# VsxProtocolDriver Python-Wrapper

3.3.2+ga52723a

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

# 1 Introduction

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

### 1.1 Supported devices

The official supported devices are the following:

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

### 1.2 Requirements

The driver is available for multiple architecture

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

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

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

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

## 2 Usage with 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 fuctionality 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 propably 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 whithin the current configuration group and how they are handled
- *parameterId*: Id of the current parameter

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

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

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

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

**Additional notes:**

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

# 5 Changelog

This is the changelog for the 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.

V3.3.2

- 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:

# 7 Hierarchical Index

## 7.1 Class Hierarchy

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

# 8 Class Index

## 8.1 Class List

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

# 9 Namespace Documentation

## 9.1 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 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

- __init__ (self, *_is_direct=True)
- __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.*
- 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

#### __init__()

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

#### __del__()

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

### 10.1.2 Member Function Documentation

#### SaveData()

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

Saves a VsxMessage to the given filename.

**Parameters**

| | |
|---|---|
| *tag* | Specify which tag from container should be saved ("*" save complete container) |
| *file_name* | Path and filename where to save the message |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

#### Save3DPointCloudData()

```
int VsxProtocolDriver.DataContainer.DataContainer.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.

**Parameters**

| | |
|---|---|
| *tag_x* | The x image tag name |
| *tag_y* | The y image tag name |
| *tag_z* | The z image tag name |
| *file_name* | Path and filename where to save the data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### GetTagList()

```
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

### GetCaptureInformation()

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

Get capture information from a dynamic container.

**Parameters**

| | |
|---|---|
| *tag* | Tag name of data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success, VsxCaptureInformation object

### GetTransformation()

```
Tuple[int, Optional[VsxTransformation]] VsxProtocolDriver.DataContainer.DataContainer.Get↩
Transformation (
              self,
             str tag )
```

Get transformation from a dynamic container.

**Parameters**

| | |
|---|---|
| *tag* | Tag name of data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success, transformation object

### GetDisparityDescriptor2()

```
Tuple[int, Optional[VsxDisparityDescriptor2]] VsxProtocolDriver.DataContainer.DataContainer.↩
GetDisparityDescriptor2 (
              self,
             str tag )
```

Get disparity descriptor from a dynamic container.

**Parameters**

| | |
|---|---|
| *tag* | Tag name of data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success, VsxDisparityDescriptor2 object

### GetOlr2CaptureInformation()

```
Tuple[int, Optional[VsxOlr2CaptureInformation]] VsxProtocolDriver.DataContainer.DataContainer.↩
GetOlr2CaptureInformation (
              self,
             str tag )
```

Get olr2 capture information from a dynamic container.

**Parameters**

| | |
|---|---|
| *tag* | Tag name of data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success, VsxOlr2CaptureInformation object

### GetOlr2ModbusData()

```
Tuple[int, Optional[VsxOlr2ModbusData]] VsxProtocolDriver.DataContainer.DataContainer.Get↩
Olr2ModbusData (
              self,
             str tag )
```

Get modbus data for olr2 sensor from a dynamic container.

**Parameters**

| | |
|---|---|
| *tag* | Tag name of data |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success, VsxOlr2ModbusData object

### 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**

| | |
|---|---|
| *image_tag* | Tag name of image data |

**Returns**

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

### GetLine()

```
Tuple[int, Optional[List[List[VsxLineCoordinate]]], Optional[dict]] VsxProtocolDriver.Data↩
Container.DataContainer.GetLine (
              self,
             str image_tag )
```

Get line data from a dynamic container.

**Parameters**

| | |
|---|---|
| *image_tag* | Tag name of line data |

**Returns**

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

### 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**

| | |
|---|---|
| *tag_result* | Name of result |

**Returns**

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

### 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**

| | |
|---|---|
| *tag_result* | Name of result |
| *xpath* | xPath definition |

**Returns**

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

### 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**

| | |
|---|---|
| *tag_result* | Name of result |
| *xpath* | xPath definition |

**Returns**

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

#### 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**

| | |
|---|---|
| *tag_result* | Name of result |
| *xpath* | xPath definition |

**Returns**

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

#### GetResultElementDouble()

```
Tuple[int, Optional[float]] VsxProtocolDriver.DataContainer.DataContainer.GetResultElement↩
Double (
              self,
             str tag_result,
             str xpath )
```

Return certain value from a result inside data container.

**Parameters**

| | |
|---|---|
| *tag_result* | Name of result |
| *xpath* | xPath definition |

**Returns**

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

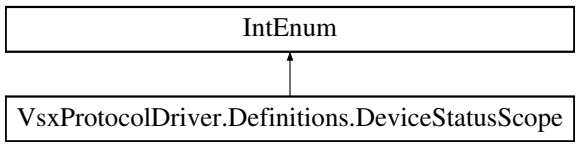
### 10.1.3 Member Data Documentation

#### data_container_handle

```
VsxProtocolDriver.DataContainer.DataContainer.data_container_handle
```

## 10.2 VsxProtocolDriver.Definitions.DeviceStatusScope Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.DeviceStatusScope:




#### Static Public Attributes

- int FULL = 0
- int MULTI = 1

### 10.2.1 Member Data Documentation

#### FULL

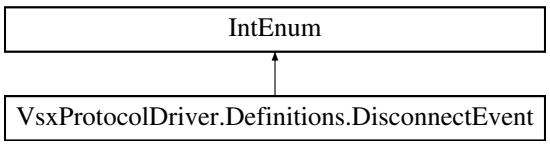
```
int VsxProtocolDriver.Definitions.DeviceStatusScope.FULL = 0  [static]
```

#### MULTI

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

## 10.3 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

### 10.3.1 Member Data Documentation

#### REMOTE_HOST_CONNECTION_CLOSED

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

#### DISCONNECT_CALLED

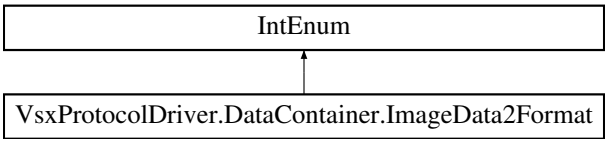
```
int VsxProtocolDriver.Definitions.DisconnectEvent.DISCONNECT_CALLED = 1  [static]
```

#### CONNECTION_ERROR

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

## 10.4 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

### 10.4.1 Member Data Documentation

#### VSX_IMAGE_DATA2_FORMAT_MONO8

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_MONO8 = 17301505
[static]
```

#### VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_CONFIDENCE8 =
17301702  [static]
```

#### VSX_IMAGE_DATA2_FORMAT_MONO12

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_MONO12 = 17825797
[static]
```

#### VSX_IMAGE_DATA2_FORMAT_MONO16

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_MONO16 = 17825799
[static]
```

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_A16

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_A16 =
17825974  [static]
```

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_B16

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_B16 =
17825975  [static]
```

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_C16

```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_C16 =
17825976  [static]
```

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_A32f

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

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f

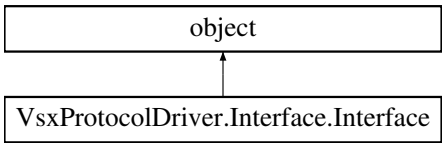
```
int VsxProtocolDriver.DataContainer.ImageData2Format.VSX_IMAGE_DATA2_FORMAT_COORD3D_B32f =
18874558  [static]
```

#### VSX_IMAGE_DATA2_FORMAT_COORD3D_C32f

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

## 10.5 VsxProtocolDriver.Interface.Interface Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.Interface:



### Public Member Functions

- 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
- 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

### 10.5.1 Member Function Documentation

#### init_driver()

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

### 10.5.2 Member Data Documentation

#### callback_on_disconnect

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

#### callback_on_session_message_received

```
VsxProtocolDriver.Interface.Interface.callback_on_session_message_received = WINFUNCTYPE(None,
c_int32, c_int32, c_int32)  [static]
```

