference

Public Member Functions

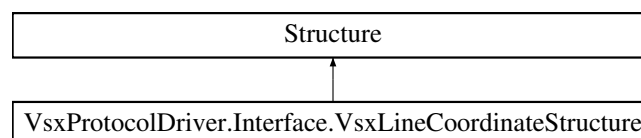
- `def __init__ (self, float c, float x, float y, float z, float q, float i)`

10.23.1 Constructor & Destructor Documentation

10.23.1.1 `__init__()` `def VsxProtocolDriver.DataContainer.VsxLineCoordinate.__init__ (`
`self,`
`float c,`
`float x,`
`float y,`
`float z,`
`float q,`
`float i)`

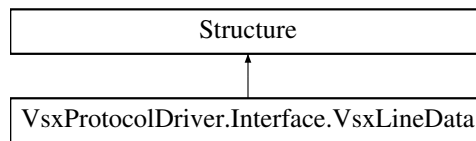
10.24 VsxProtocolDriver.Interface.VsxLineCoordinateStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxLineCoordinateStructure:



10.25 VsxProtocolDriver.Interface.VsxLineData Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxLineData:



10.26 VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation Class Reference

Public Member Functions

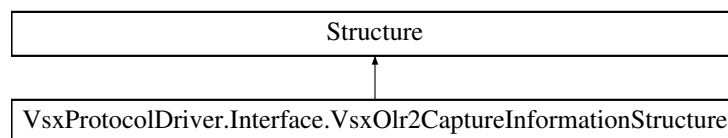
- def `__init__` (self)

10.26.1 Constructor & Destructor Documentation

10.26.1.1 `__init__()` def VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation.__init__ (
 self)

10.27 VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure:



10.28 VsxProtocolDriver.DataContainer.VsxOlr2ModbusData Class Reference

Public Member Functions

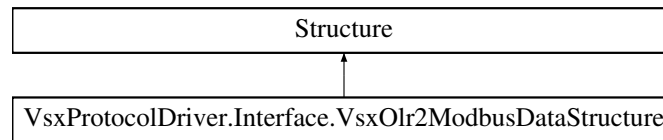
- def `__init__` (self)

10.28.1 Constructor & Destructor Documentation


```
10.28.1.1 __init__() def VsxProtocolDriver.DataContainer.VsxOlr2ModbusData.__init__ (
                self )
```

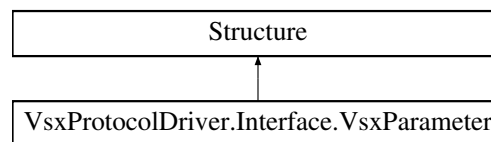
10.29 VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure:



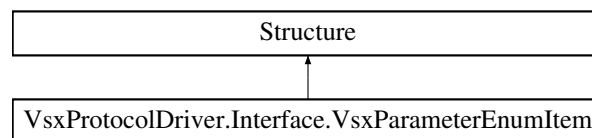
10.30 VsxProtocolDriver.Interface.VsxParameter Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameter:



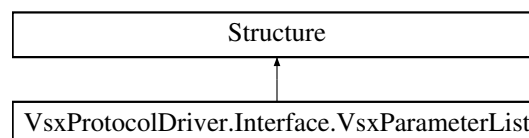
10.31 VsxProtocolDriver.Interface.VsxParameterEnumItem Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameterEnumItem:



10.32 VsxProtocolDriver.Interface.VsxParameterList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameterList:



Public Member Functions

- def `__init__` (self, `length`)

Public Attributes

- [length](#)
- [parameters](#)

10.32.1 Constructor & Destructor Documentation

10.32.1.1 `__init__()` `def VsxProtocolDriver.Interface.VsxParameterList.__init__ (self, length)`

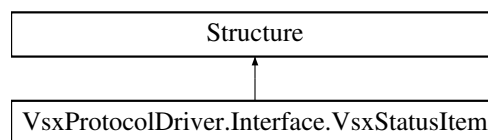
10.32.2 Member Data Documentation

10.32.2.1 `length` `VsxProtocolDriver.Interface.VsxParameterList.length`

10.32.2.2 `parameters` `VsxProtocolDriver.Interface.VsxParameterList.parameters`

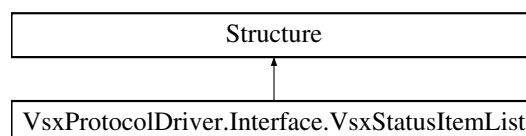
10.33 VsxProtocolDriver.Interface.VsxStatusItem Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxStatusItem:



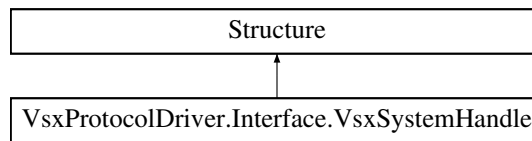
10.34 VsxProtocolDriver.Interface.VsxStatusItemList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxStatusItemList:



10.35 VsxProtocolDriver.Interface.VsxSystemHandle Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxSystemHandle:



Public Member Functions

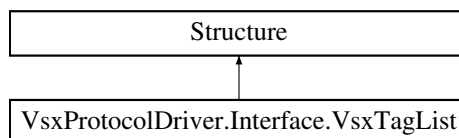
- `def __repr__(self)`

10.35.1 Member Function Documentation

10.35.1.1 `__repr__()` `def VsxProtocolDriver.Interface.VsxSystemHandle.__repr__ (self)`

10.36 VsxProtocolDriver.Interface.VsxTagList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxTagList:



10.37 VsxProtocolDriver.DataContainer.VsxTransformation Class Reference

Used to transform raw point cloud data from device.

Public Member Functions

- `def __init__(self)`

10.37.1 Detailed Description

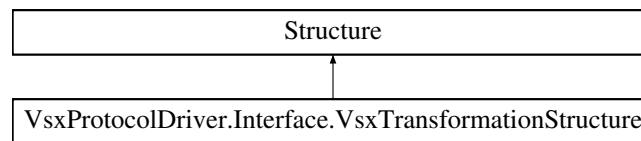
Used to transform raw point cloud data from device.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 `__init__()` `def VsxProtocolDriver.DataContainer.VsxTransformation.__init__ (`
`self)`

10.38 VsxProtocolDriver.Interface.VsxTransformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxTransformationStructure:



