VsxProtocolDriver

Allgemein

Der Treiber VsxProtocolDriver bietet den vollen Zugriff auf die Ein- und Ausgangsdaten des Sensors und ermöglicht die Einbindung in eine C#-basierte Programmierumgebung. Hierzu stellt der Treiber eine Verbindung mit dem Sensor her und übernimmt die Kommunikation entsprechend dem Kommunikationsprotokoll. Dem Benutzer werden Funktionen zur Verfügung gestellt, mit denen Parameter auf dem Sensor eingestellt, Parameterwerte vom Sensor abgefragt und ganze Parametersätze sowohl lokal als auch auf dem Sensor gespeichert und geladen werden können. Weiterhin können Sensorbilder empfangen werden. Jede Funktion enthält darüber hinaus ein Error-Objekt, welchem im Falle eines Fehlers der Funktion Informationen entnommen werden können.

Der Treiber ist in C# implementiert und benötigt als Voraussetzung .NET 5.0 oder höher.

Die Funktionen des Treibers können **synchron** oder **asynchron** verwendet werden. Hierfür muss jeweils die gewünschte Instanz unter Verwendung der Init-Funktion der jeweiligen Klassen erstellt werden. Die Klasse VsxProtocolDriver stellt den asynchronen Treiber zur Verfügung. Die Klasse VsxProtocolDriverSync stellt das synchrone Interface zur Verfügung.

In diesem Handbuch werden aus Gründen der Übersichtlichkeit nur die wichtigsten Funktionen und Variablen beschrieben. Die SDK enthält zudem weitere Funktionen, die für andere Pepperl+Fuchs-Vision-Sensoren verwendet werden. Diese Funktionen werden im zugehörigen Produkthandbuch beschrieben. Bei einigen Funktionen gibt es mehrere Deklarationsmöglichkeiten. Im Nachfolgenden werden die präferierten Funktionen fett markiert.

Hinweis!

Einbinden des NuGet

Um die SDK verwenden zu können, muss das NuGet eingebunden werden. In Visual Studio kann dies z.B. durch den NuGet-Pakte-Managers durchgeführt werden. Die SDK findet Sie auf der Produktseite des entsprechenden Sensors von Pepperl+Fuchs im Softwareordner. In der dort abgelegten ZIP-Datei befindet sich das NuGet unter dem Projektordner ext.

Synchrone und asynchrone Funktionen

Die in den Parametern verwendeten Hilfsklassen werden im Anschluss beschrieben.

Statische Funktionen

UDP-Broadcast

Gültig für:
- SmartRunner
- SmartRunner 3-D

Die statische Funktion gibt eine Liste der Vsx-Geräte zurück, welche über einen UDP-Broadcast im Netzwerk gefunden werden.

Asynchrone Funktion

```
public static Task<(bool Succ, List<Device> DeviceList, Error ErrorDesc)> UdpDeviceList()
```

Synchrone Funktion

```
public static (bool Succ, List<Device> DeviceList, Error ErrorDesc) UdpDeviceList()
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Netzwerkeinstellungen über UDP

Gültig für:
- SmartRunner
- SmartRunner 3-D

Netzwerkeinstellungen auf dem Gerät werden über UDP geändert.

Asynchrone Funktion

```
public static Task<(bool Succ, Error ErrorDesc)> SetNetworkSettingsViaUdp(string macAddress,
string ipAddress, string networkMask, string gateway)
```

Synchrone Funktion

```
public static (bool Succ, Error ErrorDesc) SetNetworkSettingsViaUdp(string macAddress, string
ipAddress, string networkMask, string gateway)
```

Mögliche Error-IDs: keine

TCP/IP-Verbindung

Gültig für:
- SmartRunner
- SmartRunner 3-D

Initialisiert eine neue Treiberinstanz, mittels derer über TCP/IP mit dem Gerät kommuniziert werden kann. Während die IP-Adresse angegeben werden muss, kann der Standard VSXPORT = 50005 verwendet werden.

Asynchrone Funktion