#### callback_on_device_status_received

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

#### release_string

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

#### get_library_version

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

#### get_error_text

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

#### init_tcp_sensor

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

#### init_serial_sensor

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

### release_sensor

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

### reconnect_tcp_device

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

### reconnect_and_login_tcp_device

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

### reconnect_serial_device

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

### connect

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

### connect_ex

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

### connect_and_login

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

### connect_ex_and_login

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

### login

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

### logout

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

### set_password

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

### get_connected

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

### disconnect

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

### register_on_disconnect

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

### deregister_on_disconnect

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

### register_on_session_message_received

```
VsxProtocolDriver.Interface.Interface.register_on_session_message_received = None  [static]
```

### deregister_on_session_message_received

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

### send_session_keep_alive

```
VsxProtocolDriver.Interface.Interface.send_session_keep_alive = None  [static]
```

### test_system

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

### test_system_ex

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

### get_wait_timeout

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

### set_wait_timeout

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

### upload_data

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

### send_firmware

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

### send_xml_data_message

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

### set_network_settings

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

### set_network_settings_via_udp

```
VsxProtocolDriver.Interface.Interface.set_network_settings_via_udp = None  [static]
```

### reset_dynamic_container_grabber

```
VsxProtocolDriver.Interface.Interface.reset_dynamic_container_grabber = None  [static]
```

### get_data_container

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

### get_cached_container

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

### release_data_container

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

### save_data

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

### save_3d_point_cloud_data

```
VsxProtocolDriver.Interface.Interface.save_3d_point_cloud_data = None  [static]
```

### get_capture_information

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

### release_capture_information

```
VsxProtocolDriver.Interface.Interface.release_capture_information = None  [static]
```

### get_transformation

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

### release_transformation

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

### get_disparity_descriptor2

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

### release_disparity_descriptor2

```
VsxProtocolDriver.Interface.Interface.release_disparity_descriptor2 = None  [static]
```

### get_olr2_capture_information

```
VsxProtocolDriver.Interface.Interface.get_olr2_capture_information = None  [static]
```

### release_olr2_capture_information

```
VsxProtocolDriver.Interface.Interface.release_olr2_capture_information = None  [static]
```

### get_olr2_modbus_data

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

### release_olr2_modbus_data

```
VsxProtocolDriver.Interface.Interface.release_olr2_modbus_data = None  [static]
```

### get_image

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

### release_image

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

### get_line

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

### release_line

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

### get_tag_list

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

### release_tag_list

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

### get_missing_container_frames_counter

```
VsxProtocolDriver.Interface.Interface.get_missing_container_frames_counter = None  [static]
```

### get_dynamic_container_queue_size

VsxProtocolDriver.Interface.Interface.get_dynamic_container_queue_size = None [static]

### get_number_of_cached_containers

VsxProtocolDriver.Interface.Interface.get_number_of_cached_containers = None [static]

### get_device_information

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

### release_device

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

### get_udp_device_list

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

### release_device_list

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

### reset_log_message_grabber

VsxProtocolDriver.Interface.Interface.reset_log_message_grabber = None [static]

### get_log_message

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

### get_log_message_queue_size

VsxProtocolDriver.Interface.Interface.get_log_message_queue_size = None [static]

### get_missing_log_messages_counter

VsxProtocolDriver.Interface.Interface.get_missing_log_messages_counter = None [static]

### set_single_parameter_value

VsxProtocolDriver.Interface.Interface.set_single_parameter_value = None [static]

### set_single_parameter_value_int32

VsxProtocolDriver.Interface.Interface.set_single_parameter_value_int32 = None [static]

### set_single_parameter_value_double

VsxProtocolDriver.Interface.Interface.set_single_parameter_value_double = None [static]

### get_single_parameter_value

VsxProtocolDriver.Interface.Interface.get_single_parameter_value = None [static]

### get_single_parameter_value_int32

VsxProtocolDriver.Interface.Interface.get_single_parameter_value_int32 = None [static]

### get_single_parameter_value_double

VsxProtocolDriver.Interface.Interface.get_single_parameter_value_double = None [static]

### load_default_parameter_set_on_device

VsxProtocolDriver.Interface.Interface.load_default_parameter_set_on_device = None [static]

### load_parameter_set_on_device

VsxProtocolDriver.Interface.Interface.load_parameter_set_on_device = None [static]

### save_parameter_set_on_device

VsxProtocolDriver.Interface.Interface.save_parameter_set_on_device = None [static]

### upload_parameter_set

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

### download_parameter_set

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

### get_parameter_list

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

### upload_parameter_list

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

### set_single_parameter_double

VsxProtocolDriver.Interface.Interface.set_single_parameter_double = None [static]

### set_single_parameter_int32

VsxProtocolDriver.Interface.Interface.set_single_parameter_int32 = None [static]

### set_single_parameter_string

VsxProtocolDriver.Interface.Interface.set_single_parameter_string = None [static]

### get_single_parameter

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

### release_parameter

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

### release_parameter_list

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

### get_result_xml

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

### get_result_element_string

VsxProtocolDriver.Interface.Interface.get_result_element_string = None [static]

### get_result_element_int32

VsxProtocolDriver.Interface.Interface.get_result_element_int32 = None [static]

### get_result_element_int64

VsxProtocolDriver.Interface.Interface.get_result_element_int64 = None [static]

### get_result_element_double

VsxProtocolDriver.Interface.Interface.get_result_element_double = None [static]

### get_all_device_status_data

VsxProtocolDriver.Interface.Interface.get_all_device_status_data = None [static]

### release_status_item_list

VsxProtocolDriver.Interface.Interface.release_status_item_list = None [static]

### register_on_device_status_received

VsxProtocolDriver.Interface.Interface.register_on_device_status_received = None [static]

### deregister_on_device_status_received

VsxProtocolDriver.Interface.Interface.deregister_on_device_status_received = None [static]

### subscribe_to_device_status_data

VsxProtocolDriver.Interface.Interface.subscribe_to_device_status_data = None [static]

### unsubscribe_to_device_status_data

VsxProtocolDriver.Interface.Interface.unsubscribe_to_device_status_data = None [static]

## 10.6 VsxProtocolDriver.Definitions.Parameter Class Reference

### Public Member Functions

- __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

### 10.6.1 Constructor & Destructor Documentation

#### __init__()

```
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 )
```

### 10.6.2 Member Data Documentation

#### settingsVersion

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

#### configId

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

#### configVersion

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

#### parameterId

`VsxProtocolDriver.Definitions.Parameter.parameterId`

#### name

`VsxProtocolDriver.Definitions.Parameter.name`

#### value

`VsxProtocolDriver.Definitions.Parameter.value`

#### valueType

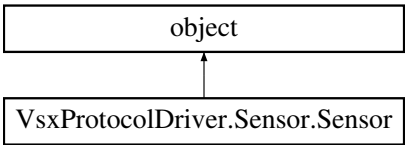
`VsxProtocolDriver.Definitions.Parameter.valueType`

#### enumItems

`VsxProtocolDriver.Definitions.Parameter.enumItems`

## 10.7 VsxProtocolDriver.Sensor.Sensor Class Reference

Inheritance diagram for VsxProtocolDriver.Sensor.Sensor:




#### Public Member Functions

- __init__ (self, *_is_direct=True)
- __del__ (self)
- Optional[bool] Handle (self)

  *Get the actual used handle from the actual sensor instance.*
- InitTcpSensor (cls, str ip_address, str plugin_name)

  *Initialize a new tcp based sensor.*
- InitSerialSensor (cls, str serial_port, int baud_rate, str sensor_type, SerialConnectionType connection_type, str plugin_name)

  *Inits 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 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.*
- SetNetworkParameter (self, str ip_address, str network_mask, str gateway)
- 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.*
- 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.*

### Static Protected Member Functions

- _OnDisconnect (c_int32 handle, c_char_p ip_address, c_int32 disconnect_event, c_char_p description)

  *Callback function for disconnect event (internal usage)*
- _OnSessionMessageReceived (c_int32 handle, c_int32 session_type, c_int32 timeout)

  *callback for "vsx_OnSessionMessageReceived" (internal usage)*
- _OnDeviceStatusReceived (c_int32 handle, c_int32 device_status_scope, POINTER(VsxStatusItemList) status_item_list)

  *callback for "vsx_OnDeviceStatusReceived" (internal)*

### Protected Attributes

- _vsx_handle
- _OnSessionMessageReceived
- _OnDeviceStatusReceived

### 10.7.1 Constructor & Destructor Documentation

#### __init__()

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

#### __del__()

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

### 10.7.2 Member Function Documentation

#### GetLibraryVersion()

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

Returns the actual library version.