Index

- `del`
 - `VsxProtocolDriver.DataContainer.DataContainer`, 12
 - `VsxProtocolDriver.Sensor.Sensor`, 38
 - `init`
 - `VsxProtocolDriver.DataContainer.DataContainer`, 12
 - `VsxProtocolDriver.DataContainer.VsxCaptureInformation`, 82
 - `VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2`, 83
 - `VsxProtocolDriver.DataContainer.VsxLineCoordinate`, 84
 - `VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation`, 85
 - `VsxProtocolDriver.DataContainer.VsxOlr2ModbusData`, 85
 - `VsxProtocolDriver.DataContainer.VsxTransformation`, 89
 - `VsxProtocolDriver.Definitions.Parameter`, 34
 - `VsxProtocolDriver.Definitions.StatusItem`, 78
 - `VsxProtocolDriver.Interface.VsxParameterList`, 87
 - `VsxProtocolDriver.Sensor.Sensor`, 38
 - `repr`
 - `VsxProtocolDriver.Interface.VsxDataContainerHandle`, 83
 - `VsxProtocolDriver.Interface.VsxSystemHandle`, 88
- BASE64
 - `VsxProtocolDriver.Definitions.ValueType`, 81
- BOOL
 - `VsxProtocolDriver.Definitions.ValueType`, 80
- `callback_on_device_status_received`
 - `VsxProtocolDriver.Interface.Interface`, 23
- `callback_on_disconnect`
 - `VsxProtocolDriver.Interface.Interface`, 23
- `callback_on_session_message_received`
 - `VsxProtocolDriver.Interface.Interface`, 23
- CANOPEN
 - `VsxProtocolDriver.Definitions.SerialConnectionType`, 60
- `configId`
 - `VsxProtocolDriver.Definitions.Parameter`, 34
- CONFIGURATION_ONLY
 - `VsxProtocolDriver.Definitions.DeviceAuthenticationMode`, 18
- `configurationClass`
 - `VsxProtocolDriver.Definitions.StatusItem`, 78
- `configVersion`
 - `VsxProtocolDriver.Definitions.Parameter`, 34
 - `VsxProtocolDriver.Definitions.StatusItem`, 78
- Connect
 - `VsxProtocolDriver.Sensor.Sensor`, 43
- `connect`
 - `VsxProtocolDriver.Interface.Interface`, 24
 - `connect_and_login`
 - `VsxProtocolDriver.Interface.Interface`, 25
 - `connect_ex`
 - `VsxProtocolDriver.Interface.Interface`, 25
 - `connect_ex_and_login`
 - `VsxProtocolDriver.Interface.Interface`, 25
 - `ConnectAndLogin`
 - `VsxProtocolDriver.Sensor.Sensor`, 43
 - `Connected`
 - `VsxProtocolDriver.Sensor.Sensor`, 43
 - `ConnectEx`
 - `VsxProtocolDriver.Sensor.Sensor`, 43
 - `ConnectExAndLogin`
 - `VsxProtocolDriver.Sensor.Sensor`, 44
 - `CONNECTION_ERROR`
 - `VsxProtocolDriver.Definitions.DisconnectEvent`, 19
 - `data_container_handle`
 - `VsxProtocolDriver.DataContainer.DataContainer`, 17
 - `deregister_on_device_status_received`
 - `VsxProtocolDriver.Interface.Interface`, 33
 - `deregister_on_disconnect`
 - `VsxProtocolDriver.Interface.Interface`, 25
 - `deregister_on_session_message_received`
 - `VsxProtocolDriver.Interface.Interface`, 26
 - `DeregisterOnDeviceStatusReceived`
 - `VsxProtocolDriver.Sensor.Sensor`, 58
 - `DeregisterOnDisconnect`
 - `VsxProtocolDriver.Sensor.Sensor`, 46
 - `DeregisterOnSessionMessageReceived`
 - `VsxProtocolDriver.Sensor.Sensor`, 47
 - `Disconnect`
 - `VsxProtocolDriver.Sensor.Sensor`, 45
 - `disconnect`
 - `VsxProtocolDriver.Interface.Interface`, 25
 - `DISCONNECT_CALLED`
 - `VsxProtocolDriver.Definitions.DisconnectEvent`, 19
 - DOUBLE
 - `VsxProtocolDriver.Definitions.ValueType`, 81
 - `download_parameter_set`
 - `VsxProtocolDriver.Interface.Interface`, 31
 - `DownloadParameterSet`
 - `VsxProtocolDriver.Sensor.Sensor`, 41
 - `DROP_OLDEST`
 - `VsxProtocolDriver.Definitions.Strategy`, 79
 - `DROP_WRITE`
 - `VsxProtocolDriver.Definitions.Strategy`, 79
 - `DynamicContainerQueueSize`
 - `VsxProtocolDriver.Sensor.Sensor`, 56
 - ENUM
 - `VsxProtocolDriver.Definitions.ValueType`, 81
 - `enumItems`
 - `VsxProtocolDriver.Definitions.Parameter`, 34
 - `ETHERNET_IP`

- VsxProtocolDriver.Definitions.SerialConnectionType, [get_result_xml](#)
[60](#)
- FLOAT
 - VsxProtocolDriver.Definitions.ValueType, [81](#)
- FULL
 - VsxProtocolDriver.Definitions.DeviceAuthenticationMode, [18](#)
 - VsxProtocolDriver.Definitions.DeviceStatusScope, [18](#)
- [get_all_device_status_data](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [get_cached_container](#)
 - VsxProtocolDriver.Interface.Interface, [27](#)
- [get_capture_information](#)
 - VsxProtocolDriver.Interface.Interface, [27](#)
- [get_connected](#)
 - VsxProtocolDriver.Interface.Interface, [25](#)
- [get_data_container](#)
 - VsxProtocolDriver.Interface.Interface, [27](#)
- [get_device_information](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_disparity_descriptor2](#)
 - VsxProtocolDriver.Interface.Interface, [28](#)
- [get_dynamic_container_queue_size](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_error_text](#)
 - VsxProtocolDriver.Interface.Interface, [24](#)
- [get_image](#)
 - VsxProtocolDriver.Interface.Interface, [28](#)
- [get_library_version](#)
 - VsxProtocolDriver.Interface.Interface, [24](#)
- [get_line](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_log_message](#)
 - VsxProtocolDriver.Interface.Interface, [30](#)
- [get_log_message_queue_size](#)
 - VsxProtocolDriver.Interface.Interface, [30](#)
- [get_missing_container_frames_counter](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_missing_log_messages_counter](#)
 - VsxProtocolDriver.Interface.Interface, [30](#)
- [get_number_of_cached_containers](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_olr2_capture_information](#)
 - VsxProtocolDriver.Interface.Interface, [28](#)
- [get_olr2_modbus_data](#)
 - VsxProtocolDriver.Interface.Interface, [28](#)
- [get_parameter_list](#)
 - VsxProtocolDriver.Interface.Interface, [31](#)
- [get_result_element_double](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [get_result_element_int32](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [get_result_element_int64](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [get_result_element_string](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [VsxProtocolDriver.Interface.Interface](#), [32](#)
- [get_single_parameter](#)
 - VsxProtocolDriver.Interface.Interface, [32](#)
- [get_single_parameter_value](#)
 - VsxProtocolDriver.Interface.Interface, [30](#)
- [get_single_parameter_value_double](#)
 - VsxProtocolDriver.Interface.Interface, [31](#)
- [get_single_parameter_value_int32](#)
 - VsxProtocolDriver.Interface.Interface, [30](#)
- [get_tag_list](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_transformation](#)
 - VsxProtocolDriver.Interface.Interface, [28](#)
- [get_udp_device_list](#)
 - VsxProtocolDriver.Interface.Interface, [29](#)
- [get_wait_timeout](#)
 - VsxProtocolDriver.Interface.Interface, [26](#)
- [GetAllDeviceStatusData](#)
 - VsxProtocolDriver.Sensor.Sensor, [57](#)
- [GetCachedContainer](#)
 - VsxProtocolDriver.Sensor.Sensor, [56](#)
- [GetCaptureInformation](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [13](#)
- [GetCurrentDeviceInformation](#)
 - VsxProtocolDriver.Sensor.Sensor, [57](#)
- [GetDataContainer](#)
 - VsxProtocolDriver.Sensor.Sensor, [56](#)
- [GetDeviceInformation](#)
 - VsxProtocolDriver.Sensor.Sensor, [57](#)
- [GetDisparityDescriptor2](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [13](#)
- [GetErrorText](#)
 - VsxProtocolDriver.Sensor.Sensor, [38](#)
- [GetImage](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [14](#)
- [GetLibraryVersion](#)
 - VsxProtocolDriver.Sensor.Sensor, [38](#)
- [GetLine](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [15](#)
- [GetLogMessage](#)
 - VsxProtocolDriver.Sensor.Sensor, [54](#)
- [GetOlr2CaptureInformation](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [14](#)
- [GetOlr2ModbusData](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [14](#)
- [GetParameterList](#)
 - VsxProtocolDriver.Sensor.Sensor, [53](#)
- [GetResultElementDouble](#)
 - VsxProtocolDriver.DataContainer.DataContainer, [17](#)
- [GetResultElementInt32](#)