```
public static VsxProtocolDriver Init(string ipAddress, int port = VSXPORT, string pluginName = "")
```

Synchrone Funktion

```
public static VsxProtocolDriverSync Init(string ipAddress, int port =VSXPORT, string pluginName = "")
```

IVsxMessage speichern

Gültig für:
- SmartRunner
- SmartRunner 3-D

Eine IVsxMessage ist eine allgemeine Schnittstelle für Daten, die zwischen PC und Sensor ausgetauscht werden.

Asynchrone Funktion

```
public static (bool Succ, Error ErrorDesc) SaveData(string filename, IVsxMessage message)
```

Synchrone Funktion

```
public static (bool Succ, Error ErrorDesc) SaveData(string filename, IVsxMessage message)
```

Mögliche Error-IDs: VSX_DRIVER_DATA_ERROR, VSX_DRIVER_INVALID_DATA_ERROR, VSX_DRIVER_SAVE_FILE_ERROR

Punktwolkedatei speichern

Gültig für:
- SmartRunner 3-D

Speichert eine Punktwolkedatei bestehend aus den Ebenen x, y und z unter dem angegebenen Dateinamen ab.

Asynchrone Funktion

```
public static (bool Succ, Error ErrorDesc) Save3DPointCloudData(string filename, VsxImageData2Message x, VsxImageData2Message y, VsxImageData2Message z)
```

Synchrone Funktion

```
public static (bool Succ, Error ErrorDesc) Save3DPointCloudData(string filename, VsxImageData2Message x, VsxImageData2Message y, VsxImageData2Message z)
```

Mögliche Error-Ids: VSX_DRIVER_DATA_ERROR, VSX_DRIVER_SAVE_FILE_ERROR

Nicht statische Funktionen

Verbindung zum Gerät herstellen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Stellt eine Verbindung zum Gerät her, unter Verwendung der mittels Init gesetzten Parameter. Als Timeout für das Öffnen der Verbindung kann der CONNECTION_TIMEOUT_MS = 1000 verwendet werden. Um nicht statische Funktionen zu verwenden, muss eine Verbindung zum Gerät bestehen.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> Connect(int timeout = CONNECTION_TIMEOUT_MS)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) Connect(int timeout =
VsxProtocolDriver.CONNECTION_TIMEOUT_MS)
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Verbindung zum Gerät trennen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Verbindung zum Gerät wird getrennt.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> Disconnect()
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) Disconnect()
```

Mögliche Error-IDs: Keine

Wiederverbindung mit übergebenden Parametern

Gültig für:
- SmartRunner
- SmartRunner 3-D

Trennt die Verbindung zum Gerät und stellt sie unter Verwendung der übergebenen Parameter erneut her. Diese Funktionen können verwendet werden, wenn zur Laufzeit die Verbindungsparameter geändert werden müssen.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> ReConnect(string ipAddress, int port = VSXPORT)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) ReConnect(string ipAddress, int port = VSXPORT)
```

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> ReConnect(string serialPort, int baudrate,
TheSensor.ConnectionType connectionType)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) ReConnect(string serialPort, int baudrate,
TheSensor.ConnectionType connectionType)
```

Mögliche Error-Ids: VSX_DRIVER_CONNECTION_ERROR

Geräteinformationen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Übermittelt Informationen über das Gerät (wie u.a. MAC-Adresse usw.).
Asynchrone Funktion
```
public Task<(bool Succ, Device CurrentDevice, Error ErrorDesc)> GetCurrentDeviceInformation()
```