#### 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**

| | |
|---|---|
| *error_code* | Input error code |

**Returns**

: error text

#### Handle()

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

Get the actual used handle from the actual sensor instance.

### InitTcpSensor()

```
VsxProtocolDriver.Sensor.Sensor.InitTcpSensor (
              cls,
             str ip_address,
             str plugin_name )
```

Initialize a new tcp based sensor.

**Parameters**

| | |
|---|---|
| *ip_address* | e.g. "192.168.2.4" |
| *plugin_name* | Additional functionality for special sensors |

**Returns**

: New instance of Sensor class

### InitSerialSensor()

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

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

**Parameters**

| | |
|---|---|
| *serial_port* | The comport of the device |
| *baud_rate* | The baudrate of the device |
| *sensor_type* | The sensor type of the device |
| *connection_type* | The connection type of the device |
| *plugin_name* | Additional functionality for special sensors |

**Returns**

:

### TestSystem()

```
Tuple[int, Optional[str], int] VsxProtocolDriver.Sensor.Sensor.TestSystem (
              self,
             str command,
             str input_value )
```

Sends a test system command to the device.

**Parameters**

| | |
|---|---|
| *command* | The test system command |
| *input_value* | Optional input value |

**Returns**

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

### TestSystemEx()

```
Tuple[int, Optional[str], int] VsxProtocolDriver.Sensor.Sensor.TestSystemEx (
              self,
             str command,
             str input_value,
             int timeout_ms )
```

Sends a test system command to the device.

**Parameters**

| | |
|---|---|
| *command* | The test system command |
| *input_value* | Optional input value |
| *timeout_ms* | Timeout for function to execute command (in ms) |

**Returns**

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

### SendFirmware()

```
int VsxProtocolDriver.Sensor.Sensor.SendFirmware (
              self,
             str file_name )
```

Sends a firmware update file to the device.

**Parameters**

| | |
|---|---|
| *file_name* | The path and filename of the firmware file |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### 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**

| | |
|---|---|
| *file_name* | The path and filename of the data file |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### UploadParameterSet()

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

Uploads a parameter file to the device.

**Parameters**

| | |
|---|---|
| *file_name* | Path and filename to upload |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### DownloadParameterSet()

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

Save the current parameter set to a file.

**Parameters**

| | |
|---|---|
| *file_name* | Path and file name to save to |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### 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**

| | |
|---|---|
| *xml_command* | Command to send |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### LoadDefaultParameterSetOnDevice()

```
int VsxProtocolDriver.Sensor.Sensor.LoadDefaultParameterSetOnDevice (
             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

### LoadParameterSetOnDevice()

```
int VsxProtocolDriver.Sensor.Sensor.LoadParameterSetOnDevice (
             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

### 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

### Connect()

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

Connect with the device.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ConnectEx()

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

Connect with the device.

**Parameters**

| | |
|---|---|
| *timeout_ms* | The timeout for a connection attempt (in ms) |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ConnectAndLogin()

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

Connect with the device.

**Parameters**

| | |
|---|---|
| *username* | username for login |
| *password* | password for login |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ConnectExAndLogin()

```
int VsxProtocolDriver.Sensor.Sensor.ConnectExAndLogin (
              self,
```

```
            str username,
            str password,
            int timeout_ms )
```

Connect with the device.

**Parameters**

| | |
|---|---|
| *username* | username for login |
| *password* | password for login |
| *timeout_ms* | The timeout for a connection attempt (in ms) |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### Login()

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

Login to the device.

**Parameters**

| | |
|---|---|
| *username* | username for login |
| *password* | password for login |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### Logout()

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

Logout from device.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### SetPassword()

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

Set new password on the device.

**Parameters**

| | |
|---|---|
| *authorization_username* | username for authorization account |
| *authorization_password* | password for authorization account |
| *username* | username for account to set new password |
| *password* | password for account |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### Connected()

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

Indicates current connection state with the device.

**Returns**

: Returns `True` or `False`

### Disconnect()

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

Disconnect with the device.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### _OnDisconnect()

```
VsxProtocolDriver.Sensor.Sensor._OnDisconnect (
             c_int32 handle,
             c_char_p ip_address,
             c_int32 disconnect_event,
             c_char_p description )  [static], [protected]
```

Callback function for disconnect event (internal usage)

**Parameters**

| | |
|---|---|
| *handle* | Returns actual used handle |
| *ip_address* | Actual used ip address |
| *disconnect_event* | Show information about disconnect type |
| *description* | Additional description |

### 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:

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

**Parameters**

| | |
|---|---|
| *func* | The function should have the following definition |

**Returns**

: Return value

### DeregisterOnDisconnect()

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

Function to deregister already existing callback function.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### _OnSessionMessageReceived()

```
VsxProtocolDriver.Sensor.Sensor._OnSessionMessageReceived (
             c_int32 handle,
             c_int32 session_type,
             c_int32 timeout )  [static], [protected]
```

callback for "vsx_OnSessionMessageReceived" (internal usage)

**Parameters**

| | |
|---|---|
| *handle* | Returns actual used handle |
| *session_type* | Session type event |
| *timeout* | Timeout, when session will end |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### RegisterOnSessionMessageReceived()

```
int VsxProtocolDriver.Sensor.Sensor.RegisterOnSessionMessageReceived (
              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:

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

**Parameters**

| | |
|---|---|
| *func* | The function should have the following definition |

**Returns**

: Return value

### DeregisterOnSessionMessageReceived()

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

Function to deregister already existing callback function.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### 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

### ReConnectTcpSensor()

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

### ReConnectTcpDevice()

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

Disconnects the device and reconnects with new connection settings.

**Parameters**

| | |
|---|---|
| *ip_address* | The new IPAddress |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ReConnectAndLoginTcpDevice()

```
int VsxProtocolDriver.Sensor.Sensor.ReConnectAndLoginTcpDevice (
              self,
             str ip_address,
             str username,
             str password )
```

Disconnects the device and reconnects with new connection settings.

**Parameters**

| | |
|---|---|
| *ip_address* | The new IPAddress |
| *username* | username for login |
| *password* | password for login |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ReConnectSerialSensor()

```
int VsxProtocolDriver.Sensor.Sensor.ReConnectSerialSensor (
              self,
             str serial_port,
             int baud_rate,
             str _,
             SerialConnectionType connection_type )
```

### 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**

| | |
|---|---|
| *serial_port* | The new serial port |
| *baud_rate* | The new baudrate |
| *connection_type* | The new connection type |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### ReleaseSensor()

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

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

### SetWaitTimeout()

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

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

**Parameters**

| | |
|---|---|
| *timeout_ms* | Time in ms |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### 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

### SetNetworkParameter()

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

### SetNetworkSettings()

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

Sets the network settings of the device.

**Parameters**

| | |
|---|---|
| *ip_address* | The new IP Address |
| *network_mask* | The new network mask |
| *gateway* | The new gateway address |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### SetNetworkSettingsViaUdp()