- VsxProtocolDriver.DataContainer.DataContainer, 16
- GetResultElementInt64
 - VsxProtocolDriver.DataContainer.DataContainer, 16
- GetResultElementString
 - VsxProtocolDriver.DataContainer.DataContainer, 15
- GetResultXml
 - VsxProtocolDriver.DataContainer.DataContainer, 15
- GetSingleDataValue
 - VsxProtocolDriver.Sensor.Sensor, 51
- GetSingleParameter
 - VsxProtocolDriver.Sensor.Sensor, 53
- GetSingleParameterValue
 - VsxProtocolDriver.Sensor.Sensor, 51
- GetSingleParameterValueDouble
 - VsxProtocolDriver.Sensor.Sensor, 52
- GetSingleParameterValueInt32
 - VsxProtocolDriver.Sensor.Sensor, 52
- GetTagList
 - VsxProtocolDriver.DataContainer.DataContainer, 13
- GetTransformation
 - VsxProtocolDriver.DataContainer.DataContainer, 13
- GetUdpDeviceList
 - VsxProtocolDriver.Sensor.Sensor, 57
- GetWaitTimeout
 - VsxProtocolDriver.Sensor.Sensor, 49
- Handle
 - VsxProtocolDriver.Sensor.Sensor, 39
- HEXSTRING
 - VsxProtocolDriver.Definitions.ValueType, 81
- init_driver
 - VsxProtocolDriver.Interface.Interface, 23
- init_serial_sensor
 - VsxProtocolDriver.Interface.Interface, 24
- init_tcp_sensor
 - VsxProtocolDriver.Interface.Interface, 24
- INITIAL_PASSWORD_REQUIRED
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- initialize_device
 - VsxProtocolDriver.Interface.Interface, 25
- InitializeDevice
 - VsxProtocolDriver.Sensor.Sensor, 44
- InitSerialSensor
 - VsxProtocolDriver.Sensor.Sensor, 39
- InitTcpSensor
 - VsxProtocolDriver.Sensor.Sensor, 39
- INT
 - VsxProtocolDriver.Definitions.ValueType, 80
- INT16
 - VsxProtocolDriver.Definitions.ValueType, 80
- IP
 - VsxProtocolDriver.Definitions.ValueType, 81
- length
 - VsxProtocolDriver.Interface.VsxParameterList, 87
- load_default_parameter_set_on_device
 - VsxProtocolDriver.Interface.Interface, 31
- load_parameter_set_on_device
 - VsxProtocolDriver.Interface.Interface, 31
- LoadDefaultParameterSetOnDevice
 - VsxProtocolDriver.Sensor.Sensor, 42
- LoadParameterSetOnDevice
 - VsxProtocolDriver.Sensor.Sensor, 42
- LOGIN
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- Login
 - VsxProtocolDriver.Sensor.Sensor, 44
- login
 - VsxProtocolDriver.Interface.Interface, 25
- LOGIN_REPLY
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- LOGIN_REQUIRED
 - VsxProtocolDriver.Definitions.SessionTypes, 60
- LogMessageQueueSize
 - VsxProtocolDriver.Sensor.Sensor, 55
- LOGOUT
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- Logout
 - VsxProtocolDriver.Sensor.Sensor, 44
- logout
 - VsxProtocolDriver.Interface.Interface, 25
- LOGOUT_REPLY
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- LONG
 - VsxProtocolDriver.Definitions.ValueType, 80
- MissingContainerFramesCounter
 - VsxProtocolDriver.Sensor.Sensor, 56
- MissingLogMessagesCounter
 - VsxProtocolDriver.Sensor.Sensor, 55
- MULTI
 - VsxProtocolDriver.Definitions.DeviceStatusScope, 18
- name
 - VsxProtocolDriver.Definitions.Parameter, 34
 - VsxProtocolDriver.Definitions.StatusItem, 78
- NONE
 - VsxProtocolDriver.Definitions.DeviceAuthenticationMode, 18
- NumberOfCachedContainers
 - VsxProtocolDriver.Sensor.Sensor, 57
- parameterId
 - VsxProtocolDriver.Definitions.Parameter, 34
- parameters
 - VsxProtocolDriver.Interface.VsxParameterList, 87
- POINT
 - VsxProtocolDriver.Definitions.ValueType, 81
- PROFIBUS
 - VsxProtocolDriver.Definitions.SerialConnectionType, 59