Synchrone Funktion
```
public (bool Succ, Device CurrentDevice, Error ErrorDesc) GetCurrentDeviceInformation()
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Featureliste

Gültig für:
- SmartRunner
- SmartRunner 3-D

Übermittelt die Liste der auf dem Gerät verfügbaren Features.
Asynchrone Funktion
```
public Task<(bool Succ, float XmlVersion, Hashtable FeatureList, Error ErrorDesc)>
GetFeatureList()
```

Synchrone Funktion
```
public (bool Succ, float XmlVersion, Hashtable FeatureList, Error ErrorDesc) GetFeatureList()
```

Mögliche Error-Ids: VSX_DRIVER_CONNECTION_ERROR

Parameterliste inkl. detallierte Informationen und aktuelle Werte

Gültig für:
- SmartRunner
- SmartRunner 3-D

Übermittelt eine Liste aller auf dem Gerät verfügbaren Parameter inklusive detaillierter Informationen und ihrer aktuellen Werte.
Asynchrone Funktion
```
public Task<(bool Succ, List<Parameter> ParameterList, Error ErrorDesc)> GetParameterList()
```

Synchrone Funktion
```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc) GetParameterList()
```

Mögliche Error-Ids: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX_DRIVER_CONNECTION_ERROR

Wert eines einzelnen Geräteparameters übermitteln

Gültig für:
- SmartRunner
- SmartRunner 3-D

Übermittelt den Wert eines einzelnen Geräteparameters.
Asynchrone Funktion
```
public Task<(bool Succ, object parameterValue, Error ErrorDesc)>
GetSingleParameterValue(Parameter parameter)

public Task<(bool Succ, object parameterValue, Error ErrorDesc)>
GetSingleParameterValue(string parameterId)

public Task<(bool Succ, object parameterValue, Error ErrorDesc)>
GetSingleParameterValue(ushort settingsVersion, ushort configVersion, string configId, string
parameterId)
```

Synchrone Funktion

```
public (bool Succ, object parameterValue, Error ErrorDesc) GetSingleParameterValue(Parameter parameter)

public (bool Succ, object parameterValue, Error ErrorDesc) GetSingleParameterValue(string parameterId)

public (bool Succ, object parameterValue, Error ErrorDesc) GetSingleParameterValue(ushort settingsVersion, ushort configVersion, string configId, string parameterId)
```

Mögliche Error-Ids: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DATA_ERROR

Wert eines einzelnen Geräteparameters setzen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Setzt den Wert eines einzelnen Parameters auf dem Gerät. Der Parameter wird durch die Übergabe der Funktionsparameter analog zur Funktion *GetSingleParameterValue* definiert. Zusätzlich wird der gewünschte Wert übergeben. Abhängig vom Wert des Parameters, der über diese Funktion gesetzt wird, ist es auch möglich, dass andere Parameter auf dem Gerät einen neuen Wert erhalten. Ist dies der Fall, so wird eine Liste mit diesen Parametern und ihrem neuen Wert über die *DependendParameters List* zurückgegeben. Ist dies nicht der Fall, so ist die Liste leer.

Asynchrone Funktion

```
public Task<(bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)>
SetSingleParameterValue(Parameter parameter, object value)

public Task<(bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)>
SetSingleParameterValue(string parameterId, object value)

public Task<(bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)>
SetSingleParameterValue(ushort settingsVersion, ushort configVersion, string configId, string parameterId, object value)
```

Synchrone Funktion

```
public (bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)
SetSingleParameterValue(Parameter parameter, object value)

public (bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)
SetSingleParameterValue(string parameterId, object value)

public (bool Succ, List<Parameter> DependendParameters, Error ErrorDesc)
SetSingleParameterValue(ushort settingsVersion, ushort configVersion, string configId, string parameterId, object value)
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DATA_ERROR

Netzwerkeinstellungen ändern

Gültig für:
- SmartRunner
- SmartRunner 3-D

Ändert die Netzwerkeinstellungen auf dem Gerät. Anschließend ist die Verbindung zum Gerät getrennt und muss mittels der "Connect"-Funktion (siehe "Verbindung zum Gerät herstellen" auf Seite 3) neu aufgebaut werden.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> SetNetworkSettings(string ipAddress, string networkMask, string gateway)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) SetNetworkSettings(string ipAddress, string networkMask, string gateway)
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Firmwaredatei an das Gerät senden