```
VsxProtocolDriver.Sensor.Sensor.SetNetworkSettingsViaUdp (
             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**

| | |
|---|---|
| *mac_address* | The mac address of the device to set |
| *ip_address* | The new IP Address |
| *network_mask* | The new network mask |
| *gateway* | The new gateway address |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### SetSingleDataValue()

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

### 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**

| | |
|---|---|
| *settings_version* | The settings version of the parameter which should be set |
| *configuration_id* | The config id of the parameter which should be set |
| *configuration_version* | The config version of the parameter which should be se |
| *parameter_id* | The id of the parameter which should be set |
| *value* | Value as string, float or integer number |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### GetSingleDataValue()

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

### 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**

| | |
|---|---|
| *settings_version* | The settings version of the parameter its value is asked for |
| *configuration_id* | The config id of the parameter its value is asked for |
| *configuration_version* | The config version of the parameter its value is asked for |
| *parameter_id* | The id of the parameter its value is asked for |

**Returns**

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

### 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**

| | |
|---|---|
| *settings_version* | The settings version of the parameter its value is asked for |
| *configuration_id* | The config id of the parameter its value is asked for |
| *configuration_version* | The config version of the parameter its value is asked for |
| *parameter_id* | The id of the parameter its value is asked for |

**Returns**

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

### 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**

| | |
|---|---|
| *settings_version* | The settings version of the parameter its value is asked for |
| *configuration_id* | The config id of the parameter its value is asked for |
| *configuration_version* | The config version of the parameter its value is asked for |
| *parameter_id* | The id of the parameter its value is asked for |

**Returns**

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

### 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

### UploadParameterList()

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

Uploads a parameter list to the device.

**Parameters**

| | |
|---|---|
| *parameter_list_data* | |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### SetSingleParameter()

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

Sets the parameter to a value on the device.

**Parameters**

| | |
|---|---|
| *parameter* | The parameter the value should be set from |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### GetSingleParameter()

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

Returns the current value of the given parameter from device.

**Parameters**

| | |
|---|---|
| *parameter* | The parameter its value is asked for |

**Returns**

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

### ResetLogMessageGrabber()

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

Starts the internal log message grabber.

**Parameters**

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

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### GetLogMessage()

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

Gets the oldest saved item and removes it internally.

**Parameters**

| | |
|---|---|
| *timeout_ms* | The maximum time in ms to try reading an item |

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### LogMessageQueueSize()

```
Optional[int] VsxProtocolDriver.Sensor.Sensor.LogMessageQueueSize (
              self )
```

Gets the current size of the log message queue.

**Returns**

: Actual log message counter

### MissingLogMessagesCounter()

```
Optional[int] VsxProtocolDriver.Sensor.Sensor.MissingLogMessagesCounter (
              self )
```

Gets the missing log messages counter for log message grabbing.

**Returns**

: Missing log messages counter

### ResetDynamicContainerGrabber()

```
int VsxProtocolDriver.Sensor.Sensor.ResetDynamicContainerGrabber (
              self,
             int buffer_size,
             int _,
             Strategy strategy )
```

### ResetDynamicContainerGrabberEx()

```
int VsxProtocolDriver.Sensor.Sensor.ResetDynamicContainerGrabberEx (
              self,
             int buffer_size,
             Strategy strategy )
```

Restarts the internal dynamic container grabber.

#### Parameters

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

#### Returns

: Returns VSX_STATUS_SUCCESS(0) on success

### GetDataContainer()

```
Tuple[int, Optional[DataContainer]] VsxProtocolDriver.Sensor.Sensor.GetDataContainer (
              self,
             int timeout_ms )
```

Gets the oldest saved item and removes it internally.

#### Parameters

| | |
|---|---|
| *timeout_ms* | The maximum time in ms to try reading an item. |

#### Returns

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

### GetCachedContainer()

```
Tuple[int, Optional[DataContainer]] VsxProtocolDriver.Sensor.Sensor.GetCachedContainer (
              self,
             int position )
```

Gets a cached dynamic container.

**Parameters**

| | |
|---|---|
| *position* | Position of the container in cache |

**Returns**

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

### MissingContainerFramesCounter()

```
Optional[int] VsxProtocolDriver.Sensor.Sensor.MissingContainerFramesCounter (
              self )
```

Gets the missing frame counter from image grabbing.

**Returns**

: Actual missing frame counter

### DynamicContainerQueueSize()

```
Optional[int] VsxProtocolDriver.Sensor.Sensor.DynamicContainerQueueSize (
              self )
```

Gets the current size of the dynamic container message queue.

**Returns**

: Returns actual queue size

### NumberOfCachedContainers()

```
Optional[int] VsxProtocolDriver.Sensor.Sensor.NumberOfCachedContainers (
              self )
```

Gets the current number of cached container messages.

**Returns**

: Returns actual cached containers

### GetCurrentDeviceInformation()

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

### 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

### 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

### 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

### _OnDeviceStatusReceived()

```
VsxProtocolDriver.Sensor.Sensor._OnDeviceStatusReceived (
            c_int32 handle,
            c_int32 device_status_scope,
            POINTER(VsxStatusItemList) status_item_list ) [static], [protected]
```

callback for "vsx_OnDeviceStatusReceived" (internal)

### 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:

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

**Parameters**

| | |
|---|---|
| *func* | The function should have the following definition: |

**Returns**

:

### DeregisterOnDeviceStatusReceived()

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

Function to deregister already existing callback function.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

### 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

### UnsubscribeToDeviceStatusData()

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

Unsubscribe status data from sensor.

**Returns**

: Returns VSX_STATUS_SUCCESS(0) on success

## 10.7.6 Member Data Documentation

### _vsx_handle

```
VsxProtocolDriver.Sensor.Sensor._vsx_handle  [protected]
```

#### _OnSessionMessageReceived

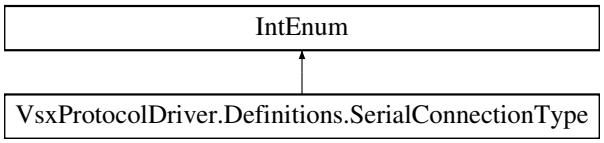
```
VsxProtocolDriver.Sensor.Sensor._OnSessionMessageReceived  [protected]
```

#### _OnDeviceStatusReceived

```
VsxProtocolDriver.Sensor.Sensor._OnDeviceStatusReceived  [protected]
```

## 10.8 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

### 10.8.1 Member Data Documentation

#### USB_SSI

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

#### PROFIBUS

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

#### PROFINET

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

#### ETHERNET_IP

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

#### RS485

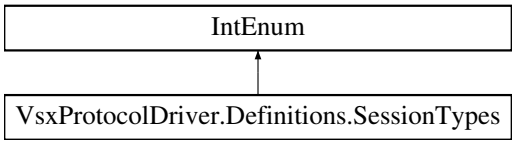
```
int VsxProtocolDriver.Definitions.SerialConnectionType.RS485 = 4  [static]
```

#### CANOPEN

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

## 10.9 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

### 10.9.1 Member Data Documentation

#### LOGIN_REQUIRED

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

#### INITIAL_PASSWORD_REQUIRED

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

#### LOGIN

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

#### LOGIN_REPLY

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

#### SET_PASSWORD

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

#### SET_PASSWORD_REPLY

```
int VsxProtocolDriver.Definitions.SessionTypes.SET_PASSWORD_REPLY = 5  [static]
```

#### TIMEOUT_ANNOUNCEMENT

```
int VsxProtocolDriver.Definitions.SessionTypes.TIMEOUT_ANNOUNCEMENT = 6  [static]
```

#### TIMEOUT

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

#### LOGOUT

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

#### LOGOUT_REPLY

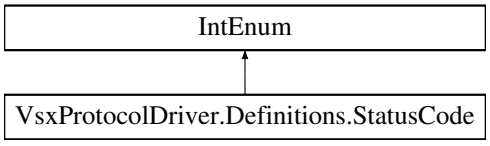
```
int VsxProtocolDriver.Definitions.SessionTypes.LOGOUT_REPLY = 9  [static]
```