- PROFINET
 - VsxProtocolDriver.Definitions.SerialConnectionType, 60
- QUAD
 - VsxProtocolDriver.Definitions.ValueType, 81
- reconnect_and_login_tcp_device
 - VsxProtocolDriver.Interface.Interface, 24
- reconnect_serial_device
 - VsxProtocolDriver.Interface.Interface, 24
- reconnect_tcp_device
 - VsxProtocolDriver.Interface.Interface, 24
- ReConnectAndLoginTcpDevice
 - VsxProtocolDriver.Sensor.Sensor, 48
- ReConnectSerialDevice
 - VsxProtocolDriver.Sensor.Sensor, 48
- ReConnectSerialSensor
 - VsxProtocolDriver.Sensor.Sensor, 48
- ReConnectTcpDevice
 - VsxProtocolDriver.Sensor.Sensor, 47
- ReConnectTcpSensor
 - VsxProtocolDriver.Sensor.Sensor, 47
- RECTANGLE
 - VsxProtocolDriver.Definitions.ValueType, 81
- register_on_device_status_received
 - VsxProtocolDriver.Interface.Interface, 33
- register_on_disconnect
 - VsxProtocolDriver.Interface.Interface, 25
- register_on_session_message_received
 - VsxProtocolDriver.Interface.Interface, 26
- RegisterOnDeviceStatusReceived
 - VsxProtocolDriver.Sensor.Sensor, 58
- RegisterOnDisconnect
 - VsxProtocolDriver.Sensor.Sensor, 46
- RegisterOnSessionMessageReceived
 - VsxProtocolDriver.Sensor.Sensor, 46
- release_capture_information
 - VsxProtocolDriver.Interface.Interface, 27
- release_data_container
 - VsxProtocolDriver.Interface.Interface, 27
- release_device
 - VsxProtocolDriver.Interface.Interface, 29
- release_device_list
 - VsxProtocolDriver.Interface.Interface, 30
- release_disparity_descriptor2
 - VsxProtocolDriver.Interface.Interface, 28
- release_image
 - VsxProtocolDriver.Interface.Interface, 28
- release_line
 - VsxProtocolDriver.Interface.Interface, 29
- release_olr2_capture_information
 - VsxProtocolDriver.Interface.Interface, 28
- release_olr2_modbus_data
 - VsxProtocolDriver.Interface.Interface, 28
- release_parameter
 - VsxProtocolDriver.Interface.Interface, 32
- release_parameter_list
 - VsxProtocolDriver.Interface.Interface, 32
- release_sensor
 - VsxProtocolDriver.Interface.Interface, 24
- release_status_item_list
 - VsxProtocolDriver.Interface.Interface, 33
- release_string
 - VsxProtocolDriver.Interface.Interface, 24
- release_tag_list
 - VsxProtocolDriver.Interface.Interface, 29
- release_transformation
 - VsxProtocolDriver.Interface.Interface, 28
- ReleaseSensor
 - VsxProtocolDriver.Sensor.Sensor, 49
- REMOTE_HOST_CONNECTION_CLOSED
 - VsxProtocolDriver.Definitions.DisconnectEvent, 19
- reset_dynamic_container_grabber
 - VsxProtocolDriver.Interface.Interface, 27
- reset_log_message_grabber
 - VsxProtocolDriver.Interface.Interface, 30
- ResetDynamicContainerGrabber
 - VsxProtocolDriver.Sensor.Sensor, 55
- ResetDynamicContainerGrabberEx
 - VsxProtocolDriver.Sensor.Sensor, 55
- ResetLogMessageGrabber
 - VsxProtocolDriver.Sensor.Sensor, 54
- RS485
 - VsxProtocolDriver.Definitions.SerialConnectionType, 60
- Save3DPointCloudData
 - VsxProtocolDriver.DataContainer.DataContainer, 12
- save_3d_point_cloud_data
 - VsxProtocolDriver.Interface.Interface, 27
- save_data
 - VsxProtocolDriver.Interface.Interface, 27
- save_parameter_set_on_device
 - VsxProtocolDriver.Interface.Interface, 31
- SaveData
 - VsxProtocolDriver.DataContainer.DataContainer, 12
- SaveParameterSetOnDevice
 - VsxProtocolDriver.Sensor.Sensor, 42
- send_firmware
 - VsxProtocolDriver.Interface.Interface, 26
- send_session_keep_alive
 - VsxProtocolDriver.Interface.Interface, 26
- send_xml_data_message
 - VsxProtocolDriver.Interface.Interface, 26
- SendFirmware
 - VsxProtocolDriver.Sensor.Sensor, 40
- SendSessionKeepAlive
 - VsxProtocolDriver.Sensor.Sensor, 47
- SendXmlDataMessage
 - VsxProtocolDriver.Sensor.Sensor, 42
- sensorTime
 - VsxProtocolDriver.Definitions.StatusItem, 79
- set_network_settings
 - VsxProtocolDriver.Interface.Interface, 27
- set_network_settings_via_udp