Gültig für:
- SmartRunner
- SmartRunner 3-D

Sendet die unter dem angegebenen Pfad und Dateinamen liegende Firmwaredatei zum Gerät. Während des laufenden Updates kann über den *FirmwareStateChannelReader* der aktuelle Status ausgelesen werden.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> SendFirmware(string fileName)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) SendFirmware(string fileName)
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DEVICE_ERROR

Parametersatz auslesen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Liest den aktuellen Parametersatz vom Gerät und speichert ihn unter dem angegebenen Pfad und Dateinamen ab.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> DownloadParameterSet(string destinationFileName)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) DownloadParameterSet(string destinationFileName)
```

Mögliche Error-IDs: VSX_DRIVER_SAVE_FILE_ERROR

Parametersatz laden

Gültig für:
- SmartRunner
- SmartRunner 3-D

Lädt einen unter dem angegebenen Pfad und Dateinamen gespeicherten Parametersatz und sendet ihn zum Gerät.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> UploadParameterSet(string sourceFileName)
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) UploadParameterSet(string sourceFileName)
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Parametersatz speichern

Gültig für:
- SmartRunner
- SmartRunner 3-D

Speichert die aktuellen Parametereinstellungen auf dem Gerät ab. Die eingestellten Werte werden bei jedem Gerätestart gesetzt.

Alle Parameterdaten werden zunächst flüchtig auf dem Gerät gespeichert und können durch einen Neustart oder durch *LoadParameterSetOnDevice* zurückgesetz werden. Erst durch *SaveParameterSetOnDevice* werden die Parameter dauerthaft gespeichert.

Asynchrone Funktion

```
public Task<(bool Succ, Error ErrorDesc)> SaveParameterSetOnDevice()
```

Synchrone Funktion

```
public (bool Succ, Error ErrorDesc) SaveParameterSetOnDevice()
```

Mögliche Error-IDs: VSX_DRIVER_CONNECTION_ERROR

Parametereinstellungen laden

Gültig für:
- SmartRunner
- SmartRunner 3-D

Lädt die mit *SaveParameterSetOnDevice* gespeicherten Parametereinstellungen auf das Gerät. Die Parameter haben dann die zuvor gespeicherten Werte. Eine aktuelle Parameterliste wird übermittelt.

Asynchrone Funktion

```
public Task<(bool Succ, List<Parameter> ParameterList, Error ErrorDesc)>
LoadParameterSetOnDevice()
```

Synchrone Funktion

```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc) LoadParameterSetOnDevice()
```

Mögliche Error-IDs: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX_DRIVER_CONNECTION_ERROR

Werkseinstellungen laden

Gültig für:
- SmartRunner
- SmartRunner 3-D

Die Werkseinstellungen aller Parameter werden auf dem Gerät geladen. Eine aktuelle Parameterliste wird übermittelt.

Asynchrone Funktion

```
public Task<(bool Succ, List<Parameter> ParameterList, Error ErrorDesc)>
LoadDefaultParameterSetFromDevice()
```

Synchrone Funktion

```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc)
LoadDefaultParameterSetFromDevice()
```

Mögliche Error-IDs: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX_DRIVER_CONNECTION_ERROR

Auslesen von Sensordaten

Gültig für:
- SmartRunner
- SmartRunner 3-D

Startet das Auslesen der Sensordaten. Die Sensordaten werden pro Trigger in *VsxDynamicContainer* gepackt. Diese können mit der Funktion *GetDynamicContainer* abgerufen werden. Die *bufferSize* gibt an, wie viele Container vom Treiber zwischengespeichert werden können, die *startCondition* ab welchem Container zwischengespeichert werden soll und die *strategy* was passieren soll, wenn der Zwischenspeicher voll ist. Bei DROP_OLDEST wird dann der älteste gespeicherte Container verworfen, bei DROP_WRITE der aktuell angekommene Container.

Asynchrone Funktion