#### UNKNOWN

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

## 10.10 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 = -0x0804C
- int VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO = -0x0804D
- int VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO = -0x0804E
- int VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO = -0x0804F
- int VSX_STATUS_ERROR_MAC_ADDRESS_ZERO = -0x08050
- int VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND = -0x08051
- int VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO = -0x08052
- int VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO = -0x08053
- 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

### 10.10.1 Member Data Documentation

#### VSX_STATUS_SUCCESS

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

#### VSX_STATUS_ERROR_DRIVER_INIT

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

#### VSX_STATUS_ERROR_DRIVER_TIMEOUT

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

#### VSX_STATUS_ERROR_DRIVER_SAVE_FILE

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

#### VSX_STATUS_ERROR_DRIVER_DATA

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

#### VSX_STATUS_ERROR_DRIVER_CONNECTION

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_CONNECTION = -0x5  [static]
```

### VSX_STATUS_ERROR_DRIVER_INVALID_DATA

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

### VSX_STATUS_ERROR_DRIVER_DEVICE

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

### VSX_STATUS_ERROR_DRIVER_LOAD_FILE

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_LOAD_FILE = -0x8  [static]
```

### VSX_STATUS_ERROR_SESSION

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_SESSION = -0x9  [static]
```

### VSX_STATUS_ERROR_STRING

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_STRING = -0x0A  [static]
```

### VSX_STATUS_ERROR_VERSION

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VERSION = -0x0B  [static]
```

### VSX_STATUS_ERROR_DRIVER_GENERAL

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_DRIVER_GENERAL = -0x1000  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_SYSTEM =
-0x8001  [static]
```

### VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_ZERO

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

### VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_NOT_AVAILABLE

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

### VSX_STATUS_ERROR_MISSING_IP_ADDRESS_DECLARATION

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

### VSX_STATUS_ERROR_MISSING_SERIALPORT_DECLARATION

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

### VSX_STATUS_ERROR_VSX_SYSTEM_HANDLE_POINTER_ZERO

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

### VSX_STATUS_ERROR_CONFIGURATION_ID_ZERO

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

### VSX_STATUS_ERROR_PARAMETER_ID_ZERO

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

### VSX_STATUS_ERROR_VALUE_ZERO

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

### VSX_STATUS_ERROR_COMMAND_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_COMMAND_ZERO = -0x800B  [static]
```

### VSX_STATUS_ERROR_INPUT_VALUE_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_INPUT_VALUE_ZERO = -0x800C  [static]
```

### VSX_STATUS_ERROR_OUTPUT_VALUE_POINTER_ZERO

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

### VSX_STATUS_ERROR_OUTPUT_VALUE_NOT_ZERO

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

### VSX_STATUS_ERROR_VALUE_POINTER_ZERO

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

### VSX_STATUS_ERROR_VALUE_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VALUE_NOT_ZERO = -0x8010  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_SYSTEM

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

### VSX_STATUS_ERROR_XML_COMMAND_ZERO

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

### VSX_STATUS_ERROR_FILENAME_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_FILENAME_ZERO = -0x8013  [static]
```

### VSX_STATUS_ERROR_STRING_POINTER_ZERO

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

### VSX_STATUS_ERROR_STRING_ZERO

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

### VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_↩
POINTER_ZERO = -0x8016  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_ALLOCATE_VSX_DATA_↩
CONTAINER = -0x8017  [static]
```

### VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_↩
ZERO = -0x8018  [static]
```

### VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_ZERO

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

### VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_AVAILABLE

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DATA_CONTAINER_HANDLE_NOT_↩
AVAILABLE = -0x801A  [static]
```

### VSX_STATUS_ERROR_IMAGE_TAG_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_IMAGE_TAG_ZERO = -0x801B  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_VSX_DATA_CONTAINER
= -0x801C  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_ID_IN_↩
DATA_CONTAINER = -0x801D  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_IMAGE_TAG_TO_↩
DATA_FORMAT = -0x801E  [static]
```

### VSX_STATUS_ERROR_POINT_Z_ID_ZERO

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

### VSX_STATUS_ERROR_POINT_Y_ID_ZERO

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

### VSX_STATUS_ERROR_POINT_X_ID_ZERO

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

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_IN_↩
DATA_CONTAINER = -0x8022  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_IN_↩
DATA_CONTAINER = -0x8023  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_IN_↩
DATA_CONTAINER = -0x8024  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_X_ID_TO_↩
DATA_FORMAT = -0x8025  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Y_ID_TO_↩
DATA_FORMAT = -0x8026  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_POINT_Z_ID_TO_↩
DATA_FORMAT = -0x8027  [static]
```

### VSX_STATUS_ERROR_LOG_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_LOG_POINTER_ZERO = -0x8028  [static]
```

### VSX_STATUS_ERROR_LOG_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_LOG_NOT_ZERO = -0x8029  [static]
```

### VSX_STATUS_ERROR_RESULT_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_RESULT_NOT_ZERO = -0x802A  [static]
```

### VSX_STATUS_ERROR_RESULT_POINTER_ZERO

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

### VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_IN_↩
DATA_CONTAINER = -0x802C  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_RESULT_ID_TO_↩
DATA_FORMAT = -0x802D  [static]
```

### VSX_STATUS_ERROR_VERSION_POINTER_ZERO

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

### VSX_STATUS_ERROR_VERSION_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VERSION_NOT_ZERO = -0x802F  [static]
```

### VSX_STATUS_ERROR_VSX_IMAGE_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_IMAGE_NOT_ZERO

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

### VSX_STATUS_ERROR_UNDEFINED_STRATEGY_VALUE

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

### VSX_STATUS_ERROR_UNDEFINED_CONNECTION_TYPE_VALUE

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

### VSX_STATUS_ERROR_XPATH_ZERO

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

### VSX_STATUS_ERROR_INVALID_DATA_FORMAT

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

### VSX_STATUS_ERROR_NO_ELEMENT_FOUND

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

### VSX_STATUS_ERROR_RESULT_TAG_ZERO

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

### VSX_STATUS_ERROR_TAG_ZERO

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

### VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_TAG_IN_DATA_↩
CONTAINER = -0x8039  [static]
```

### VSX_STATUS_ERROR_IP_ADDRESS_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_IP_ADDRESS_ZERO = -0x803A  [static]
```

### VSX_STATUS_ERROR_NETWORK_MASK_ZERO

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

### VSX_STATUS_ERROR_GATEWAY_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_GATEWAY_ZERO = -0x803C  [static]
```

### VSX_STATUS_ERROR_EXCEPTION_THROWN

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

### VSX_STATUS_ERROR_VSX_DEVICE_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_DEVICE_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_IMAGE_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_IMAGE_ZERO = -0x8040  [static]
```

### VSX_STATUS_ERROR_VSX_DEVICE_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DEVICE_ZERO = -0x8041  [static]
```

### VSX_STATUS_ERROR_VSX_DEVICE_LIST_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_DEVICE_LIST_ZERO

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

### VSX_STATUS_ERROR_VSX_TAG_LIST_ZERO

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

### VSX_STATUS_ERROR_VSX_TAG_LIST_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_TAG_LIST_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_PARAMETER_LIST_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_PARAMETER_LIST_ZERO

