SpcInterfaceControl

(iID® reader library)

Product:	microsensys iID®3000 PRO RFID interfaces microsensys iID®4000 UHF RFID interfaces
Product Code:	**-**
Revision:	**1.0**
Date:	2021-03-18

API - Definition for Windows® PC, macOS and Linux

iID® reader library is a .NETStandard library implementation supporting microsensys RFID interfaces. This document describes the software API provided by this library.

This documentation describes the whole software API valid for many microsensys RFID interfaces supporting SPC communication mode.
A lot of applications are based on standard SPC RFID operations, where we recommend usage of *SPCInterfaceControl*.

Class name:	**iIDReaderLibrary.SpcInterfaceControl**
Tested devices:	MICROSOFT® Windows 7/64bit, 8.1/32bit/64bit, 10/64bit
Supported host interface:	RS232 serial, USB, Bluetooth SPP, LAN
Version:	1.0
Release date:	2021-03-18
Dependencies:	.NETStandard v2.0, System.IO.Ports v5.0
NuGet Package-ID:	Microsensys.iIDReaderLibrary.SpcInterfaceControl

NuGet-Link:
https://www.nuget.org/packages/Microsensys.iIDReaderLibrary.SpcInterfaceControl/

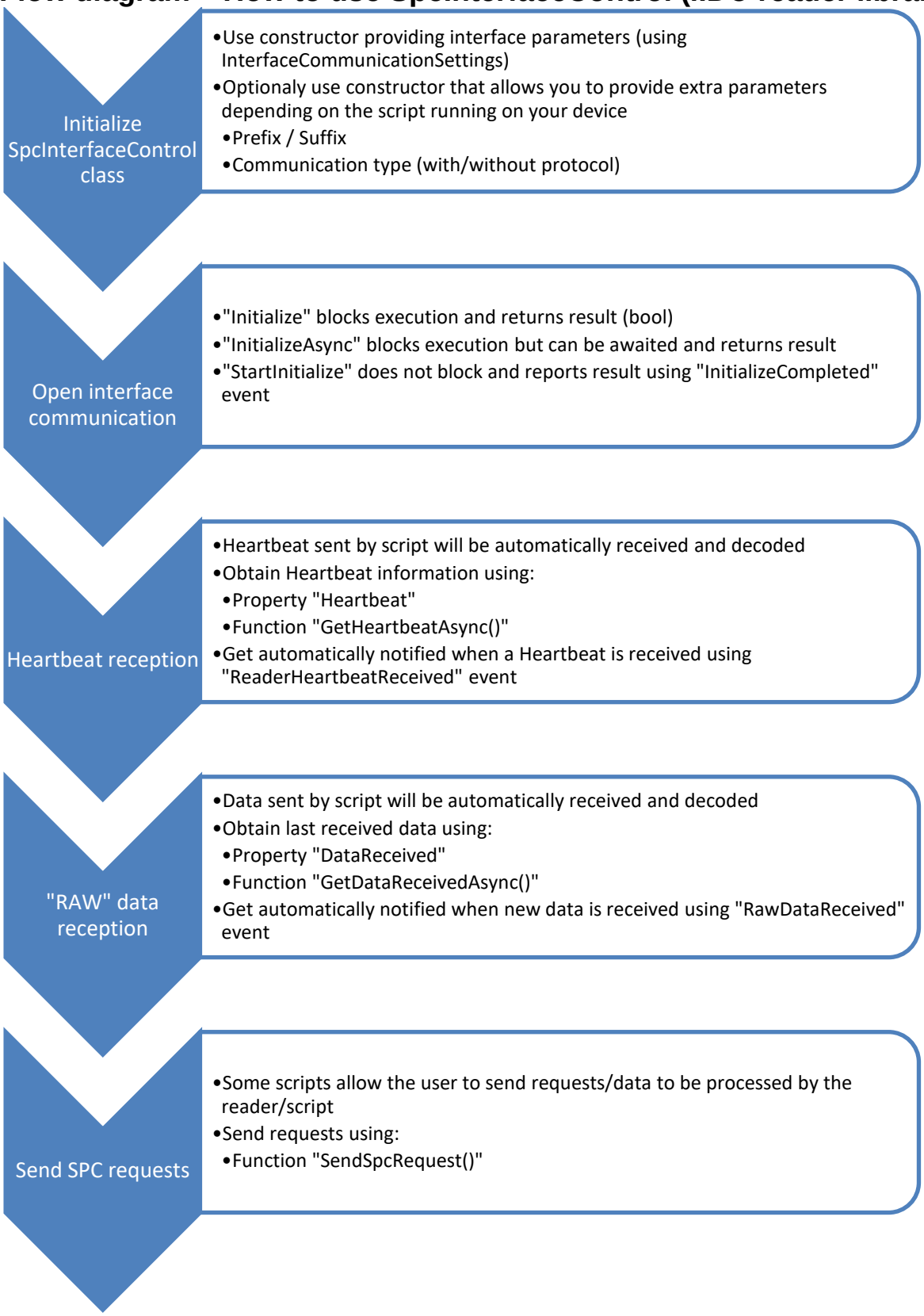
Content

Short history

This chapter includes a short history of modifications of *SpcInterfaceControl*.

Date	Reason	Modification	Release Date / Version	FileName
2021-03	- first release		1.0	official NuGet release

Flow diagram – How to use SpcInterfaceControl (iID® reader library)



Initialization

Constructors

This chapter includes all the available constructors, which are necessary for initialization of the class. All of them provide details on host-side peripheral interface to be used.

public SpcInterfaceControl (InterfaceCommunicationSettings _commSettings, string _dataPrefix, string _dataSuffix, bool _useProtocol)

Parameter :

_commSettings – instance of *InterfaceCommunicationSettings* containing host-side peripheral interface details (for example name of serial port, where RFID interface is connected)
_dataPrefix – Prefix inserted by script before each communication package.
_dataSuffix – Suffix appended by script at the end of each communication package.
_useProtocol – If true, SpcInterfaceControl will expect the received data to be provided using one of supported protocols and will use it to encode the SPC requests.

Result :

success – instance of *SpcInterfaceControl*
error - Exception

Description :

Initializes *SpcInterfaceControl* class using provided peripheral interface parameters, prefix, suffix and communication type.

public SpcInterfaceControl (InterfaceCommunicationSettings _commSettings, string _dataPrefix, string _dataSuffix)

Parameter :

_commSettings – instance of *InterfaceCommunicationSettings* containing host-side peripheral interface details (for example name of serial port, where RFID interface is connected)
_dataPrefix – Prefix inserted by script before each communication package.
_dataSuffix – Suffix appended by script at the end of each communication package.