```
public void ResetDynamicContainerGrabber(int bufferSize,
PF.Foundation.Communication.Vsx.Strategy strategy =
PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Synchrone Funktion

```
public void ResetDynamicContainerGrabber(int bufferSize,
PF.Foundation.Communication.Vsx.Strategy strategy =
PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Mögliche Error-IDs: Keine

Ausgabe des ältesten zwischengespeicherten DynamicContainers

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt den ältesten zwischengespeicherten *DynamicContainer* (siehe "Auslesen von Sensordaten" auf Seite 7) zurück. Diese Funktion gibt nach *timeoutMs* Millisekunden einen Fehler zurück, wenn bis dahin kein neuer DynamicContainer verfügbar ist.

Asynchrone Funktion

```
public Task<(bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, Error
ErrorDesc)> GetDynamicContainer(int timeoutMs = System.Threading.Timeout.Inifinite)
```

Synchrone Funktion

```
public (bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, Error
ErrorDesc) GetDynamicContainer(int timeoutMs = Timeout.Infinite)
```

Mögliche Error-IDs: VSX_DRIVER_INIT_ERROR, VSX_DRIVER_TIMEOUT_ERROR

Auslesen von Logdaten

Gültig für:
- SmartRunner
- SmartRunner 3-D

Startet das Auslesen von Logdaten neu. Die Logdaten können mit der Funktion *GetLogMessage* abgerufen werden. Die *bufferSize* gibt an, wie viele Logdatenmessages vom Treiber zwischengespeichert werden können. Die *typeMask* gibt an, welche Logdatentypen vom Gerät übertragen werden sollen und die *strategy* gibt an was passieren soll, wenn der Zwischenspeicher voll ist. Bei DROP_OLDEST wird die älteste gespeicherte Logmessage verworfen und bei DROP_WRITE die aktuell angekommene Logmessage.

Asynchrone Funktion

```
public void ResetLogMessageGrabber(int bufferSize, int typeMask,
PF.Foundation.Communication.Vsx.Strategy strategy =
PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Synchrone Funktion

```
public void ResetLogMessageGrabber(int bufferSize, int typeMask,
PF.Foundation.Communication.Vsx.Strategy strategy =
PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Mögliche Error-IDs: Keine

Ausgabe der ältesten zwischengespeicherten LogMessage

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt die ältesten zwischengespeicherte *LogMessage* (siehe "Auslesen von Logdaten" auf Seite 8) zurück. Diese Funktion gibt nach *timeoutMs* Millisekunden einen Fehler zurück, wenn bis dahin keine neue Logmessage verfügbar ist.

Asynchrone Funktion

```
public Task<(bool Succ, VsxLogMessage LogMessage, int NumberOfDiscardedItems,
Error ErrorDesc)> GetLogMessage(int timeoutMs = System.Threading.Timeout.Inifinite)
```

Synchrone Funktion

```
public (bool Succ, VsxLogMessage LogMessage, int NumberOfDiscardedItems, Error ErrorDesc)
GetLogMessage(int timeoutMs = Timeout.Infinite)
```

Mögliche Error-IDs: VSX_DRIVER_INIT_ERROR, VSX_DRIVER_TIMEOUT_ERROR

Properties

Informationen zum Firmwareupdate

Gültig für:
- SmartRunner
- SmartRunner 3-D

Der *FirmwareStateChannelReader* liefert Informationen über den aktuellen Status des Firmwareupdates.

```
public ChannelReader<FirmwareState> FirmwareStateChannelReader
```

Verbindungsstatus

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt den Verbindungsstatus an.