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

### VSX_STATUS_ERROR_VSX_PARAMETER_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_POINTER_↩
ZERO = -0x804A  [static]
```

### VSX_STATUS_ERROR_VSX_STATUS_ITEM_LIST_ZERO

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

### VSX_STATUS_ERROR_VSX_STATUS_ITEM_NOT_ZERO

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

### VSX_STATUS_ERROR_ERROR_TEXT_POINTER_ZERO

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

### VSX_STATUS_ERROR_ERROR_TEXT_NOT_ZERO

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

### VSX_STATUS_ERROR_ON_DISCONNECT_CALLBACK_ZERO

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

### VSX_STATUS_ERROR_MAC_ADDRESS_ZERO

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

### VSX_STATUS_ERROR_VSX_CACHED_CONTAINER_NOT_FOUND

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

### VSX_STATUS_ERROR_VSX_PARAMETER_LIST_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_PARAMETER_POINTER_ZERO

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

### VSX_STATUS_ERROR_VSX_PARAMETER_ZERO

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

### VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_↩
POINTER_ZERO = -0x08055  [static]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_DESCRIPTOR2_TAG

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_DISPARITY_↩
DESCRIPTOR2_TAG = -0x08056  [static]
```

### 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]
```

### VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_NOT_↩
ZERO = -0x08058  [static]
```

### VSX_STATUS_ERROR_VSX_DISPARITY_DESCRIPTOR2_ZERO

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

### VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_TRANSFORMATION_POINTER_ZERO
= -0x0805A  [static]
```

### 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]
```

### VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_TAG

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_TRANSFORMATION_↩
TAG = -0x0805C  [static]
```

### VSX_STATUS_ERROR_VSX_TRANSFORMATION_NOT_ZERO

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

### VSX_STATUS_ERROR_VSX_TRANSFORMATION_ZERO

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

### VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION_TAG

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_CAPTURE_INFORMATION↩
_TAG = -0x0805F  [static]
```

### 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]
```

### VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_POINTER_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_POINTER↩
_ZERO = -0x08061  [static]
```

### VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_NOT_ZERO
= -0x08062  [static]
```

### VSX_STATUS_ERROR_VSX_CAPTURE_INFORMATION_ZERO

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

### VSX_STATUS_ERROR_VSX_LINE_DATA_POINTER_ZERO

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

### VSX_STATUS_ERROR_LINE_DATA_TAG_ZERO

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

### VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA_CONTAINER

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_ID_IN_DATA↩
_CONTAINER = -0x08066  [static]
```

#### VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_DATA_FORMAT

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_UNABLE_TO_FIND_LINE_TAG_TO_↩
DATA_FORMAT = -0x08067  [static]
```

#### VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_LINE_NOT_ZERO = -0x08068
[static]
```

#### VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_VSX_LINE_DATA_ZERO = -0x08069
[static]
```

#### VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_MISSING_LOGIN_PASSWORD = -0x0806A
[static]
```

#### VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_MISSING_LOGIN_USERNAME = -0x0806B
[static]
```

#### VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_CALLBACK_ZERO

```
int VsxProtocolDriver.Definitions.StatusCode.VSX_STATUS_ERROR_ON_SESSION_MESSAGE_RECEIVED_↩
CALLBACK_ZERO = -0x0806C  [static]
```

## 10.11 VsxProtocolDriver.Definitions.StatusItem Class Reference

### Public Member Functions

- __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

### 10.11.1 Constructor & Destructor Documentation

#### __init__()

```
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 )
```

### 10.11.2 Member Data Documentation

#### settingsVersion

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

#### configurationClass

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

#### configVersion

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

#### statusItemId

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

#### name

```
VsxProtocolDriver.Definitions.StatusItem.name
```

#### value

```
VsxProtocolDriver.Definitions.StatusItem.value
```

#### valueType

```
VsxProtocolDriver.Definitions.StatusItem.valueType
```

#### time

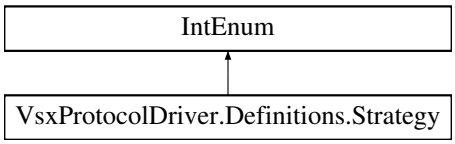
`VsxProtocolDriver.Definitions.StatusItem.time`

#### sensorTime

`VsxProtocolDriver.Definitions.StatusItem.sensorTime`

## 10.12 VsxProtocolDriver.Definitions.Strategy Class Reference

Inheritance diagram for VsxProtocolDriver.Definitions.Strategy:




#### Static Public Attributes

- int DROP_OLDEST = 0
- int DROP_WRITE = 1

### 10.12.1 Member Data Documentation

#### DROP_OLDEST

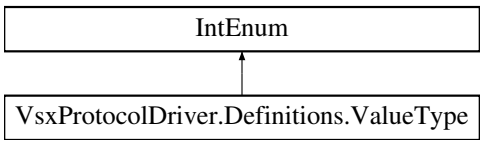
```
int VsxProtocolDriver.Definitions.Strategy.DROP_OLDEST = 0  [static]
```

#### DROP_WRITE

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

## 10.13 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

## 10.13.1 Member Data Documentation

### BOOL

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

### INT

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

### LONG

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

### UINT

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

### INT16

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

### FLOAT

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

### DOUBLE

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

### STRING

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

### HEXSTRING

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

### BASE64

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

### ENUM

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

### IP

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

### RECTANGLE

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

### QUAD

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

### POINT

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

### UNKNOWN

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

## 10.14 VsxProtocolDriver.DataContainer.VsxCaptureInformation Class Reference

CaptureInformation contains information identifying and describing the captured image.

### Public Member Functions

- __init__ (self)

### 10.14.1 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).

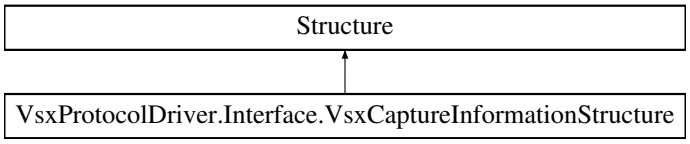
### 10.14.2 Constructor & Destructor Documentation

#### __init__()

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

## 10.15 VsxProtocolDriver.Interface.VsxCaptureInformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxCaptureInformationStructure:




### Static Protected Attributes

- list _fields_

### 10.15.1 Member Data Documentation

#### _fields_

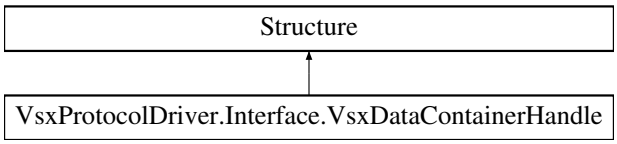
```
list VsxProtocolDriver.Interface.VsxCaptureInformationStructure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('triggerCounter', c_uint64),
                ('parameterId', c_uint64),
                ('jobId', c_uint64),
                ('rotaryEncoder', c_int64),
                ('frameCounter', c_uint64),
                ('timestamp', c_uint64),
                ('exposureTime', c_uint32),
                ('gain', c_uint32),
                ('illumination', c_uint8),
                ('triggerSource', c_uint8),
                ]
```

## 10.16 VsxProtocolDriver.Interface.VsxDataContainerHandle Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDataContainerHandle:




### Public Member Functions

- __repr__ (self)

### Public Attributes

- handle

### Static Protected Attributes

- list _fields_ = [('handle', c_int32)]

### 10.16.1 Member Function Documentation

#### __repr__()

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

### 10.16.2 Member Data Documentation

#### _fields_

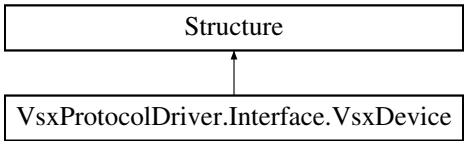
```
list VsxProtocolDriver.Interface.VsxDataContainerHandle._fields_ = [('handle', c_int32)]  [static],
[protected]
```

#### handle