Result :

success – instance of *SpcInterfaceControl*
error - Exception

Description :

Initializes *SpcInterfaceControl* class using provided peripheral interface parameters, prefix, suffix. Communication is performed without protocol.

public SpcInterfaceControl (InterfaceCommunicationSettings _commSettings, bool _useProtocol)

Parameter :

_commSettings – instance of *InterfaceCommunicationSettings* containing host-side peripheral interface details (for example name of serial port, where RFID interface is connected)
_useProtocol – If true, SpcInterfaceControl will expect the received data to be provided using one of supported protocols and will use it to encode the SPC requests.

Result :

success – instance of *SpcInterfaceControl*
error - Exception

Description :

Initializes *SpcInterfaceControl* class using provided peripheral interface parameters and communication type. Prefix is defined to ":", suffix to "@".

public SpcInterfaceControl (InterfaceCommunicationSettings _commSettings)

Parameter :

_commSettings – instance of *InterfaceCommunicationSettings* containing host-side peripheral interface details (for example name of serial port, where RFID interface is connected)

Result :

success – instance of *SpcInterfaceControl*
error - Exception

Description :

Initializes *SpcInterfaceControl* class using provided peripheral interface parameters. Prefix is defined to ":", suffix to "@". Communication is performed without protocol.

Initialization functions

This chapter includes all the functions to handle the interface communication initialization, as well as how to close the interface communication.

public bool Initialize ()

Parameter :
-
Result :
success – returns true after the communication interface is open and communication with reader is established.
error – returns “false”.
Description :
Initializes interface communication, opening the communication port and trying to communicate with the reader using the defined peripheral interface parameters, interface type and protocol type.

public async Task<bool> InitializeAsync ()

Parameter :
-
Result :
success – returns true after the communication interface is open and communication with reader is established.
error – returns “false”.
Description :
Asynchronous version of *Initialize()* function. This function may be awaited upon.

public void StartInitialize ()

Parameter :
-
Result :
-
Description :
Starts the initialization process in a separate thread. This function completes without blocking and waiting for the process result. It initializes interface communication, opening the communication port and trying to communicate with the reader using the defined peripheral interface parameters, interface type and protocol type. Once the process is finished, the result is provided using the *InitializeCompleted* event

Blocking functionality

Reader Heartbeat

This chapter describes how to get Heartbeat information sent by the script running in the RFID reader.