```
public bool Connected { get; }
```

Timeout

Gültig für:
- SmartRunner
- SmartRunner 3-D

Timeout gibt an (in ms), wie lange bei einer Anfrage an das Gerät auf dessen Antwort gewartet wird. Der Standardwert ist DEFAULT_ETHERNET_TIMEOUT_MS bzw. DEFAULT_SERIAL_TIMEOUT_MS je nach Verbindungsart.

```
public int WaitTimeout { get; set; }
```

Anzahl verworfener DynamicContainer bzw. LogMessages

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt an, wie viele DynamicContainer bzw. LogMessages seit dem letzten Reset-Grabber verworfen wurden, nachdem kein Platz mehr im Zwischenspeicher war.

DynamicContainer

```
public int MissingContainerFramesCounter { get; }
```

LogMessages

```
public int MissingLogMessagesCounter { get; }
```

Anzahl zwischengespeicherter DynamicContainer bzw. LogMessages

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt an, wie viele Dynamic Container bzw. Log Messages vom Treiber zwischengespeichert werden können.

DynamicContainer

```
public int DynamicContainerQueueSize { get; }
```

LogMessages

```
public int LogMessageQueueSize { get; }
```

Welche DynamicContainer bzw. LogMessages werden verworfen

Gültig für:
- SmartRunner
- SmartRunner 3-D

Gibt an, welche DynamicContainer bzw. LogMessages verworfen werden sollen, sofern der Zwischenspeicher voll ist. DROP_OLDEST verwirft die älteste gespeicherte, DROP_WRITE verwirft die aktuell angekommene.

DynamicContainer

```
public Strategy DynamicContainerGrabberStrategy { get; }
```

LogMessages

```
public Strategy LogMessageGrabberStrategy { get; }
```

Events

Verbindungsverlust

Gültig für:
- SmartRunner
- SmartRunner 3-D

Das Event wird ausgelöst, sobald der Treiber einen Verbindungsverlust zum Gerät feststellt. Parameter sind die IP-Adresse des zuvor verbundenen Geräts und eine Meldung, warum die Verbindung verloren gegangen ist. Dieses Event wird nur ausgelöst, wenn eine TCP/IP-Verbindung verwendet wird.

```
public event Action<string, string> OnDisconnect
```

Hilfsklassen

Geräteinformationen (PF.Types.Device)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet Informationen über das aktuell verbundene Gerät.

```
public string PhysicalAddress;

public int PhysicalPort;

public string IpAddress;

public string NetworkMask;

public string Gateway;

public string MacAddress;

public string Identifier;

public string FirmwareVersion;

public string SensorType;
```

Fehlerinformationen (PF.Types.Error)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet Informationen über einen aufgetretenen Fehler.

```
public ErrorId Id;

public string Tag;

public string Message;
```

Folgende Error-IDs sind möglich

Gültig für:
- SmartRunner
- SmartRunner 3-D

```
VSX_DRIVER_NO_ERROR = 0x0,

VSX_DRIVER_INIT_ERROR = -0x1,

VSX_DRIVER_TIMEOUT_ERROR = -0x2,

VSX_DRIVER_SAVE_FILE_ERROR = -0x3,

VSX_DRIVER_DATA_ERROR = -0x4,

VSX_DRIVER_CONNECTION_ERROR = -0x5,

VSX_DRIVER_INVALID_DATA_ERROR = -0x6,

VSX_DRIVER_DEVICE_ERROR = -0x7,

VSX_DRIVER_GENERAL_ERROR = -0x1000
```

Parameter (PF.VsxDriver.Parameter)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet Informationen über einen Geräteparameter. Wichtige Eigenschaften sind hier die Informationen über Version und IDs, welche für das Setzen eines Parameters benötigt werden. Eine weitere Eigenschaft ist mit Value der aktuelle Parameterwert. Nicht jede Eigenschaft wird bei jedem Parameter verwendet.