```
VsxProtocolDriver.Interface.VsxDataContainerHandle.handle
```

## 10.17 VsxProtocolDriver.Interface.VsxDevice Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDevice:




#### Static Protected Attributes

- list _fields_

### 10.17.1 Member Data Documentation

#### _fields_

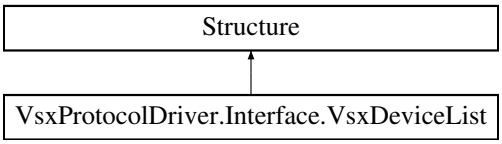
```
list VsxProtocolDriver.Interface.VsxDevice._fields_  [static], [protected]
```

**Initial value:**

```
=  [('ipAddress', c_char_p),
                ('networkMask', c_char_p),
                ('gateway', c_char_p),
                ('macAddress', c_char_p),
                ('firmwareVersion', c_char_p),
                ('sensorType', c_char_p),
                ('sensorName', c_char_p),
                ('busy', c_int32),
                ('deviceVsxVersionMajor', c_int32),
                ('deviceVsxVersionMinor', c_int32),
                ('comPort', c_char_p),
                ('baudrate', c_int32),
                ('headAddress', c_char_p),
                ('isLoginNeeded', c_int32),
                ]
```

## 10.18 VsxProtocolDriver.Interface.VsxDeviceList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDeviceList:




### Static Protected Attributes

- list _fields_

### 10.18.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxDeviceList._fields_  [static], [protected]
```

**Initial value:**

```
=  [('length', c_int32),
               ('devices', POINTER(VsxDevice)),
               ]
```

## 10.19 VsxProtocolDriver.DataContainer.VsxDisparityDescriptor2 Class Reference

### Public Member Functions

- __init__ (self)

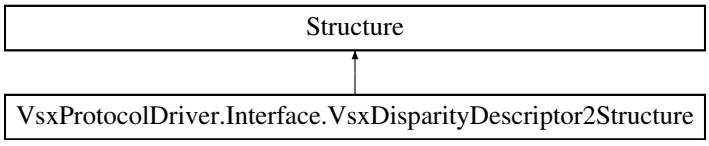
### 10.19.1 Constructor & Destructor Documentation

#### __init__()

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

## 10.20 VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure:



### Static Protected Attributes

- list _fields_

### 10.20.1 Member Data Documentation

#### _fields_

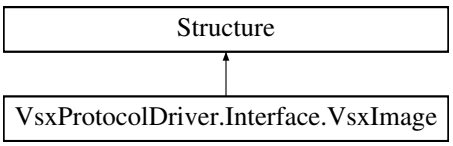
```
list VsxProtocolDriver.Interface.VsxDisparityDescriptor2Structure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('focalLength', c_double),
                ('principalPointU', c_double),
                ('principalPointV', c_double),
                ('baseline', c_double),
                ('offsetLeftRectifiedToDisparityU', c_double),
                ('offsetLeftRectifiedToDisparityV', c_double),
                ]
```

## 10.21 VsxProtocolDriver.Interface.VsxImage Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxImage:




### Static Protected Attributes

- list _fields_

### 10.21.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxImage._fields_  [static], [protected]
```

**Initial value:**

```
=  [('raw_data', c_void_p),
                ('format', c_int32),
                ('width', c_int32),
                ('height', c_int32),
                ('line_pitch', c_int32),
                ('frame_counter', c_long),
                ('coordinate_scale', c_double),
                ('coordinate_offset', c_double),
                ('axis_min', c_double),
                ('axis_max', c_double),
                ('invalid_data_value', c_double),
                ]
```

## 10.22 VsxProtocolDriver.DataContainer.VsxLineCoordinate Class Reference

### Public Member Functions

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

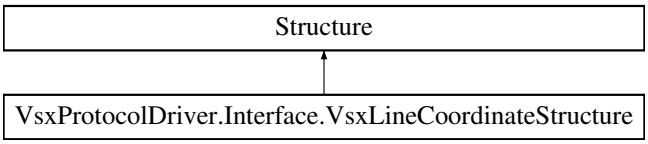
### 10.22.1 Constructor & Destructor Documentation

#### __init__()

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

## 10.23 VsxProtocolDriver.Interface.VsxLineCoordinateStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxLineCoordinateStructure:




### Static Protected Attributes

- list _fields_

### 10.23.1 Member Data Documentation

#### _fields_

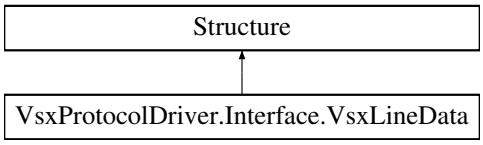
```
list VsxProtocolDriver.Interface.VsxLineCoordinateStructure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('c', c_float),
               ('x', c_float),
               ('y', c_float),
               ('z', c_float),
               ('q', c_float),
               ('i', c_float),
               ]
```

## 10.24 VsxProtocolDriver.Interface.VsxLineData Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxLineData:




### Static Protected Attributes

- list _fields_

### 10.24.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxLineData._fields_  [static], [protected]
```

**Initial value:**

```
=  [('lines', c_void_p),
               ('format', c_uint16),
               ('width', c_uint16),
               ('countLines', c_uint16),
               ('frameCounter', c_uint16),
               ('minX', c_float),
               ('maxX', c_float),
               ('minZ', c_float),
               ('maxZ', c_float),
               ]
```

## 10.25 VsxProtocolDriver.DataContainer.VsxOlr2CaptureInformation Class Reference

### Public Member Functions

- __init__ (self)

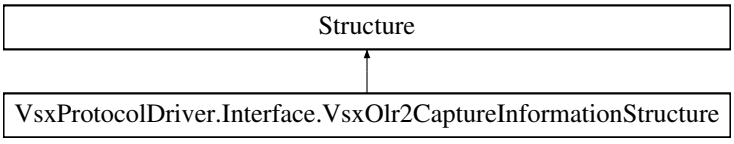
### 10.25.1 Constructor & Destructor Documentation

#### __init__()

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

## 10.26 VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure:



#### Static Protected Attributes

- list _fields_

### 10.26.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxOlr2CaptureInformationStructure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('frameCounter', c_uint64),
                ('triggerCounter', c_uint64),
                ('currentPosition', c_double),
                ('ioState', c_uint64),
                ('timestamp', c_uint64),
                ('lmaExposureTime1', c_uint32),
                ('lmaExposureTime2', c_uint32),
                ('lmbExposureTime1', c_uint32),
                ('lmbExposureTime2', c_uint32),
                ('lmaRoiOffsetX', c_uint16),
                ('lmaRoiLengthX', c_uint16),
                ('lmaRoiOffsetZ', c_uint16),
                ('lmaRoiLengthZ', c_uint16),
                ('lmbRoiOffsetX', c_uint16),
                ('lmbRoiLengthX', c_uint16),
                ('lmbRoiOffsetZ', c_uint16),
                ('lmbRoiLengthZ', c_uint16),
                ('autoTriggerFrameRate', c_uint16),
                ('triggerSource', c_uint8),
                ]
```

## 10.27 VsxProtocolDriver.DataContainer.VsxOlr2ModbusData Class Reference

#### Public Member Functions

- __init__ (self)

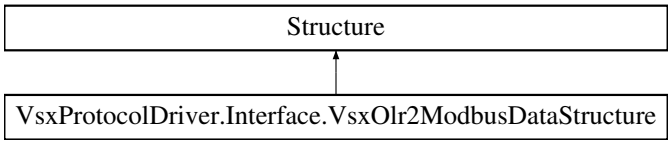
### 10.27.1 Constructor & Destructor Documentation

#### __init__()

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

## 10.28 VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure:



### Static Protected Attributes

- list _fields_

### 10.28.1 Member Data Documentation

#### _fields_

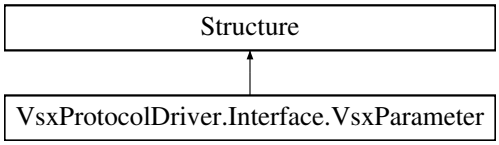
```
list VsxProtocolDriver.Interface.VsxOlr2ModbusDataStructure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('activationTimer', c_uint16),
                ('compareBuffer', c_uint16),
                ('targetPosition', c_uint16),
                ('robotData', c_uint16 * 13),
                ]
```

## 10.29 VsxProtocolDriver.Interface.VsxParameter Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameter:




### Static Protected Attributes

- list _fields_

### 10.29.1 Member Data Documentation

#### _fields_

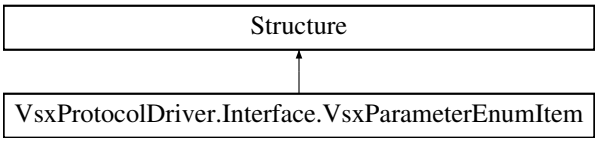
```
list VsxProtocolDriver.Interface.VsxParameter._fields_  [static], [protected]
```

**Initial value:**