- VsxProtocolDriver.Interface.Interface, 27
- SET_PASSWORD
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- set_password
 - VsxProtocolDriver.Interface.Interface, 25
- SET_PASSWORD_REPLY
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- set_single_parameter_double
 - VsxProtocolDriver.Interface.Interface, 31
- set_single_parameter_int32
 - VsxProtocolDriver.Interface.Interface, 31
- set_single_parameter_string
 - VsxProtocolDriver.Interface.Interface, 32
- set_single_parameter_value
 - VsxProtocolDriver.Interface.Interface, 30
- set_single_parameter_value_double
 - VsxProtocolDriver.Interface.Interface, 30
- set_single_parameter_value_int32
 - VsxProtocolDriver.Interface.Interface, 30
- set_wait_timeout
 - VsxProtocolDriver.Interface.Interface, 26
- SetNetworkParameter
 - VsxProtocolDriver.Sensor.Sensor, 49
- SetNetworkSettings
 - VsxProtocolDriver.Sensor.Sensor, 50
- SetNetworkSettingsViaUdp
 - VsxProtocolDriver.Sensor.Sensor, 50
- SetPassword
 - VsxProtocolDriver.Sensor.Sensor, 45
- SetSingleDataValue
 - VsxProtocolDriver.Sensor.Sensor, 50
- SetSingleParameter
 - VsxProtocolDriver.Sensor.Sensor, 53
- SetSingleParameterValue
 - VsxProtocolDriver.Sensor.Sensor, 51
- settingsVersion
 - VsxProtocolDriver.Definitions.Parameter, 34
 - VsxProtocolDriver.Definitions.StatusItem, 78
- SetWaitTimeout
 - VsxProtocolDriver.Sensor.Sensor, 49
- statusItemId
 - VsxProtocolDriver.Definitions.StatusItem, 78
- STRING
 - VsxProtocolDriver.Definitions.ValueType, 81
- subscribe_to_device_status_data
 - VsxProtocolDriver.Interface.Interface, 33
- SubscribeToDeviceStatusData
 - VsxProtocolDriver.Sensor.Sensor, 58
- test_system
 - VsxProtocolDriver.Interface.Interface, 26
- test_system_ex
 - VsxProtocolDriver.Interface.Interface, 26
- TestSystem
 - VsxProtocolDriver.Sensor.Sensor, 40
- TestSystemEx
 - VsxProtocolDriver.Sensor.Sensor, 40
- time
 - VsxProtocolDriver.Definitions.StatusItem, 79
- TIMEOUT
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- TIMEOUT_ANNOUNCEMENT
 - VsxProtocolDriver.Definitions.SessionTypes, 61
- UINT
 - VsxProtocolDriver.Definitions.ValueType, 80
- UNKNOWN
 - VsxProtocolDriver.Definitions.SessionTypes, 61
 - VsxProtocolDriver.Definitions.ValueType, 81
- unsubscribe_to_device_status_data
 - VsxProtocolDriver.Interface.Interface, 33
- UnsubscribeToDeviceStatusData
 - VsxProtocolDriver.Sensor.Sensor, 59
- upload_data
 - VsxProtocolDriver.Interface.Interface, 26
- upload_parameter_list
 - VsxProtocolDriver.Interface.Interface, 31
- upload_parameter_set
 - VsxProtocolDriver.Interface.Interface, 31
- UploadData
 - VsxProtocolDriver.Sensor.Sensor, 41
- UploadParameterList
 - VsxProtocolDriver.Sensor.Sensor, 53
- UploadParameterSet
 - VsxProtocolDriver.Sensor.Sensor, 41
- USB_SSI
 - VsxProtocolDriver.Definitions.SerialConnectionType, 59
- value
 - VsxProtocolDriver.Definitions.Parameter, 34
 - VsxProtocolDriver.Definitions.StatusItem, 79
- valueType
 - VsxProtocolDriver.Definitions.Parameter, 34
 - VsxProtocolDriver.Definitions.StatusItem, 79
- VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_COORD3D_A16
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f
 - VsxProtocolDriver.DataContainer.ImageData2Format, 21
- VSX_IMAGE_DATA2_FORMAT_COORD3D_B16
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f
 - VsxProtocolDriver.DataContainer.ImageData2Format, 21
- VSX_IMAGE_DATA2_FORMAT_COORD3D_C16
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f
 - VsxProtocolDriver.DataContainer.ImageData2Format, 21
- VSX_IMAGE_DATA2_FORMAT_MONO12

- VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_MONO16
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_IMAGE_DATA2_FORMAT_MONO8
 - VsxProtocolDriver.DataContainer.ImageData2Format, 20
- VSX_STATUS_ERROR_COMMAND_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VSX_STATUS_ERROR_DRIVER_CONNECTION
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_DRIVER_DATA
 - VsxProtocolDriver.Definitions.StatusCode, 64
- VSX_STATUS_ERROR_DRIVER_DEVICE
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_DRIVER_GENERAL
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_DRIVER_INIT
 - VsxProtocolDriver.Definitions.StatusCode, 64
- VSX_STATUS_ERROR_DRIVER_INVALID_DATA
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_DRIVER_LOAD_FILE
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_DRIVER_SAVE_FILE
 - VsxProtocolDriver.Definitions.StatusCode, 64
- VSX_STATUS_ERROR_DRIVER_TIMEOUT
 - VsxProtocolDriver.Definitions.StatusCode, 64
- VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 74
- VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 74
- VSX_STATUS_ERROR_EXCEPTION_THROWN
 - VsxProtocolDriver.Definitions.StatusCode, 72
- VSX_STATUS_ERROR_FILENAME_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 67
- VSX_STATUS_ERROR_GATEWAY_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 72
- VSX_STATUS_ERROR_IMAGE_TAG_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 68
- VSX_STATUS_ERROR_INPUT_VALUE_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VSX_STATUS_ERROR_INVALID_DATA_FORMAT
 - VsxProtocolDriver.Definitions.StatusCode, 71
- VSX_STATUS_ERROR_IP_ADDRESS_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 72
- VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 76
- VSX_STATUS_ERROR_LOG_NOT_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 70
- VSX_STATUS_ERROR_LOG_POINTER_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 70
- VSX_STATUS_ERROR_MAC_ADDRESS_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 74
- VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VsxProtocolDriver.Definitions.StatusCode, 77
- VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME
 - VsxProtocolDriver.Definitions.StatusCode, 77
- VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VSX_STATUS_ERROR_NETWORK_MASK_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 72
- VSX_STATUS_ERROR_NO_ELEMENT_FOUND
 - VsxProtocolDriver.Definitions.StatusCode, 71
- VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 74
- VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK
 - VsxProtocolDriver.Definitions.StatusCode, 77
- VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 67
- VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 67
- VSX_STATUS_ERROR_PARAMETER_ID_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 66
- VSX_STATUS_ERROR_POINT_X_ID_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 69
- VSX_STATUS_ERROR_POINT_Y_ID_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 69
- VSX_STATUS_ERROR_POINT_Z_ID_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 69
- VSX_STATUS_ERROR_RESULT_NOT_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 70
- VSX_STATUS_ERROR_RESULT_POINTER_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 70
- VSX_STATUS_ERROR_RESULT_TAG_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 71
- VSX_STATUS_ERROR_SESSION
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_STRING
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_STRING_POINTER_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 67
- VSX_STATUS_ERROR_STRING_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 67
- VSX_STATUS_ERROR_TAG_ZERO
 - VsxProtocolDriver.Definitions.StatusCode, 71
- VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER
 - VsxProtocolDriver.Definitions.StatusCode, 68
- VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM
 - VsxProtocolDriver.Definitions.StatusCode, 65
- VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION
 - VsxProtocolDriver.Definitions.StatusCode, 76
- VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION
 - VsxProtocolDriver.Definitions.StatusCode, 76
- VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR
 - VsxProtocolDriver.Definitions.StatusCode, 75
- VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR
 - VsxProtocolDriver.Definitions.StatusCode, 75
- VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_CONTAINER
 - VsxProtocolDriver.Definitions.StatusCode, 68
- VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_FORMAT
 - VsxProtocolDriver.Definitions.StatusCode, 68

VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_VSX_CONTAINER VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 76 VsxProtocolDriver.Definitions.StatusCode, 67

VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 77 VsxProtocolDriver.Definitions.StatusCode, 68

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 72

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 72

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 72

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_VSX_STATUS_ERROR_VSX_DEVICE_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 72

VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 69 VsxProtocolDriver.Definitions.StatusCode, 75

VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 70 VsxProtocolDriver.Definitions.StatusCode, 74

VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 70 VsxProtocolDriver.Definitions.StatusCode, 75

VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER_VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 71 VsxProtocolDriver.Definitions.StatusCode, 71

VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORM_MATRIX_IN_DATA_CONTAINER_VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 75 VsxProtocolDriver.Definitions.StatusCode, 70

VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORM_MATRIX_VSX_STATUS_ERROR_VSX_IMAGE_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 75 VsxProtocolDriver.Definitions.StatusCode, 72

VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER_VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 68 VsxProtocolDriver.Definitions.StatusCode, 76

VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM_VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 67 VsxProtocolDriver.Definitions.StatusCode, 77

VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 71 VsxProtocolDriver.Definitions.StatusCode, 77

VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE_VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 71 VsxProtocolDriver.Definitions.StatusCode, 74

VSX_STATUS_ERROR_VALUE_NOT_ZERO VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 67 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VALUE_POINTER_ZERO VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 67 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VALUE_ZERO VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 66 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VERSION VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 65 VsxProtocolDriver.Definitions.StatusCode, 74

VSX_STATUS_ERROR_VERSION_NOT_ZERO VSX_STATUS_ERROR_VSX_PARAMETER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 70 VsxProtocolDriver.Definitions.StatusCode, 74

VSX_STATUS_ERROR_VERSION_POINTER_ZERO VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 70 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND_VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 74 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 76 VsxProtocolDriver.Definitions.StatusCode, 73

VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE
 VsxProtocolDriver.Definitions.StatusCode, 76 VsxProtocolDriver.Definitions.StatusCode, 66

VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 76 VsxProtocolDriver.Definitions.StatusCode, 65

VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 68 VsxProtocolDriver.Definitions.StatusCode, 66

VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 68 VsxProtocolDriver.Definitions.StatusCode, 66

VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 73
 VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 73
 VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 73
 VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 75
 VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 75
 VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 76
 VSX_STATUS_ERROR_XML_COMMAND_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 67
 VSX_STATUS_ERROR_XPATH_ZERO
 VsxProtocolDriver.Definitions.StatusCode, 71
 VSX_STATUS_SUCCESS
 VsxProtocolDriver.Definitions.StatusCode, 64
 VsxProtocolDriver, 9
 VsxProtocolDriver.DataContainer, 10
 VsxProtocolDriver.DataContainer.DataContainer, 11
 __del__, 12
 __init__, 12
 data_container_handle, 17
 GetCaptureInformation, 13
 GetDisparityDescriptor2, 13
 GetImage, 14
 GetLine, 15
 GetOlr2CaptureInformation, 14
 GetOlr2ModbusData, 14
 GetResultElementDouble, 17
 GetResultElementInt32, 16
 GetResultElementInt64, 16
 GetResultElementString, 15
 GetResultXml, 15
 GetTagList, 13
 GetTransformation, 13
 Save3DPointCloudData, 12
 SaveData, 12
 VsxProtocolDriver.DataContainer.ImageData2Format, 19
 VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8, 20
 VSX_IMAGE_DATA2_FORMAT_COORD3D_A16, 20
 VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f, 21
 VSX_IMAGE_DATA2_FORMAT_COORD3D_B16, 20
 VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f, 21
 VSX_IMAGE_DATA2_FORMAT_COORD3D_C16, 20
 VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f, 21
 VSX_IMAGE_DATA2_FORMAT_MONO12, 20
 VSX_IMAGE_DATA2_FORMAT_MONO16, 20
 VSX_IMAGE_DATA2_FORMAT_MONO8, 20
 VsxProtocolDriver.DataContainer.VsxCaptureInformation, 82
 __init__, 82
 VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2, 83
 __init__, 83
 VsxProtocolDriver.DataContainer.VsxLineCoordinate, 84
 __init__, 84
 VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation, 85
 __init__, 85
 VsxProtocolDriver.DataContainer.VsxOlr2ModbusData, 85
 __init__, 85
 VsxProtocolDriver.DataContainer.VsxTransformation, 88
 __init__, 89
 VsxProtocolDriver.Definitions, 10
 VsxProtocolDriver.Definitions.DeviceAuthenticationMode, 17
 CONFIGURATION_ONLY, 18
 FULL, 18
 NONE, 18
 VsxProtocolDriver.Definitions.DeviceStatusScope, 18
 FULL, 18
 MULTI, 18
 VsxProtocolDriver.Definitions.DisconnectEvent, 19
 CONNECTION_ERROR, 19
 DISCONNECT_CALLED, 19
 REMOTE_HOST_CONNECTION_CLOSED, 19
 VsxProtocolDriver.Definitions.Parameter, 33
 __init__, 34
 configId, 34
 configVersion, 34
 enumItems, 34
 name, 34
 parameterId, 34
 settingsVersion, 34
 value, 34
 valueType, 34
 VsxProtocolDriver.Definitions.SerialConnectionType, 59
 CANOPEN, 60
 ETHERNET_IP, 60
 PROFIBUS, 59
 PROFINET, 60
 RS485, 60
 USB_SSI, 59
 VsxProtocolDriver.Definitions.SessionTypes, 60
 INITIAL_PASSWORD_REQUIRED, 61
 LOGIN, 61
 LOGIN_REPLY, 61
 LOGIN_REQUIRED, 60
 LOGOUT, 61
 LOGOUT_REPLY, 61
 SET_PASSWORD, 61
 SET_PASSWORD_REPLY, 61
 TIMEOUT, 61