```
ushort settingsVersion;

ushort configVersion;

string configId;

string parameterId;

string name;

Vsx.ParameterTypes type;

Vsx.ValueTypes valueType;

bool enable;

bool visible;

object min;

object max;

object step;

string userLevel;

object value;

object defaultValue;

string unit;

List<ItemTuple> items;
```

Informationen zum Firmwareupdate (PF.Foundation.Communication.Vsx.Device.FirmwareState)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet Informationen über den aktuellen Status eines laufenden Firmwareupdates.

```
public int Id;

public string Tag;

public string Message;
```

Liste der IVsxMessage (PF.Foundation.Communication.Vsx.Message.VsxDynamicContainerMessage : IVsxMessage)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet eine Liste von IVsxMessages, welche wiederum gesendete Daten vom Gerät beinhalten. Die enthaltenen Messages werden in der Liste mittels eines String identifiziert. Die möglichen Messages sind gerätespezifisch. Beispiel zum VsxDynamic-ContainerMessage.

```
public bool ContainsMessage(string tag)

public IVsxMessage GetMessage(string tag)
```

Bilddaten (PF.Foundation.Communication.Vsx.Message.VsxImageData2Message : IVsxMessage)

Gültig für:
- SmartRunner
- SmartRunner 3-D

Beinhaltet Bilddaten eines speziellen Bildes. Abhängig davon, ob die einzelnen Bildwerte bytes oder floats sind, sind diese im jeweiligen Array ImageData oder ImageDataFloats gespeichert.

```
public ImageData2Format;

public int Width;

public int Height;

public int LinePitch;

public long FrameCounter;

public double CoordinateScale;

public double CoordinateOffset;

public double AxisMin;

public double AxisMax;

public double InvalidDataValue;

public byte[] ImageData;

public float[] ImageDataFloats;
```

PF.Foundation.Communication.Vsx.Message.VsxDisparityDescriptorMessage : IVsxMessage

Gültig für:
- SmartRunner
- SmartRunner 3-D

```
public double FocalLength;

public double PrincipalPointU;

public double PrincipalPointV;

public double Baseline;
```

PF.Foundation.Communication.Vsx.Message.VsxTransformationMessage : IVsxMessage

Gültig für:
- SmartRunner
- SmartRunner 3-D

```
public double TranslationTX;

public double TranslationTY;

public double TranslationTZ;

public double QuaternionQ0;

public double QuaternionQ1;

public double QuaternionQ2;