public ReaderHeartbeat Heartbeat

Accessors :
get / -
Description :
(Property) Returns an instance of *ReaderHeartbeat* representing heartbeat information last received from reader returns immediately with the result.

public async Task<ReaderHeartbeat> GetHeartbeatAsync ()

Parameter :
-
Result :
success – returns instance of *ReaderHeartbeat* representing heartbeat information received from reader.
error – *Exception*.
Description :
Blocks execution until a new Heartbeat is received from script. This function can be awaited.

public async Task<ReaderHeartbeat> GetHeartbeatAsync (CancellationToken _cancelToken)

Parameter :
_cancelToken – instance of CancellationToken that may be used to cancel asynchronous call.
Result :
success – returns instance of *ReaderIDInfo* representing information received from reader.
error – *Exception*.
Description :
Blocks execution until a new Heartbeat is received from script. This function can be awaited. Token provided as parameter can be used to cancel the asynchronous call.

RAW Data

This chapter describes how to get RAW data sent by the script running in the RFID reader.

Note: Once data recovered from SpcInterfaceControl, the received data is reset, so that one call to obtain RAW data only succeeds when new data successfully is received by SpcInterfaceControl since last recovered. This applies for all the functions/properties in this chapter.

public RawDataReceived DataReceived

Accessors :
get / -

Description :
(Property) Returns an instance of *RawDataReceived* representing the last received RAW data (or *null* if nothing received) and returns immediately with the result. Calling this property resets the received data, so that one call to this property only succeeds when data successfully received since last recovered by the software.

public async Task<RawDataReceived> GetDataReceivedAsync ()

Parameter :
-

Result :
success – returns instance of *RawDataReceived* representing raw data last received from reader.
error – *Exception*.

Description :
Blocks execution until new RAW data is received from script. This function can be awaited. Once data recovered from *SpcInterfaceControl*, the received data is reset, so that one call to obtain RAW data only succeeds when new data successfully is received by *SpcInterfaceControl* since last recovered.

public async Task<RawDataReceived> GetDataReceivedAsync (CancellationToken _cancelToken)

Parameter :
_cancelToken – instance of CancellationToken that may be used to cancel asynchronous call.

Result :
success – returns instance of *RawDataReceived* representing raw data last received from reader.
error – *Exception*.

Description :
Blocks execution until new RAW data is received from script. This function can be awaited. Token provided as parameter can be used to cancel the asynchronous call.

SPC Request

This chapter describes functions to send SPC requests to the script running in the RFID reader.

Note: SPC request must be implemented in the running script. Sending an SPC request is always allowed by *SpcInterfaceControl*, but only if the script supports it, will be able to receive the request and perform the request. Please check the script documentation.

public void SendSpcRequest (string _spcRequest)

Parameter :
_spcRequest – string containing the SPC request to send to the script.
Result :
-
Description :
Sends the provided SPC request to the running script.

public void SendSpcRequest (byte[] _spcRequestBytes)

Parameter :
_spcRequest – byte array containing the SPC request to send to the script.
Result :
-
Description :
Sends the provided SPC request to the running script.

Classes, Enumerations, Events and Delegates

SpcInterfaceControl properties

This chapter includes all the available properties, which can be used to get the status of the initialization process and configure some parameters.

public bool IsInitialized

Accessors :
get / -

Description :
Returns the status of the communication interface. Result is *true* if the communication interface is still open, *false* otherwise.

public bool IsInitializing

Accessors :
get / -

Description :
Returns the status of the initialization process. Result is *true* if the initialization process is still running, *false* otherwise.

public ICommunicationHandler CommunicationHandler

Accessors :
get / -

Description :
Returns the internal instance of *ICommunicationHandler* used to communicate with the RFID reader. This property may be used to initialize a new instance of *SpcInterfaceControl* that communicates using the same host-side peripheral interface.

public string DataPrefix

Accessors :
get / set

Description :
Access the selected prefix currently being used in *SpcInterfaceControl* class to decode the data received from the script running in the reader.

public string DataSuffix

Accessors :
get / set

Description :
Access the selected suffix currently being used in *SpcInterfaceControl* class to decode the data received from the script running in the reader.