- TIMEOUT_ANNOUNCEMENT, [61](#)
- UNKNOWN, [61](#)
- VsxProtocolDriver.Definitions.StatusCode, [62](#)
- VSX_STATUS_ERROR_COMMAND_ZERO, [66](#)
- VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO, [66](#)
- VSX_STATUS_ERROR_DRIVER_CONNECTION, [65](#)
- VSX_STATUS_ERROR_DRIVER_DATA, [64](#)
- VSX_STATUS_ERROR_DRIVER_DEVICE, [65](#)
- VSX_STATUS_ERROR_DRIVER_GENERAL, [65](#)
- VSX_STATUS_ERROR_DRIVER_INIT, [64](#)
- VSX_STATUS_ERROR_DRIVER_INVALID_DATA, [65](#)
- VSX_STATUS_ERROR_DRIVER_LOAD_FILE, [65](#)
- VSX_STATUS_ERROR_DRIVER_SAVE_FILE, [64](#)
- VSX_STATUS_ERROR_DRIVER_TIMEOUT, [64](#)
- VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO, [74](#)
- VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO, [74](#)
- VSX_STATUS_ERROR_EXCEPTION_THROWN, [72](#)
- VSX_STATUS_ERROR_FILENAME_ZERO, [67](#)
- VSX_STATUS_ERROR_GATEWAY_ZERO, [72](#)
- VSX_STATUS_ERROR_IMAGE_TAG_ZERO, [68](#)
- VSX_STATUS_ERROR_INPUT_VALUE_ZERO, [66](#)
- VSX_STATUS_ERROR_INVALID_DATA_FORMAT, [71](#)
- VSX_STATUS_ERROR_IP_ADDRESS_ZERO, [72](#)
- VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO, [76](#)
- VSX_STATUS_ERROR_LOG_NOT_ZERO, [70](#)
- VSX_STATUS_ERROR_LOG_POINTER_ZERO, [70](#)
- VSX_STATUS_ERROR_MAC_ADDRESS_ZERO, [74](#)
- VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION, [66](#)
- VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD, [77](#)
- VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME, [77](#)
- VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION, [66](#)
- VSX_STATUS_ERROR_NETWORK_MASK_ZERO, [72](#)
- VSX_STATUS_ERROR_NO_ELEMENT_FOUND, [71](#)
- VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO, [74](#)
- VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK_ZERO, [77](#)
- VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO, [67](#)
- VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO, [67](#)
- VSX_STATUS_ERROR_PARAMETER_ID_ZERO, [66](#)
- VSX_STATUS_ERROR_POINT_X_ID_ZERO, [69](#)
- VSX_STATUS_ERROR_POINT_Y_ID_ZERO, [69](#)
- VSX_STATUS_ERROR_POINT_Z_ID_ZERO, [69](#)
- VSX_STATUS_ERROR_RESULT_NOT_ZERO, [70](#)
- VSX_STATUS_ERROR_RESULT_POINTER_ZERO, [70](#)
- VSX_STATUS_ERROR_RESULT_TAG_ZERO, [71](#)
- VSX_STATUS_ERROR_SESSION, [65](#)
- VSX_STATUS_ERROR_STRING, [65](#)
- VSX_STATUS_ERROR_STRING_POINTER_ZERO, [67](#)
- VSX_STATUS_ERROR_STRING_ZERO, [67](#)
- VSX_STATUS_ERROR_TAG_ZERO, [71](#)
- VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER, [68](#)
- VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM, [65](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION, [76](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION, [76](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTION, [75](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTION, [75](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_CONTAINER, [68](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_CONTAINER, [68](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA_CONTAINER, [76](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_DATA_CONTAINER, [77](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_DATA_CONTAINER, [69](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_DATA_CONTAINER, [70](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_DATA_CONTAINER, [70](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER, [71](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_INFORMATION, [75](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_INFORMATION, [75](#)
- VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER, [75](#)

[68](#) VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO,
 VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM, [74](#)
[67](#) VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO,
 VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VALUE,
[71](#) VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO,
 VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE, [73](#)
[71](#) VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO,
 VSX_STATUS_ERROR_VALUE_NOT_ZERO, [67](#) [73](#)
 VSX_STATUS_ERROR_VALUE_POINTER_ZERO, VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO,
[67](#) [74](#)
 VSX_STATUS_ERROR_VALUE_ZERO, [66](#) VSX_STATUS_ERROR_VSX_PARAMETER_ZERO,
 VSX_STATUS_ERROR_VERSION, [65](#) [74](#)
 VSX_STATUS_ERROR_VERSION_NOT_ZERO, VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO,
[70](#) [73](#)
 VSX_STATUS_ERROR_VERSION_POINTER_ZERO, VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO,
[70](#) [73](#)
 VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_AVAILABLE, VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO,
[74](#) [73](#)
 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE,
[76](#) [66](#)
 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO,
[76](#) [65](#)
 VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO,
[76](#) [66](#)
 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO,
[68](#) [66](#)
 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO,
[68](#) [73](#)
 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO,
[67](#) [73](#)
 VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO,
[68](#) [73](#)
 VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO, VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO,
[72](#) [75](#)
 VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO, VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO,
[73](#) [75](#)
 VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO, VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO,
[72](#) [76](#)
 VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO, VSX_STATUS_ERROR_XML_COMMAND_ZERO,
[72](#) [67](#)
 VSX_STATUS_ERROR_VSX_DEVICE_ZERO, [72](#) VSX_STATUS_ERROR_XPATH_ZERO, [71](#)
 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR1_VSX_STATUS_SUCCESS, [64](#)
[75](#) VsxProtocolDriver.Definitions.StatusItem, [77](#)
 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO,
[74](#) configurationClass, [78](#)
 VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO, [78](#)
[75](#) name, [78](#)
 VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO, sensorTime, [79](#)
[71](#) settingsVersion, [78](#)
 VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO, statusItemId, [78](#)
[70](#) time, [79](#)
 VSX_STATUS_ERROR_VSX_IMAGE_ZERO, [72](#) value, [79](#)
 VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO, valueType, [79](#)
[76](#) VsxProtocolDriver.Definitions.Strategy, [79](#)
 VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO, DROP_OLDEST, [79](#)
[77](#) DROP_WRITE, [79](#)
 VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO, VsxProtocolDriver.Definitions.ValueType, [80](#)
[77](#) BASE64, [81](#)