public double QuaternionQ3;
```

SmartRunner

DynamicContainer

Ein vom Sensor empfangener DynamicContainer kann die folgenden Messages enthalten:

TagName	Typ	Status	Beschreibung
"Image"	ImageData[Mono8]	optional	Bilddaten
"Line"	LineMessage	optional	Liniendaten

Aufbau des Typs LineMessage

Haupteigenschaft

Line

Gibt eine Punktliste zurück. Jeder Punkt besteht aus Bild-, Weltkoordinaten und Qualitätsinformationen. Die Punkte sind vom Typ *Coordinate*. Im Detail sieht die Punktinformation wie folgt aus:

- ContentValue: Reserviert für zukünftige Verwendung
- ImageCoordinate: Bildkoordinaten in Subpixeln
- Intensity: Wert der mit dem Punktgrauwert korreliert
- LineId: Immer 0 in Einzelliniensystemen
- Quality: Punktqualität in Prozent
- SegmentId: Wenn unterstützt Liniensegmentnummer ansonsten 0
- Valid: Wahr wenn Punkt gültig ist
- WoorldCoordinate: Weltkoorrdinaten in mm

Nebeneigenschaften (Metainformationen)

Metainformationen geben zusätzliche Informationen zu den Daten.

MinX

Minimal möglicher X-Wert der Gerätefamilie, üblicherweise negativ in µm.

MaxX

Maximaler möglicher X-Wert der Gerätefamilie, üblicherweise positiv in µm.

MinZ

Minimal möglicher Z-Wert der Gerätefamilie, positiv in µm.

MaxZ

Maximaler möglicher Z-Wert der Gerätefamilie, positiv in µm.

FrameCounter

Aktueller Bildzähler, zusammengehörige Daten haben den gleichen Zähler.

ScaleXYZ, ScaleC

Skalierungsfaktoren für die interne Berechnung der Liniendaten.

Width

Maximale Anzahl der möglichen Linienpunkte.

Format

Reserviert für zukünftige Verwendung.

Status

Reserviert für zukünftige Verwendung.

Lines

Gibt eine Liste von Linien zurück. In der Regel enthält die Liste nur eine Linie. Jede Linie besteht aus einer Liste von Punkten, die Punkte der ersten Linie können direkt über Line abgerufen werden.

SmartRunner 3-D Stereo

Ein vom Sensor empfangener DynamicContainer kann die folgenden Messages enthalten:

TagName	Typ	Status	Beschreibung
"LeftRaw"	ImageData2[Mono8]	optional	Linkes Rohbild
"RightRaw"	ImageData2[Mono8]	optional	Rechtes Rohbild
"LeftRectified"	ImageData2[Mono8]	optional	Linkes rektifiziertes Bild
"RightRectified"	ImageData2[Mono8]	optional	Rechtes rektifiziertes Bild
"DisparityC"	ImageData2[Coord3D_C16]	optional	Disparitätsbild (1)
"DisparityDesc"	DisparityDescriptor	optional	Erforderlich für Disparitätsbild (1)
"CalibratedA"[a]	ImageData2[Coord3D_A32f]	optional	X-Ebene des kalibrierten Rasters
"CalibratedB"[b]	ImageData2[Coord3D_B32f]	optional	Y-Ebene des kalibrierten Rasters
"CalibratedC"[c]	ImageData2[Coord3D_C32f]	optional	Z-Ebene des kalibrierten Rasters
"Transformation"	Transformation	optional	Umwandlung von XYZ
"Confidence"	ImageData2[Confidence8]	optional	Konfidenzkarte (2)

a. Die drei Layer "CalibratedA", "CalibratedB" und "CalibratedC" werden geräteseitig nicht gesendet. Die Funktion "ProcessDeviceData" (VsxDynamicContainerMessage container) erzeugt diese drei Messages und berechnet die jeweilige Ebene der "PointCloud".

b. Die drei Layer "CalibratedA", "CalibratedB" und "CalibratedC" werden geräteseitig nicht gesendet. Die Funktion "ProcessDeviceData" (VsxDynamicContainerMessage container) erzeugt diese drei Messages und berechnet die jeweilige Ebene der "PointCloud".

c. Die drei Layer "CalibratedA", "CalibratedB" und "CalibratedC" werden geräteseitig nicht gesendet. Die Funktion "ProcessDeviceData" (VsxDynamicContainerMessage container) erzeugt diese drei Messages und berechnet die jeweilige Ebene der "PointCloud".

SmartRunner 3-D Time-of-Flight

Ein vom Sensor empfangener DynamicContainer kann die folgenden Messages enthalten:

TagName	Typ	Status	Beschreibung
"Amplitude"	ImageData2[Mono16]	optional	Amplitude des Signals
"Depth"	ImageData2[Mono16]	optional	Tiefenkarte
"CalibratedA"	ImageData2[Coord3D_A16]	optional	X-Ebene des kalibrierten Rasters
"CalibratedB"	ImageData2[Coord3D_B16]	optional	Y-Ebene des kalibrierten Rasters
"CalibratedC"	ImageData2[Coord3D_C16]	optional	Z-Ebene des kalibrierten Rasters
"Transformation"	Transformation	optional	Umwandlung von XYZ
"Confidence"	ImageData2[Confidence8]	optional	Konfidenzkarte (1)