```
=  [('valueString', c_char_p),
                ('valueInt', c_int32),
                ('valueDouble', c_double),
                ('valueType', c_int32),
                ('name', c_char_p),
                ('parameterId', c_char_p),
                ('configId', c_char_p),
                ('configVersion', c_int32),
                ('settingsVersion', c_int32),
                ('enumItemListLength', c_int32),
                ('enumItemList', POINTER(VsxParameterEnumItem))
                ]
```

## 10.30 VsxProtocolDriver.Interface.VsxParameterEnumItem Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameterEnumItem:




### Static Protected Attributes

- list _fields_

### 10.30.1 Member Data Documentation

#### _fields_

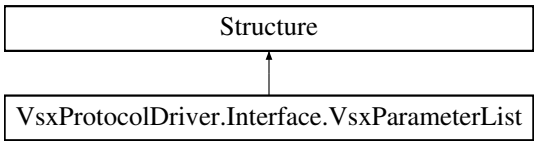
```
list VsxProtocolDriver.Interface.VsxParameterEnumItem._fields_  [static], [protected]
```

**Initial value:**

```
=  [('id', c_char_p),
               ('name', c_char_p)]
```

## 10.31 VsxProtocolDriver.Interface.VsxParameterList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxParameterList:




### Public Member Functions

- __init__ (self, length)

### Public Attributes

- length
- parameters

### Static Protected Attributes

- list _fields_

### 10.31.1 Constructor & Destructor Documentation

#### __init__()

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

### 10.31.2 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxParameterList._fields_  [static], [protected]
```

**Initial value:**

```
=  [('length', c_int32),
              ('parameters', POINTER(VsxParameter)),
              ]
```

#### length

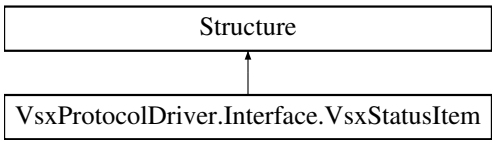
```
VsxProtocolDriver.Interface.VsxParameterList.length
```

#### parameters

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

## 10.32 VsxProtocolDriver.Interface.VsxStatusItem Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxStatusItem:




#### Static Protected Attributes

- list _fields_

### 10.32.1 Member Data Documentation

#### _fields_

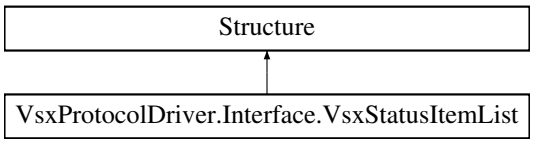
```
list VsxProtocolDriver.Interface.VsxStatusItem._fields_  [static], [protected]
```

**Initial value:**

```
=  [('valueString', c_char_p),
                ('valueInt', c_int32),
                ('valueDouble', c_double),
                ('valueType', c_int32),
                ('name', c_char_p),
                ('statusItemId', c_char_p),
                ('configurationClass', c_char_p),
                ('configVersion', c_int32),
                ('settingsVersion', c_int32),
                ('time', c_uint64),
                ('sensorTime', c_uint64),
                ]
```

## 10.33 VsxProtocolDriver.Interface.VsxStatusItemList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxStatusItemList:




**Static Protected Attributes**

- list _fields_

### 10.33.1 Member Data Documentation

#### _fields_

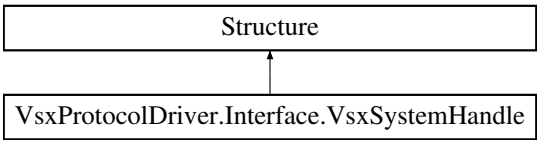
```
list VsxProtocolDriver.Interface.VsxStatusItemList._fields_  [static], [protected]
```

**Initial value:**

```
=  [('length', c_int32),
                ('statusItems', POINTER(VsxStatusItem)),
                ]
```

## 10.34 VsxProtocolDriver.Interface.VsxSystemHandle Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxSystemHandle:



### Public Member Functions

- __repr__ (self)

### Public Attributes

- handle

### Static Protected Attributes

- list _fields_ = [('handle', c_int32)]

#### 10.34.1 Member Function Documentation

#### __repr__()

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

#### 10.34.2 Member Data Documentation

#### _fields_

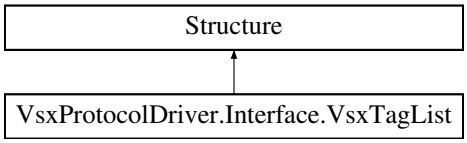
```
list VsxProtocolDriver.Interface.VsxSystemHandle._fields_ = [('handle', c_int32)]  [static],
[protected]
```

#### handle

```
VsxProtocolDriver.Interface.VsxSystemHandle.handle
```

## 10.35 VsxProtocolDriver.Interface.VsxTagList Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxTagList:




### Static Protected Attributes

- list _fields_

### 10.35.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxTagList._fields_  [static], [protected]
```

**Initial value:**

```
=  [('length', c_int32),
                ('tags', POINTER(c_char_p)),
                ]
```

## 10.36 VsxProtocolDriver.DataContainer.VsxTransformation Class Reference

Used to transform raw point cloud data from device.

### Public Member Functions

- __init__ (self)

### 10.36.1 Detailed Description

Used to transform raw point cloud data from device.

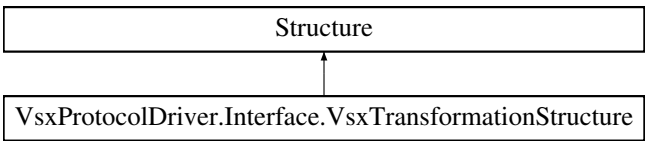
### 10.36.2 Constructor & Destructor Documentation

#### __init__()

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

## 10.37 VsxProtocolDriver.Interface.VsxTransformationStructure Class Reference

Inheritance diagram for VsxProtocolDriver.Interface.VsxTransformationStructure:




### Static Protected Attributes

- list _fields_

### 10.37.1 Member Data Documentation

#### _fields_

```
list VsxProtocolDriver.Interface.VsxTransformationStructure._fields_  [static], [protected]
```

**Initial value:**

```
=  [('translationTX', c_double),
                ('translationTY', c_double),
                ('translationTZ', c_double),
                ('quaternionQ0', c_double),
                ('quaternionQ1', c_double),
                ('quaternionQ2', c_double),
                ('quaternionQ3', c_double),
                ]
```

# Index