- BOOL, [80](#)
- DOUBLE, [81](#)
- ENUM, [81](#)
- FLOAT, [81](#)
- HEXSTRING, [81](#)
- INT, [80](#)
- INT16, [80](#)
- IP, [81](#)
- LONG, [80](#)
- POINT, [81](#)
- QUAD, [81](#)
- RECTANGLE, [81](#)
- STRING, [81](#)
- UINT, [80](#)
- UNKNOWN, [81](#)
- VsxProtocolDriver.Interface, [10](#)
- VsxProtocolDriver.Interface.Interface, [21](#)
 - callback_on_device_status_received, [23](#)
 - callback_on_disconnect, [23](#)
 - callback_on_session_message_received, [23](#)
 - connect, [24](#)
 - connect_and_login, [25](#)
 - connect_ex, [25](#)
 - connect_ex_and_login, [25](#)
 - deregister_on_device_status_received, [33](#)
 - deregister_on_disconnect, [25](#)
 - deregister_on_session_message_received, [26](#)
 - disconnect, [25](#)
 - download_parameter_set, [31](#)
 - get_all_device_status_data, [32](#)
 - get_cached_container, [27](#)
 - get_capture_information, [27](#)
 - get_connected, [25](#)
 - get_data_container, [27](#)
 - get_device_information, [29](#)
 - get_disparity_descriptor2, [28](#)
 - get_dynamic_container_queue_size, [29](#)
 - get_error_text, [24](#)
 - get_image, [28](#)
 - get_library_version, [24](#)
 - get_line, [29](#)
 - get_log_message, [30](#)
 - get_log_message_queue_size, [30](#)
 - get_missing_container_frames_counter, [29](#)
 - get_missing_log_messages_counter, [30](#)
 - get_number_of_cached_containers, [29](#)
 - get_olr2_capture_information, [28](#)
 - get_olr2_modbus_data, [28](#)
 - get_parameter_list, [31](#)
 - get_result_element_double, [32](#)
 - get_result_element_int32, [32](#)
 - get_result_element_int64, [32](#)
 - get_result_element_string, [32](#)
 - get_result_xml, [32](#)
 - get_single_parameter, [32](#)
 - get_single_parameter_value, [30](#)
 - get_single_parameter_value_double, [31](#)
 - get_single_parameter_value_int32, [30](#)
- get_tag_list, [29](#)
- get_transformation, [28](#)
- get_udp_device_list, [29](#)
- get_wait_timeout, [26](#)
- init_driver, [23](#)
- init_serial_sensor, [24](#)
- init_tcp_sensor, [24](#)
- initialize_device, [25](#)
- load_default_parameter_set_on_device, [31](#)
- load_parameter_set_on_device, [31](#)
- login, [25](#)
- logout, [25](#)
- reconnect_and_login_tcp_device, [24](#)
- reconnect_serial_device, [24](#)
- reconnect_tcp_device, [24](#)
- register_on_device_status_received, [33](#)
- register_on_disconnect, [25](#)
- register_on_session_message_received, [26](#)
- release_capture_information, [27](#)
- release_data_container, [27](#)
- release_device, [29](#)
- release_device_list, [30](#)
- release_disparity_descriptor2, [28](#)
- release_image, [28](#)
- release_line, [29](#)
- release_olr2_capture_information, [28](#)
- release_olr2_modbus_data, [28](#)
- release_parameter, [32](#)
- release_parameter_list, [32](#)
- release_sensor, [24](#)
- release_status_item_list, [33](#)
- release_string, [24](#)
- release_tag_list, [29](#)
- release_transformation, [28](#)
- reset_dynamic_container_grabber, [27](#)
- reset_log_message_grabber, [30](#)
- save_3d_point_cloud_data, [27](#)
- save_data, [27](#)
- save_parameter_set_on_device, [31](#)
- send_firmware, [26](#)
- send_session_keep_alive, [26](#)
- send_xml_data_message, [26](#)
- set_network_settings, [27](#)
- set_network_settings_via_udp, [27](#)
- set_password, [25](#)
- set_single_parameter_double, [31](#)
- set_single_parameter_int32, [31](#)
- set_single_parameter_string, [32](#)
- set_single_parameter_value, [30](#)
- set_single_parameter_value_double, [30](#)
- set_single_parameter_value_int32, [30](#)
- set_wait_timeout, [26](#)
- subscribe_to_device_status_data, [33](#)
- test_system, [26](#)
- test_system_ex, [26](#)
- unsubscribe_to_device_status_data, [33](#)
- upload_data, [26](#)
- upload_parameter_list, [31](#)

- upload_parameter_set, 31
- VsxProtocolDriver.Interface.VsxCaptureInformationStructure, 82
- VsxProtocolDriver.Interface.VsxDataContainerHandle, 82
- __repr__, 83
- VsxProtocolDriver.Interface.VsxDevice, 83
- VsxProtocolDriver.Interface.VsxDeviceList, 83
- VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure, 84
- VsxProtocolDriver.Interface.VsxImage, 84
- VsxProtocolDriver.Interface.VsxLineCoordinateStructure, 84
- VsxProtocolDriver.Interface.VsxLineData, 85
- VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure, 85
- VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure, 86
- VsxProtocolDriver.Interface.VsxParameter, 86
- VsxProtocolDriver.Interface.VsxParameterEnumItem, 86
- VsxProtocolDriver.Interface.VsxParameterList, 86
 - __init__, 87
 - length, 87
 - parameters, 87
- VsxProtocolDriver.Interface.VsxStatusItem, 87
- VsxProtocolDriver.Interface.VsxStatusItemList, 87
- VsxProtocolDriver.Interface.VsxSystemHandle, 88
 - __repr__, 88
- VsxProtocolDriver.Interface.VsxTagList, 88
- VsxProtocolDriver.Interface.VsxTransformationStructure, 89
- VsxProtocolDriver.Sensor, 11
- VsxProtocolDriver.Sensor.Sensor, 35
 - __del__, 38
 - __init__, 38
 - Connect, 43
 - ConnectAndLogin, 43
 - Connected, 45
 - ConnectEx, 43
 - ConnectExAndLogin, 44
 - DeregisterOnDeviceStatusReceived, 58
 - DeregisterOnDisconnect, 46
 - DeregisterOnSessionMessageReceived, 47
 - Disconnect, 45
 - DownloadParameterSet, 41
 - DynamicContainerQueueSize, 56
 - GetAllDeviceStatusData, 57
 - GetCachedContainer, 56
 - GetCurrentDeviceInformation, 57
 - GetDataContainer, 56
 - GetDeviceInformation, 57
 - GetErrorText, 38
 - GetLibraryVersion, 38
 - GetLogMessage, 54
 - GetParameterList, 53
 - GetSingleDataValue, 51
 - GetSingleParameter, 53
 - GetSingleParameterValue, 51
 - GetSingleParameterValueDouble, 52
 - GetSingleParameterValueInt32, 52
 - GetUdpDeviceList, 57
 - GetWaitTimeout, 49
 - Handle, 39
 - InitializeDevice, 44
 - InitSerialSensor, 39
 - InitTcpSensor, 39
 - LoadDefaultParameterSetOnDevice, 42
 - LoadParameterSetOnDevice, 42
 - Login, 44
 - LogMessageQueueSize, 55
 - Logout, 44
 - MissingContainerFramesCounter, 56
 - MissingLogMessagesCounter, 55
 - NumberOfCachedContainers, 57
 - ReConnectAndLoginTcpDevice, 48
 - ReConnectSerialDevice, 48
 - ReConnectSerialSensor, 48
 - ReConnectTcpDevice, 47
 - ReConnectTcpSensor, 47
 - RegisterOnDeviceStatusReceived, 58
 - RegisterOnDisconnect, 46
 - RegisterOnSessionMessageReceived, 46
 - ReleaseSensor, 49
 - ResetDynamicContainerGrabber, 55
 - ResetDynamicContainerGrabberEx, 55
 - ResetLogMessageGrabber, 54
 - SaveParameterSetOnDevice, 42
 - SendFirmware, 40
 - SendSessionKeepAlive, 47
 - SendXmlDataMessage, 42
 - SetNetworkParameter, 49
 - SetNetworkSettings, 50
 - SetNetworkSettingsViaUdp, 50
 - SetPassword, 45
 - SetSingleDataValue, 50
 - SetSingleParameter, 53
 - SetSingleParameterValue, 51
 - SetWaitTimeout, 49
 - SubscribeToDeviceStatusData, 58
 - TestSystem, 40
 - TestSystemEx, 40
 - UnsubscribeToDeviceStatusData, 59
 - UploadData, 41
 - UploadParameterList, 53
 - UploadParameterSet, 41