SpcInterfaceControl delegates

This chapter describes delegates that define events used in this *SpcInterfaceControl* class.

public delegate void InitializeCompletedEventHandler (object _sender, bool _portOpen)

Parameter :
_sender – object instance where the event handler is attached.
_portOpen – true if the communication port is open as result of *Initialize* function.
Result :
-
Description :
Represents a method that will handle the result to an Initialize process.

public delegate void ReaderHeartbeatEventHandler (object _sender, ReaderHeartbeat _heartbeat)

Parameter :
_sender – object instance where the event handler is attached.
_heartbeat – instance of *ReaderHeartbeat* just received by *SpcInterfaceControl*
Result :
-
Description :
Represents a method that will handle the received Heartbeat information.

public delegate void RawDataReceivedEventHandler (object _sender, RawDataReceived _rawData)

Parameter :
_sender – object instance where the event handler is attached.
_rawData – instance of *RawDataReceived* just received by *SpcInterfaceControl*
Result :
-
Description :
Represents a method that will handle the received RAW data.

SpcInterfaceControl events

This chapter describes events defined in *SpcInterfaceControl* class.

public event InitializeCompletedEventHandler InitializeCompleted

Description :

This event is raised when Initialize process is completed after calling *StartInitialize* function.

public event ReaderHeartbeatEventHandler ReaderHeartbeatReceived

Description :

This event is raised when a Heartbeat is received by *SpcInterfaceControl*.

public event RawDataReceivedEventHandler RawDataReceived

Description :

This event is raised when new RAW data is received by *SpcInterfaceControl*.

Utils classes

This chapter describes classes used as return types of functions described above.

iIDReaderLibrary.SpcInterfaceFunctions.ReaderHeartbeat

This class contains information about the Reader.

Type	Property name	Description
DateTime	Timestamp	Instance of DateTime representing the exact time when the Heartbeat represented in this class instance was received
int	ReaderID	Reader-ID (Serial number) of the RFID reader
BatStatus	BatteryStatus	Enum value representing the device battery status
string	ScriptName	Name of the script running in the RFID reader

iIDReaderLibrary.SpcInterfaceFunctions.RawDataReceived

This class contains information about the RAW data received from the script running in RFID reader.

Type	Property name	Description
DateTime	Timestamp	Instance of DateTime representing the exact time when the Heartbeat represented in this class instance was received
int	ReaderID	Reader-ID (Serial number) of the RFID reader
string	Data	Text representing the data received from the script running in the RFID reader

iIDReaderLibrary.Utils.InterfaceCommunicationSettings

This class provides a series of functions needed to initialize the interface communication.

ICommunicationHandler InitCommHandler ()

Parameter :
-

Result :
Instance of class that implements *ICommunicationHandler* interface.

Description :
Initializes a new instance of a class that implements *ICommunicationHandler* and returns it as a result.

static string[] GetAvailablePortNames ()

Parameter :
-

Result :
Array of string containing port names.

Description :
Reads the serial port names available in host device and populates array of string. These names can be used to initialize a new *SpcInterfaceControl* instance.

static InterfaceCommunicationSettings GetForUsbDevice ()

Parameter :

-

Result :

Instance of *InterfaceCommunicationSettings* class for USB communication.

Description :

Initializes a new instance of *InterfaceCommunicationSettings* with default baud rate of 57600 to pass to *SpcInterfaceControl* constructor.

static InterfaceCommunicationSettings GetForSerialDevice (int _portType, string _portName)

Parameter :

_portType – number representing the port type. See *PortTypeEnum*
_portName – name of the serial port (not needed for USB port type)

Result :

Instance of *InterfaceCommunicationSettings* class representing the given parameters.

Description :

Initializes a new instance of *InterfaceCommunicationSettings* with default baud rate of 57600 to pass to *SpcInterfaceControl* constructor.

static InterfaceCommunicationSettings GetForLanDevice (int _portType, IPEndPoint _localEndPoint, IPEndPoint _remoteEndPoint, uint _pcanLocalAddr, uint _pcanRemoteAddr)

Parameter :

_portType – number representing the port type. See *PortTypeEnum*
_localEndPoint – instance of System.Net.IPEndPoint representing the host device IP and port.
_remoteEndPoint – instance of System.Net.IPEndPoint representing the reader IP and port number.
_pcanLocalAddr – PCAN address to use for host device.
_pcanRemoteAddr – PCAN address of the reader.

Result :

Instance of *InterfaceCommunicationSettings* class representing the given parameters.

Description :

Initializes a new instance of *InterfaceCommunicationSettings* to pass to *SpcInterfaceControl* constructor.

Enumerations

This chapter describes enumerations that are used in *SpcInterfaceControl*.

iIDReaderLibrary.SpcInterfaceFunctions.BatStatus

This enumeration contains values for possible battery status values.

Name
UNKNOWN
PERFECT
GOOD
MEDIUM
LOW